UNIVERSIDAD DE COSTA RICA
SISTEMA DE ESTUDIOS DE POSGRADO

APLICACIÓN DE CIENCIA DE DATOS PARA PREDICCIÓN DEL TRAFICO

Trabajo final de investigación aplicada sometido a la consideración de la Comisión del Programa de Estudios de Posgrado en Ingeniería Civil para optar al grado y título de Maestría Profesional en Ingeniería de Transportes y Vías

NATALIA RAQUEL MARIN VILLALOBOS

Ciudad Universitaria Rodrigo Facio, Costa Rica

2021

Dedicatoria

A Tomás y Esteban, son lo que más amo en el mundo.

"Bienaventurado el varón que no anduvo en consejo de malos,
Ni estuvo en camino de pecadores,
Ni en silla de escarnecedores se ha sentado;

Sino que en la Ley de Jehová esta su delicia,
Y en su Ley medita de día y de noche.

Será como árbol plantado junto a corrientes de aguas,
Que da su fruto a su tiempo,
Y su hoja no cae,
Y todo lo que hace, prosperará"

Salmo 1:1-3

¡Recuérdelo siempre!

Agradecimientos

Al eterno y único Dios, gracias por darme todo lo que necesito en el mundo y en el cielo para ser realmente feliz en las bases de la cruz y otros cientos de detalles, aquí uno de ellos, concedidos por tu amor que nunca falla y se renueva cada mañana.

A Ze, gracias por tantos días y noches que te encargaste de los chicos, de la casa y de mí para cumplir esta meta. Sin tu ayuda y apoyo esto seguiría siendo un sueño. Gracias por creer en nosotros en la luz y en la oscuridad, entre victorias y fracasos, entre risas y lágrimas.

A Tomás y Esteban, gracias por tratar de compartir y apoyar mis deseos de superación, aunque a su corta edad no lo entiendan por completo. Gracias por llorar mi ausencia y extrañar mis abrazos, si Dios me lo permite, les juro recompensarlo.

A mis padres, Edgar y Maritza, gracias por ser incondicionales, sin fallar un solo segundo, aun antes de conocerme.

A mis hermanos César, Daniela y Roberto, gracias por ser mi referencia de bondad, superación y empeño, son el mejor regalo de nuestros padres.

A Rodrigo, gracias por tu ayuda, empeño y dedicación a este proyecto, pero, sobre todo, por una amistad invaluable que trasciende el tiempo y la distancia.

A mis amigos y compañeros de maestría: Jairo, Olver, Martín, Jorge y Juan Carlos, gracias por tantas horas de esfuerzo, apoyo y enseñanzas. Espero que sigamos en contacto. Jairo, usted delo por un hecho.

A mi Comité Asesor, gracias por su apoyo y dedicación a este proyecto y aún más por las lecciones dadas a lo largo de esta maestría han sido y siempre serán de gran provecho.

Al resto de mi familia y amigos, gracias por darme esos escapes a pies descalzos, llenos de risas y cariño, con aroma a café, miel y carbón.

Al equipo CONVIA, gracias por el apoyo, la milla extra y por ser parte de la materialización de todo lo aprendido en este proceso en función de ser cada día mejores.

A la memoria del Ing. Jorge Picado Abarca, siempre le recordaré como mi primer tutor y quien me introdujo al fascinante mundo de la ingeniería vial y cuya dedicación me ha dado tantas oportunidades y privilegios.

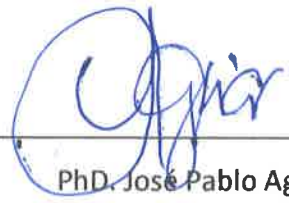
Hoja de aprobación

"Este trabajo final de investigación aplicada fue aceptado por la Comisión del Programa de Estudios de Posgrado en Ingeniería Civil de la Universidad de Costa Rica, como requisito parcial para optar al grado y título de Maestría Profesional en Ingeniería de Transportes y Vías"

Dr. Alberto Serrano Pacheco

Representante del Decano

Sistema de Estudios de Posgrado



PhD. José Pablo Aguiar Moya

Lector

M.Sc. Tania Ávila Esquivel

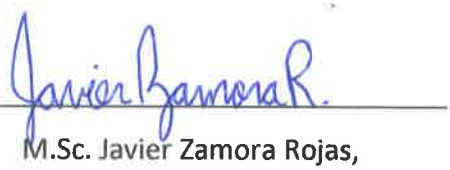
Profesora Guía

PhD. Jonathan Agüero Valverde

Lector

Mag. Josué Quesada Campos

Lector



M.Sc. Javier Zamora Rojas,

Representante del Director del Programa de Posgrado en Ingeniería Civil

Natalia Raquel Marín Villalobos

Sustentante

Contenido

Dedicatoria ii

Agradecimientos iii

Hoja de aprobación iv

Contenido v

Resumen viii

Índice de Figuras ix

Índice de Tablas x

1. Generalidades 1

1.1. Introducción 1

1.2. Justificación de realizar el trabajo 2

1.2.1. Problema específico 2

1.2.2. Importancia 7

1.3. Objetivos 9

1.3.1. Objetivo General 9

1.3.2. Objetivos Específicos 9

1.4. Delimitaciones del problema 10

1.4.1. Alcances 10

1.4.2. Limitaciones 11

1.5. Metodología 12

1.6. Antecedentes teóricos y prácticos del problema 14

1.7. Marco Teórico 19

1.7.1. Ciencia de datos 20

1.7.2. Regresión lineal múltiple 22

1.7.3. Regresión de Ridge 25
1.7.4. Árbol de regresión 25
1.7.5. Series de tiempo 27
1.7.6. Análisis de componentes 28
1.7.7. Importancia de las variables 31
2. Definición y descripción de las variables 33
2.1. Descripción de la variable dependiente 33
2.2. Descripción de las variables independientes 40
2.2.1. Variables Internas 40
2.2.2. Variables Externas 50
3. Modelo de proyección de tráfico 60
3.1. Definición de las series de modelos 60
3.2. Construcción del modelo 64
3.2.1. Revisión de la variable dependiente 64
3.2.2. Revisión de las variables independientes 66
3.2.2.1. Reducción de dimensión por componentes principales PCA 66
3.2.2.2. Importancia de las variables 68
3.2.3. División de la base de datos 69
3.3. Resultados de las series 69
3.3.1. Resultados de la Serie de Modelos A 70
3.3.2. Resultados de la Serie de Modelos B 73
3.3.3. Resultados de la Serie de Modelos C 76
3.4. Comparación de las series y modelos evaluadas 79
3.5. Modelo Final 81

3.6. Análisis de elasticidad 83

4. Conclusiones 88

Referencias bibliográficas: 91

Apéndice A: Memoria de cálculo Python para la Serie de Modelos A 94

Apéndice B: Memoria de cálculo Python para la Serie de Modelos B 123

Apéndice C: Memoria de cálculo Python para la Serie de Modelos C 154

Apéndice D: Memoria de cálculo Python para la Serie de Modelos D 177

Resumen

El presente proyecto utiliza herramientas de ciencia de datos para generar un modelo de predicción de tráfico basado en el desarrollo de diferentes usos de suelo y su aporte en la generación de viajes de una zona, así como algunos indicadores económicos generales. De esta manera se espera que el tráfico vehicular no solo se estime mediante una proyección geométrica basada en datos históricos del tráfico en una sección de la red vial, logrando así cuantificar el impacto de un desarrollo particular o una situación económica especifica, en el Tránsito Promedio Diario (TPD) del lugar de análisis.

Específicamente, se considera la Radial Siquiares – Coyol como sitio de estudio, sin embargo, es posible que, bajo una adecuada calibración del modelo, este pueda ser utilizado en otros sectores del territorio nacional que compartan características sociodemográficas y económicas similares a las de la zona del Coyol cercana a dicha radial.

Se escoge la zona del Coyol de Alajuela, particularmente los distritos de La Garita, Turrucares, San José y San Antonio porque son los aledaños a la radial en estudio y estos lugares ha experimentado un crecimiento muy acelerado del tráfico vehicular debido al desarrollo industrial y económico que presenta la zona desde la apertura de la Radial Coyol en el año 2012 como parte del proyecto de Concesión San José – Caldera, adicionado a las excelentes condiciones territoriales de la zona.

Si bien es necesario el desarrollo de nuevas fuentes de emprendimiento y empleo, este debe ser sostenible en cuanto la estructura vial que demanda. De acuerdo con estudios realizados a la fecha, la zona presenta problemas serios de congestión vehicular, lo cual es preocupante porque aún queda mucho potencial de desarrollo en el sitio.

Siendo que esta es una de las zonas que mayor aporta a la economía del país y su potencial de desarrollo es muy amplio, la determinación de los efectos del desarrollo urbano y los cambios producidos por la dinámica económica del país sobre la infraestructura vial que sirve al sector, es fundamental para demostrar la necesidad de priorización de soluciones viales que permitan mantener el crecimiento que tanto favorece a nuestro país.

Entre las variables se incluyen están: diferentes usos de suelo, aspectos demográficos y algunos indicadores económicos. Siendo el uso de suelo industrial y los indicadores económicos los que tienen mayor influencia en el modelo, según los resultados obtenidos de la aplicación de las herramientas de ciencia de datos, por lo que se logra presentar un modelo que considera más que una tendencia histórica y se ajusta al desarrollo y a la elasticidad de la condición económica en la que vivimos lo cual permite valorar diferentes alternativas y priorizar proyectos.

Índice de Figuras

Figura 1. 1: Mapa de ubicación de La Radial Siquiares – Coyol.....2
Figura 1. 2. Colas vehiculares y niveles de servicio en las rotondas del Intercambio Coyol..3
Figura 1. 3. Colas vehiculares en la Radial Coyol Rotonda Sur del Intercambio Coyol.....4
Figura 1. 4. Colas vehiculares desde San José en la Rotonda Norte del Intercambio Coyol..4
Figura 1. 5. Desarrollo de la zona de El Coyol en los últimos 10 años.....5
Figura 1. 6. Extracto del mapa del Plan Regulador de la Municipalidad de Alajuela 6
Figura 1. 7. Representación de la capacidad vial de la radial y su relación con el TPD proyectado.....8
Figura 1. 8. Descripción de la metodología 14
Figura 1. 9. Ciclo de Vida Analítico Iterativo.....20
Figura 1. 10. Proceso del ciclo de vida analítico 21
Figura 1. 11. Enfoque gráfico de los supuestos de los modelos de regresión lineal.....24
Figura 1. 12. Tipos de ajustes en las regresiones 25
Figura 1. 13. Explicación geométrica de los PCA 30

Figura 2. 1. Comportamiento del tráfico vehicular de livianos sobre Radial Coyol 36
Figura 2. 2. Comportamiento del tráfico buses sobre Radial Coyol 37
Figura 2. 3. Comportamiento del tráfico vehículos sobre Radial Coyol 38
Figura 2.4. Gráfico resultante de la limpieza de datos de la variable dependiente mediante la aplicación Anaconda 39
Figura 2.5. Gráfico resultante de la variable dependiente con los datos faltantes imputados 39
Figura 2. 6. Ubicación de los distritos a considerar en el análisis 41
Figura 2.7. Gráfico del Comportamiento de las variables internas medidas mensualmente 46
Figura 2. 8. Población por distrito y total de los distritos considerados 47
Figura 2. 9. Distancia entre la cabecera distrital y la radial.....49

Figura 2. 10. Comportamiento de las variables externas medidas mensualmente – relacionadas con cantidad de vehículos 52
Figura 2. 11. Comportamiento de las variables externas medidas mensualmente – relacionadas con indicadores económicos 55
Figura 2. 12. Comportamiento de las variables externas medidas mensualmente – relacionadas con el precio de los combustibles y tipo de cambio del dólar 58

Figura 3. 1. Esquema del procedimiento de aplicación, análisis y selección del modelo de proyección 63
Figura 3. 2. Comprobación de normalidad de la variable dependiente 65
Figura 3. 3. Representación gráfica del árbol de regresión 73
Figura 3. 4. Gráfico de los valores predichos y estimados para la base de entrenamiento. 82
Figura 3. 5. Gráfico de los valores predichos y estimados para la base de prueba 83

Índice de Tablas

Tabla 1. 1. Cálculo de la capacidad a saturación de la Radial Coyol 7
Tabla 1. 2. TPD y tasas de crecimiento anual observadas en la radial 8

Tabla 2. 1. Promedio mensual del TPD de vehículos livianos de la base de datos original .. 36
Tabla 2. 2. Promedio mensual del TPD de buses de la base de datos original 37
Tabla 2. 3. Promedio mensual del TPD de vehículos pesados de la base de datos original 38
Tabla 2.4. Cantidad de unidades habitaciones tipo casa tramitadas por mes 42
Tabla 2.5. Cantidad de m^2 tipo condominio tramitadas por mes 43
Tabla 2.6. Cantidad de industrias en m^2 tramitadas por mes 43
Tabla 2.7 Cantidad de comercio en m^2 tramitados por mes 44
Tabla 2.8. Cantidad de servicios en m^2 tramitados por mes 45
Tabla 2. 9. Población del área de influencia 47
Tabla 2. 10. Factores de ajuste población/ distancia 48
Tabla 2. 11. TPD de la Ruta Nacional 1 para el periodo de análisis 51

Tabla 2. 12. Cantidad de vehículos importados por mes en Costa Rica 51
Tabla 2. 13. Índice Mensual de Precio al Consumidor 53
Tabla 2. 14. Índice Mensual de Actividad Económica 54
Tabla 2. 15. Precio de la gasolina súper 54
Tabla 2. 16. Precio de la gasolina regular 56
Tabla 2. 17. Precio del diésel 56
Tabla 2. 18. Tipo de cambio del dólar 57
Tabla 2. 19. Variables independientes consideradas 59

Tabla 3. 1. Reducción de variables por PCA 67
Tabla 3. 2. Resultado de la aplicación del método “Importancia de las Variables” 68
Tabla 3. 3. Resumen de resultados estadísticos de la Regresión lineal múltiple 70
Tabla 3. 4. Resumen de resultados estadísticos de la Regresión lineal de Ridge 71
Tabla 3. 5. Resumen de valores propuestos y utilizados para los hiperparámetros del árbol de regresión para la Serie de Modelos A 72
Tabla 3. 6. Resumen de resultados estadísticos de la Regresión lineal múltiple 74
Tabla 3. 7. Resumen de resultados estadísticos de la Regresión lineal de Ridge 75
Tabla 3. 8. Resumen de resultados estadísticos de la Regresión lineal múltiple 77
Tabla 3. 9. Resumen de resultados estadísticos de la Regresión lineal de Ridge 78
Tabla 3. 10. Comparación de los R^2 de los diferentes modelos y series evaluadas 79
Tabla 3. 11. Resultados del TPD estimado por el 80
Tabla 3. 12. Resultados de la Prueba F 80
Tabla 3. 13. Resultados estadísticos del modelo final 81
Tabla 3. 14. Escenarios para ser analizados 84
Tabla 3. 15. Resultados de la aplicación del Modelo Final 85

UNIVERSIDAD DE COSTA RICA

SEP Sistema de Estudios de Posgrado

Autorización para digitalización y comunicación pública de Trabajos Finales de Graduación del Sistema de Estudios de Posgrado en el Repositorio Institucional de la Universidad de Costa Rica.

Yo, Natalia Raquel Marín Villalobos, con cédula de identidad 205680452, en mi condición de autor del TFG titulado Aplicación de la ciencia de datos para predicción del tráfico

Autorizo a la Universidad de Costa Rica para digitalizar y hacer divulgación pública de forma gratuita de dicho TFG a través del Repositorio Institucional u otro medio electrónico, para ser puesto a disposición del público según lo que establezca el Sistema de Estudios de Posgrado. **SI** [ ] **NO** * [X]

***En caso de la negativa favor indicar el tiempo de restricción:** 2 **año (s).**

Este Trabajo Final de Graduación será publicado en formato PDF, o en el formato que en el momento se establezca, de tal forma que el acceso al mismo sea libre, con el fin de permitir la consulta e impresión, pero no su modificación.

Manifiesto que mi Trabajo Final de Graduación fue debidamente subido al sistema digital Kerwá y su contenido corresponde al documento original que sirvió para la obtención de mi título, y que su información no infringe ni violenta ningún derecho a terceros. El TFG además cuenta con el visto bueno de mi Director (a) de Tesis o Tutor (a) y cumplió con lo establecido en la revisión del Formato por parte del Sistema de Estudios de Posgrado.

INFORMACIÓN DEL ESTUDIANTE:

Nombre Completo: Natalia Raquel Marín Villalobos.

Número de Carné: A02481 Número de cédula: 205680452.

Correo Electrónico: nmarin@convia.net.

Fecha: 29 de setiembre de 2021. Número de teléfono: 88314634.

Nombre del Director (a) de Tesis o Tutor (a): Ing. Tania Ávila Esquivel.

FIRMA ESTUDIANTE

Nota: El presente documento constituye una declaración jurada, cuyos alcances aseguran a la Universidad, que su contenido sea tomado como cierto. Su importancia radica en que permite abreviar procedimientos administrativos, y al mismo tiempo genera una responsabilidad legal para que quien declare contrario a la verdad de lo que manifiesta, puede como consecuencia, enfrentar un proceso penal por delito de perjurio, tipificado en el artículo 318 de nuestro Código Penal. Lo anterior implica que el estudiante se vea forzado a realizar su mayor esfuerzo para que no sólo incluya información veraz en la Licencia de Publicación, sino que también realice diligentemente la gestión de subir el documento correcto en la plataforma digital Kerwá.

1. Generalidades

1.1. Introducción

Los métodos de “Ciencia de datos” son una herramienta que integra matemática avanzada, estadística, análisis de datos e informática para el desarrollo de algoritmos, que pueden ser utilizados para predecir cambios en los volúmenes vehiculares de la red vial tras la ocurrencia eventos a los que el comportamiento de los volúmenes vehiculares de la red vial es sensible. Los modelos de proyección de tráfico son fundamentales para el diagnóstico y predicción del comportamiento de los flujos vehiculares, con lo cual se pueden tomar decisiones inteligentes de inversión en infraestructura, desarrollo comercial, industrial, empresarial, entre otros. El Coyol de Alajuela es una zona de un alto crecimiento económico debido a que, por sus características territoriales, ha experimentado un desarrollo principalmente industrial acelerado, el cual está vinculado estrechamente con el crecimiento de la flota vehicular en sus alrededores.

Aún con todo el crecimiento experimentado en los últimos años en esta zona, se sigue teniendo un altísimo potencial de desarrollo, estando en la mira de inversionistas de gran escala. Para un país en desarrollo, se requiere lograr un crecimiento socioeconómico y mejor calidad de vida de la población acompañado de las mejores condiciones para la apertura y realización de actividades económicas y comerciales, la capacidad adecuada de la infraestructura vial es una de las condiciones indispensables para dicho desarrollo. “Un buen control de la economía de un país se ve reflejado en la calidad de sus sistemas viales, de ahí la importancia de mantener en estado óptimo las carreteras como médula del desarrollo de un país” (Katiyar Quiros, 2018). Esta investigación puede convertirse en una herramienta de control para el emplazamiento de desarrollos comerciales e industriales, considerando además otros aspectos sociales y económicos del sector o el país en general.

1.2. Justificación de realizar el trabajo

Para justificar el presente trabajo se presenta el problema específico y la importancia de contar con una herramienta de predicción para la toma de decisiones importantes en cuanto inversión en infraestructura vial o bien la factibilidad de continuar con el desarrollo de una zona bajo las condiciones de la red vial que le acoge en la actualidad.

1.2.1. Problema específico

En la actualidad, ante una infraestructura vial limitada, excluyente y con un alto nivel de congestión vehicular, la zona de El Coyol queda sin capacidad de reserva para enfrentar el flujo vehicular producto de nuevos desarrollos, los cuales seguirán surgiendo pese a las dificultades de los permisos de accesos y disponibilidad de agua, lo cual implicará más congestión y limita el ansiado desarrollo de nuestro país, debilitando oportunidades de crecimiento y bienestar para sus habitantes. En la Figura 1.1 se muestra la ubicación de la radial en estudio y algunos puntos de relevancia sobre ella.

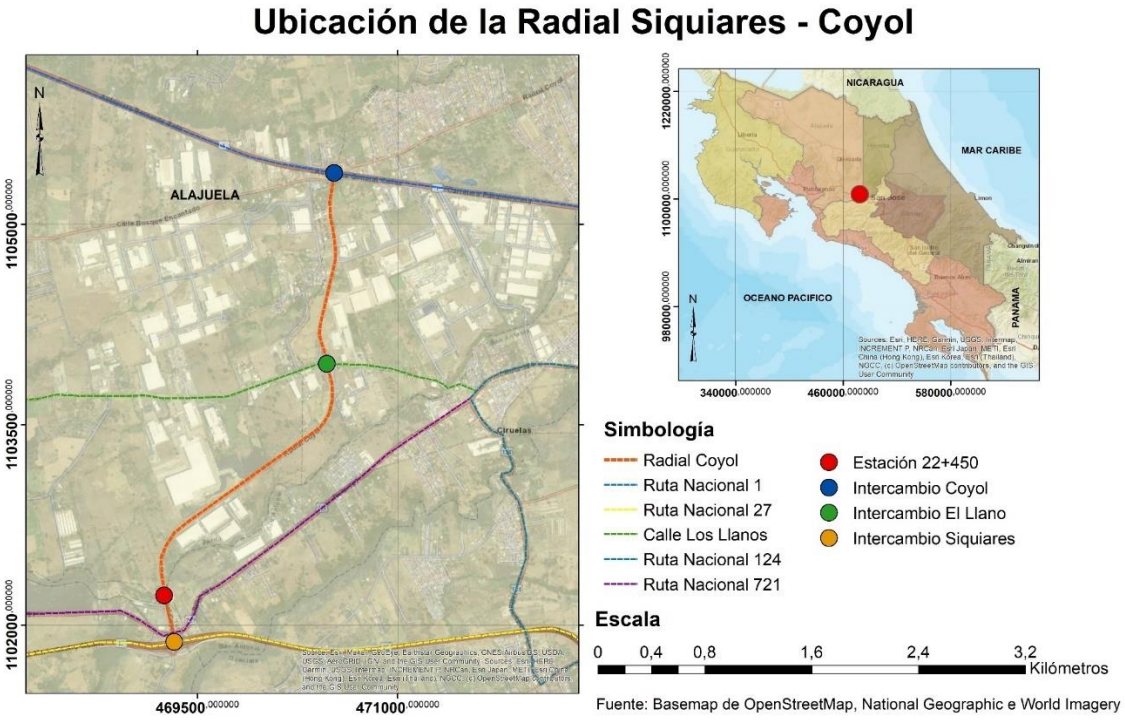


Figura 1. 1: Mapa de ubicación de La Radial Siquiares – Coyol

De acuerdo con estudios realizados por el autor (Marín, 2019), en la zona de El Coyol, se tienen demoras de hasta 40 minutos para ingresar a la Radial Coyol y colas superiores a los 3 km en el ingreso desde San José y desde la Radial Coyol. En la Figura 1.2 se muestra gráficamente mediciones realizadas y en las Figuras 1.3 y 1.4 se muestra dicha congestión mediante fotografías en el sitio. Totalmente concordante con el crecimiento de la flota vehicular generalizado para Costa Rica, donde desde 1990 ya se alcanzaban relaciones volumen/capacidad de 1,1 (Castro, Picado, & Rodríguez, 2018). Cabe mencionar que la congestión vehicular implica en Costa Rica, un gasto estimado en un 3.8% del PIB (Programa Estado de la Nación, 2018) y que el 75% de las emisiones de dióxido de carbono (CO2) provienen de dicha congestión (Universidad de Costa Rica, 2018). Sin dejar de lado la más grave externalidad negativa del transporte, los accidentes de tránsito que para el año 2015 ocuparon la octava causa de muerte de los costarricenses (Instituto Nacional de Estadística y Censo, 2020).

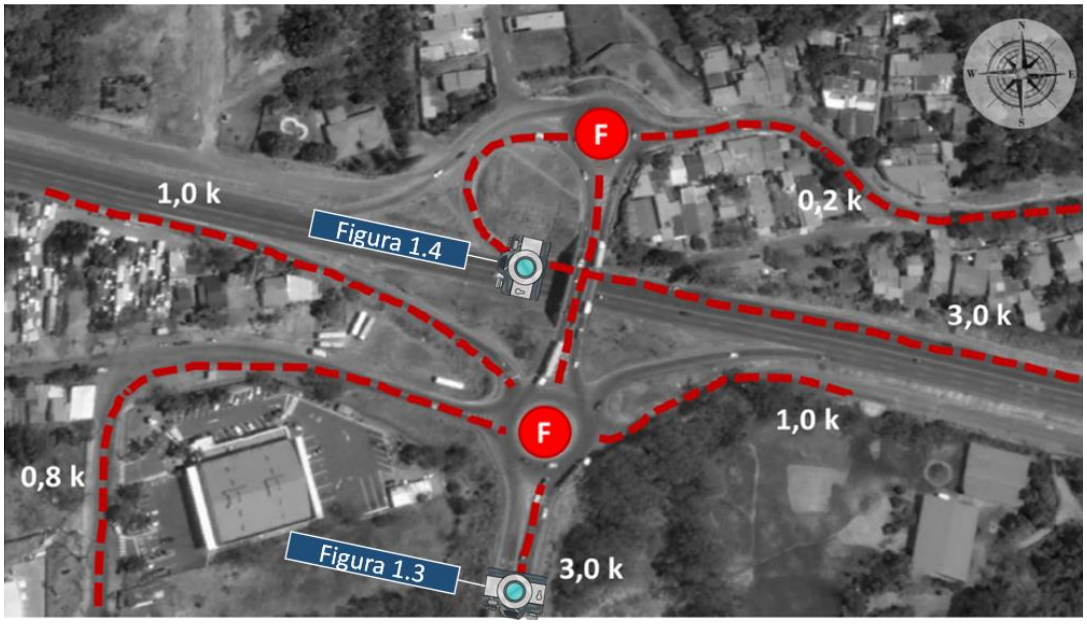


Figura 1. 2. Colas vehiculares y niveles de servicio en las rotondas del Intercambio Coyol

Fuente: (Marín, 2019)

Figura 1. 3. Colas vehiculares en la Radial Coyol Rotonda Sur del Intercambio Coyol

Fuente: (Marín, 2019)

Figura 1. 4. Colas vehiculares desde San José en la Rotonda Norte del Intercambio Coyol

Fuente: (Marín, 2019)

Esto se debe a un crecimiento acelerado del sector industrial y empresarial, producto de las excelentes condiciones territoriales de la zona como: topografía principalmente regular, clima cálido, cercanía con rutas importantes, aproximación con el Puerto de Caldera y al Aeropuerto Internacional Juan Santamaría, entre otras. En la Figura 1.5 se muestra una imagen área de la zona con el paso de los años donde claramente se percibe el aumento de área techada. Esto es concordante con el Plan Regulador de la Municipalidad de Alajuela, en el cual se destinó una fracción importante de terreno al desarrollo industrial tal como se puede observar en la Figura 1.6 que presenta un extracto del mapa de dicho plan.

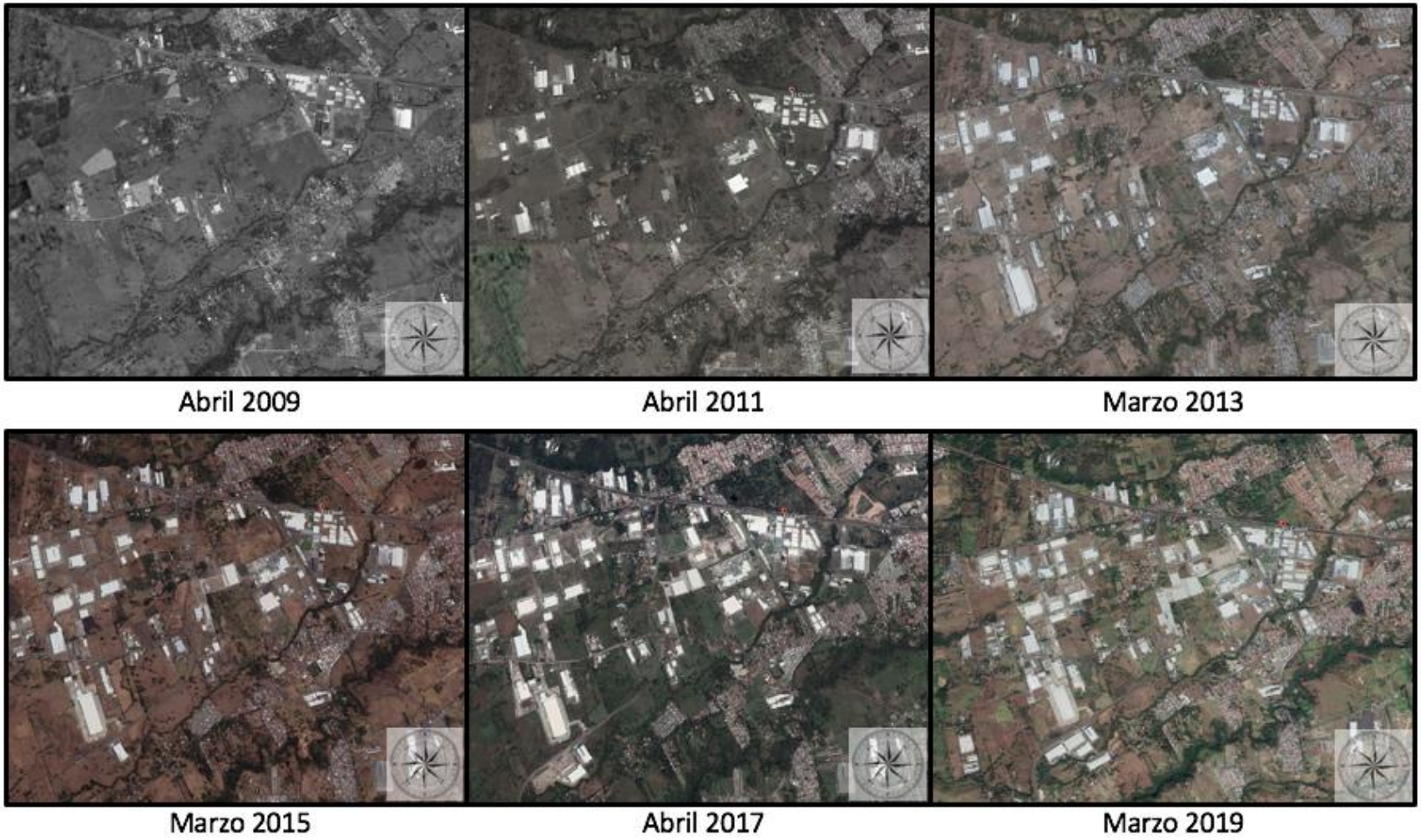


Figura 1. 5. Desarrollo de la zona de El Coyol en los últimos 10 años

Fuente: Elaboración propia utilizando las imágenes de Google Earth

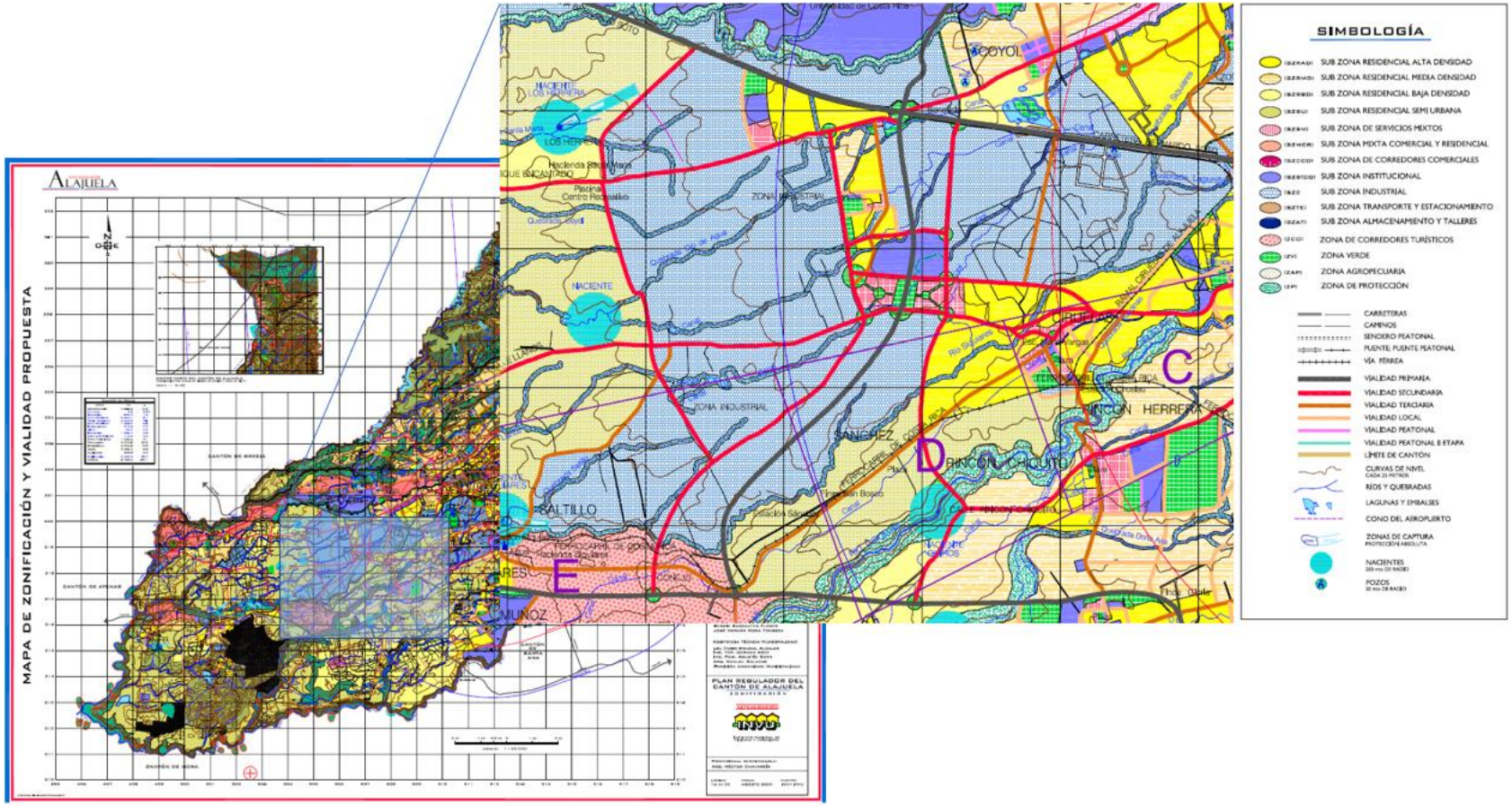


Figura 1. 6. Extracto del mapa del Plan Regulador de la Municipalidad de Alajuela

Fuente: (Municipalidad de Alajuela, 2019)

1.2.2. Importancia

Cabe mencionar que tras hacer una estimación de la capacidad de la radial de acuerdo con la metodología del Manual de Capacidad de Carreteras (Transportation Research Board, 2016) se tiene que la radial tiene una capacidad a saturación aproximada de 41872 veh/d. En la Tabla 1.1 se muestra un resumen del cálculo de capacidad por hora por carril (veh/h/c), este cálculo se basa en la capacidad ideal de un carril según la metodología indicada estimado en 1700 veh/h/c, la cual debe ser multiplicada por el factor de ajuste por pendiente y por el factor de ajuste por pesados, que a su vez dependen del volumen vehicular en la hora pico máxima (alrededor de 1000 veh/h en ambos sentidos para las horas pico, específicamente el percentil 85 de los datos horarios indica 953 veh/h) y del tipo de terreno (ondulado), lo cual fue definido en el levantamiento de campo realizado, obteniéndose la capacidad por carril que debe ser multiplicada por la cantidad de carriles, dos en este caso y dividida entre el factor de concentración máxima horaria obtenido del Anuario de Tránsito 2019 (Secretaria de Planificación Sectorial, 2020). Como se verá más adelante, este valor es muy importante en la definición del modelo de proyección de tránsito.

Tabla 1. 1. Cálculo de la capacidad a saturación de la Radial Coyol

Capacidad ideal (veh(/h)	Factor por pendiente	Factor por pesados	Cantidad de Carriles	Capacidad real (veh/h)	Factor de concentración horaria	Capacidad real diaria (veh/d)
1700	0,93	0,87	2	2764	0,066	**41872**

Al comparar el perfil 85 de los datos de volumen horario con la capacidad, queda claro que la estimación de esta es adecuada. En la Tabla 1.2 se muestran los registros históricos anuales de la Estación 22+450 (Ver ubicación en la Figura 1.1), según la base de datos aportada por Global Vía para esta investigación, adicionalmente se muestra la proyección de tráfico según la tendencia logística dada por estos datos históricos y la capacidad de la vía indicada en la Tabla 1.1. Adicionalmente se presentan las tasas de crecimiento que se han presentado en la zona y se esperarían tener en los próximos años. Finalmente, en Figura 1.7 se muestra la capacidad representada por una línea roja horizontal y el crecimiento

esperado de la zona de modo que, según esta proyección, para el año 2044 la radial estaría alcanzando su capacidad de saturación para una relación V/C superior a 0,9.

Tabla 1. 2. TPD y tasas de crecimiento anual observadas en la radial

Año Registro TPD	TPD Observada	Tasa Crecimiento Observada (i)
2013	11309	
2014	13087	15,72%
2015	13282	1,49%
2016	14919	12,32%
2017	16695	11,90%
2018	16537	-0,95%
2019	16848	1,88%

Fuente: Base de datos Global Vía

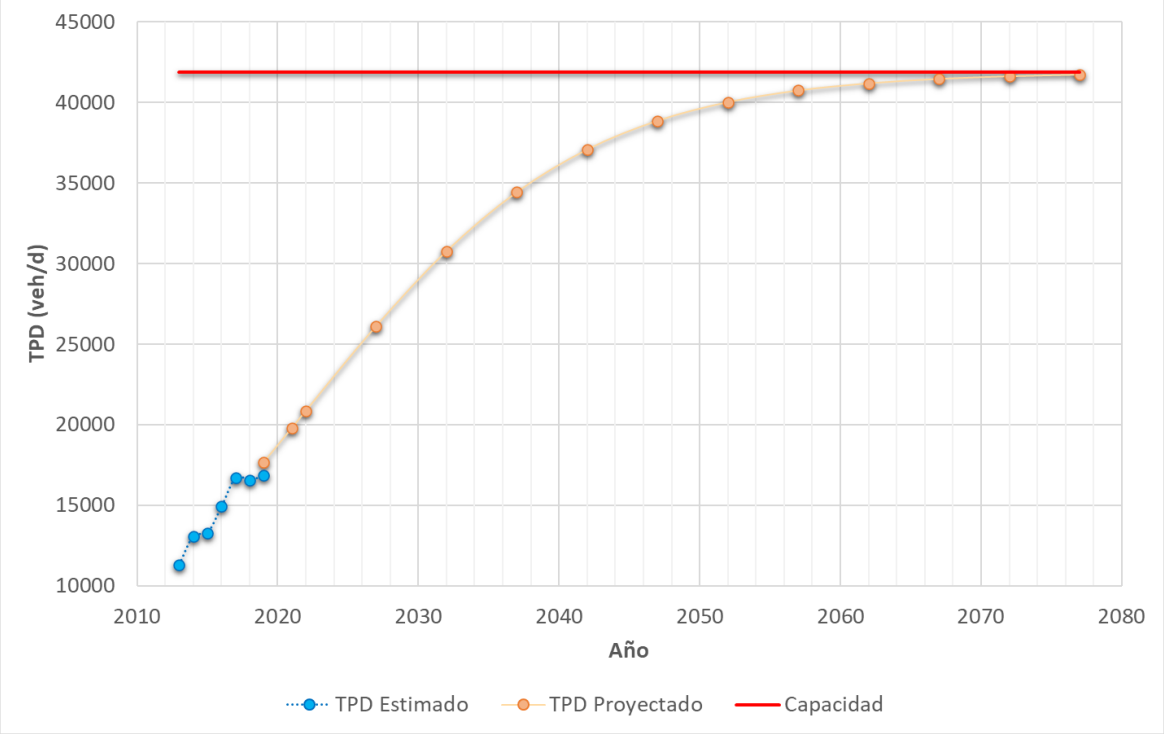


Figura 1. 7. Representación de la capacidad vial de la radial y su relación con el TPD proyectado

Esta situación es alarmante por la importancia económica que tiene esta zona para el país, solo la zona franca Coyol Free Zone generó al país $1.430 millones en exportaciones en

2017, esto equivale al 28% de las exportaciones de todo el régimen de zonas francas y al 0,9% del Producto Interno Bruto del país (Coyol Free Zone, 2019).

De ahí la importancia de que se cuente con una herramienta de estimación de tráfico que permita tomar decisiones de priorización e inversión, tanto en infraestructura vial como en el desarrollo de diferentes usos de suelo de manera sostenible.

Adicionalmente, es posible que esta herramienta se pueda calibrar para otras zonas del país siendo que Costa Rica es un país pequeño y mantiene esquemas similares de red vial y generación de viales en la mayoría del territorio nacional.

1.3. Objetivos

Los objetivos que persigue el proyecto son:

1.3.1. Objetivo General

Aplicar los métodos de ciencia de datos para modelar y predecir el comportamiento vehicular en la Radial Coyol en el Coyol de Alajuela.

1.3.2. Objetivos Específicos

1. Elaborar un modelo para la proyección del tráfico que permita a los desarrolladores y la Administración valorar la factibilidad de un proyecto en función del tráfico que genera sobre la ruta en análisis, su vinculación con el uso, protección y mantenimiento de la capacidad vial de reserva, considerando la posibilidad de más proyectos inmuebles o diversos escenarios socioeconómicos de manera simultánea como estrategias de gestión de carreteras.
2. Cuantificar la relevancia de cada variable en la predicción del tráfico para identificar que usos del suelo o situaciones socioeconómicas pueden impactar más la zona y por el contrario que otros usos pueden aplazar el punto de saturación de la red vial.
3. Determinar la aplicabilidad del modelo para realizar un análisis de rentabilidad de un proyecto ante un posible aumento de su exposición comercial debido al efecto de otro

desarrollo que empieza a emplazarse, dada la posibilidad de modelar numerosos escenarios.

1.4. Delimitaciones del problema

Para definir las bases en las que se realizó el proyecto, se enumeran una serie de alcances y limitaciones que se presentan a continuación:

1.4.1. Alcances

1. El modelo desarrollado en este trabajo no es en sí mismo una solución a la congestión vehicular de la zona, es una herramienta para la toma de decisiones de inversión tanto en la adecuación de la infraestructura vial como en crecimiento urbano, comercial e industrial ante diferentes escenarios de desarrollo y variaciones socioeconómicas ya sean valoradas de manera independiente o en conjunto.
2. El análisis se realiza únicamente con tráfico vehicular sobre la Radial Coyol de Alajuela en ambos sentidos como variable dependiente.
3. Los aforos vehiculares se registran desde el año 2009 hasta el año 2019 mediante equipos de conteo mecanizado en custodia de Global Vía y consideran el sentido y la composición vehicular por tipo de vehículo (liviano, autobuses y pesado) para luego ser superpuestas en un valor total de TPD. Cabe mencionar que pese a que la radial completa se inauguró en junio del año 2012 (Mendez, 2012), antes de esa fecha existía parcialmente y por eso es que se registran datos precedentes del año 2012.
4. Las variables internas corresponden a los permisos de construcción registrados mensualmente en el Colegio Federado de Ingenieros y Arquitectos (CFIA) en los distritos de San José, San Antonio, La Garita y Turrucares del cantón de Alajuela. Al ser permisos de construcción se consideran en operación seis meses después de la fecha de registro para proyectos de vivienda y un año después para proyectos de comercio, servicios e industria.
5. Se utiliza Python como lenguaje de programación de los modelos de simulación y la aplicación Anaconda para correr dichos modelos.

6. El uso de herramientas de ciencias de datos es básico y aunque en muchos casos carezca de una fácil interpretación, son necesarios para lograr construir los modelos.

7. Cada serie se compone de un modelo de regresión lineal, un modelo de regresión de Ridge y un modelo de árbol de regresión.

8. Se generan diferentes series de modelos hasta cumplir con las expectativas del autor basados en la obtención del estadístico Coeficiente de Determinación (R^2) tanto para el modelo de entrenamiento como el de prueba. La definición de R^2 para los tres modelos seleccionados es la es la proporción de la varianza total de la variable explicada por la regresión.

9. La capacidad teórica de la radial se basa en levantamientos funcionales de la vía en análisis realizados por el autor y se considera constante en el tiempo, aunque esto no toma en cuenta el efecto del derrame las horas pico.

1.4.2. Limitaciones

1. Se cuenta con diez años de registro de TPD sobre la radial, sin embargo, hay posibilidad de vincularlo con usos de suelo a partir del año 2015 hasta el año 2019, lo que implica únicamente 60 registros mensuales utilizables. Esto representa una dificultad para el uso de algoritmos de ciencia de datos, que por definición funcionan mucho mejor para bases de datos masivas, sin embargo, no lo hace imposible.

2. Dentro del periodo de análisis (2015 - 2019), se encontraron faltantes de datos en la variable dependiente debido a afectaciones al equipo de medición, por lo que los datos de TPD requirieron ajustes para estimar los datos faltantes.

3. La cuantificación de las variables se realiza mediante información brindada por terceros por lo que la exactitud de la información y prontitud de obtención no depende únicamente del autor.

4. Dado que la información se da a nivel de permisos se consideró que todos ellos se llegaron a ejecutar y que la operación de dichos usos se dio entre seis meses y un año después de dado el permiso según el uso explicado en los alcances.

5. El tránsito promedio diario (TPD), el cual constituye la variable dependiente en los modelos a realizar, se analiza desde el año 2009 hasta el año 2019. El extremo inferior se debe a la existencia de la radial y el extremo superior a cerrar el análisis con el tiempo suficiente para realizar el proyecto en los términos requeridos en la modalidad de maestría que se aplica y considerando también las variaciones viales que se han tenido desde el primer semestre del 2020 debido a las restricciones sanitarias impuestas por la Administración en respuesta a enfrentar la pandemia por COVID-19.

1.5. Metodología

La metodología por seguir para el desarrollo de este proyecto es la que se presenta en la Figura 1.8 y básicamente sigue los siguientes pasos:

Paso 1: Revisión bibliográfica

Se inicia con la revisión bibliográfica la cual incluye: estudio de los principales conceptos de ciencia de datos, sus aplicaciones tanto en ingeniería de transportes como en otros ámbitos del desarrollo humano, los métodos de análisis de proyecciones de tráfico, las tendencias más utilizadas, y la revisión de proyectos con objetivos similares.

Paso 2: Definición del objetivo

Una vez clara la teoría alrededor de estos conceptos, se requiere definir el objetivo de la investigación, porque este propósito marca la dirección bajo la que se obtienen y ordenan las variables tanto dependientes o independientes.

Paso 3: Obtención de la variable dependiente

Posteriormente, se procede a la obtención de la variable dependiente del modelo, las cuales son obtenidos de las bases de datos de Global Vía concesionario de la Ruta Nacional 27 y que posee una estación de conteo sobre la radial en estudio (ver Figura 1.1). Esta información se procesa y ordena para obtener los valores numéricos de la variable dependiente.

Paso 4: Obtención de las variables independientes

Luego se requieren obtener las variables independientes relacionadas con los usos de suelo en las cercanías de la radial, para ello se solicita la información a la Unidad de Permisos de

Construcción de la Municipalidad de Alajuela y al Colegio Federado de Ingenieros y Arquitectos (CFIA), la información se procesa y se analiza para definir las variables independientes a ser utilizadas en el modelo. Adicionalmente, entre las variables independientes, se consideran otras variables relacionadas con indicadores económicos.

Paso 5: Elaboración de los modelos

Las bases de datos resultantes deben integrarse y ordenarse de manera que sean consistentes y asegurar que se cuenta con todos los registros. Adicionalmente deben revisarse temas de normalidad de la variable dependiente y cantidad de variables acorde con la cantidad de registros de la variable dependiente.

Teniendo las variables dependientes e independientes integradas, ordenadas y analizadas en una sola base de datos, se procede a la elaboración de una serie de modelos. Para lo cual, aleatoriamente se toma 90% de los datos, los que se conocen como datos de entrenamiento y es con ellos con los que se generan los modelos. En el Apartado 3.2.3 se retoma la discusión sobre este porcentaje de selección subjetiva.

Paso 6: Comprobación del modelo

El restante 10% de los datos, conocidos como datos de prueba, se reservan para evaluar los modelos y comparar los resultados con datos reales. A ambos conjuntos (entrenamiento y prueba) se les determina el valor de correlación estadística (R^2) para medir la exactitud de cada modelo. Si los resultados no se consideran satisfactorios, se prueba con otra serie de modelos o se realizan más ajustes a la base de datos hasta alcanzar las expectativas de satisfacción de alguno de los modelos y se escoge el que presente los mejores estadísticos.

Paso 7: Aplicaciones del modelo

Si los resultados se consideran acertados para alguno de los modelos o varios de ellos, se analizan las aplicaciones que puede dársele a este, como por ejemplo la estimación del efecto en el TPD sobre la Radial Coyol de un mismo uso de suelo en diferentes ubicaciones o el impacto en el TPD de la Radial al variar algún indicador económico.

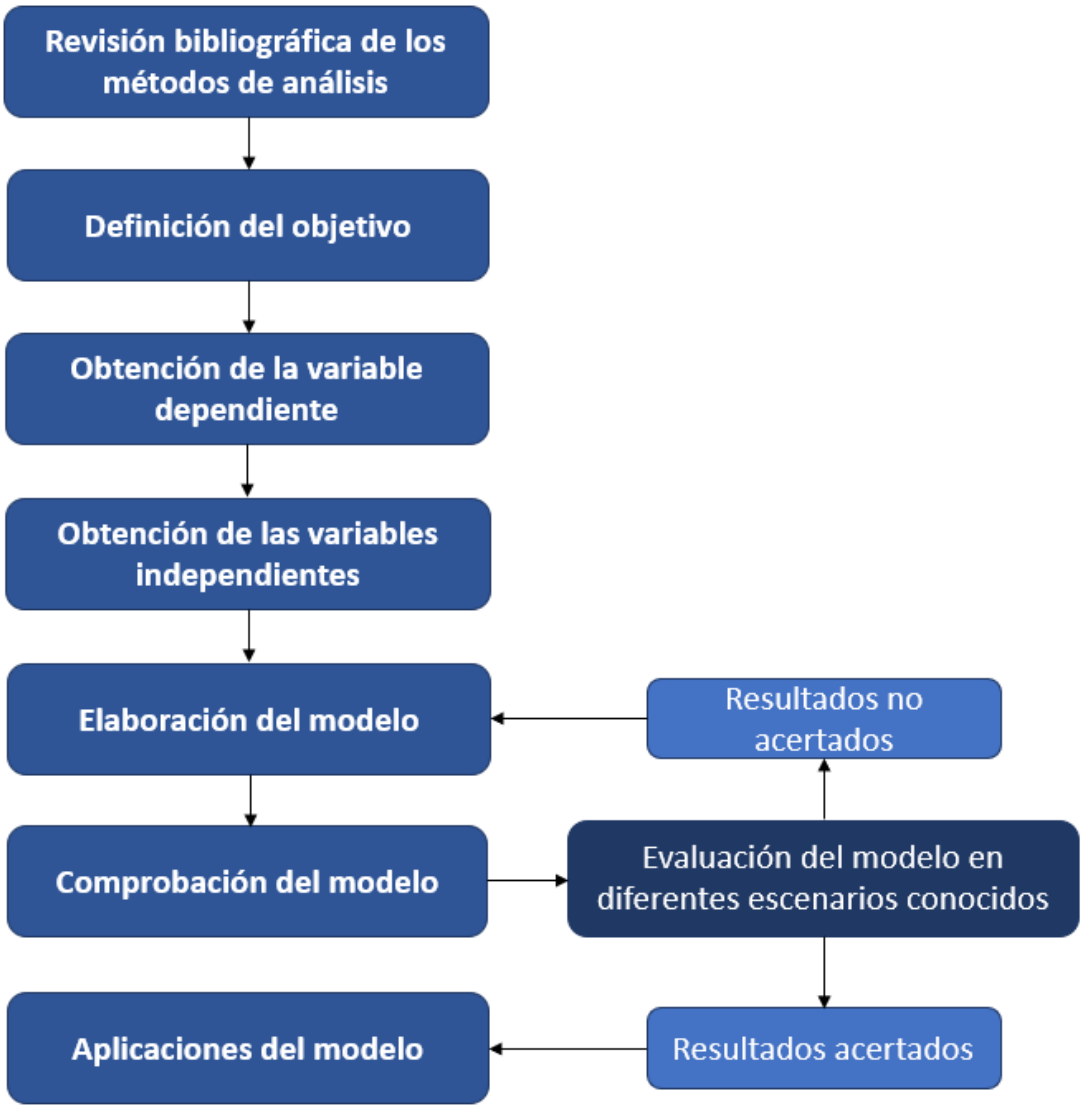


Figura 1. 8. Descripción de la metodología

1.6. Antecedentes teóricos y prácticos del problema

La Radial Coyol fue construida por el concesionario de la Ruta Nacional 27 como parte del contrato de concesión de dicha ruta y puesta en operación el 29 de junio de 2012 (Mendez, 2012). Se extiende por aproximadamente 4,1 km y conecta dos de las rutas más importantes del país: La Ruta Nacional 1 y la Ruta Nacional 27 mediante los Intercambios Coyol y Siquiares respectivamente (ver Figura 1.1).

Esta radial, a nivel general, es bidireccional con un carril por sentido, sin embargo, el emplazamiento de accesos a diferentes desarrollos ha cambiado su configuración funcional

al emplazar carriles centrales de giros izquierdos y canalizaciones para giros derechos en algunos sectores.

Tanto su alineamiento vertical como horizontal es ondulado asociado a una velocidad de diseño de 80 km/h. Según la información suministrada para esta investigación por Global Vía, el Tránsito Promedio Diario (TPD) para el año 2019 (estimado utilizando registros hasta setiembre del año 2019), es de 15122 con un 11,2% de vehículos pesados según lo reportó la Estación 22+450 ubicada aproximadamente 400 m al norte de la Rotonda Norte del Intercambio Siquiares sobre la radial indicada.

Como se mencionó anteriormente, se estimó que esta radial tiene una capacidad total de 41872 vehículos por día y también se estimó que tiene una velocidad de flujo libre de 95,5 km/h, mientras que el promedio de velocidad de ruedo se calculó en 75 km/h, lo que implica que por lo general solamente se alcanza un 78% de la velocidad de flujo libre. El porcentaje de tiempo siguiendo a otro vehículo alcanzó un valor de 73%. Con base en estos resultados y siguiendo a metodología del Highway Capacity Manual (Transportation Research Board, 2016) y del Manual Centroamericano de Normas para el Diseño Geométrico de Carreteras con enfoque de Gestión de Riesgo y Seguridad Vial (Secretaría de Integración Económica Centroamericana (SIECA), 2011), a la radial se le asigna un nivel de servicio C para su condición actual.

Respecto a la Ruta Nacional 1, en las cercanías de la radial tiene un TPD proyectado al año 2019 de 29564 vehículos con un porcentaje de pesados de 18,14%, según lo reporta la Estación 122. Respecto a la Ruta Nacional 27 en las cercanías de la radial tiene un TPD proyectado al año 2018 de 20587 con un porcentaje de pesados de 9,47%, según lo reporta la Estación 30+620 (Secretaría de Planificación Sectorial, Proceso de Planificación Estratégica Multimodal de Servicios de Infraestructura y de Transporte, 2018).

En ingeniería de transportes se distinguen cuatro métodos para estimar volúmenes vehiculares a futuro, basados siempre en series históricas de Tránsito Promedio Diario (TPD) de una carretera en específico, para la cual se utilizan regresiones lineales y curvilíneas, (Cal y Mayor R. & Cárdenas G., 2007). Estos métodos son:

- ✓ Regresión lineal simple (línea de mínimos cuadrados).
- ✓ Regresión curvilínea tipo exponencial
- ✓ Regresión curvilínea tipo potencial
- ✓ Regresión curvilínea tipo logarítmica

En estos métodos están basados los modelos que típicamente se utilizan para la estimación de tráfico a futuro. El más sencillo de ellos es extrapolar el ritmo del crecimiento observado en años anteriores, suponiendo que se mantiene en el futuro próximo. Sin embargo, este modelo, conocido como Tendencia geométrica, se limita a un futuro máximo de cinco años (Kraemer , y otros, 2004).

Existen otros modelos como son: Tendencia curva logística y Tendencia curva exponencial, todos ellos se basan en las series históricas de tránsito para calibrar los parámetros de las tendencias los cuales se pueden estimar por regresión lineal una vez realizadas algunas transformacionales logarítmicas. La tendencia de curva logística se caracteriza por tener una asíntota la cual representa la capacidad máxima de la vía y es estimada mediante el Manual de Capacidad de Carreteras (Transportation Research Board, 2016) como se describió anteriormente.

Con modelos de simulación se puede predecir el comportamiento de los flujos vehiculares, lo cual es muy valioso para la toma de decisiones inteligentes en cuanto a inversión y priorización de los recursos.

Sin embargo, es sabido que el tráfico vehicular está totalmente vinculado con otras variables que pueden acelerar o reducir la tasa de crecimiento, como la apertura de nuevas conexiones viales, desarrollo inmobiliario, aumento de la tenencia vehicular, aumento del producto interno bruto (PIB), entre otros, que no precisamente sigue una tendencia lineal, logarítmica o exponencial. De acuerdo con Austroads (Austroads, 2019) hay múltiples factores que impactarán en el crecimiento real del tráfico en comparación con las estimaciones realizadas, incluidos los desarrollos territoriales locales y regionales, los cambios demográficos, la economía en general y los incrementos de la flota de vehículos.

Es por lo que, para la estimación del tráfico vehicular futuro se ha intentado introducir algunas otras variables que permitan a los modelos adaptarse a eventos particulares. Por ejemplo, la Ing. Lady Diana Torres Sanabria realizó y evaluó varios modelos entre los que incluyó: población, tenencia vehicular, producción bruta de la industria manufacturera, actividad edificadora, energía eléctrica consumida, cantidad de sociedades constituidas, producto interno bruto e índice de precios al consumidor (Torres Sanabria, 2007) en la Ciudad de Pereira, Colombia.

Igualmente, en Costa Rica el Ing. German Valverde González presentó un modelo macroeconómico de crecimiento de tráfico vehicular en el que se consideró el producto interno bruto per cápita para estimar el índice de motorización y con este indicador se puede estimar la tenencia vehicular tras multiplicarlo por el número de habitantes (Valverde González, 2010).

De esta forma, tomando en cuenta las variables adecuadas se puede generar un modelo de predicción de tráfico que permita la consideración de otros eventos y mejore las estimaciones de tráfico, que es información vital en el proceso de diseño de la infraestructura vial y que además permita tomar decisiones de priorización de esta o bien estrategias de inversión privada.

En la actualidad, el acceso a la información es casi ilimitado y el procesamiento de esta es parte de la rutina de la mayoría de los entes tanto estatales como privados (aunque en muchos casos la obtención de la información no es automática y puede ser engorrosa). Por lo que un modelo de predicción puede alimentarse de gran cantidad de datos y ahí es donde se vinculan los modelos de predicción de tráfico con la ciencia de datos.

La ciencia de datos es una rama de la Inteligencia Artificial, la que nació en los años 50 (Russell & Norvig, 1994) y ha tomado un auge importante en los últimos años debido a los avances tecnológicos, la abundancia de información y que puede ser aplicado en cualquier área del desarrollo humano, desde la salud, economía, finanzas, educación hasta preferencias de artículos personales, música, sitios de interés, entre otros (SAS, 2020).

La ingeniería vial no se escapa de lograr aplicaciones de esta ciencia. En general, se ha utilizado para buscar rutas alternas con menores tiempos de demora para optimizar

recorridos y agilizar el trasporte de recursos (incluyendo recursos humanos) y mercancías (Penta Analytics, 2019). En este caso se procura que el modelo prediga el crecimiento de los flujos vehiculares tras cambios puntuales en la red de manera individual o una superposición de ellos. (Mata, 2019).

Así, por ejemplo, la ciencia de datos tiene aplicaciones en muchas áreas de la ciencia y la tecnología, combinándose entre ellas para encontrar el mejor uso de las grandes bases de datos que se han ido formando. Algunas de estas aplicaciones se enlistan a continuación:

Gobierno: los gobiernos pueden gestionar programas de riesgos y anticipar demandas de recursos de sus diferentes aristas.

Comercio y negocios: mediante esta metodología se logra predecir las tendencias de compra de los consumidores, clasificarlos bajo diferentes categorías, recomendarle sus preferencias, aumentar tasas de respuesta de las campañas de mercado, programar ofertas, mantenimiento de clientes, entre otros.

Educación y tecnología: nuevos métodos de enseñanza, descubriendo de diferentes capacidades y habilidades humanas, así como el reconocimiento de imágenes y voz permite automatizar cada día más los dispositivos inteligentes convirtiéndose en parte fundamental del modo de vida de las personas, así como ser parte del ocio de las personas teniendo aplicaciones en redes sociales y juegos.

Salud: el análisis de característica de diferentes individuos puede ayudar a diagnosticar enfermedades y atenderlas a tiempo, así como identificar efectos positivos y negativos de medicamentos durante sus pruebas clínicas.

Economía y finanzas: puede predecir el comportamiento de la Bolsa, establecer precios de venta y compra óptimos, clasificar individuos u operaciones según un perfil de riesgos, detectar fraudes y problemas de seguridad cibernética.

Transportes: desde las predicciones de tráfico, que son el objetivo de este estudio, hasta automóviles totalmente automatizados, pasando por sistemas de optimización de trasporte de carga y redes de viales que incluyen puntos de control con semáforos para la optimización de tiempos de recorrido.

Como se mencionó antes, debido a la cantidad de datos que generalmente manejan los modelos, se requiere el uso de algoritmos matemáticos y capacidad de análisis que solamente puede lograrse con equipos de cómputo y lenguajes de programación y/o softwares especializados para esto, como por ejemplo: R, Matlab y Julia.

En este proyecto se utilizó como lenguaje de programación Python el cual tiene una curva de aprendizaje moderada, con programación versátil multiplataforma y multiparadigma que se destaca por su código legible y limpio. Por lo que es utilizado por diversas escuelas de todo el mundo para iniciarse en la programación, pero también es utilizado por grandes compañías para el análisis de datos como Google, Facebook y YouTube. Python nació a inicios de los años 90 y fue desarrollado por Guido Van Rossum, ingeniero holandés que trabajaba en el Centro de Investigación de Ciencias de la Computación en Ámsterdam (Robledano, 2019).

Como interfaz de este lenguaje se utiliza la aplicación Anaconda, la cual fue desarrollada por Red Hat y su lanzamiento inicial fue en el año 1999. Es una aplicación gratuita y posee una interfaz gráfica muy sencilla para el usuario, pero con un gran potencial porque facilita la integración y automatización de los algoritmos en los procesos operativos (Anaconda, 2021).

1.7. Marco Teórico

Dentro de la teoría a revisar para generar los modelos de proyección de tránsito se deben considerar los principales conceptos de ciencia de datos, algoritmos de minería de datos como: regresión lineal, regresión de Ridge y árboles de regresión, así como algunas técnicas

de análisis, por ejemplo: series de tiempo, análisis de componentes e importancia de las variables. A continuación, se describen sus principales conceptos.

1.7.1. Ciencia de datos

La ciencia de datos es el estudio de información recopilada con un propósito. Este objetivo define la forma de ordenar dicha información para la generación de modelos que permiten predecir comportamientos. Es evidente que, conforme aumenten estos, el modelo mejora, lo que lo convierte en un ciclo de vida analítico interactivo que incluye la recolección de los datos, el descubrimiento de nuevos conocimientos que se pueden tener de los datos coleccionados y la implementación del conocimiento adquirido a la solución de problemas prácticos. En la Figura 1.9 se muestra la combinación de estas tres partes del ciclo.

En la parte de "Recolección de los datos", es importante que esté claro cuál es el objetivo del modelo para que lo datos recolectados converjan a ello, la clave es preguntarse para qué se quiere el modelo. También en esta parte se deben preparar los datos, bajo las condiciones actuales de avance tecnológico y equipos y herramientas de trabajo, el obtener datos no es difícil, pero si se deben preparar (tabular, ordenar y filtrar) de acuerdo con el objetivo previsto.

Figura 1. 9. Ciclo de Vida Analítico Iterativo

Fuente: (SAS, 2020)

Una vez preparados los datos, se inicia la etapa de “Descubrimiento”, la cual inicia con la exploración de los datos para verificar, fortalecer o ajustar en función del objetivo determinado y finaliza con la modelación de los datos, utilizando, los ya mencionados algoritmos matemáticos en equipos de cómputo de alta capacidad, que logren predecir de manera precisa los datos para el cumplimiento de la meta prevista.

Con los modelos construidos, se pasa a la etapa de “Implementación”, esta puede ser la etapa más difícil del proceso, porque es cuando los modelos se enfrentan a la realidad al ser implantados en un sistema. No obstante, debe entenderse que los modelos fabricados y la información obtenida de ellos se realizaron para ser utilizados, es decir para actuar y tomar decisiones que, según los modelos, son las mejores.

Evidentemente se debe evaluar los resultados, que es el paso más importante, para determinar si realmente el modelo logró predecir correctamente y las decisiones tomadas en función de ello fueron productivas y como se mencionó antes, cada día se generan más datos que pueden fortalecer o ajustar los modelos por lo que el proceso vuelve a iniciar con los nuevos datos. En la Figura 1.10 se muestra este proceso.

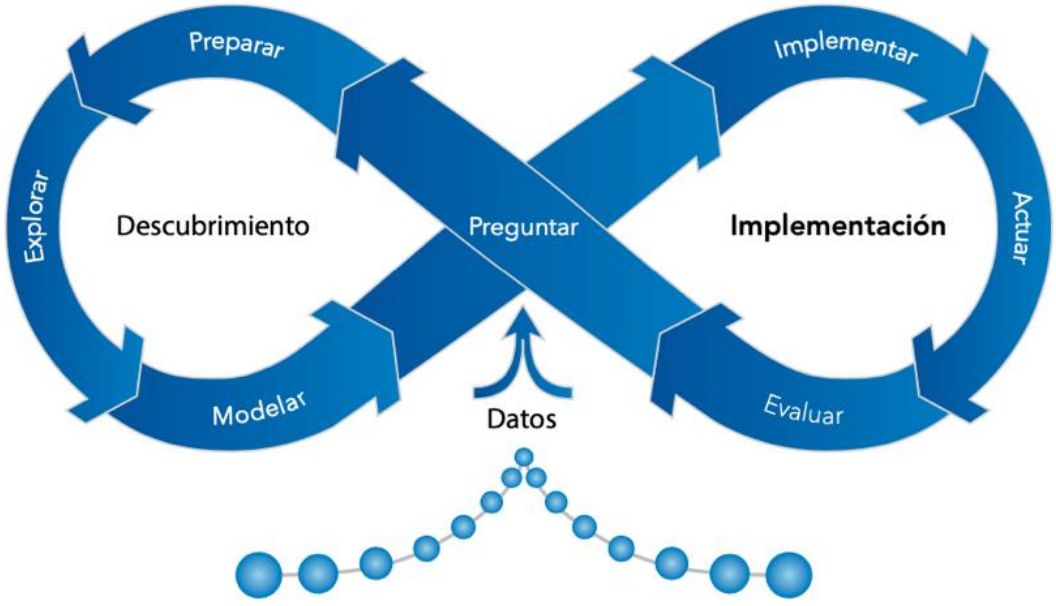


Figura 1. 10. Proceso del ciclo de vida analítico

Fuente: (SAS, 2020)

1.7.2. Regresión lineal múltiple

La regresión lineal múltiple es modelo matemático lineal o linealizable que permite relacionar una variable dependiente con más de una variable independiente (Montero Granados, 2016). Es importante recordar que una relación estadística no puede por sí misma implicar en forma lógica una casualidad (Benavente, Otero, & Javiera, 2007). El modelo general es el que se muestra en la Ecuación 1.1.

$$y = \beta_0 + \beta_1 x_{1j} + \beta_2 x_{2j} + \cdots + \beta_k x_{kj} + u_j \quad \text{Ecuación 1.1}$$

Se exponen a continuación algunos de los principales conceptos de la regresión lineal:

Magnitud determinista: cada vez que se mide se da el mismo resultado.

Magnitud estocástica: cada vez que se mide da un resultado diferente.

Correlación: es una medida de la similitud de la variabilidad de dos magnitudes estocásticas. Tiene como ventaja que se calcula de manera sencilla, pero como desventaja que muchas veces no es suficiente para comprender la relación entre ambas variables (Montero Granados, 2016).

Variable endógena: se refiere a la variable dependiente.

Variables exógenas: se refiere a las variables independientes.

Variables continuas: cuando se pueden contar o medir.

Variables discretas: cuando no se pueden contar y representan por lo general una característica, cuando tienen sola dos posibilidades se les llama dicótomas.

Para asegurar que la regresión lineal implique el menor error posible, se utiliza el Método de Mínimos Cuadrados (MCO) que se basa en minimizar (utilizando derivación) la expresión la sumatoria del cuadrado de los errores, en función de los coeficientes de la regresión y de esa forma encontrar los coeficientes que aseguran que el error sea mínimo. Adicionalmente, los modelos preferiblemente deben cumplir los siguientes supuestos:

- ✓ El modelo debe ser de la forma indicada en la Ecuación 1.1

- ✓ Las variables deben ser fijas, es decir, no son estocásticas
- ✓ El valor medio del error es igual a cero $E(\varepsilon) = 0$
- ✓ Homogeneidad de la varianza de los errores $Var(\varepsilon) = \sigma^2$
- ✓ No correlación de los errores $Cov(\varepsilon) = 0$
- ✓ La covarianza entre las variables y los errores es cero
- ✓ El número de observaciones debe ser mayor al número de parámetros por estimar
- ✓ Se requiere variabilidad de los valores de las variables
- ✓ Distribución normal de los errores $\varepsilon \sim$ Normal $(0, \sigma^2)$

Cada uno de estos supuestos puede demostrarse gráficamente cuando se cuenta con suficientes datos. En la Figura 1.11 se muestran dichas demostraciones gráficas. El cumplimiento de estos supuestos facilita la estimación del objetivo buscado, pero de no cumplirse, existen técnicas de corrección que permiten su adecuación sin problema.

Es muy importante también revisar la correlación entre las variables a utilizar, esto se mide mediante el coeficiente de correlación, este coeficiente mide el grado de asociación lineal entre dos variables tratadas de manera simétrica, a diferencia con el análisis de regresión en donde la variable dependiente es aleatoria y tiene una distribución de probabilidad y las variables explicativas o independientes toman valores fijos. Algunas precauciones con el coeficiente de correlación son (Benavente, Otero, & Javiera, 2007):

- ✓ Tener suficiente cantidad de variables
- ✓ Mantener relaciones lineales entre las variables
- ✓ La correlación no implica casualidad es solo una relación estadística.
- ✓ La correlación puede indicar una relación espuria
- ✓ La correlación muestral es una variable aleatoria y por lo tanto el coeficiente por sí solo no garantiza una relación entre las series.

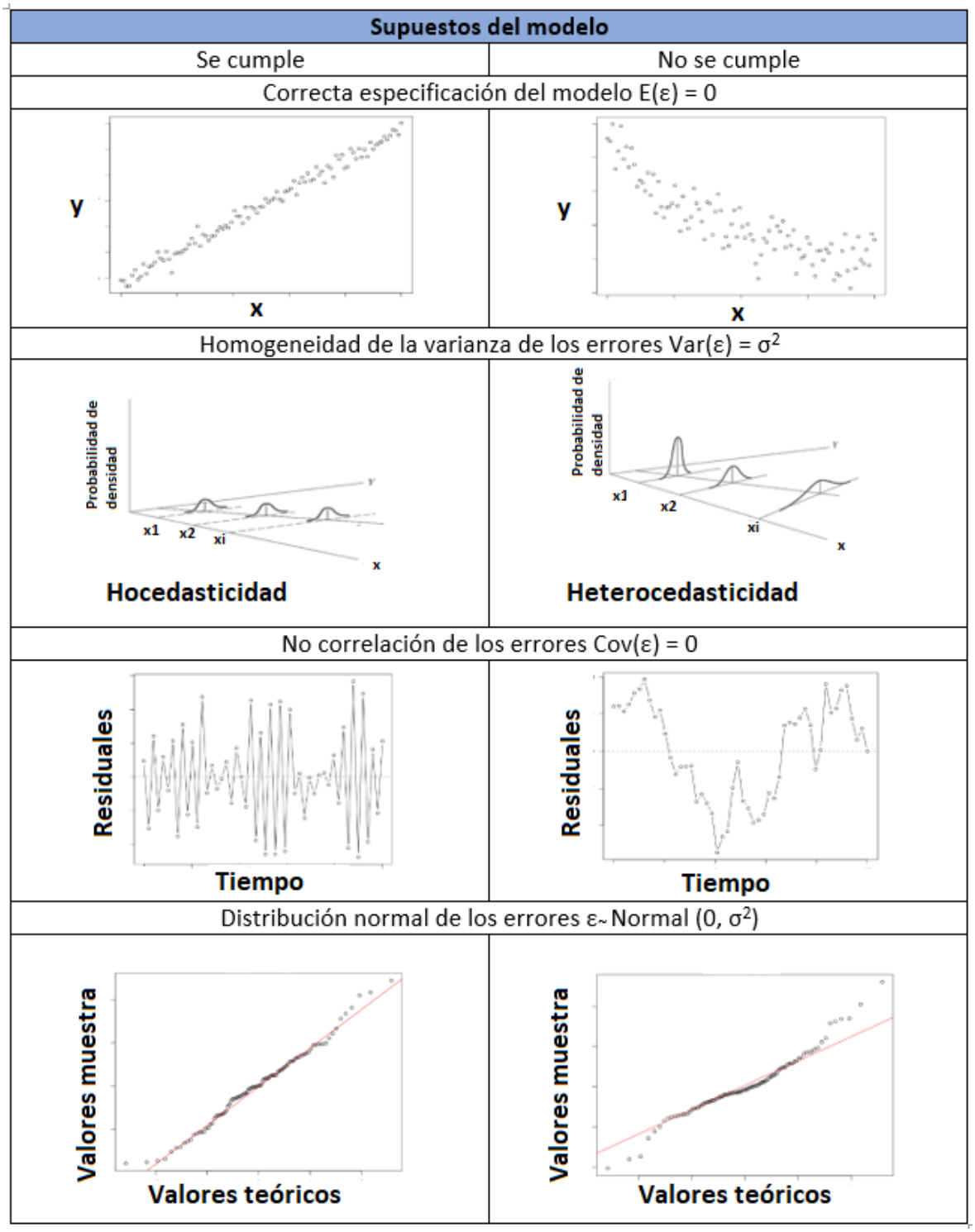


Figura 1. 11. Enfoque gráfico de los supuestos de los modelos de regresión lineal

Fuente: Elaboración propia utilizando información de (Behar, 2003), (Montgomery, Peck, & Vinning, 2002), (Draper & Smith, 1998) y (Rawlings, Pantula, & Dickey, 1998)

1.7.3. Regresión de Ridge

La regresión de Ridge sigue los mismos principios y características de una regresión lineal, siendo un caso particular de esta. Sin embargo, se agrega el concepto de regularización. Una regresión lineal tiende a generar como modelo un polinomio de alto grado para ajustarse a los datos analizados, sin embargo, se ha notado que en cuanto más se ajuste a los datos usados para generar el modelo, al aplicarlo con nuevos datos se tiene una importante deficiencia del modelo, es decir no generaliza adecuadamente (Maklin, 2019).

La regularización es un método para evitar el sobreajuste del modelo donde se hacen tender a cero los coeficientes de las variables de orden superior introduciendo un sesgo, no se puede abusar de la regularización porque se genera un subajuste como lo muestra la Figura 1.12. En los lenguajes de programación la regularización se realiza mediante el uso de un parámetro α, donde si α es igual a uno, la regresión de Ridge y la lineal son exactamente iguales, así entonces el valor de α dependerá del criterio del modelador según sus objetivos.

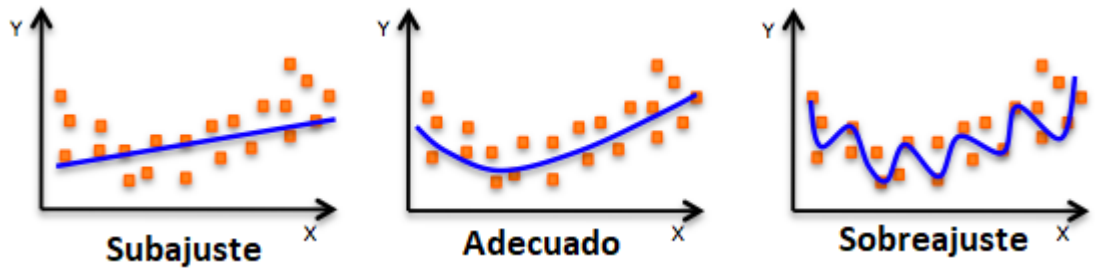


Figura 1. 12. Tipos de ajustes en las regresiones

Fuente: (Epicalsoft-Instance-Blog, 2019)

1.7.4. Árbol de regresión

En general, un Árbol de regresión es un método analítico que, a través de una representación esquemática de las alternativas disponibles, facilita la toma de mejores decisiones, especialmente cuando existen múltiples opciones.

Propiamente dentro del ámbito de ciencia de datos, según (Ferrero & López, 2021), un árbol de regresión es un modelo predictivo supervisado (porque para que aprenda el modelo se

necesita una variable dependiente en el conjunto de entrenamiento), que divide el espacio de los predictores agrupando observaciones con valores similares para la variable respuesta o dependiente. Su asociación con la ciencia de datos y el aprendizaje automatizado se inició en el año 1984 por Leo Breiman, Jerome Friedman, Richard Olshen y Charles Stone.

Los árboles de regresión están formados por nodos de diferentes niveles y su lectura se realiza en una sola dirección. El primer nodo o nodo principal es el objeto de estudio y de él se desprenden conectores ("split" en el lenguaje de programación utilizado) que llegan a los nodos del siguiente nivel denominados hojas ("leaf" en el lenguaje de programación utilizado). La cantidad de niveles que tenga el árbol de decisión se conoce como profundidad del árbol de decisión ("depth" en el lenguaje de programación utilizado). A mayor cantidad de estos elementos más complejo es el árbol y existe una mayor probabilidad de diferentes resultados.

Entre las ventajas de los árboles de regresión se encuentran que no es preciso que se cumplan una serie de supuestos como en la regresión lineal (linealidad, normalidad de los residuos, homogeneidad de la varianza, entre otros). Sirven tanto para variables dependientes cualitativas como cuantitativas, como para variables predictoras o independientes numéricas y categóricas. Además, no necesita variables binarias, aunque a veces mejoran el modelo. Permiten relaciones no lineales entre las variables explicativas y la variable dependiente y pueden servir para categorizar variables numéricas.

También presentan algunas desventajas como son que tienden al sobreajuste de los datos, son muy influenciados por los valores atípicos por lo que estos deben eliminarse. No suelen ser muy eficientes como modelos de regresión clásicos. Árboles demasiado complejos resta capacidad de interpretación. Se pueden crear árboles sesgados si una de las clases es más numerosa que otra y se pierde información cuando se utilizan para categorizar una variable numérica continua.

Estos tres últimos algoritmos se escogieron como posibles modelos por dos razones:

1. Son los algoritmos que se ajustan a una base de datos pequeña como la que se tiene en este caso.

2. Permiten una interpretación más sencilla para el razonamiento matemático del ser humano.

Existen otros algoritmos como: Random Forest, SVR SUPPORT VECTOR REGRESSOR y XGBOOST REGRESSOR, fueron probados inicialmente, pero se descartaron porque no son interpretables y su mayor efectividad se logra con bases de datos de mucho más grandes (millones de observaciones).

1.7.5. Series de tiempo

Las series de tiempo son la forma da tabular variables que son medidas en el tiempo. El análisis de estas variables en relación con el tiempo permite llevar a cabo pronósticos aplicables en diferentes ramas profesionales y tecnologías. Las propiedades o componentes de una serie de tiempo son:

- Tendencia: es una medida a largo plazo del comportamiento de las variables, por ejemplo, si con el tiempo se da un incremento, decremento o se mantiene el volumen de la variable.
- Ciclo: es una medida a mediano y largo plazo que muestra un comportamiento cíclico de la muestra.
- Estacionalidad: es una medida a mediano y corto plazo que está relacionada con eventos anuales, mensuales, semanales, entre otras, como, por ejemplo: el clima, el inicio de ciclo lectivo, final y principio de año, vacaciones, entre otros.
- Aleatoriedad: es un comportamiento errático de la serie que se da por un evento inesperado como: una guerra, huelgas, desastres naturales y otros.

Las series de tiempo pueden tener uno solo de estos componentes, varios de ellos o todos (Reyes Aguilar, 2007). Según el objetivo propuesto, al analizar los datos se pueden encontrar comportamientos que son similares una serie de tiempo, por lo que pueden ser utilizadas para estimar valores faltantes en las bases de datos.

1.7.6. Análisis de componentes

De acuerdo con Jaadi (Jaadi, 2021), “el análisis de componentes principales, o PCA, es un método de reducción de dimensionalidad que se utiliza a menudo para reducir la dimensionalidad de grandes conjuntos de datos, transformando un gran conjunto de variables en uno más pequeño que aún contiene la mayor parte de la información del conjunto grande. La reducción de la cantidad de variables de un conjunto de datos se obtiene naturalmente a expensas de la precisión, pero el truco en la reducción de la dimensionalidad es cambiar un poco de precisión por simplicidad”.

El proceso para llevar a cabo la definición de PCA se da en cuatro pasos básicos, el primero es la estandarización, el segundo es el cálculo de la matriz de covarianza, el tercero es el cálculo de los autovectores y autovalores de la matriz de covarianza y finalmente la determinación del vector de características. A continuación, se explica cada uno de ellos:

Paso 1: Estandarización

La varianza es uno de los conceptos más importantes en la determinación de los PCA, porque el método se basa en ella y por lo tanto es muy sensible a los valores de este estadístico que las variables tengan. Entonces es fundamental estandarizar (restar la media y dividir entre la desviación estándar para cada valor de cada variable) las variables de que cada una de ellas contribuya por igual en análisis y evitar resultados sesgados.

Paso 2: Cálculo de la matriz de covarianza

La matriz de covarianza es la herramienta que permite visualizar como las variables del conjunto de datos varían entre sí y entre ellas, de modo que se define si existe una relación entre ellas, lo cual muchas veces es probable y se tiene información redundante.

La matriz de covarianza es una matriz simétrica p×p (donde p es el número de dimensiones) que tiene como entradas las covarianzas asociadas con todos los pares posibles de las variables iniciales. Dado que la covarianza de una variable consigo misma es su varianza y la covarianza es conmutativa, las entradas de la matriz de covarianza son simétricas con respecto a la diagonal principal (varianza de las variables inicial), lo que significa que las porciones triangulares superior e inferior de la matriz son iguales.

Se debe recordar que lo importante es la matriz de covarianza es el signo resultante, si es positivo, entonces: las dos variables aumentan o disminuyen juntas, es decir están correlacionadas. Si es negativo, uno aumenta cuando el otro disminuye (correlación inversa)

Paso 3: Cálculo los autovectores y autovalores de la matriz de covarianza

En primer lugar, se debe entender el concepto de componentes principales. Según Jaadi (Jaadi, 2021), "los componentes principales son nuevas variables que se construyen como combinaciones lineales o mezclas de las variables iniciales. Estas combinaciones se realizan de tal manera que las nuevas variables (es decir, componentes principales) no están correlacionadas y la mayor parte de la información dentro de las variables iniciales se comprime en los primeros componentes. Entonces, la idea es que los datos de ***p*** dimensiones le brindan ***p*** componentes principales, pero PCA intenta poner la máxima información posible en el primer componente, luego la máxima información restante en el segundo y así sucesivamente".

Jaadi (Jaadi, 2021) explica también que, geométricamente hablando (Ver Figura 1.13), los componentes principales representan las direcciones de los datos que explican una cantidad máxima de varianza, es decir, las líneas que capturan la mayor parte de la información de los datos. La relación entre la varianza y la información aquí es que, cuanto mayor es la varianza llevada por una línea, mayor es la dispersión de los puntos de datos a lo largo de ella y cuanto mayor es la dispersión a lo largo de una línea, más información tiene.

El eje que logra capturar la mayor varianza posible del conjunto de datos es denominado primer componente principal, en la Figura 1.13 se muestra como el eje que se proyecta del cuadrante inferior izquierdo al cuadrante superior derecho. El segundo componente principal representa la siguiente varianza más alta y se calcula exactamente igual que el primero, pero con la condición de que sean ortogonales (Eje que se proyecta del cuadrante inferior derecho al cuadrante superior izquierdo en la Figura 1.13).

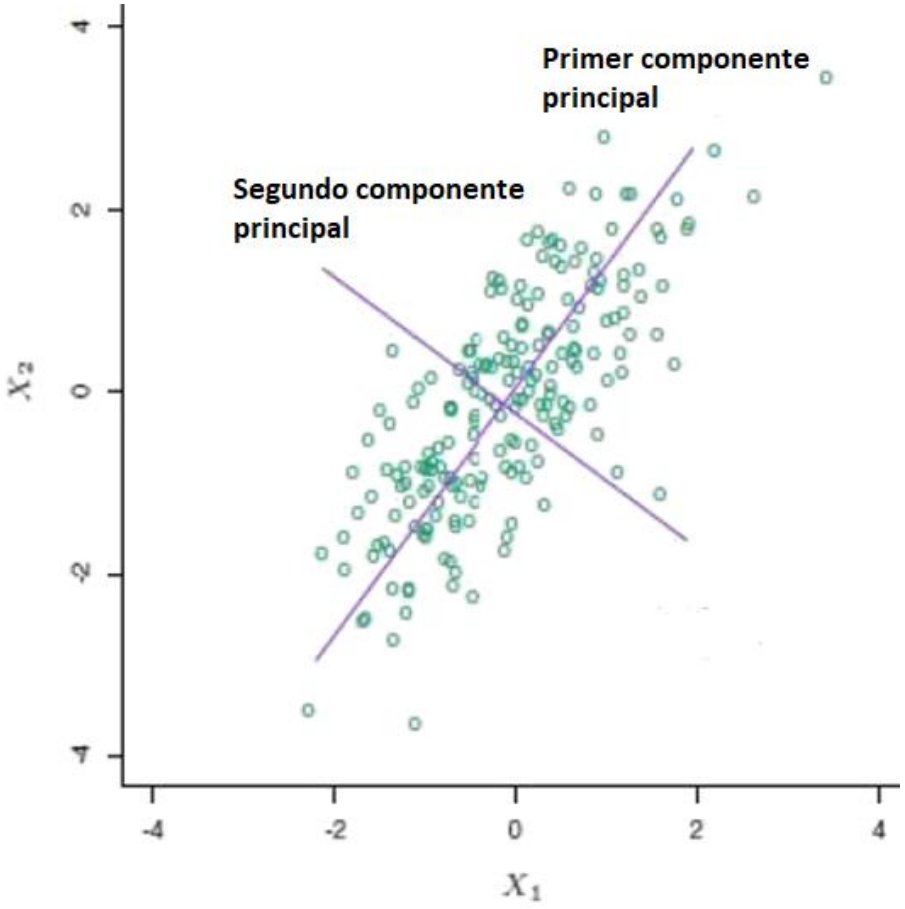


Figura 1. 13. Explicación geométrica de los PCA

Fuente: (TIBCO, 2021)

Existen tantos componentes principales como variables, por lo que el procedimiento se sigue repitiendo hasta el último componente el cual es el que representa la menor varianza posible en el conjunto de datos.

Al calcular los componentes principales cada uno tendrá un valor que representa el porcentaje de varianza que capturó. De esta forma se puede seleccionar él o los componentes que logren captar un porcentaje razonable de las variables y desechar los demás y de esta forma se reduce la dimensión de la base de datos o bien simplemente se asegura que las nuevas variables utilizadas no están correlacionadas. Una limitación de este procedimiento es que las variables resultantes no tienen una interpretación fácil ni un significado real ya que están construidas como combinaciones lineales de las variables

iniciales. Sin embargo, las aplicaciones para generar los modelos permiten mantener como dato de entrada las variables originales.

Los vectores y los valores propios son los conceptos de álgebra lineal que se necesitan calcular a partir de la matriz de covarianza para determinar los componentes principales de los datos. Es así como los vectores propios de la matriz de covarianza son en realidad las direcciones de los ejes donde hay más varianza y por lo tanto a los que llamamos componentes principales y los valores propios son simplemente los coeficientes adjuntos a los vectores propios, que dan la cantidad de varianza transportada en cada componente principal, de modo que cada vector propio tiene su valor propio. Al clasificar sus vectores propios en orden de sus valores propios, de mayor a menor, obtiene los componentes principales en orden de importancia. Para calcular el porcentaje de varianza contabilizado por cada componente, dividimos el valor propio de cada componente por la suma de los valores propios (Jaadi, 2021).

Paso 4: Vector de características

Se conoce como Vector de características la matriz resultante tras seleccionar cuales componentes principales se mantienen y cuales se desechan. Lo que conlleva a la reducción de las dimensiones de la base de datos sin afectar significativamente el contenido de la base. Según Joaquín Amat (Joaquín Amat, 2017), no existe una respuesta o método único que permita identificar cual es el número óptimo de componentes principales a utilizar. Una forma de proceder muy extendida consiste en evaluar la proporción de varianza explicada acumulada y seleccionar el número de componentes mínimo a partir del cual el incremento deja de ser sustancial.

1.7.7. Importancia de las variables

El método de Importancia de las variables es una forma de seleccionar las variables más representativas para el modelo. Este método se basa en la teoría de árboles de regresión y lo que estima es un valor relativo del aporte de cada variable al modelo. Este es utilizado

cuando se requiere reducir el número de variables independientes en el modelo manteniendo las que aportan más información a este.

Una vez presentadas las generalidades de la investigación, en el siguiente capítulo se procede a describir las variables dependientes e independientes a utilizar.

2. Definición y descripción de las variables

En este capítulo se hace referencia a lo relacionado con las variables, tanto la dependiente como las independientes, su concepto, cuáles se incluyen en esta investigación y una descripción de los hallazgos encontrados en la exploración realizada sobre ellas, que permite su delimitación y agrupamiento.

2.1. Descripción de la variable dependiente

La variable dependiente es la variable que se quiere predecir. Para realizar el modelo de proyección de tráfico, se necesita tener una serie de estos datos asociados a las variables independientes suficientemente amplia para que el modelo aprenda las relaciones entre ambos y pueda estimar la variable dependiente para diferentes escenarios de interés.

En este caso, la variable dependiente corresponde al Tránsito Promedio Diario (TPD) de la Radial Siquiares – Coyol. Estos datos se obtuvieron de la estación de medición PK22450 ubicada a aproximadamente 400 m al norte de la Ronda Norte del Intercambio Siquiares sobre la misma radial (Ver Figura 1.1). Esta estación es administrada por Autopistas del Sol como concesionario de la Ruta Nacional 27 y dicha radial. Es importante revisar si el Intercambio las Llanos ubicado aproximadamente a la mitad de la radial (Ver Figura 1.1), distorsiona considerablemente el TPD de modo que esta estación sea solo representativa de la parte sur de la radial y no de la parte norte. Sin embargo, mediante la revisión de aforos vehiculares realizados por la autora en la zona, se determinó que la relación de volumen norte/sur es de 56/44 lo cual sugiere que no hay restricciones en el uso de esta estación para analizar la radial completa. Este resultado es congruente que con el hecho de que, en la actualidad, hacia el norte de la radial existe más desarrollo que hacia el sur.

La información suministrada incluye, el volumen horario desde abril del año 2010 hasta el 31 de diciembre del año 2019, para los diferentes tipos de vehículos (livianos, autobuses, 2 y 3 ejes, 4 ejes, 5 o más ejes, estos últimos tres se agruparon como pesados), el sentido de

circulación y la presencia o no de congestión vehicular, para un total de 757566 datos, los cuales fueron tabulados y ordenados para una más fácil comprensión.

En primera instancia se ordenó la información y se realizó un primer resumen de agrupamiento, estimando el Tránsito Promedio Diario Anual (TPDA) por mes, cabe mencionar que el TPDA difiere en al TPD en que el primero es la sumatoria de todos los TPD de un punto durante un año y divididos entre los días del año (365). En las Tabla 2.1 a la 2.3 se muestra la información indicada y la cual es graficada para su mejor comprensión y análisis (Ver Figura 2.1 a la 2.3) según el tipo de vehículo.

De esta primera síntesis de información se aprecia que en el año 2010 el valor de TPD mensual es prácticamente nulo y aumenta ligeramente en el año 2011, da un salto medio en el año 2012 y es muy significativo el cambio al llegar el año 2013. Esto se debe a que fue justo en junio del año 2012 que fue abierta al público la radial en estudio (Mendez, 2012). De esta forma, los registros anteriores a julio del año 2012 deben ser excluidos como variables dependientes porque realmente no son representativos del comportamiento del tránsito bajo condiciones normales de uso de la radial.

Por otra parte, en el año 2018 se da un problema en lectura de los datos (posiblemente por una falla del equipo tras la sustracción delictiva de uno de sus componentes según informó Autopistas del Sol) y los valores caen a menos de la mitad de los valores obtenidos en el año 2017. En una revisión mensual se logró apreciar que el problema de lectura inició en julio del año 2017 y se prolongó hasta octubre del año 2018, razón por la que en el año 2017 se inicia la curva de descenso en las Figuras 2.1 a la 2.3.

En una primera instancia los datos faltantes se obtuvieron por estimaciones utilizando la tendencia logística, sin embargo, este método implica muchos supuestos ya que da estimaciones para TPDA y estas deben subdividirse en datos mensuales y luego por tipo de vehículo lo que induce a errores. Como se discutirá más adelante, los primeros prototipos del modelo se realizaron con esa base de datos y los resultados no fueron adecuados por lo que se interpreta que esta forma de estimación de los datos faltantes no fue adecuada.

Al introducir la base de datos cruda en la aplicación Anaconda y con el lenguaje de programación Python, se realizó nuevamente la limpieza y análisis de los datos y se

encontró un comportamiento muy similar a las series de tiempo, por lo que se procedió a calcular los datos faltantes utilizando las características de estacionalidad y tendencia de las series de tiempo explicados anteriormente.

El procedimiento es seguido es el siguiente, primero se estiman los índices de estacionalidad para los datos válidos, luego se destacionaliza la serie de datos para dejar únicamente la tendencia con la cual se puede realizar una regresión lineal y calcular los datos faltantes, que finalmente se les aplica el índice de estacionalidad según corresponda al mes y una vez estimados los datos faltantes, se imputaron en la base de datos, en la Figura 2.4 se muestra el hallazgo al respecto y en la Figura 2.5 se muestra la base de datos ya con la imputación realizada cuyos resultados son satisfactorios para continuar el análisis con ellos.

También se notó que existió una disminución en la cantidad de vehículos pesados en los últimos meses del año 2019, específicamente en octubre, noviembre y diciembre. Sin embargo, no se encontró una razón aparente registrada. Por lo que se asume que puede deberse a un fallo en la calibración del equipo porque justo esos meses, el flujo vehicular reportado como buses presenta un aumento en comparación con otros años de análisis. Esta distorsión no se consideró significativa en la construcción del modelo, porque finalmente la variable dependiente se utilizó el TPD completo.

Por otra parte, se considera que la presentación mensual de las variables dependientes es adecuada para asociarla con variables independientes relacionadas con inicios de operación de industrias, empresas, comercios, servicios, entre otros, cuyos valores se presentan en la siguiente sección.

Tabla 2. 1. Promedio mensual del TPD de vehículos livianos de la base de datos original

Tránsito Promedio Diario Vehículos Livianos											
Año		2010	2011	2012	2013	2014	2015	2016	2017	2018	2019
Mes	Enero	0	160	108	7668	10136	11106	13290	15446	14330	15340
	Febrero	0	189	121	8144	10439	11692	13515	17906	2530	16047
	Marzo	0	219	146	7575	10442	11817	12773	16557	2841	15831
	Abril	87	240	114	7854	9224	11393	13230	14748	2818	14487
	Mayo	121	203	90	7744	8613	11402	13195	15210	2783	14740
	Junio	100	100	289	7745	9421	11689	13435	13766	2842	14890
	Julio	0	52	4518	7845	9963	12326	13751	13661	2840	15047
	Agosto	0	28	5789	8699	9921	12181	13496	12619	2840	15011
	Setiembre	0	47	5962	8891	9819	11949	13541	11033	2842	14756
	Octubre	0	40	6432	9097	10145	12477	13240	7538	0	14747
	Noviembre	0	44	8107	9860	10622	12954	13737	10501	7670	15609
	Diciembre	0	127	7624	10122	11708	13659	14881	9426	7755	10862

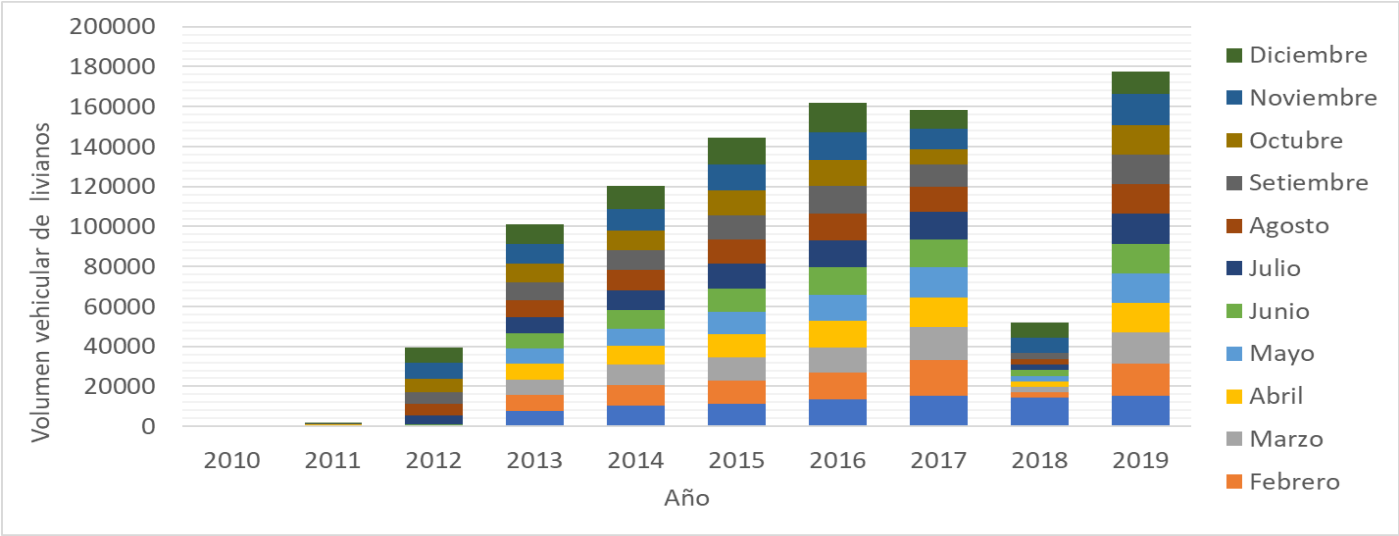


Figura 2. 1. Comportamiento del tráfico vehicular de livianos sobre Radial Coyol

Tabla 2. 2. Promedio mensual del TPD de buses de la base de datos original

Tránsito Promedio Diario Buses											
Año		2010	2011	2012	2013	2014	2015	2016	2017	2018	2019
Mes	Enero	0	80	3	167	361	367	339	382	370	854
	Febrero	0	106	2	207	279	311	360	439	64	1026
	Marzo	0	94	4	168	305	319	379	410	70	733
	Abril	34	104	2	181	401	318	446	386	69	513
	Mayo	54	92	2	179	265	365	394	415	71	440
	Junio	39	48	5	177	229	333	415	364	70	434
	Julio	0	26	104	189	244	404	467	333	70	399
	Agosto	0	25	136	220	233	352	421	302	70	386
	Setiembre	0	4	126	231	233	334	354	261	70	409
	Octubre	0	3	146	239	236	386	345	220	0	456
	Noviembre	0	3	188	240	254	349	373	300	196	425
	Diciembre	0	3	197	220	324	451	392	231	184	262

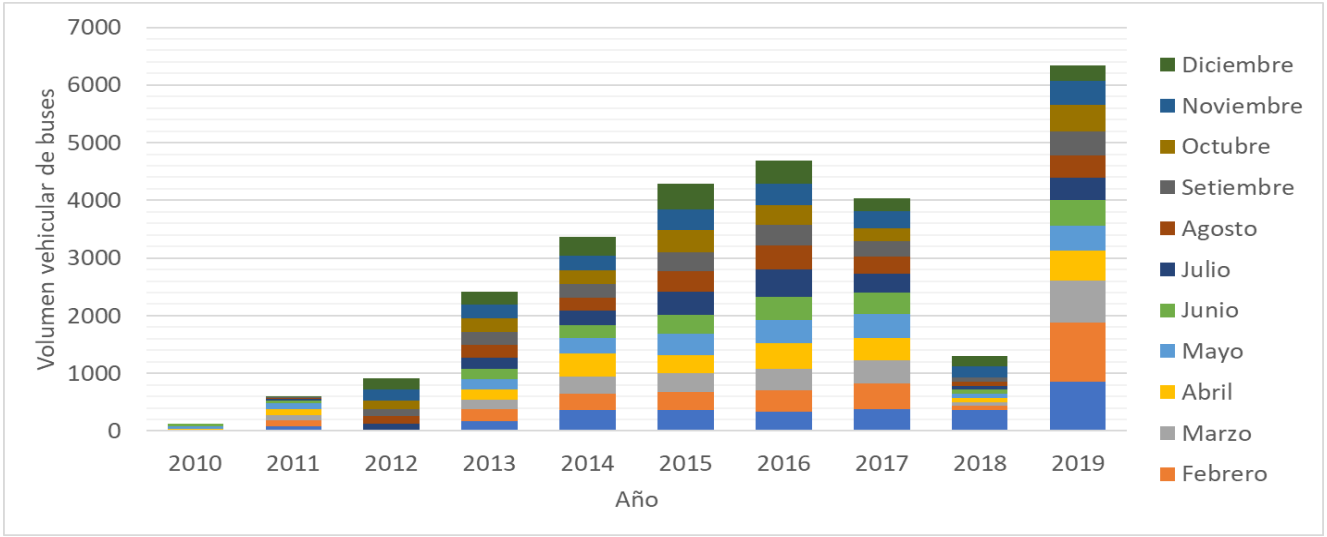


Figura 2. 2. Comportamiento del tráfico buses sobre Radial Coyol

Tabla 2. 3. Promedio mensual del TPD de vehículos pesados de la base de datos original

Tránsito Promedio Diario Pesados											
Año		2010	2011	2012	2013	2014	2015	2016	2017	2018	2019
Mes	Enero	0	342	603	746	3491	3641	4565	6724	1252	5844
	Febrero	0	174	668	814	3231	3550	4519	6561	208	5691
	Marzo	0	202	584	836	3822	4282	4692	6168	234	6060
	Abril	46	275	632	706	3080	3748	5075	6338	229	5606
	Mayo	49	312	620	721	3101	3929	5045	7090	240	6130
	Junio	38	132	616	700	2994	4001	5033	5940	233	5769
	Julio	0	49	626	754	3391	4410	5368	5326	234	6285
	Agosto	0	27	673	713	3225	4059	5191	5063	234	5988
	Setiembre	0	7	676	745	3277	4147	5012	4248	233	3632
	Octubre	0	4	703	785	3547	4492	5513	3653	0	725
	Noviembre	0	4	784	752	3425	4517	6046	3970	573	1006
	Diciembre	0	2	738	815	3732	5056	6568	3382	534	803

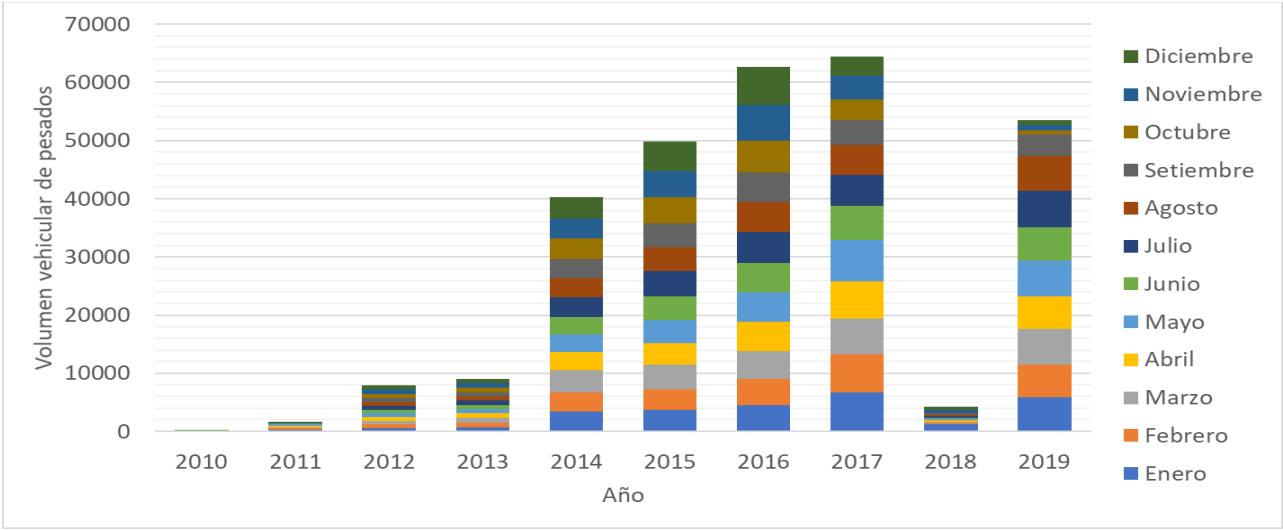


Figura 2. 3. Comportamiento del tráfico vehículos sobre Radial Coyol

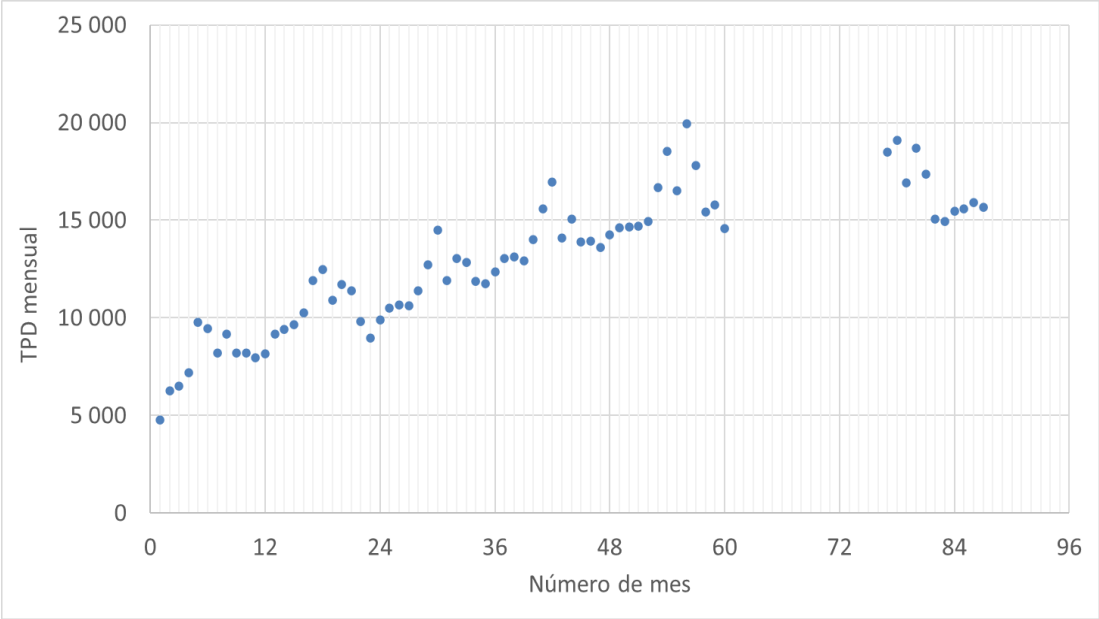


Figura 2.4. Gráfico resultante de la limpieza de datos de la variable dependiente mediante la aplicación Anaconda

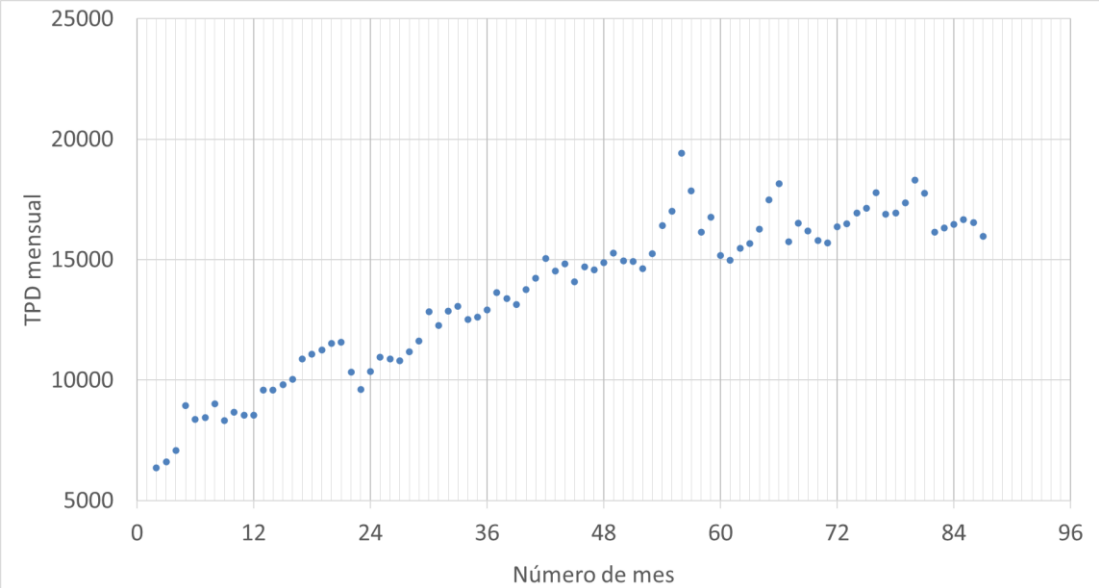


Figura 2.5. Gráfico resultante de la variable dependiente con los datos faltantes imputados

2.2. Descripción de las variables independientes

Las variables independientes son una serie de datos cuyo valor no depende de otras variables, de ahí su nombre, su función en el modelo es establecer relaciones con la variable dependiente de modo que se puedan predecir valores de esta última para diferentes escenarios. En este caso, las variables independientes corresponden a aquellas variables que puedan estar relacionadas con el crecimiento del tráfico vehicular en la zona de análisis. Para efectos de esta investigación variables independientes se segregaron en variables internas y variables externas de acuerdo con su naturaleza. Es decir, si son propias del sector del Coyol o si son datos generales para el país respectivamente.

Entre las variables internas se tiene: la cantidad de viviendas, los metros cuadrados de industria, comercio y servicios. Así como los datos de población y distancia entre la cabecera del distrito y un punto central en la radial. Estos últimos no se utilizan como variables propias sino para estimar un factor de afectación a las variables anteriores por población y distancia.

Entre las variables externas se consideraron el TPD de la Ruta Nacional 1, la cantidad de vehículos importados en Costa Rica, índices económicos como el Índice Mensual de Precios al Consumidor (IPC), cuyo cambio porcentual es la inflación y el Índice Mensual de Actividad Económica (IMAE). También se consideró el precio de los combustibles más utilizados: gasolina súper, gasolina regular y diésel y finalmente el tipo de cambio del dólar estadunidense al colón, nuestra moneda local. En los próximos párrafos se realiza la descripción de cada una de ellas.

2.2.1. Variables Internas

Se refiere a variables propias de los distritos San José, San Antonio, Turrucares y La Garita del cantón de Alajuela, estos son los distritos que se consideran en el análisis por ser los más cercanos a la radial, en la Figura 2.6 se muestra la ubicación de estos distritos respecto a la Radial Siquiares – Coyol.

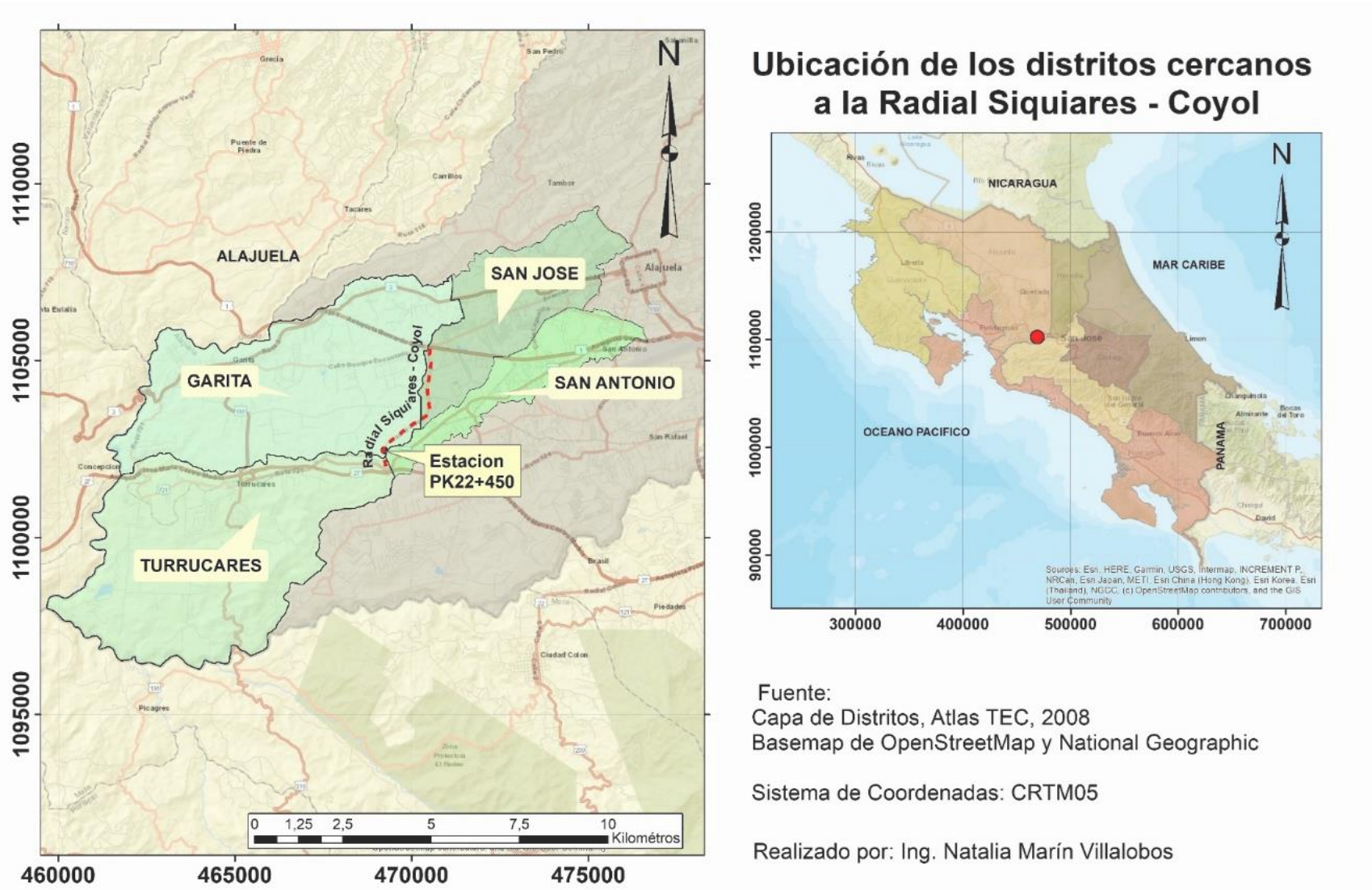


Figura 2. 6. Ubicación de los distritos a considerar en el análisis

Los datos mensuales que se muestran a continuación (Tablas 2.4 a la 2.7) corresponden a los permisos tramitados en los cuatro distritos en el Colegio Federado de Ingenieros y Arquitectos (CFIA) de donde también se obtuvo la información.

Número de Viviendas: registra la cantidad de permisos de construcción de vivienda por mes, incluye diferentes tipos de vivienda como: casas, apartamentos, condominios, edificios residenciales de diferentes características, en fin, para efectos del ordenamiento de la base de datos se clasifican en casas (medidas en unidades) y condominios (medidas en m^2). En las Tablas 2.4 y 2.5 se muestran los valores totales obtenidos, aclarando que en el modelo se introducen segregados bajo estas dos categorías para cada distrito, lo que implica cuatro variables para cada categoría, pero debido a que en Turrucares no se presentaron registros de condominios, para el uso vivienda se tienen entonces siete variables en total (Ver Tabla 3.1).

Tabla 2.4. Cantidad de unidades habitaciones tipo casa tramitadas por mes

Vivienda tipo casas (unidades)						
Año		2015	2016	2017	2018	2019
Mes	Enero	93	32	52	62	28
	Febrero	66	57	41	37	17
	Marzo	51	65	46	46	44
	Abril	77	45	24	37	50
	Mayo	46	41	28	33	32
	Junio	20	49	45	16	23
	Julio	39	42	38	38	44
	Agosto	71	51	33	19	32
	Setiembre	79	54	63	43	22
	Octubre	19	88	14	45	26
	Noviembre	37	64	66	24	25
	Diciembre	29	77	37	26	53

Fuente: Colegio Federado de Ingenieros y Arquitectos

Tabla 2.5. Cantidad de m^2 tipo condominio tramitadas por mes

Vivienda tipo condominios (m^2)						
Año		2015	2016	2017	2018	2019
Mes	Enero	0	0	0	11406	0
	Febrero	23339	3176	0	0	0
	Marzo	0	0	0	0	0
	Abril	61	0	0	0	0
	Mayo	0	0	6906	0	0
	Junio	0	2403	0	0	0
	Julio	0	7665	0	0	0
	Agosto	0	0	0	20445	0
	Setiembre	0	3183	0	0	0
	Octubre	19046	0	0	34705	0
	Noviembre	0	0	0	0	0
	Diciembre	0	418	0	0	0

Fuente: Colegio Federado de Ingenieros y Arquitectos

Área Industrial: indica la cantidad de permisos de construcción de área industrial en metros cuadrados, incluye: plantas industriales o de producción, bodegas y similares. En la Tabla 2.6 se muestran los valores totales obtenidos. Estas se consideran en una sola categoría y por lo que generan tres variables una por cada distrito analizado a excepción del distrito de San José que no registró permisos de construcción de este uso durante el periodo de análisis (Ver Tabla 3.1).

Tabla 2.6. Cantidad de industrias en m^2 tramitadas por mes

Industria (m^2)						
Año		2015	2016	2017	2018	2019
Mes	Enero	4410	1318	0	2751	9722
	Febrero	6000	941	0	19120	17277
	Marzo	1286	2465	3157	0	28129
	Abril	1856	61487	13028	11781	13089
	Mayo	422	183	6763	77	39377
	Junio	327	875	5409	1003	4949
	Julio	16353	813	12278	692	0
	Agosto	0	10405	3296	5157	40089
	Setiembre	0	29120	76	0	0
	Octubre	0	18732	15120	4227	105
	Noviembre	2990	26965	3	11210	18255
	Diciembre	10546	7200	13972	30586	40

Fuente: Colegio Federado de Ingenieros y Arquitectos

Área Comercial: registra la cantidad de permisos de construcción de área comercial en metros cuadrados, incluye: locales comerciales, centros comerciales, supermercados y bancos. En la Tabla 2.7 se muestran los valores totales obtenidos. Estas se consideran en una sola categoría y por lo que generan cuatro variables una por cada distrito analizado durante el periodo de análisis (Ver Tabla 3.1).

Área de servicios: registra la cantidad de permisos de construcción de área de servicios en metros cuadrados. Básicamente se incluyen oficinas de diferentes tipos. En la Tabla 2.8 se muestran los valores obtenidos. Estas se consideran en una sola categoría y por lo que generan cuatro variables una por cada distrito analizado durante el periodo de análisis (Ver Tabla 3.1).

Cabe mencionar que en la base de datos se encontraron otros usos de suelo del tipo servicios como por ejemplo calderas, estaciones de servicio y telecomunicaciones. Pero estos usos no se considerar generadores de viajes.

Tabla 2.7 Cantidad de comercio en m^2 tramitados por mes

Comercio (m^2)						
Año		2015	2016	2017	2018	2019
Mes	Enero	510	1514	6590	66	54
	Febrero	1927	18527	62	8	677
	Marzo	44	0	1555	199	0
	Abril	324	105	4612	4346	2084
	Mayo	1262	73	97	265	267
	Junio	419	227	931	2293	460
	Julio	991	891	1210	945	244
	Agosto	856	3437	760	24194	698
	Setiembre	133	395	5769	229	10875
	Octubre	503	252	0	1949	851
	Noviembre	83	39	1428	538	0
	Diciembre	84	0	19319	0	175

Fuente: Colegio Federado de Ingenieros y Arquitectos

Tabla 2.8. Cantidad de servicios en m^2 tramitados por mes

Servicios (m^2)						
Año		2015	2016	2017	2018	2019
Mes	Enero	0	0	333	0	0
	Febrero	0	109	0	0	0
	Marzo	0	0	0	0	0
	Abril	0	141	0	0	0
	Mayo	543	0	0	54	0
	Junio	0	0	0	0	0
	Julio	0	0	232	2175	0
	Agosto	0	0	0	0	184
	Setiembre	153	0	0	0	0
	Octubre	0	78	0	0	0
	Noviembre	0	0	0	0	0
	Diciembre	0	0	0	0	0

Fuente: Colegio Federado de Ingenieros y Arquitectos

En la Figura 2.7 se muestra el comportamiento de estas variables mes a mes en el periodo de análisis. Para efectos del modelo, la información presentada fue traspuesta seis meses para el caso de viviendas y un año para el resto de los usos de suelo considerando que es este lapso necesario para que el permiso otorgado tenga un efecto en el tráfico vehicular.

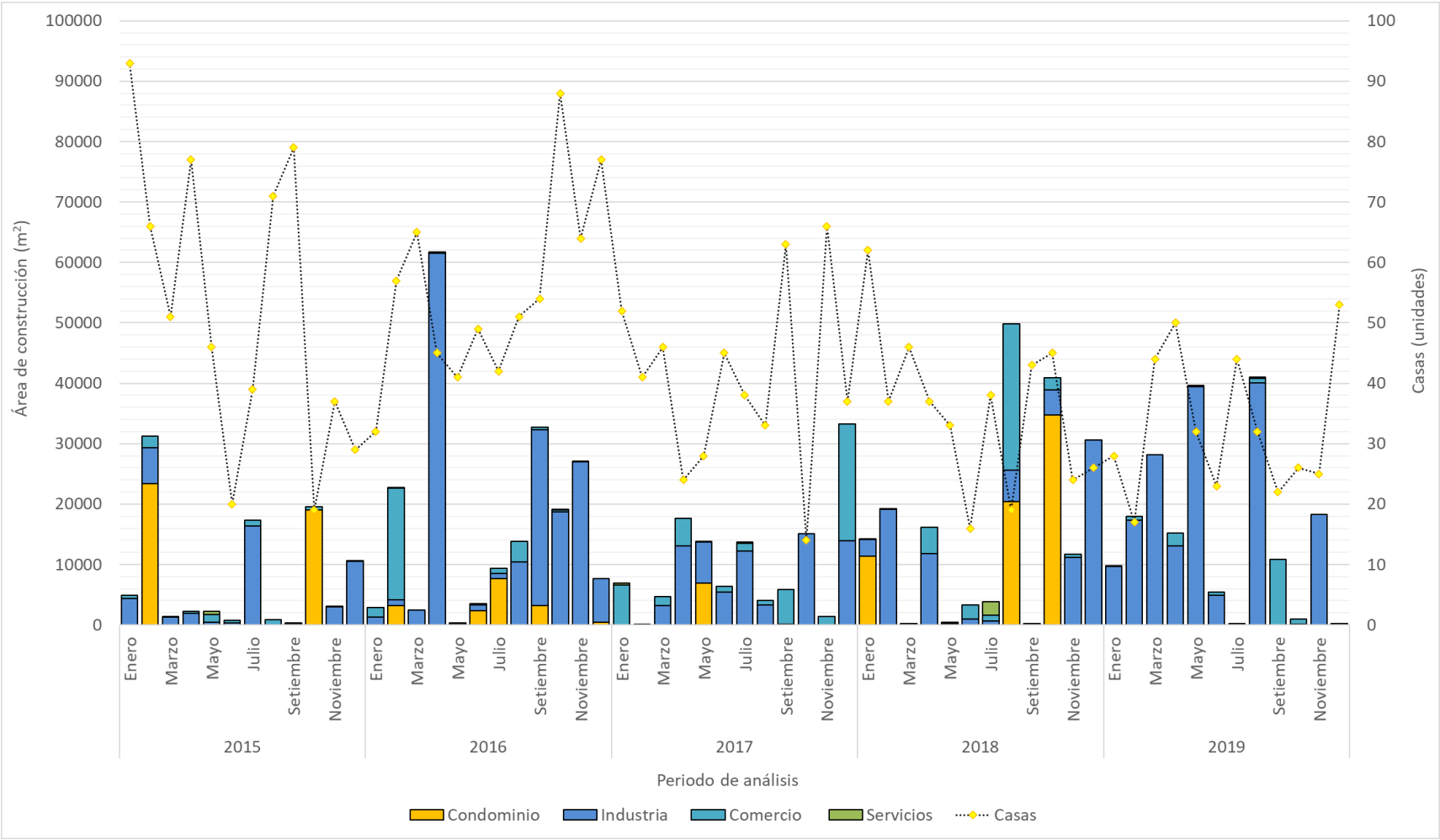


Figura 2.7. Gráfico del Comportamiento de las variables internas medidas mensualmente

Fuente: Elaboración propia utilizando los datos del CFIA

Desde el punto de vista demográfico y espacial se consideraron las siguientes variables internas:

Población: se considera la población total del área de influencia, es decir la población de los distritos considerados según el Instituto Nacional de Estadística y Censo (Instituto Nacional de Estadística y Censo, 2020). En la Tabla 2.9 se muestran los resultados obtenidos los cuales son anuales. Para un mejor detalle de estos resultados se muestra la Figura 2.8 donde gráficamente se muestra el aporte de población en cada uno de los distritos considerados.

Tabla 2. 9. Población del área de influencia

Población (habitantes)					
Año	**2015**	**2016**	**2017**	**2018**	**2019**
La Garita	8637	8768	8894	9017	9139
San Antonio	28541	29004	29464	29916	30356
San José	47751	48491	49224	49944	50654
Turrucares	8807	8893	8975	9056	9134

Fuente: Instituto Nacional de Estadística y Censo

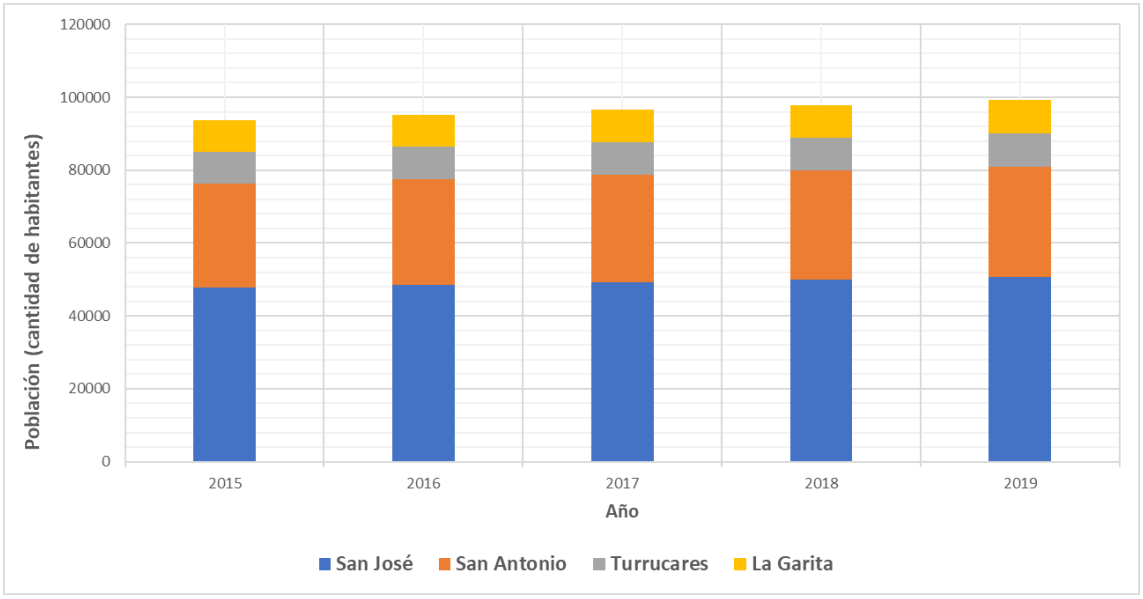


Figura 2. 8. Población por distrito y total de los distritos considerados

Fuente: Elaboración propia utilizando los datos del INEC

Distancia a la radial: otro aspecto importante para considerar es la distancia entre el lugar del desarrollo y la radial analizada, para la medición de esta distancia se tomó la ubicación de la cabecera distrital y un punto medio de la radial. Utilizando la aplicación de Street Maps se logra determinar dicha distancia. En algunas ocasiones se presentaron dos alternativas de recorrido por lo que se escogió la que representaba menos tiempo (no precisamente corresponde a la menor distancia). En la Figura 2.9 se muestran los recorridos, tiempos y distancia y la selección realizada.

Es de esperar que a mayor población se tenga un mayor efecto en la radial y que por el contrario la distancia sea inversamente proporcional al efecto del tráfico en la radial, por lo que el cociente población / distancia se considera como un factor de ajuste para los usos de suelo indicados en las Tablas de la 2.4 a la 2.8. En la Tabla 2.10 se muestran los factores de ajuste población/ distancia a utilizar.

Tabla 2. 10. Factores de ajuste población/ distancia

Factor de Población/Distancia					
Distrito	**2015**	**2016**	**2017**	**2018**	**2019**
San José	8,84	8,98	9,12	9,25	9,38
San Antonio	5,95	6,04	6,14	6,23	6,32
Turrucares	1,11	1,13	1,14	1,15	1,16
La Garita	0,84	0,85	0,86	0,88	0,89

En la Figura 2.9 se muestra el comportamiento de las variables internas consideradas mediante un gráfico que expone las variables medidas en metros como columnas apiladas y las refiere al eje vertical principal del gráfico (lo que también permute visualizar el total de m^2 construidos por mes independientemente de la categoría) mientras que las medidas en unidades se muestran con una línea y se refieren al eje vertical secundario. Para las casas y condominios se utilizan colores similares como indicación que pertenecen a la misma categoría de variables.

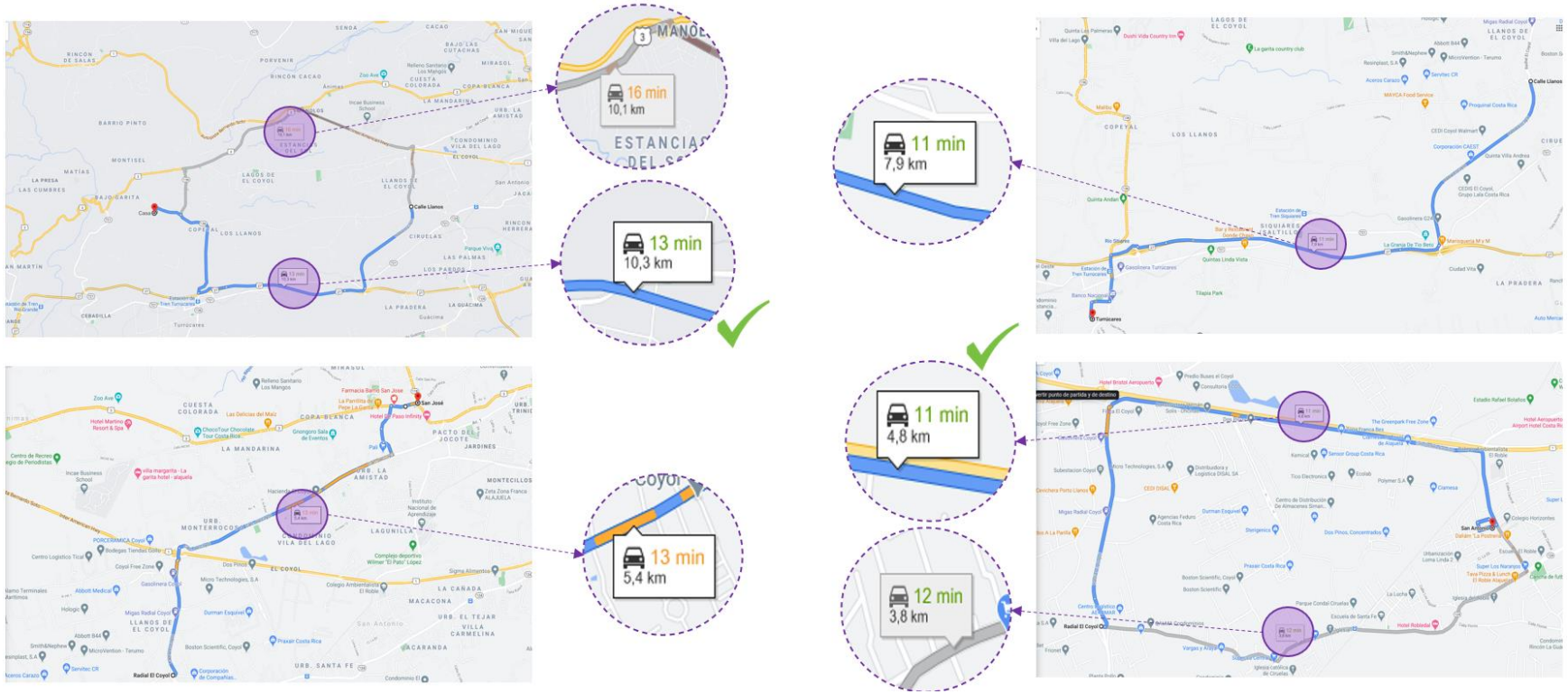


Figura 2. 9. Distancia entre la cabecera distrital y la radial

Fuente: Elaboración propia con base los datos de la aplicación Streets Maps

2.2.2. Variables Externas

Son variables externas a los distritos analizados y corresponden a mediciones a nivel nacional (Ver Tabla 3.1), son consideradas porque, por su naturaleza, sugieren inferir en la tasa de motorización o bien al tráfico vehicular de una zona:

TPD Ruta Nacional 1: se refiere al Tránsito Promedio Diario (TPD) de la Ruta Nacional 1 en su promedio mensual, la estación a utilizar en este caso es la Estación Agres ubicada frente al Hotel Aeropuerto en la Ruta Nacional 1. Esta información fue obtenida de la base de datos del MOPT (Ministerio de Obras Públicas y Transportes, 2020). Sin embargo, la instalación del equipo de medición fue a partir del año 2017, por lo que se debe proyectar hacia el pasado los valores faltantes para ser utilizados en la base de datos.

Es considerada como una posible variable porque es la ruta nacional más cercana hacia el norte de la radial en estudio. De hecho, como se mostró en la Figura 1.1, estas interceptan en el intercambio vial conocido como Intercambio Coyol.

El procedimiento para estimar los datos faltantes es similar al realizado inicialmente para corregir la variable dependiente, partiendo del uso de la curva logística para proyección de tráfico para la cual se requiere el cálculo de la capacidad de la ruta y los parámetros de la regresión, con la diferencia de que se comporta mucho mejor al ser una extrapolación y no una interpolación como en el caso de la variable dependiente, siendo que para la mayoría de los meses (excepto junio y setiembre) los R^2, tanto simple como ajustado rondan valores entre 0,8 y 0,95 y el estadístico t de student supera los valores ±2,96. Dado que solamente se tienen tres años de registro utilizables (del año 2017 al 2019) los p-valores indican que no se tiene la suficiente representatividad estadística, sin embargo, es con la única información que se cuenta. Las estaciones con más registros distan considerablemente de la radial en estudio.

Finalmente, con el valor de la capacidad y los parámetros de la regresión, se estiman los datos faltantes, se tabulan mostrándose en la Tabla 2.11 y se grafican según se muestra en la Figura 2.10. Cabe mencionar que este cálculo se pudo revisar con series de tiempo. pero

al resultar una variable sin importancia en el modelo, según se verá más adelante, se dejó a ese nivel el cálculo.

Tabla 2. 11. TPD de la Ruta Nacional 1 para el periodo de análisis

Tránsito Promedio Diario Ruta Nacional 1						
Año		2015	2016	2017	2018	2019
Mes	Enero	28801	34772	38724	44153	47128
	Febrero	24989	32808	38757	44231	48279
	Marzo	28100	34836	40413	43456	48663
	Abril	35621	38538	40372	43165	46026
	Mayo	35293	38429	40864	43211	46113
	Junio	39645	41546	42983	43679	47183
	Julio	38532	41153	42840	44474	48021
	Agosto	38746	40940	42116	44252	46985
	Setiembre	37881	39763	41367	41157	46154
	Octubre	35390	38777	40762	44198	46955
	Noviembre	39350	42201	43558	46478	48848
	Diciembre	36742	41048	42996	47257	49108

Fuente: Ministerio de Obras Públicas y Transportes

Cantidad de vehículos en Costa Rica (Veh en CR): Se refiere a la cantidad de vehículos matriculados en Costa Rica tanto nuevos como usados. Se incluyen todos los tipos de vehículos (livianos, pesados y buses) y el dato se obtuvo de la página web del Ministerio de Hacienda. En la Tabla 2.12 se muestran los datos obtenidos (Ministerio de Hacienda, 2020) y estos son graficados en la Figura 2.10 junto con el TPD de la Ruta Nacional 1.

Tabla 2. 12. Cantidad de vehículos importados por mes en Costa Rica

Cantidad de vehiculos matriculados en Costa Rica						
Año		2015	2016	2017	2018	2019
Mes	Enero	7490	6179	6449	5851	4362
	Febrero	4787	6727	5735	5811	4153
	Marzo	6971	8466	8967	6740	6185
	Abril	6589	9132	8038	7150	5019
	Mayo	5943	7140	8168	5986	4764
	Junio	6081	7314	6427	5223	4146
	Julio	6881	6770	5702	5104	4232
	Agosto	5794	7191	5666	4666	3829
	Setiembre	5601	6759	6540	5927	4872
	Octubre	7149	7053	6526	6209	5433
	Noviembre	8511	8810	8652	6633	5848
	Diciembre	7203	7319	6257	4500	4368

Fuente: Ministerio de Hacienda

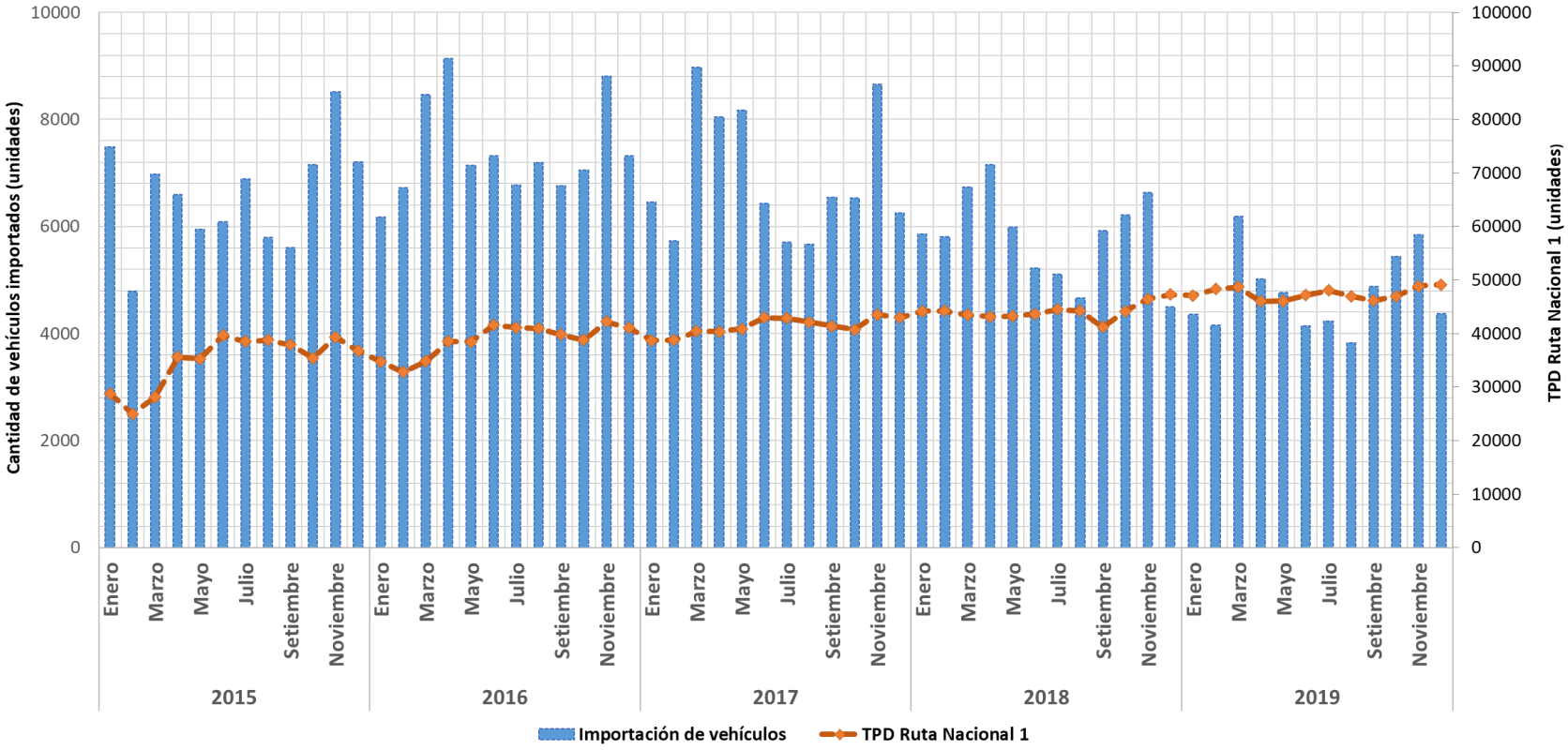


Figura 2. 10. Comportamiento de las variables externas medidas mensualmente – relacionadas con cantidad de vehículos

Fuente: Elaboración propia utilizando los datos de Planificación Sectorial del MOPT y Ministerio de Hacienda

En el extremo sur se ubica la Ruta Nacional 27 de la cual también se buscó información histórica para considerar su relación con el TPD de la radial en estudio. No obstante, la estación con información más cercana a la Radial Siquiares – Coyol es la Estación PK30+620 (ubicada en el puente del Río Grande sobre la Ruta Nacional 27) la cual dista a más de ocho kilómetros de esta radial por lo que no se consideró oportuno tomarla en cuenta.

Indicadores económicos: se consideran los indicadores económicos de Índice Mensual de Precio al Consumidor (IPC) y el Índice Mensual de Actividad Económica, con estos valores se pretende analizar el efecto de la economía del país en el tráfico. En las Tablas 2.13 y 2.14 se muestra los valores recopilados los cuales fueron obtenidos de la página web del Banco Central de Costa Rica (Banco Central de Costa Rica, 2020) y la Figura 2.11 se muestra gráficamente su comportamiento en el periodo de análisis.

Tabla 2. 13. Índice Mensual de Precio al Consumidor

Indice Mensual de Precio al Consumidor (IPC)						
Año		2015	2016	2017	2018	2019
Mes	Enero	99,96	99,79	100,44	102,86	104,58
	Febrero	99,80	99,76	100,85	103,08	104,66
	Marzo	99,94	98,86	100,42	103,05	104,51
	Abril	99,87	98,95	100,58	102,97	105,10
	Mayo	99,57	99,14	100,80	102,85	105,24
	Junio	100,00	99,12	100,88	103,03	105,53
	Julio	99,57	100,04	101,21	103,32	106,30
	Agosto	99,48	100,08	100,99	103,25	106,20
	Setiembre	99,22	99,60	101,24	103,48	106,09
	Octubre	98,94	99,49	101,80	103,86	106,02
	Noviembre	98,93	99,49	101,97	104,30	106,24
	Diciembre	99,12	99,87	102,45	104,52	106,11

Fuente: Banco Central de Costa Rica

Tabla 2. 14. Índice Mensual de Actividad Económica

Indice Mensual de Actividad Economica (IMAE)						
Año		2015	2016	2017	2018	2019
Mes	Enero	107,1	113,5	117,2	121,1	123,1
	Febrero	107,9	114,4	117,0	119,5	122,2
	Marzo	111,6	115,3	121,5	124,9	127,2
	Abril	104,3	110,0	113,0	117,6	118,0
	Mayo	108,6	113,1	117,9	123,1	123,1
	Junio	108,5	111,9	118,7	122,0	122,5
	Julio	109,7	112,5	116,0	120,3	122,0
	Agosto	109,2	111,6	114,7	118,8	120,2
	Setiembre	107,8	112,6	114,8	116,2	120,3
	Octubre	111,0	115,8	119,7	122,2	126,2
	Noviembre	114,3	121,0	123,9	125,5	128,7
	Diciembre	112,8	118,6	122,6	123,9	127,6

Fuente: Banco Central de Costa Rica

Precio de los combustibles: se consideran los precios de los combustibles más utilizados, estos son: gasolina súper, gasolina regular y diésel. En las Tablas de la 2.15 a la 2.17 se muestra los valores recopilados los cuales fueron obtenidos de la página web de la Refinadora Costarricense de Petróleo (Refinadora Costarricense de Petróleo, 2020) la Figura 2.12 se muestra gráficamente su comportamiento en el periodo de análisis.

Tabla 2. 15. Precio de la gasolina súper

Precio de combustible Gasolina Súper (Precio en colones costarricenses)						
Año		2015	2016	2017	2018	2019
Mes	Enero	537,0	559,0	592,0	604,0	596,0
	Febrero	536,0	559,0	620,0	629,0	582,0
	Marzo	570,0	445,0	593,0	653,0	581,0
	Abril	601,0	463,0	600,0	642,0	618,0
	Mayo	600,0	502,0	622,0	662,0	691,0
	Junio	643,0	518,0	597,0	680,0	684,0
	Julio	672,0	602,0	584,0	677,0	659,0
	Agosto	676,0	590,0	577,0	677,0	692,0
	Setiembre	635,0	566,0	602,0	681,0	632,0
	Octubre	563,0	579,0	632,0	680,0	616,0
	Noviembre	559,0	581,0	605,0	693,0	635,0
	Diciembre	559,0	562,0	628,0	637,0	634,0

Fuente: Refinadora Costarricense de Petróleo

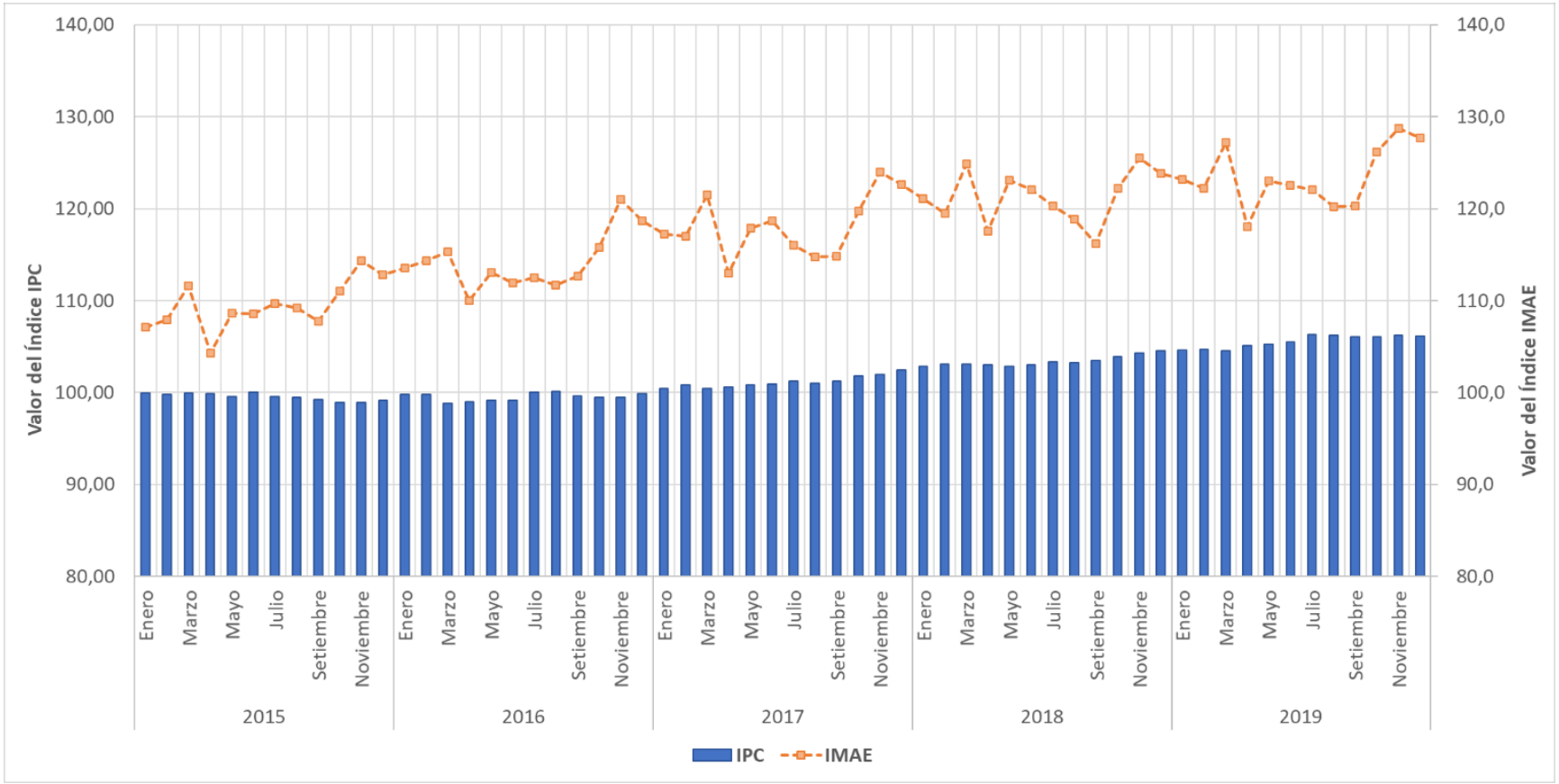


Figura 2. 11. Comportamiento de las variables externas medidas mensualmente – relacionadas con indicadores económicos

Fuente: Elaboración propia utilizando los datos del Banco Central de Costa Rica

Tabla 2. 16. Precio de la gasolina regular

Precio de combustible Gasolina Plus 91 (Precio en colones costarricenses)						
Año		2015	2016	2017	2018	2019
Mes	Enero	503,7	536,0	562,0	593,0	581,0
	Febrero	503,0	536,0	591,0	608,0	568,0
	Marzo	545,0	424,0	567,0	624,0	565,0
	Abril	574,0	446,0	571,0	614,0	602,0
	Mayo	575,0	483,0	597,0	633,0	669,0
	Junio	610,0	498,0	569,0	653,0	665,0
	Julio	636,0	575,0	559,0	652,0	638,0
	Agosto	639,0	563,0	555,0	655,0	666,0
	Setiembre	603,0	541,0	576,0	663,0	609,0
	Octubre	540,0	554,0	603,0	663,0	597,0
	Noviembre	536,0	555,0	581,0	673,0	617,0
	Diciembre	536,0	525,0	605,0	626,0	618,0

Fuente: Refinadora Costarricense de Petróleo

Tabla 2. 17. Precio del diésel

Precio de combustible Diésel (Precio en colones costarricenses)						
Año		2015	2016	2017	2018	2019
Mes	Enero	474,0	449,0	475,0	514,0	534,0
	Febrero	474,0	448,0	489,0	537,0	525,0
	Marzo	493,0	345,0	484,0	546,0	526,0
	Abril	500,0	358,0	475,0	531,0	542,0
	Mayo	478,0	348,0	470,0	540,0	568,0
	Junio	514,0	380,0	450,0	562,0	571,0
	Julio	498,0	451,0	443,0	556,0	527,0
	Agosto	477,0	448,0	449,0	555,0	543,0
	Setiembre	451,0	443,0	492,0	565,0	526,0
	Octubre	458,0	467,0	507,0	586,0	533,0
	Noviembre	449,0	468,0	513,0	598,0	552,0
	Diciembre	449,0	427,0	526,0	587,0	547,0

Fuente: Refinadora Costarricense de Petróleo

Tipo de cambio del dólar: la economía de Costa Rica es muy sensible al cambio del tipo de dólar, por lo que se incluyen estos valores como variable independiente. El valor mensual corresponde al promedio diario del mes correspondiente. En la Tabla 2.18 se muestran los valores recopilados los cuales fueron obtenidos de la página web del Banco Central de Costa

Rica (Banco Central de Costa Rica, 2020) y la Figura 2.12 se muestra gráficamente su comportamiento en el periodo de análisis junto con el precio de los combustibles

Tabla 2. 18. Tipo de cambio del dólar

Tipo de cambio de venta del dólar US al colón costarricense (Precios en colones)						
Año		2015	2016	2017	2018	2019
Mes	Enero	543,74	542,79	561,15	571,97	609,65
	Febrero	542,93	543,01	566,74	573,72	613,30
	Marzo	539,26	541,73	566,28	570,17	606,92
	Abril	538,13	542,34	568,48	568,25	602,92
	Mayo	538,78	543,77	580,53	568,41	595,21
	Junio	541,86	549,65	576,92	570,43	589,81
	Julio	540,72	554,56	575,37	570,30	580,19
	Agosto	540,47	556,99	577,71	571,03	571,86
	Setiembre	541,24	558,49	578,67	582,43	581,26
	Octubre	540,77	559,73	574,20	595,81	584,50
	Noviembre	539,73	560,68	570,43	615,02	579,72
	Diciembre	540,63	558,82	569,64	604,30	571,80

Fuente: Banco Central de Costa Rica

Finalmente, en la Tabla 2.19 se muestran las variables a considerar en los modelos. Estas se subdividen en siete categorías: Vivienda, Industrial, Comercial, Servicios, Económicas, Combustible y Cantidad de vehículos. Es de mucho interés considerar las variables segregadas por distrito para la aplicación de factor de ajuste por población/distancia, explicado en el Apartado 2.2.1.

En total suman 26 variables lo cual implica la utilización de algunas técnicas de reducción de variables explicadas en el Capítulo 1 y aplicadas al caso en el Capítulo 3 al construir los modelos.

Se debe mencionar que las variables independientes internas relacionadas con los usos de suelo, se presentan de forma acumulada para ser consistentes con la naturaleza del TPD como un valor total para cada mes, ya que este que acumula los efectos de las distintas ejecuciones de obra descritas en los permisos de construcción utilizados.

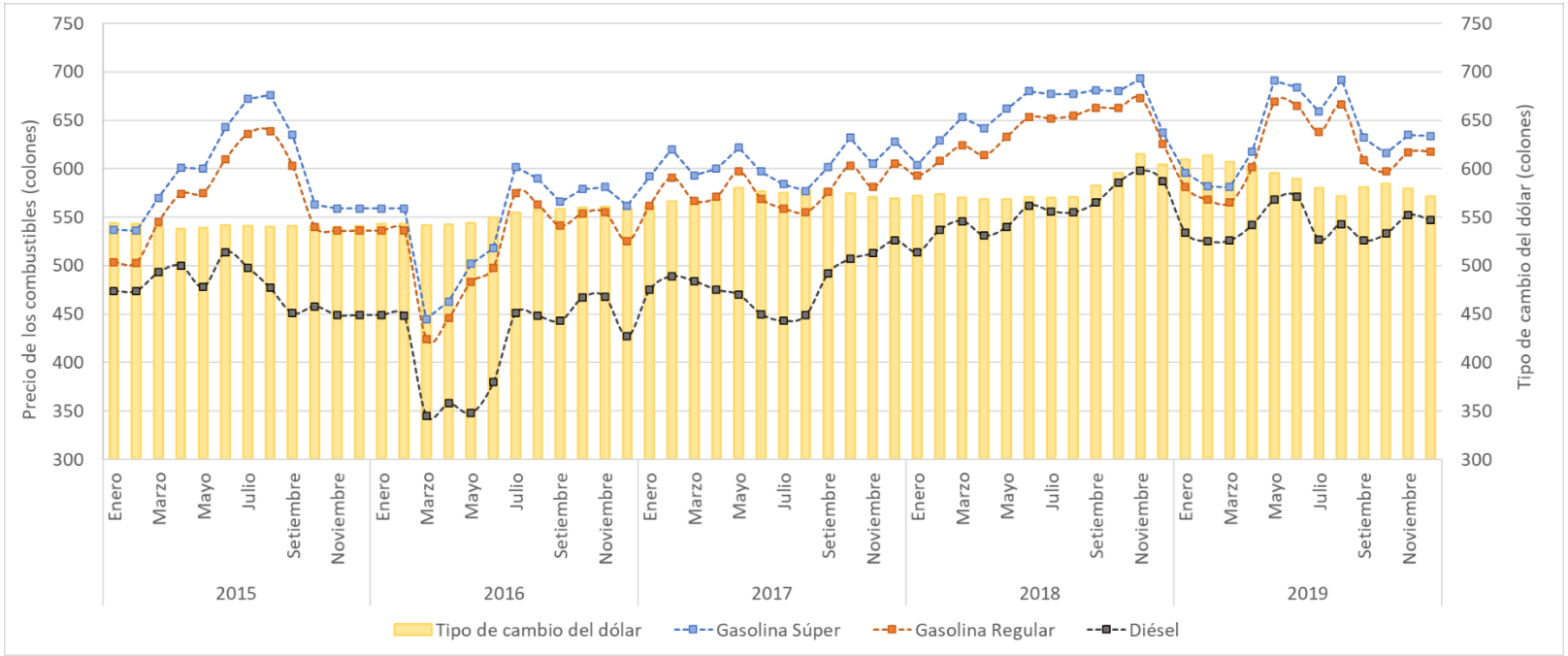


Figura 2. 12. Comportamiento de las variables externas medidas mensualmente – relacionadas con el precio de los combustibles y tipo de cambio del dólar

Fuente: Elaboración propia utilizando los datos del Banco Central de Costa Rica y RECOPE

Tabla 2. 19. Variables independientes consideradas

Categoría	Variable	Cantidad de variables
Vivienda	Garita Vivienda Acumulado	7
	San Antonio Vivienda Acumulado	
	San José Vivienda Acumulado	
	Turrucares Vivienda Acumulado	
	Garita Condominio Acumulado	
	San Antonio Condominio Acumulado	
	San José Condominio Acumulado	
Industrial	Garita Industrial Acumulado	3
	San Antonio Industrial Acumulado	
	Turrucares Industrial Acumulado	
Comercial	Garita Comercial Acumulado	4
	San Antonio Comercial Acumulado	
	San José Comercial Acumulado	
	Turrucares Comercial Acumulado	
Servicios	Garita Servicios Acumulado	4
	San Antonio Servicios Acumulado	
	San José Servicios Acumulado	
	Turrucares Servicios Acumulado	
Económicas	IPC	3
	IMAE	
	Tipo de cambio	
Combustible	Gasolina súper	3
	Gasolina regular	
	Diesel	
Cantidad de vehículos	TPD RN1	1
	Importación de vehículos a CR	1
Total de variables		26

Una vez definidas y analizadas las posibles variables, en el siguiente capítulo se presentan los modelos de proyección de tráfico

3. Modelo de proyección de tráfico

En este capítulo se presentan los modelos de proyección de tráfico realizados, partiendo de la revisión de datos que se mostró en el capítulo anterior para la construcción del modelo. Existen numerosos modelos o algoritmos que pueden utilizarse para estimar proyecciones de tráfico, para determinar cuál utilizar, se deben tener la totalidad de la información clara, ordenada y completa. Con ello, se logra iniciar las corridas del sistema bajo alguno de los algoritmos predispuestos para ello. Hasta no realizar las primeras modelaciones, no se puede escoger un modelo en específico, ya que es parte de la calibración del sistema, determinar el que mejor se adapte a las variables encontradas y por ello se trabaja con series de modelos para una condición especifica de la base de datos. En este proyecto se definen tres series las cuales serán descritas a continuación y se consideran tres modelos para cada serie: regresión lineal múltiple, regresión lineal de Ridge y árbol de regresión, cuya teoría se explicó en el Capítulo 1 y más adelante se detalla en los resultados de cada modelo para cada serie.

3.1. Definición de las series de modelos

Serie de Modelos A: La forma básica de esta serie de modelos es del tipo lineal como se muestra en la Ecuación 3.1.

$$Y_{ji} = \sum_{1}^{n} \beta_0 + \beta_n X_{ji} + u_{ji} \qquad \text{Ecuación 3.1}$$

Donde:

Y_{ji}: es el TPD del mes *j* del año *i*, que equivale a la variable dependiente

X_{ji}: son las variables independientes (n) del mes *j* del año *i* para considerar en el modelo.

β_n: son los parámetros de la ecuación a calcularse por regresión lineal.

u_{ji}: es el error asociado a la regresión.

Una vez descrita la Serie de Modelos A, en el Apartado 3.3.1 se presentan los resultados de este modelo que incluyen la forma del modelo, el resumen de los resultados estadísticos para los modelos de regresión lineal y sus equivalentes en el modelo de árbol de decisión.

Serie de Modelos B: Para la Serie de Modelos B se considera utilizar como base la tendencia logística para la proyección de volúmenes vehiculares la cual se muestra en la Ecuación 3.2 pero considerando que el exponente incluye todas las variables que quieren ser valoradas en el modelo. De esta forma se incorpora al modelo la capacidad de la vía (K).

$$TPD_{ji} = \frac{K}{1 + e^{\sum_1^n \beta_n X_{ji} + \beta_0}} \qquad \text{Ecuación 3.2}$$

Donde:

TPD_{ji}: es el tránsito promedio diario del mes *j* para el año *i* equivalente a la variable dependiente.

K: es el valor de la capacidad de la vía

X_{ji}: son las variables independientes (n) para considerar mes *j* para el año *i*.

β_n: son los parámetros de la ecuación a calcularse por regresión lineal.

Realizando una transformación logística se obtiene la expresión mostrada en la Ecuación 3.3 y finalmente en la Ecuación 3.4.

$$\sum_1^n \beta_n X_{ji} + \beta_0 = Ln\left(\frac{K}{TPD_{ji}} - 1\right) \approx T_{ji} \qquad \text{Ecuación 3.3}$$

$$T_{ji} \approx \sum_1^n \beta_n X_{ji} + \beta_0 \qquad \text{Ecuación 3.4}$$

Donde T_{ji} representa la transformada logarítmica del mes *j* del año *i*. El valor de la capacidad (K) fue estimado en el Capítulo 1 con un valor de 41872 veh/día. Lo anterior devuelve al modelo una expresión lineal para el cálculo de los parámetros de la regresión que una vez calculados vuelven a presentarse en el modelo original.

Una vez descrita la Serie de Modelos B, en el Apartado 3.3.2 se presentan los resultados de este modelo, que incluyen, igual que el anterior, la forma del modelo, el resumen de los resultados estadísticos para los modelos de regresión lineal y sus equivalentes en el modelo de árbol de decisión.

Se decide realizar un último modelo afectado las variables independientes internas por el factor población/distancia, discutido en el capítulo anterior, lo que genera la Serie de Modelos C.

Serie de Modelos C: Esta serie se define haciendo un ajuste en la Serie de Modelos B, el cual consiste en la aplicación del factor de ajuste por población/distancia al modelo anterior en las variables independientes internas ya que se había dejado de lado desde la Serie de Modelos A, por no haberse encontrado significancia en ese momento, pero bajo las nuevas características de los modelos y el ajuste en la base de datos desde la Serie de Modelos B, se considera la opción de su aplicación. En la Figura 3.1 se muestra un esquema del procedimiento de aplicación, análisis y selección del modelo de proyección, el cual es un resumen de lo comentado hasta el momento.

Una vez descrita la Serie de Modelos C, en el Apartado 3.3.3 se presentan los resultados de este modelo, de manera semejante a las otras dos series de modelos descritas antes.

No obstante, antes de mostrar los resultados de cada serie, se detalla el procedimiento de construcción del modelo, el cual es general para las tres series descritas.

Con los resultados estadísticos de las diferentes series y modelos se determina el Modelo Final. En la Figura 3.1 se muestra en forma esquemática el procedimiento seguido para la determinación de las series de modelos, cabe mencionar que cada serie de modelos representa una mejora de la anterior en cuando consideración de efectos que pretenden acercar más el modelo a la realidad como son el uso de una ecuación tipo logística y por lo tanto la consideración de la capacidad de la vía y el efecto del tamaño de la población y la distancia a la que el centro poblacional está del punto de medición, por lo que pese a que se muestran los resultados de todas las series se considera que Serie de Modelos C es la más completa y la base para la determinación del modelo final.

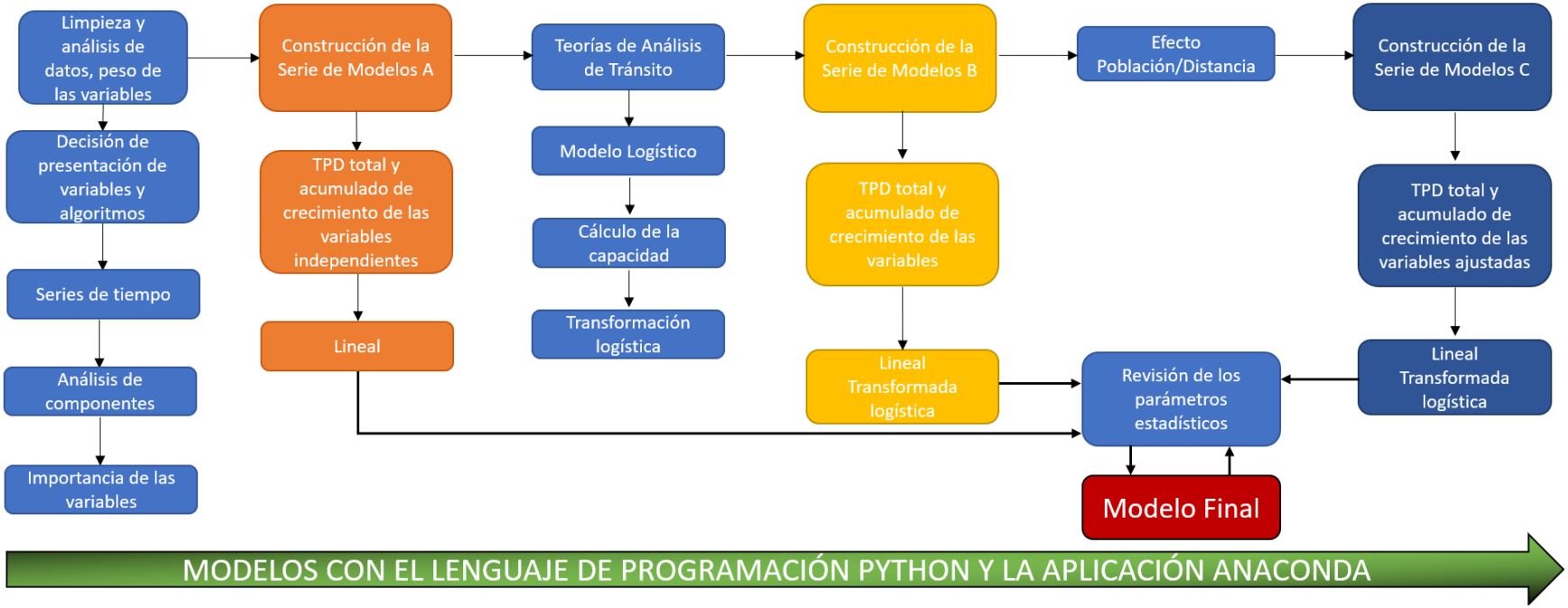


Figura 3. 1. Esquema del procedimiento de aplicación, análisis y selección del modelo de proyección

3.2. Construcción del modelo

La construcción del modelo se refiere a todos los ajustes y revisión de la base de datos previos a la determinación de los coeficientes de las regresiones y estadísticos de cada modelo en función del objetivo buscado y según las características de la base de datos. Como primer paso se debe realizar la revisión de la variable dependiente, luego se revisan las variables independientes y se plantea la división de los datos para el entrenamiento y las pruebas.

3.2.1. Revisión de la variable dependiente

Como se indicó en el Capítulo 1, es preferible que los modelos de regresión lineal cumplan con varios supuestos, entre ellos: la correcta especificación del modelo ($E(\varepsilon)=0$), la homogeneidad de la varianza de los errores ($Var(\varepsilon)=\sigma^2$), no correlación de los errores ($Cov(\varepsilon)=0$) y una distribución normal de los errores (ε-Normal $(0, \sigma^2)$. Por lo que mediante el lenguaje de programación Python se procede a revisar la normalidad y la ausencia de datos extremos de la variable dependiente.

La ausencia de valores extremos se verifica mediante un gráfico de cajas y la normalidad se verifica mediante la prueba de Shapiro – Wilk cuya hipótesis nula se define de la siguiente forma:

H_0: Los datos son normales

H_1: Los datos no son normales

El resultado del estadístico de la prueba de Shapiro – Wilk fue de 0,97809 con un p-valor de 0,35314 para la Serie de Modelos A (TPD no transformado logarítmicamente) y de 0,97398 con un p-valor de 0,22743 para las series de Modelos B y C (TPD transformado logarítmicamente) por lo que se acepta la hipótesis nula y por lo tanto se determina que la variable dependiente sigue un comportamiento normal para todas las series. En la Figura 3.2 se muestran como ejemplo, los gráficos obtenidos del análisis para las Series de Modelos B y C, de donde se muestra gráficamente que, en efecto, variable dependiente tiene un comportamiento normal y no presenta datos extremos.

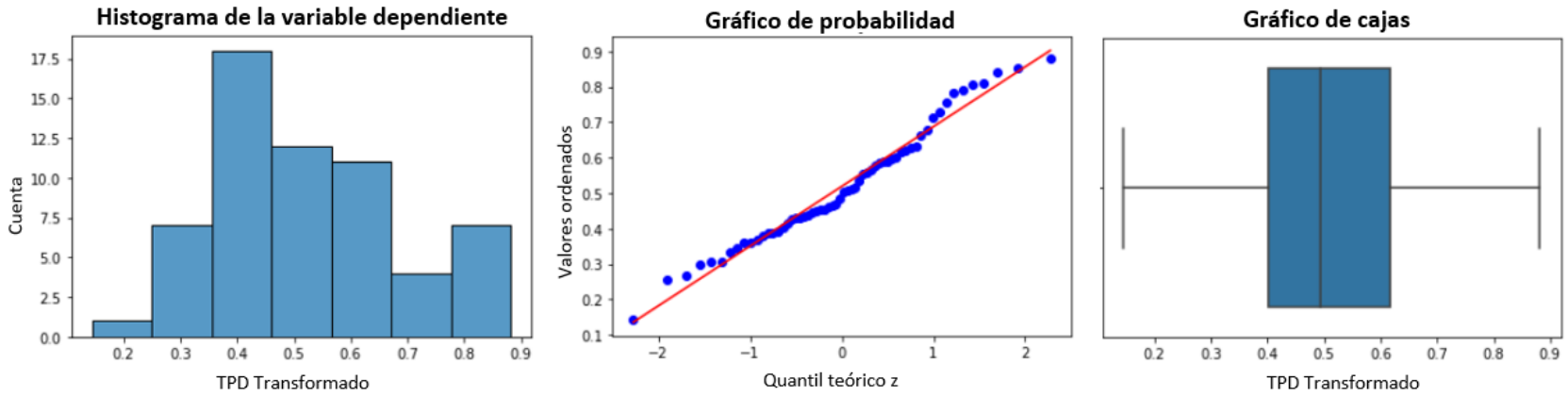


Figura 3. 2. Comprobación de normalidad de la variable dependiente

Una vez revisados los supuestos de normalidad en la variable dependiente se procede a revisar las variables independientes.

3.2.2. Revisión de las variables independientes

La revisión de las variables independientes consta de verificar cuáles de ellas son realmente significativas en el modelo. Se debe recordar que, debido a las limitaciones de la información de TPD y permisos de construcción de los diferentes usos de suelo, se obtuvieron únicamente 60 registros correspondientes a los doce meses de los años comprendidos entre el 2015 y el 2019. Por otra parte, se consideraron 26 variables las cuáles se pueden categorizar de la forma en que se mostró en la Tabla 2.19

Como puede notarse la cantidad de variables es excesiva para la cantidad de observaciones de la variable dependiente pues se recomienda que sea entre el 10% y el 20% de los registros, es decir estar en un rango de seis a doce variables por lo que se establece como meta escoger solamente nueve variables. Para no perder ninguna de las 26 variables se propone el uso del método de reducción de dimensión por componentes principales PCA el cual se presentó en el Capítulo 1 de este informe.

3.2.2.1. Reducción de dimensión por componentes principales PCA

Recordando lo indicado en el Capítulo 1 y de acuerdo con Jaadi (Jaadi, 2021), “el análisis de componentes principales, o PCA, es un método de reducción de dimensionalidad que se utiliza a menudo para reducir el tamaño de grandes conjuntos de datos, transformando un gran conjunto de variables en uno más pequeño que aún contiene la mayor parte de la información del conjunto grande. La reducción de la cantidad de variables de un conjunto de datos se obtiene naturalmente a expensas de la precisión, pero el truco en la reducción de la dimensionalidad es cambiar un poco de precisión por simplicidad”, el cual puede ser solicitado al lenguaje de programación Python, considerando las categorías descritas en la Tabla 2.19 para obtener PCA por categorías y la estimación sea más acertada.

Según se explicó antes, el proceso para llevar a cabo la definición de PCA se da en cuatro pasos básicos, el primero es la estandarización, el segundo es el cálculo de la matriz de

covarianza, el tercero es el cálculo de los autovectores y autovalores de la matriz de covarianza y finalmente la determinación del vector de características. La cantidad de componentes principales por categoría está relaciona con el porcentaje de varianza que se desee captar, en este caso se estimó adecuado un 90%, es decir se consideró el número de componentes principales necesario para asegurar que al menos el 90% de varianza estaba considerado. De esta forma en la Tabla 3.1 se muestran los componentes principales obtenidos para cada categoría analizada y se nombran como se muestra en la misma tabla. Adicionalmente, se indica el porcentaje captado de la varianza correspondiente a cada componente y el total para la verificación de llegar a la cantidad de componentes principales necesarios para lograr capturar el 90% de la varianza.

Tabla 3. 1. Reducción de variables por PCA

Categoría	Variable	Cantidad de variables	Cantidad de PCA	Porcentaje de varianza capturado
Vivienda	Garita Vivienda Acumulado	7	3	0,9897
	San Antonio Vivienda Acumulado			
	San José Vivienda Acumulado			
	Turrucares Vivienda Acumulado			
	Garita Condominio Acumulado		PC Vivienda 1	0,8780
	San Antonio Condominio Acumulado		PC Vivienda 2	0,0894
	San José Condominio Acumulado		PC Vivienda 3	0,0223
Industrial	Garita Industrial Acumulado	3	1	1,0000
	San Antonio Industrial Acumulado			
	Turrucares Industrial Acumulado		PC Industrial 1	1,0000
Comercial	Garita Comercial Acumulado	4	2	0,9747
	San Antonio Comercial Acumulado			
	San José Comercial Acumulado		PC Comercial 1	0,9109
	Turrucares Comercial Acumulado		PC Comercial 2	0,0638
Servicios	Garita Servicios Acumulado	4	2	0,9033
	San Antonio Servicios Acumulado			
	San José Servicios Acumulado		PC Servicios 1	0,7816
	Turrucares Servicios Acumulado		PC Servicios 2	0,1217
Económicas	IPC	3	2	0,9350
	IMAE		PC Economía 1	0,8556
	Tipo de cambio		PC Economía 2	0,0794
Combustible	Gasolina súper	3	1 (0,9460)	0,9460
	Gasolina regular			
	Diesel		PC Combustible 1	0,9460
Cantidad de vehículos	TPD RN1	1	1	No aplica
	Importación de vehículos a CR	1	1	No aplica
Total de variables		26	13	No aplica

Se debe recordar que en este procedimiento las variables resultantes no tienen una interpretación fácil ni un significado real ya que están construidas como combinaciones lineales de las variables iniciales. Sin embargo, las aplicaciones para generar los modelos permiten mantener como dato de entrada las variables originales.

Cabe mencionar que el TPD RN 1 y los Vehículos importados no se incluyeron en la reducción de dimensiones por PCA por no pertenecer a una categoría grupal. Es así como se reduce el número de variables de 26 a 13 sin perder información significativa de ninguna de ellas. No obstante, siguen siendo más que las recomendadas por lo que se procede a utilizar adicionalmente el método "Importancia de las Variables" el cual se describe en el siguiente apartado.

3.2.2.2. Importancia de las variables

El método de Importancia de las variables se expuso en el Capítulo 1. Al aplicar el algoritmo a las variables dependientes resultantes del apartado anterior, e indicarle que presente las nueve variables más importantes se obtienen los resultados que se muestran en la Tabla 3.2. Para cada serie de modelos el peso de la variable es diferente por lo que estos se pueden encontrar en los códigos de los apéndices según el modelo.

Tabla 3. 2. Resultado de la aplicación del método "Importancia de las Variables"

Variables seleccionadas
PC Vivienda 1
PC Vivienda 2
PC Vivienda 3
PC Industrial 1
PC Comercial 1
PC Servicios 1
PC Economía 1
PC Economía 2
PC Combustible 1

De esta forma la cantidad resultante de variables, utilizando ambos métodos, está en la media del rango recomendado de acuerdo con la cantidad de registros y bajo los métodos utilizados se asegura que las variables escogidas y presentadas en la Tabla 3.2 sean las más representativas para el modelo.

El peso relativo de estas variables depende de la Serie de Modelos que se analice. Los pesos relativos de cada serie se muestran en el apéndice asociado a cada modelo en la sección correspondiente del código (Sección 5).

3.2.3. División de la base de datos

Para corroborar los resultados del modelo, la base de datos se divide aleatoriamente en dos partes, la primera es utilizada para entrenar el modelo y es típico seleccionar entre un 70% y un 90% de los datos y la segunda parte es utilizada para probar el modelo y es la diferencia al porcentaje seleccionado. La decisión sobre los porcentajes a utilizar es subjetiva y depende del criterio del analista. Sin embargo, la recomendación general dada en foros de discusión al respecto, por ejemplo, en el sitio web de Data Science (Data Science, 2017) es que entre más amplia sea la base se puede optar por el límite inferior del rango, pero por el contrario para bases pequeñas, como es el caso en análisis, se debe escoger la mayor cantidad posible de los datos en la parte de entrenamiento. Es por ello por lo que, en este caso se seleccionó un 90% de los datos para el entrenamiento del modelo.

Es importante entonces revisar tanto los resultados del entrenamiento como de la prueba, los resultados se dan en función del estadístico R^2 y el análisis de este es la base para la escogencia del modelo.

Una vez que la base ha sido dividida, está lista para ser someterla a los algoritmos descritos en el Capítulo 1, por lo que en los siguientes apartados se muestran los resultados obtenidos.

3.3. Resultados de las series

A continuación, se describen los resultados obtenidos para cada una de las series de modelos descritas y los algoritmos seleccionados.

3.3.1. Resultados de la Serie de Modelos A

Para la Serie de Modelos A se obtuvieron los resultados que se muestran a continuación para cada uno de los algoritmos seleccionados.

Resultados del Modelo de Regresión Lineal Múltiple: Bajo la Serie de Modelos A y la base de datos asociada a esta serie, se realizó el modelo de regresión lineal. De este modelo se obtiene un R^2 de 0,7815, un R^2 ajustado de 0,7347 y un estadístico F de 16,69 sobre las nueve variables utilizadas. En la Tabla 3.3 se muestra el resumen de resultados estadísticos para cada una de las variables.

Tabla 3. 3. Resumen de resultados estadísticos de la Regresión lineal múltiple para la Serie de Modelos A

Variable	Coeficientes	Error estándar	t - Valor	p - Valor
Intercepto	15812,86	96,34	164,14	0,0000
PC Economía 1	-924,43	257,88	-3,58	0,0008
PC Industrial 1	-1797,73	463,22	-3,88	0,0003
PC Servicios 1	-566,8	346,18	-1,64	0,1077
PC Vivienda 1	-1024,85	486,31	-2,11	0,0400
PC Vivienda 2	-8,27	214,11	-0,04	0,9693
PC Combustible 1	17,05	85,25	0,20	0,8421
PC Comercial 1	-404,61	357,13	-1,13	0,2625
PC Vivienda 3	626,32	372,93	1,68	0,0992
PC Economía 2	-314,07	228,64	-1,37	0,1756

De estos resultados se encuentra que además del intercepto, las variables con representatividad estadística son PC Economía 1, PC Industrial 1 y PC Vivienda 1. Por otra parte, el R^2 del entrenamiento fue de 0,7815 y el R^2 ajustado es de 0,7347 con un error cuadrático medio (RMSE) de 695 veh/d (Cercano a un 4,6% del TPD actual de la radial). Al valorar este modelo con la base de datos de prueba se obtiene un R^2 de 0,8418 y un error cuadrático medio (RMSE) de 574,22 veh/d (Cercano a un 3,9% del TPD actual de la radial). El R^2 ajustado en caso de la base de prueba, no se estima porque para esta base de datos tiene solamente seis registros y deja de tener sentido su estimación.

En el Anexo A se muestra el gráfico de los valores predichos contra los valores estimados tanto para el entrenamiento como para la prueba para este modelo.

Resultados del Modelo de Regresión de Ridge: En el caso del Modelo de Regresión de Ridge para la Serie de Modelos A, se obtienen los mismos valores de R^2, R^2 ajustado y de estadístico F sobre las nueve variables utilizadas que para el caso de regresión lineal múltiple ya que este es un caso particular de la regresión lineal múltiple. No obstante, la diferencia radica en los coeficientes de la regresión y en los parámetros estadísticos asociados. En la Tabla 3.4 se muestra el resumen de resultados estadísticos para cada una de las variables.

Tabla 3. 4. Resumen de resultados estadísticos de la Regresión lineal de Ridge para la Serie de Modelos A

Variable	Coeficientes	Error estándar	t - Valor	p - Valor
Intercepto	15812,86	98,51	160,51	0,0000
PC Economía 1	-911,81	263,70	-3,46	0,0011
PC Industrial 1	-1014,73	473,68	-2,14	0,037
PC Servicios 1	-483,83	354,00	-1,37	0,1777
PC Vivienda 1	-394,04	497,29	-0,79	0,4318
PC Vivienda 2	-335,27	218,95	-1,53	0,1319
PC Combustible 1	-37,62	87,18	-0,43	0,6679
PC Comercial 1	-214,15	365,19	-0,59	0,5602
PC Vivienda 3	291,41	381,35	0,76	0,4483
PC Economía 2	-155,06	233,80	-0,66	0,5102

Similar al caso de la regresión lineal, de estos resultados se encuentra que además del intercepto, las variables con representatividad estadística son PC Economía 1 y PC Industrial 1, excluyéndose PC Vivienda 1.

De esta forma, el R^2 del entrenamiento fue de 0,7815 y el R^2 ajustado es de 0,7347 con un error cuadrático medio (RMSE) de 710,39 veh/d (Cercano a un 4,7% del TPD actual de la radial). Al valorar este modelo con la base de datos de prueba se obtiene un R^2 de 0,8282 y un error cuadrático medio (RMSE) de 598,40 veh/d (Cercano a un 4,0% del TPD actual de la radial). Se tienen la misma consideración para el R^2 ajustado del caso anterior para la base de prueba.

En el Anexo A se muestra el gráfico de los valores predichos contra los valores estimados tanto para el entrenamiento como para la prueba para este modelo.

Los resultados de la Regresión Lineal de Ridge son muy similares a los de la regresión lineal simple con la diferencia muy significativa que la Regresión Lineal de Ridge ya está regularizada y por lo tanto evita el subajuste y sobreajuste del modelo y por lo tanto generaliza mucho mejor y esto implica resultados más acertados al poner en práctica el modelo.

Resultados del Modelo de Árbol de Regresión: Como se mencionó antes, otro de los modelos utilizados fue el Modelo de Árbol de Regresión, que igualmente con la base de datos asociada a la Serie de Modelos A, se realizaron los ejercicios de entrenamiento y prueba.

Cabe mencionar que, para el algoritmo de árbol de regresión, no se tiene una estructura lineal del modelo, es decir, no tiene coeficientes y por lo tanto no tiene estadísticos asociados a ellos. Solamente se puede estimar el R^2, el R^2 ajustado y el error cuadrático medio (RMSE).

Como se explicó en el Capítulo 1, la estructura de este modelo es un esquema basado en los conceptos de profundidad máxima, cantidad de conectores mínimos y cantidad de hojas mínimas, los cuales también se conocen como hiperparámetros. En la Tabla 3.5 se muestran los valores probados de estos hiperparámetros y los escogidos como óptimos por el algoritmo. La representación gráfica de la Tabla 3.6 se muestra en la Figura 3.3.

Tabla 3. 5. Resumen de valores propuestos y utilizados para los hiperparámetros del árbol de regresión para la Serie de Modelos A

Hiperparámetro	Posibles valores	Valor seleccionado
Profundidad máxima	2, 3, 4 y 5	2
Conexiones mínimas	2, 3 y 4	2
Hojas mínimas	1 y 2	1

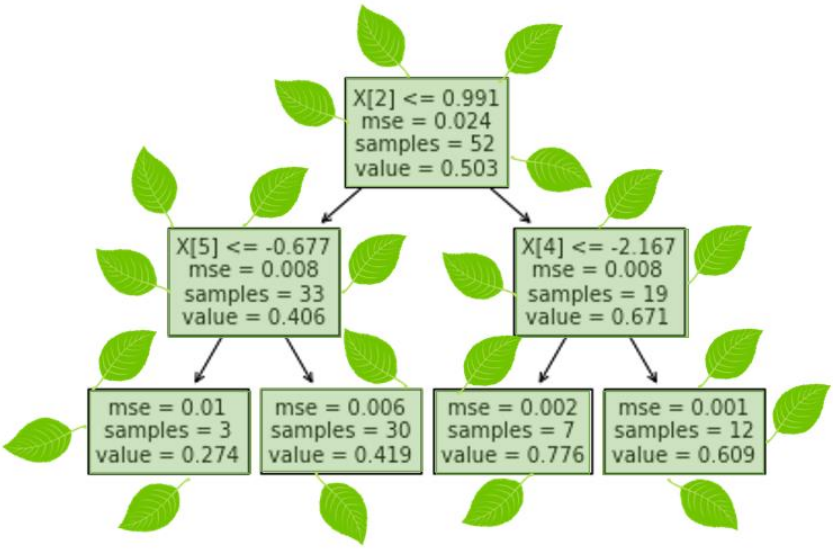


Figura 3. 3. Representación gráfica del árbol de regresión

De este modelo, para el entrenamiento, se obtiene un R^2 de 0,7399, un R^2 ajustado de 0,6867 y un error cuadrático medio (RMSE) de 690,41 veh/d (Cercano a un 4,6% del TPD actual de la radial). De manera similar al modelo anterior, con la base de datos de prueba se obtiene un R^2 de 0,7884 y un error cuadrático medio (RMSE) de 664 veh/d (Cercano a un 4,4% del TPD actual de la radial). Se tienen la misma consideración para el R^2 ajustado del caso anterior para la base de prueba. Los gráficos de valores predichos contra los valores estimados se muestran en el Apéndice A para este modelo al igual que los demás de la Serie de Modelos A.

3.3.2. Resultados de la Serie de Modelos B

Se presenta la Serie de Modelos B, el cual debe ser una mejor aproximación de la realidad que la Serie de Modelos A al considerar un modelo logístico y por lo tanto la capacidad vial de la radial. De este modelo se obtuvieron los resultados que se muestran a continuación para cada uno de los algoritmos seleccionados.

Resultados del Modelo de Regresión Lineal Múltiple: Bajo la Serie de Modelos B y la base de datos asociada a esta serie, se realizó el modelo de regresión lineal. De este modelo se obtiene un R^2 de 0,8024, un R^2 ajustado de 0,7601 y un estadístico F de 18,95 sobre las

nueve variables utilizadas. En la Tabla 3.6 se muestra el resumen de resultados estadísticos para cada una de las variables. Efectivamente se obtienen mejores resultados que la Serie de Modelos A.

Tabla 3. 6. Resumen de resultados estadísticos de la Regresión lineal múltiple para la Serie de Modelos B

Variable	Coeficientes	Error estándar	t - Valor	p - Valor
Intercepto	0,5025	0,01	52,27	0,0000
PC Servicios 1	0,0563	0,03	1,63	0,1091
PC Vivienda 1	0,1058	0,05	2,18	0,0338
PC Economía 1	0,0923	0,03	3,59	0,0008
PC Industrial 1	0,1810	0,05	3,92	0,0003
PC Vivienda 2	0,0039	0,02	0,18	0,8566
PC Comercial 1	0,0412	0,04	1,16	0,2528
PC Combustible 1	-0,0015	0,01	-0,18	0,8599
PC Vivienda 3	-0,0636	0,04	-1,71	0,0933
PC Economía 2	0,0316	0,02	1,39	0,1720

De estos resultados se encuentra que además del intercepto, nuevamente las variables con representatividad estadística son PC Economía 1, PC Industrial 1 y PC Vivienda 1. Por otra parte, el R^2 del entrenamiento fue de 0,8024 y el R^2 ajustado es de 0,7601 con un error cuadrático medio (RMSE) de 0,0693 equivalente a 700 veh/d (Cercano a un 4,7% del TPD actual de la radial). Al valorar este modelo con la base de datos de prueba se obtiene un R^2 de 0,8524 y un error cuadrático medio (RMSE) de 0,583 equivalente a 585 veh/d (Cercano a un 3,9% del TPD actual de la radial).

Al igual que en la Serie de Modelos A, el R^2 ajustado en caso de la base de prueba, no se estima porque para esta base de datos tiene solamente seis registros y deja de tener sentido su estimación. En el Anexo B se muestra el gráfico de los valores predichos contra los valores estimados tanto para el entrenamiento como para la prueba para este modelo.

Para el entrenamiento y la prueba de este modelo, los resultados estadísticos son mejores que la Serie de Modelos A correspondiente.

Resultados del Modelo de Regresión de Ridge: En el caso del Modelo de Regresión de Ridge para la Serie de Modelos B, se obtienen un valor de R^2 de 0,7932, un R^2 ajustado de 0,7489 y un estadístico F de 17,90 sobre las nueve variables utilizadas que igualmente superan su par en la Serie de Modelos A.

En la Tabla 3.7 se muestra el resumen de resultados estadísticos para cada una de las variables. Similar al caso de la regresión lineal, de estos resultados se encuentra que además del intercepto, las variables con representatividad estadística son PC Economía 1 y PC Industrial 1, excluyéndose PC Vivienda 1.

Por otra parte, el R^2 del entrenamiento fue de 0,7932 y el R^2 ajustado es de 0,7489 con un error cuadrático medio (RMSE) de 0,0709 equivalente a 727 veh/d (Cercano a un 4,8% del TPD actual de la radial). Al valorar este modelo con la base de datos de prueba se obtiene un R^2 de 0,8405 y un error cuadrático medio (RMSE) de 0,0606 equivalente a 624 veh/d (Cercano a un 4,2% del TPD actual de la radial). Se tienen la misma consideración para el R^2 ajustado del caso anterior para la base de prueba.

Tabla 3. 7. Resumen de resultados estadísticos de la Regresión lineal de Ridge para la Serie de Modelos B

Variable	Coeficientes	Error estándar	t - Valor	p - Valor
Intercepto	0,5025	0,01	51,10	0,0000
PC Servicios 1	0,0484	0,04	1,37	0,1764
PC Vivienda 1	0,0416	0,05	0,84	0,4062
PC Economía 1	0,0912	0,03	3,47	0,0011
PC Industrial 1	0,1020	0,05	2,15	0,0357
PC Vivienda 2	0,0369	0,02	1,69	0,0978
PC Comercial 1	0,0218	0,04	0,60	0,5523
PC Combustible 1	0,0040	0,01	0,46	0,6487
PC Vivienda 3	-0,0295	0,04	-0,78	0,4413
PC Economía 2	0,0156	0,02	0,67	0,5068

En el Anexo B se muestra el gráfico de los valores predichos contra los valores estimados tanto para el entrenamiento como para la prueba para este modelo.

Resultados del Modelo de Árbol de Regresión: En el caso de la Serie de Modelos B, la estructura e hiperparámetros son iguales a la Serie de Modelos A. Sin embargo, los resultados de este modelo, para el entrenamiento, son un R^2 de 0,7567, un R^2 ajustado de 0,7069 y un error cuadrático medio (RMSE) de 0,0690 equivalente a 706 veh/d (Cercano a un 4,7% del TPD actual de la radial).

De manera similar al modelo anterior, con la base de datos de prueba se obtiene un R^2 de 0,7991 y un error cuadrático medio (RMSE) de 0,0690 equivalente a 706 veh/d (Cercano a un 4,7% del TPD actual de la radial). Se tienen la misma consideración para el R^2 ajustado del caso anterior para la base de prueba.

3.3.3. Resultados de la Serie de Modelos C

Seguidamente se presenta la Serie de Modelos C, el cual también debe ser una mejor aproximación de la realidad que la Serie de Modelos A y B al considerar no solo un modelo logístico y por lo tanto la capacidad vial de la radial, sino también el factor población distancia. De este modelo se obtuvieron los resultados que se muestran a continuación para cada uno de los algoritmos seleccionados.

Resultados del Modelo de Regresión Lineal Múltiple: Bajo la Serie de Modelos C y la base de datos asociada a esta serie, se realizó el modelo de regresión lineal. De este modelo se obtiene un R^2 de 0,8032, un R^2 ajustado de 0,7611 y un estadístico F de 19,05 sobre las nueve variables utilizadas. En la Tabla 3.8 se muestra el resumen de resultados estadísticos para cada una de las variables. Efectivamente se obtienen mejores resultados que la Serie de Modelos A y B.

De estos resultados se encuentra que además del intercepto y otra vez las variables con representatividad estadística son PC Economía 1, PC Industrial 1 y PC Vivienda 1. Por otra parte, el R^2 del entrenamiento fue de 0,8032 y el R^2 ajustado es de 0,7611 con un error cuadrático medio (RMSE) de 0,0692 equivalente a 700 veh/d (Cercano a un 4,7% del TPD actual de la radial). Al valorar este modelo con la base de datos de prueba se obtiene un R^2

de 0,8558 y un error cuadrático medio (RMSE) de 0,576 equivalente a 590 veh/d (Cercano a un 3,9% del TPD actual de la radial).

Tabla 3. 8. Resumen de resultados estadísticos de la Regresión lineal múltiple para la Serie de Modelos C

Variable	Coeficientes	Error estándar	t - Valor	p - Valor
Intercepto	0,5025	0,01	52,39	0,0000
PC Economía 1	0,0910	0,03	3,57	0,0008
PC Vivienda 1	0,1051	0,05	2,17	0,0349
PC Servicios 1	0,0599	0,04	1,71	0,0934
PC Industrial 1	0,1839	0,05	3,89	0,0003
PC Comercial 1	0,0420	0,04	1,16	0,2531
PC Vivienda 2	0,0061	0,02	0,27	0,7901
PC Combustible 1	-0,0006	0,01	-0,07	0,9416
PC Vivienda 3	-0,0595	0,04	-1,50	0,1400
PC Economía 2	0,0325	0,02	1,43	0,1578

Al igual que en la Serie de Modelos A y B, el R^2 ajustado en caso de la base de prueba, no se estima porque para esta base de datos tiene solamente seis registros y deja de tener sentido su estimación. En el Anexo C se muestra el gráfico de los valores predichos contra los valores estimados tanto para el entrenamiento como para la prueba para este modelo.

Para el entrenamiento y la prueba de este modelo, los resultados estadísticos son mejores que la Serie de Modelos A y B correspondiente.

Resultados del Modelo de Regresión de Ridge: En el caso del Modelo de Regresión de Ridge para la Serie de Modelos C, se obtienen un valor de R^2 de 0,7941, un R^2 ajustado de 0,7500 y un estadístico F de 18 sobre las nueve variables utilizadas que igualmente superan su par en la Serie de Modelos A y B.

En la Tabla 3.9 se muestra el resumen de resultados estadísticos para cada una de las variables. Similar al caso de la regresión lineal, de estos resultados se encuentra que además del intercepto, las variables con representatividad estadística son PC Economía 1 y PC Industrial 1, excluyéndose PC Vivienda 1.

Tabla 3. 9. Resumen de resultados estadísticos de la Regresión lineal de Ridge para la Serie de Modelos C

Variable	Coeficientes	Error estándar	t - Valor	p - Valor
Intercepto	0,5025	0,01	51,21	0,0000
PC Economía 1	0,0907	0,03	3,48	0,0010
PC Vivienda 1	0,0403	0,05	0,81	0,4197
PC Servicios 1	0,0498	0,04	1,39	0,1702
PC Industrial 1	0,1027	0,05	2,12	0,0385
PC Comercial 1	0,0219	0,04	0,59	0,5582
PC Vivienda 2	0,0413	0,02	1,76	0,0846
PC Combustible 1	0,0048	0,01	0,56	0,5806
PC Vivienda 3	-0,0226	0,04	-0,56	0,5798
PC Economía 2	0,0163	0,02	0,7	0,4849

Por otra parte, el R^2 del entrenamiento fue de 0,7941 y el R^2 ajustado es de 0,7500 con un error cuadrático medio (RMSE) de 0,0708 equivalente a 727 veh/d (Cercano a un 4,8% del TPD actual de la radial). Al valorar este modelo con la base de datos de prueba se obtiene un R^2 de 0,8396 y un error cuadrático medio (RMSE) de 0,0607 equivalente a 624 veh/d (Cercano a un 4,2% del TPD actual de la radial). Se tienen la misma consideración para el R^2 ajustado del caso anterior para la base de prueba.

En el Anexo C se muestra el gráfico de los valores predichos contra los valores estimados tanto para el entrenamiento como para la prueba para este modelo.

Resultados del Modelo de Árbol de Regresión: En el caso de la Serie de Modelos C, la estructura e hiperparámetros son iguales a la Serie de Modelos A y B. Sin embargo, los resultados de este modelo, para el entrenamiento, son un R^2 de 0,7676, un R^2 ajustado de 0,7201 y un error cuadrático medio (RMSE) de 0,0677 equivalente a 693 veh/d (Cercano a un 4,6% del TPD actual de la radial).

De manera similar al modelo anterior, con la base de datos de prueba se obtiene un R^2 de 0,7372 y un error cuadrático medio (RMSE) de 0,0777 equivalente a 795 veh/d (Cercano a un 5,3% del TPD actual de la radial). Se tienen la misma consideración para el R^2 ajustado del caso anterior para la base de prueba.

3.4. Comparación de las series y modelos evaluadas

Para hacer una comparación más sencilla de los resultados de las series y modelos, se presenta la Tabla 3.10 donde se muestran los resultados del R^2 para cada uno de los modelos evaluados.

Tabla 3. 10. Comparación de los R^2 de los diferentes modelos y series evaluadas

Modelo	Serie de Modelos		
	A	B	C
Regresión Lineal Múltiple	0,7815	0,8024	0,8032
Regresión Lineal de Ridge	0,7815	0,7932	0,7941
Árbol de regresión	0,7399	0,7567	0,7676

De la tabla anterior se puede deducir fácilmente que, con forme se agregaron detalles al modelo, generando cada una de las series, los R^2 mejoran (lectura horizontal de la tabla) y si se compara los modelos para las diferentes en cada serie se determina que el Árbol de Regresión es el modelo menos acertado y las ligeras diferencias entre el modelo de Regresión Lineal Múltiple y el Modelo de Regresión de Ridge sugieren que se escoja Ridge que cuenta con la regularización de los coeficientes de la regresión y por lo tanto elimina posibles subajustes y sobre ajustes.

Así los resultados, el modelo a escoger sería el Modelo de Regresión de Ridge de la Serie de Modelos C. Sin embargo, se decide probar mediante la Prueba F, si las varianzas de los resultados de la aplicación del modelo de Regresión de Ridge son lo suficientemente diferentes en cada uno de los modelos para descartar el uso de una serie u otra. En la Tabla 3.11 se muestran los resultados de la aplicación de Modelo de Regresión de Ridge al conjunto de datos de prueba y el cuadrado de la desviación estándar (S^2).

Tabla 3. 11. Resultados del TPD estimado por el Modelo de Regresión de Ridge para todas las series

Muestra	Identificador	TPD Estimado por Ridge		
		Serie A	Serie B	Serie C
1	20182	16452	16447	16455
2	20181	16614	16607	15615
3	20189	16610	16612	16605
4	20176	16093	16076	16070
5	20162	15032	15008	15045
6	20155	13087	13085	13076
S^2		1928244	1929733	1710197

Se aplica la Prueba F para una confianza del 95% (análisis de dos colas), cinco grados de libertad dado que para cada serie se tienen seis muestras. En la Tabla 3.12 se presentan los resultados de la Prueba F para las siguientes hipótesis:

Primera prueba: comparación de Serie A con Serie C

H_0: $\sigma_A = \sigma_C$

H_1: $\sigma_A \neq \sigma_C$

Segunda prueba: comparación de Serie A con Serie B

H_0: $\sigma_A = \sigma_B$

H_1: $\sigma_A \neq \sigma_B$

Tercera prueba: comparación de Serie B con Serie C

H_0: $\sigma_B = \sigma_C$

H_1: $\sigma_B \neq \sigma_C$

Tabla 3. 12. Resultados de la Prueba F

Límites del Intervalo de Confianza			
Límite superior del Intervalo de Confianza para F			0,140
Límite inferior del Intervalo de Confianza para F			7,146
Prueba F	**Resultado**	**Referencia**	**Conclusión**
F_{A-C}	1,1275	Dentro del rango	No se puede rechazar la H_0
F_{A-B}	0,9992		
F_{B-C}	1,1284		

De esta manera se concluye que, estadísticamente no hay diferencia en aplicar una serie u otra. Esto es muy importante porque, no restringe el modelo a conseguir información o hacer estimaciones para las cuales se requiere un importante tiempo de gestión o datos de campo que no siempre son sencillos de estimar. Adicionalmente, al poder usar como base la Serie de Modelos A, que no depende de la capacidad vial de vía, se pueden hacer estimaciones por tipo de vehículo.

Para continuar con la determinación del modelo final, se parte del Modelo de Regresión de Ridge para la Serie de Modelos C, porque, pese a lo anterior, este se considera el más completo.

3.5. Modelo Final

Como se mencionó antes, partiendo del Modelo de Regresión de Ridge para la Serie C, se van eliminando las variables que no son estadísticamente significativas según el t-valor y el p-valor hasta lograr una significancia del 90% de todas las variables que se incluyen en el modelo con este valor se asegura que el primer componente principal de cada una de las variables de pendientes internas quede dentro del modelo. De modo que las variables a utilizar en el modelo final son las que se muestran en la Tabla 3.14.

Para este modelo se tiene un R^2 de 0,7545, un R^2 ajustado de 0,7336 y un estadístico F de 36,11 sobre las cuatro variables utilizadas. En la Tabla 3.14 también se muestra el resumen de resultados estadísticos para cada una de las variables donde se demuestra que todas las variables son altamente significativas y la forma del modelo es la que se presenta en la Ecuación 3.5

Tabla 3. 13. Resultados estadísticos del modelo final

Variable	Coeficientes	Error estándar	t - Valor	p - Valor
Intercepto	0,5025	0,01	46,90	0,00000
PC Economía 1 (PCE)	0,0730	0,03	2,94	0,00500
PC Servicios 1 (PCS)	0,0823	0,03	2,85	0,00632
PC Vivienda 1 (PCV)	0,0458	0,02	1,96	0,05501
PC Industrial 1 (PCI)	0,1501	0,03	4,79	0,00002

$$TPD = \frac{K}{1 + e^{(0{,}0730PCE + 0{,}0823PCS + 0{,}0458PCV + 0{,}1501PCI + 0{,}5025)}}$$

Ecuación 3.5:

Donde:

TPD: es el tránsito promedio diario

K: es el valor de la capacidad de la vía

PCE, PCS, PCV y PCI: componentes principales definidos en la Tabla 3.13.

Por otra parte, el R^2 del entrenamiento fue de 0,7545 y el R^2 ajustado es de 0,7336 con un error cuadrático medio (RMSE) de 0,0773 equivalente a 792 veh/d (Cercano a un 5,3% del TPD actual de la radial). Al valorar este modelo con la base de datos de prueba se obtiene un R^2 de 0,8653 y un error cuadrático medio (RMSE) de 0,0557 equivalente a 570 veh/d (Cercano a un 3,8% del TPD actual de la radial).

En las Figuras 3.4 y 3.5 se muestra el gráfico de los valores predichos contra los valores estimados tanto para el entrenamiento como para la prueba para este modelo, los cuales corroboran los resultados obtenidos.

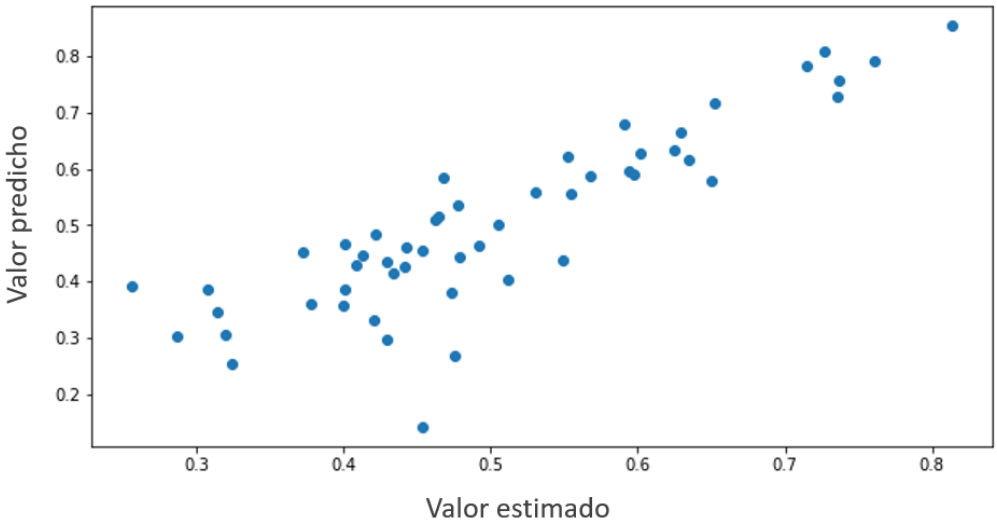


Figura 3. 4. Gráfico de los valores predichos y estimados para la base de entrenamiento Modelo Final

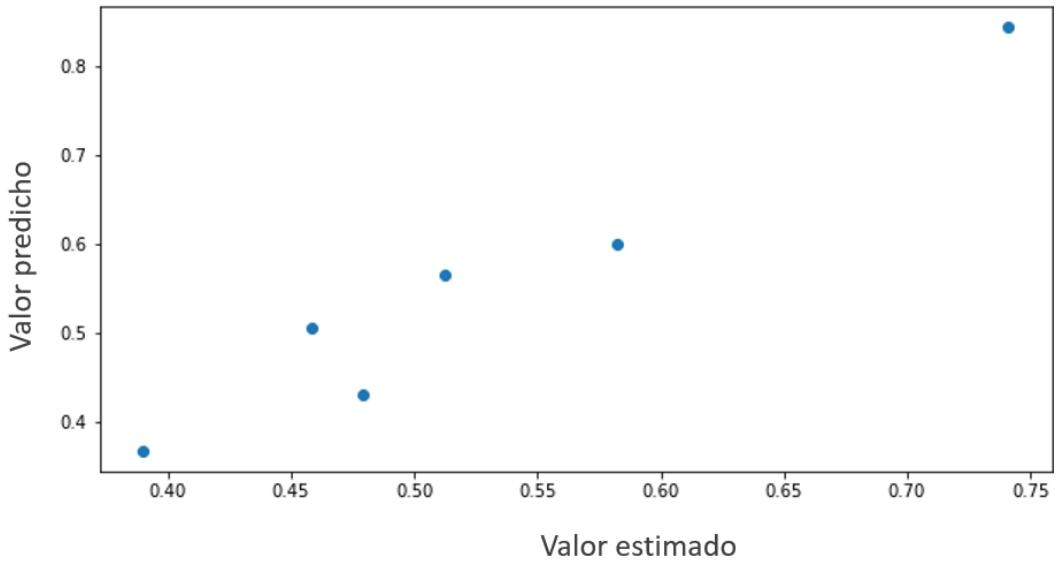


Figura 3. 5. Gráfico de los valores predichos y estimados para la base de prueba Modelo Final

Una vez determinado el modelo final, se procede a realizar una aplicación de este en el siguiente apartado.

3.6. Análisis de elasticidad

A continuación, se presenta un análisis de elasticidad del modelo para diferentes escenarios. El modelo permite realizar una superposición de efectos, sin embargo, para una valoración más sencilla, se realiza un escenario para cada alteración de uso de suelo o indicador más importante en el modelo tal como se describe en la Tabla 3.14. donde se describen 17 escenarios diferentes para ser analizados.

También se debe indicar que en algunos usos de suelo se mantuvo la misma afectación para valorar el impacto del factor población/distancia que es parte de este modelo, pero no en todos los usos de suelo se puede mantener la misma afectación porque esta debe ser quedar dentro de los rangos que se ha desarrollado cada uso en los distritos ya que este tipo de modelos es bueno para interpolar, pero no para extrapolar de manera abrupta. La escogencia de la afectación está sujeta a la subjetividad del autor, pero razonada con los valores de la base de datos y la lógica general.

Tabla 3. 14. Escenarios para ser analizados

Escenario	Variable	Afectación
1	Garita Vivienda Acumulado	25 unidades más
2	San Antonio Vivienda Acumulado	25 unidades más
3	San José Vivienda Acumulado	25 unidades más
4	Turrucares Vivienda Acumulado	25 unidades más
5	Garita Condominio Acumulado	1000 m^2 más
6	San Antonio Condominio Acumulado	1000 m^2 más
7	San José Condominio Acumulado	500 m^2 más
8	Garita Industrial Acumulado	10000 m^2 más
9	San Antonio Industrial Acumulado	10000 m^2 más
10	Turrucares Industrial Acumulado	2000 m^2 más
11	Garita Servicios Acumulado	100 m^2 más
12	San Antonio Servicios Acumulado	250 m^2 más
13	San José Servicios Acumulado	250 m^2 más
14	Turrucares Servicios Acumulado	100 m^2 más
15	IPC	110
16	IMAE	100
17	Tipo de cambio	650 colones

El código de este modelo final se muestra en el Apéndice D de este informe y en la Tabla 3.15 se muestran los resultados del análisis realizado al comparar los TPD obtenidos en cada escenario con el dato de TPD obtenido para el mes de diciembre de 2019.

Es importante mencionar que el diferencial entre el TPD base y los datos de TPD que arroja el modelo no son la generación de viajes para el uso de suelo en análisis, sino que corresponde a la afectación en TPD de la Radial Coyol.

Tabla 3. 15. Resultados de la aplicación del Modelo Final

Resultados Modelo Final						
Resultado del modelo					Regresión de Ridge	
					TPD	
Estimación de TPD para diciembre 2019 (último dato histórico)					16807	
Variable tipo	**Escenario**	**Variación**	**Efecto**	**Distrito**	**TPD**	**ΔTPD**
Variables internas	1	Variación de cantidad de casas	25 casas más	Garita	16847	40
	2		25 casas más	San Antonio	16840	33
	3		25 casas más	San José	16820	13
	4		25 casas más	Turrucares	16845	38
	5	Variación de m^2 condominios	1000m^2 más	Garita	16816	9
	6		1000m^2 más	San Antonio	16816	9
	7		500m^2 más	San José	16821	14
	8	Variación de m^2 industria	10000m^2 más	Garita	16619	-188
	9		10000m^2 más	San Antonio	16728	-79
	10		2000m^2 más	Turrucares	16460	-347
	11	Variación de m^2 servicios	100m^2 más	Garita	17178	371
	12		250m^2 más	San Antonio	17312	505
	13		250m^2 más	San José	16895	88
	14		100m^2 más	Turrucares	17163	356
Variables externas	15	Variación IPC	Alcanza un valor de 110	General	17480	673
	16	Variación IMAE	Alcanza un valor de 100	General	14827	-1980
	17	Variación tipo de cambio	Alcanza un valor de 650	General	18371	1564

Analizando los resultados, cada variable presenta una afectación diferente en el modelo y seguidamente se analiza cada una de ellas.

Para el caso de las viviendas los cuatro distritos presentan valores positivos y muy similares a los que la generación de viajes de una residencia pueda aportar por lo que se entiende que las viviendas dan un aporte directo a la radial como la vía de acceso para entrar o salir de la zona del Coyol.

De forma similar a la anterior, al tratarse de condominios, medidos en m^2, podría tratarse de cuatro a seis unidades habitacionales para el caso de la Garita y San Antonio y de dos a tres unidades habitacionales para el caso de San José. Para los dos primeros el resultado es congruente a la generación de viajes típica del uso residencia. Para el caso de San José, sí muestra un efecto muy significativo para este uso de suelo, lo cual puede deberse a que es el distrito con mayor población y desarrollo urbano.

La valoración de la industria implica una reducción del TPD sobre la Radial, esto se debe a que, al crecer estos usos de suelo en cada distrito, es menos frecuente el uso de la radial porque la apertura de estos desarrollos crea oportunidades de trabajo en la vecindad donde se emplazan.

En el caso de los servicios se genera un aumento considerable del TPD sobre la radial. Esto se debe a que es un uso de suelo mucho más dinámico que los demás donde durante largas jornadas se dan servicios puntuales que provocan una mayor cantidad de viajes.

Por otra parte, un aumento en el IPC genera un aumento significativo en el TPD de la radial ya que es el efecto de una mayor demanda de bienes y servicios. Por el contrario, una caída del IMAE a su base, es decir a un valor de 100, implica una caída abrupta en la economía del país y por lo tanto es lógico que el TPD caiga de manera que lo indica el modelo. La variación en el tipo de cambio al aumentar el precio del dólar puede influir en aumentar los viajes sobre la radial porque la mayoría de las empresas de la zona son multinacionales y por lo tanto un aumento en el precio del dólar puede generar mayor inversión y con ello más oportunidades de trabajo y crecimiento.

Como se puede ver, la mayoría de los resultados se pueden explicar bajo cierta lógica. Sin embargo, no existe un modelo perfecto y este en particular requiere la inclusión de muchos

más registros de TPD y las variables independientes asociadas para dar mejores resultados, pues es evidente que la precisión del modelo está estrechamente relacionada con la cantidad de datos que lo sustenten. Los pocos datos obligan a utilizar técnicas de reducción de dimensiones y aunque se procura perder la mínima información, el modelo se ve restringido.

Aun con estas limitaciones, se considera que el Modelo Final, logra captar la esencia del objetivo de esta investigación.

Una vez seleccionado y probado el modelo, en el siguiente capítulo se consideran una serie de conclusiones de este proyecto.

4. Conclusiones

A continuación, se enlistan una serie de conclusiones sobre la investigación realizada:

1. Se logra determinar un modelo para predecir el comportamiento del tráfico vehicular en la Radial Coyol utilizando métodos de ciencia de datos incluyendo variables de desarrollo inmobiliario, aspectos demográficos e índices económicos.

2. El modelo escogido está basado en el algoritmo de Regresión de Ridge dado que fue el que presentó los mejores resultados considerando que en sí mismo incluye una regularización, lo que desestima al modelo de Regresión Lineal Múltiple. El modelo basado en el algoritmo de árbol de regresión se desestima porque no generaliza adecuadamente.

3. La base del modelo escogido fue la Serie de Modelos C, cuya forma del modelo es una ecuación logística, que, para la aplicación de los algoritmos seleccionados, debe aplicarse una transformación logística pero que permite modelar de manera real el comportamiento de tráfico vehicular y además incluye el factor población/distancia y esto la convierte en una serie de modelos muy completa. Esto es congruente al comparar los resultados del coeficiente de determinación R^2 de todas las series. Sin embargo, se determinó que no existe diferencia estadísticamente significativa para descartar las otras series de modelos realizadas.

4. Al ser estadísticamente considerable la Serie de Modelos A, pueden realizarse submodelos que estimen el TPD para determinado tipo de vehículo lo que permite la determinación de impactos a este nivel y siendo que la zona esta muy influenciada por la industria es interesante entender el efecto propio de los vehículos pesados que tienen tanta influencia en el diseño geométrico y de pavimentos de rutas.

5. Los resultados del entrenamiento de los datos son muy importantes, pero no se puede dejar de lado el análisis de los resultados de la prueba del modelo ya que es fundamental demostrar que el modelo generaliza adecuadamente.

6. El modelo elaborado permite la estimación del efecto vehicular sobre la radial de diferentes usos de suelo y escenarios socioeconómicos, ya sea de manera independiente o simultánea.

7. Las variables económicas, el desarrollo habitacional, los servicios y la industria son las variables que afectan mayormente el tráfico en la radial. El efecto de las variables económicas es relativo a la productividad del país. En el caso de las viviendas y los servicios, su impacto es aumentar el tráfico vehicular. Mientras que el desarrollo de la industria hace que se resuelvan los viajes en cada distrito y se disminuya el tráfico sobre la radial.

8. Los efectos que pueden valorarse con este modelo son aquellos que se encuentren dentro de rango de valores de la base de datos suministrada para cada variable. Ya que estos modelos son muy buenos en la interpolación, pero generalmente son poco precisos y deficientes en la extrapolación.

9. Al utilizar valores acumulados, de cierta forma se está extrapolando para encontrar resultados, por lo que con más razón los efectos deben ser razonados dentro de la lógica del comportamiento histórico, para que la extrapolación no vaya más allá de los valores acumulados en tres o cuatro meses para la variable a analizar.

10. El modelo es aplicable para estimar la exposición comercial de un eventual desarrollo ante el crecimiento colateral de la zona, lo que permite hacer algunos análisis de rentabilidad de proyectos en función del tráfico vehicular que enfrenta y siendo que es aplicable un modelo más sencillo (alguna de las opciones de la Serie

de Modelos A, por ejemplo) es mucho más rentable, simple y eficiente la prestación de servicios de consultoría en este posible frente de trabajo.

11. La precisión del modelo está estrechamente relacionada con la cantidad de registros, lo cual representa una importante limitación en esta investigación ya que no hace tanto tiempo se cuenta con bases de datos digitales y de fácil acceso. Para ampliar la base de datos y tener un mayor provecho de las herramientas de ciencia de datos es necesario que los diferentes gestores de la administración pública y privada incluyan dentro de sus servicios la facilitación de datos de manera rápida y accesible y se continue en la línea de la digitalización de bases de datos.

Referencias bibliográficas:

Anaconda. (14 de julio de 2021). *Blog*. Obtenido de Anaconda Web site: http://www.anaconda.com

Austroads. (23 de Abril de 2019). *austroads.com.au/publications/road-desing/agrd02.* Obtenido de austroads.com.au: http://austroads.com.au/publications/road-desing/agrd02

Banco Central de Costa Rica. (12 de Octubre de 2020). *Banco Central de Costa Rica Estadísticas.* Obtenido de Banco Central de Costa Rica: http://www.bccr.fi.cr

Behar, R. (2003). *Validación de supuestos en el modelo de regresión* (Primera ed.). Colombia: Universidad del Valle.

Benavente, J. M., Otero, A., & Javiera, V. (2007). *Econometría I.* Chile: Facultad de Economía y Negocios de la Universidad de Chile.

Cal y Mayor R., R., & Cárdenas G., J. (2007). *Ingeniería de Trásito: Fundamentos y Aplicaciones.* México, DF: Alfa y Omega Grupo Editor, S.A de C.V.

Castro, L., Picado, G., & Rodríguez, S. (2018). Evolución Historica de la Modelación de Demanda de Transporte Urbano en Costa Rica. *Infraestrcutura Vial*, 4-47.

Coyol Free Zone. (19 de Diciembre de 2019). *Coyol Free Zone*. Obtenido de Coyol Free Zone: http://www.coyolfz.com

Data Science. (Marzo de 2017). *Statistics - Train and test data split*. Obtenido de Data Science Web site: http://www.datascience.stackexchange.com

Draper, N., & Smith, H. (1998). *Applied regressión analysis* (Tercera ed.). New York: John Wiley & Sons.

Epicalsoft-Instance-Blog. (27 de Febrero de 2019). *2019/02/azure-machine-learning-sobreajuste-y.html.* Obtenido de Epicalsoft blogspot Web Site: http://epicalsoft.blogspot.com/

Ferrero, R., & López, J. (15 de Julio de 2021). *blog-dat/que-son-los-arboles-de-decision-y-para-que-sirven.* Obtenido de Maxima Formación Web side: http://www.maximaformacion.es

Instituto Nacional de Estadística y Censo. (14 de Octubre de 2020). *Estadísticas*. Obtenido de Instituto Nacional de Estadística y Censo: http://www.inec.cr

Instituto Nacional de Estadística y Censo. (20 de Abril de 2020). *INEC*. Obtenido de Página Web INEC: http://www.inec.go.co

Jaadi, Z. (13 de Julio de 2021). *data-science/step-step-explanation-principal-component-analysis.* Obtenido de Builtin: http://www.builtin.com

Joaquín Amat, R. (Junio de 2017). *documentos/35_principal_component_analysis.* Obtenido de Ciencia de datos Web site: https://www.cienciadedatos.net

Katiyar Quiros, I. (2018). *Análisis del desempeño de geosintéticos como sistemas de intercapa en pavimentos flexibles.* San José, Costa Rica: Universidad de Costa Rica.

Kraemer , C., Pardillo, J., Rocci, S., Romana, M., Sánchez Blanco, V., & del Val, M. Á. (2004). *Ingeniería de Carreteras.* España: McGraW-Hill.

Maklin, C. (18 de Agosto de 2019). *ridge-regression-python-example-f015345d936b.* Obtenido de Towards data science Wen side: https://towardsdatascience.com/ridge-regression-python-example-f015345d936b

Marín, N. (2019). *Análisis Vial Intercambio Coyol.* Estudio de Tráfico, Heredia, Costa Rica.

Mata, R. (7 de setiembre de 2019). Aplicación de aprendizaje automatizado en transportes. (N. Marín, Entrevistador)

Mendez, D. (29 de junio de 2012). Emergencia obligó a Autopistas del Sol a abrir la Radial El Coyol. *La Nación*.

Ministerio de Hacienda. (12 de Octubre de 2020). *Estadística Ministerio de Hacienda.* Obtenido de Minsiterio de Hacienda: http://www.hacienda.go.cr

Ministerio de Obras Públicas y Transportes. (29 de Abril de 2020). *Ministerio de Obras Públicas y Transportes*. Obtenido de Página Web Ministerio de Obras Públicas y Transportes: http://www.mopt.go.cr

Montero Granados, R. (2016). *Modelos de Regresión Lineal Múltiple.* España: Universidad de Granada.

Montgomery, D., Peck, E., & Vinning, G. (2002). *Introducción al análisis de regresión lineal* (Tercera ed.). México: CECSA.

Municipalidad de Alajuela. (19 de Diciembre de 2019). *Plan Regulador*. Obtenido de Página Web Municipalidad de Alajuela: http://www.munialajuela.go.cr

Orange. (8 de Diciembre de 2020). *Orange*. Obtenido de Orange: http://www.orange.biolab.si

Penta Analytics. (19 de Diciembre de 2019). *Aplicaciones de Machine Learnig en el Transporte y la Logística*. Obtenido de Página web de Penta Analytics.cl: http://www.analytics.cl

Programa Estado de la Nación. (2018). *Estado de la Nación.* San José.

Rawlings, J., Pantula, S., & Dickey, D. (1998). *Applied Regression Analysis: A Research Tool* (Segunda ed.). New York: Springer - Verlag.

Refinadora Costarricense de Petróleo. (20 de Octubre de 2020). *RECOPE*. Obtenido de Historicos: http://www.recope.go.cr

Reyes Aguilar, P. (15 de Julio de 2007). *Metodología de análisis con series de tiempo.* México. Obtenido de fdocuments.ec: https://fdocuments.ec/document/seriesdetiempocomp.html

Robledano, A. (23 de Setiembre de 2019). *Lenguajes de Programación*. Obtenido de OpenWebinars: http://www.openwebinars

Russell, S., & Norvig, P. (1994). *Inteligencia artificial, un enfoque moderno* (Segunda ed.). Madrid, España: Pearson Educación, S.A.

SAS. (26 de Octubre de 2020). *Data- minig.* Obtenido de SAS: http://www.sas.com

SAS. (26 de Octubre de 2020). *Machine Learning, una expresión de la Inteligencia Artificial.* Obtenido de SAS: http://www.sas.com

Secretaría de Integración Económica Centroaméricana (SIECA). (2011). *Manual Centroaméricano de Normas para el Diseño Geométrico de Carreteras con enfoque de Gestión de Riesgo y Seguridad Vial.* América Central.

Secretaria de Planificación Sectorial. (2020). *Anuario de Transito 2019.* San José, Costa Rica : MOPT.

Secretaría de Planificación Sectorial, Proceso de Planificación Estratégica Multimodal de Servicios de Infraestructura y de Transporte. (2018). *Anuario de información de tránsito 2018.* San José.

TIBCO. (15 de Julio de 2021). *https://docs.tibco.com/pub/sfire-dsc/6.5.0/doc/html/TIB_sfire-dsc_user-guide/GUID-BF34695C-47FD-44CA-9002-8A3FA69FB4E0.html*. Obtenido de Tibco Web site: https://docs.tibco.com

Torres Sanabria, L. (2007). *Formulación y evaluación de posibles modelos de crecimiento del transito en la ciudad de Pereira en función de las variables indirectas medibles.* Pereira, Colombia.

Transportation Research Board. (2016). *Highway Capacity Manual.* Washington D.C: Transportation Research Board.

Universidad de Costa Rica. (25 de Setiembre de 2018). *Noticias.* Obtenido de Universidad de Costa Rica: http://www.ucr.ac.cr/noticias

Valverde González, G. (2010). Proyección de Tráficos mediante un modelo microeconómico. *Tecnología MOPT*, 1-14.

Apéndice A: Memoria de cálculo Python para la Serie de Modelos A

SERIE A - MODELO RADIAL COYOL

1. CONFIGURACION GENERAL

1.1. Importamos las Librerias a utilizar

In [1]:

```
# Librerías básicas
import pandas as pd
import numpy as np
import warnings

# Librerias Machine Learnign & Estadísticas
from sklearn import linear_model
from sklearn.tree import DecisionTreeRegressor
from scipy import stats
import statsmodels
import statsmodels.api as sm
import statsmodels.formula.api as smf
import statsmodels.stats.api as sms
from statsmodels.compat import lzip
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import StandardScaler

# Visualización de Datos
import seaborn as sns
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import pylab

# Métricas de Evaluación
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error

# Otras
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
from sklearn.decomposition import PCA
import math
```

1.2. Cargamos el set de datos base

In [2]:
```
#Carga inicial del Data Set
data = pd.read_excel('C:/Users/Rodrigo Mata/Desktop/Proyecto Coyol/FINAL/DATASET_FINAL_NATALIA.xls', sheet_name='BASEN3')

#Seleccionados ID como index
data =data.set_index('ID')
```

1.3. Cargamos el set de deploy

In [3]:
```
#Cargamos base de referencia
datos = pd.read_excel('C:/Users/Rodrigo Mata/Desktop/Proyecto Coyol/FINAL/DATASET_FINAL_NATALIA.xls', sheet_name='APLICAC

#Seleccionados ID como index
datos =datos.set_index('ID')
```

1.4. Limpieza final de los datos

In [4]:
```
#Eliminamos columna YEAR
data=data.drop('YEAR', axis=1)
```

2. Definición de la Variable Dependiente (Target)

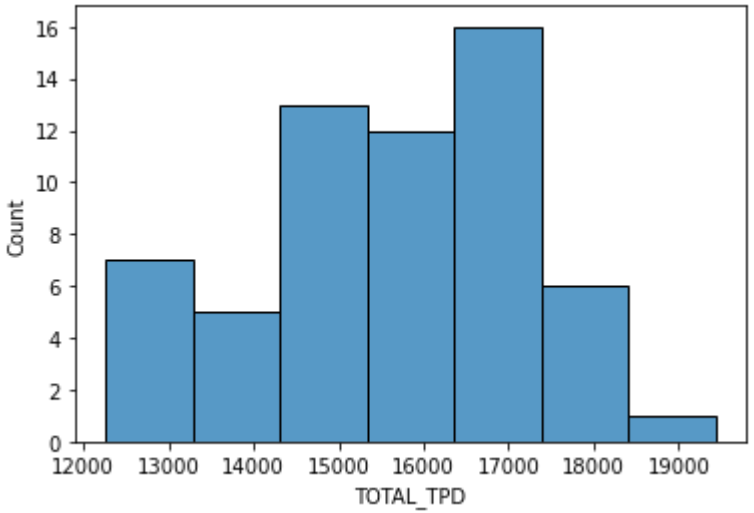
2.1. Seleccion de la Variable Independiente

In [5]:
```
#Definimos la variable independiente como el Total TPD Transformado
target = 'TOTAL_TPD'

#Generamos el vector final con la Variable Independiente
y = data[target]
```

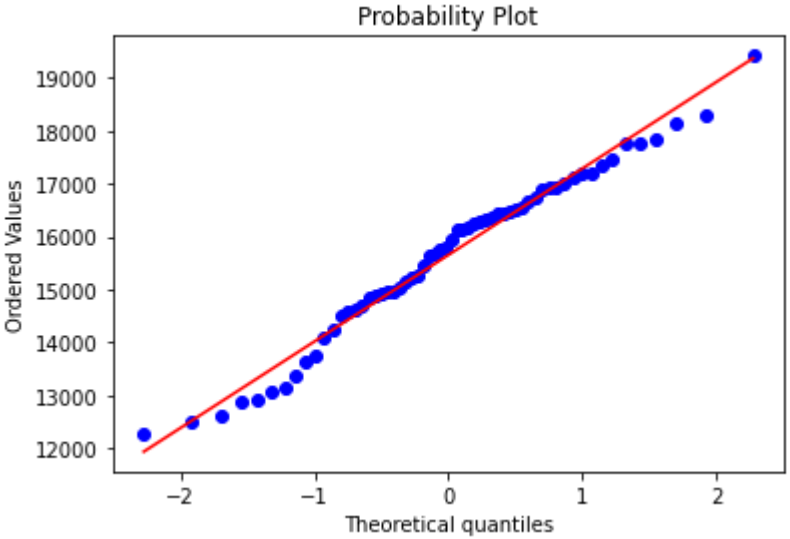
2.2. Normalidad de la Variable Independiente

In [6]:
```
#Metodo 1: Histograma
sns.histplot(data[target])
```

Out[6]: <AxesSubplot:xlabel='TOTAL_TPD', ylabel='Count'>



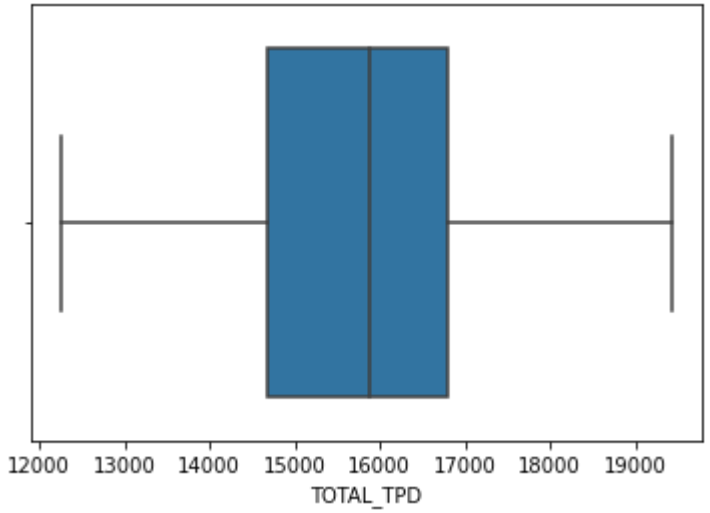
In [7]:
```
#metodo 2: Quantile-Quantile Plot
stats.probplot(data[target], dist='norm', plot=pylab)
pylab.show()
```



In [8]:
```
#Metodo 3: Box Plot
sns.boxplot(data[target])
```

```
C:\Users\Rodrigo Mata\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable a
s a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments wit
hout an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[8]: <AxesSubplot:xlabel='TOTAL_TPD'>



In [9]:

```
# Prueba de Normalidad de Variable Independiente
# Prueba Shapiro-Wilk
shapiro_test = stats.shapiro(data[target])
shapiro_test

if shapiro_test[1]> 0.05:
    print(shapiro_test)
    print('Normal')

else:
    print(shapiro_test)
    print('No Normal')
```

```
ShapiroResult(statistic=0.9780883193016052, pvalue=0.3531355857849121)
Normal
```

2.3. Filtros de Outliers en Variable Independiente

In [10]:

```
#Definimos el valor de Z para definición de valores extremos
Z = 3
```

In [11]:
```
#Tamaño de muestra inicial
print('Tamaño de Muestra Inicial:' + str(y.shape))
```

```
Tamaño de Muestra Inicial:(60,)
```

In [12]:
```
#Eliminamos valores extremos que estén fuera del rango de Z Desviaciones Estandar
upper_lim = y.mean () + y.std () * Z
lower_lim = y.mean () - y.std () * Z

y = y[(y < upper_lim) & (y > lower_lim)]
```

In [13]:
```
#Tamaño de muestra final
print('Tamaño de Muestra luego de eliminar oultiers:'+ str(y.shape))
```

```
Tamaño de Muestra luego de eliminar oultiers:(60,)
```

3. Definición de Variables Independientes (Features)

3.1. Selección de Variables Independientes

In [14]:
```
#Lista de Variables Independientes a revisar
lista_features = ['DIESEL', 'GAS_SUPER', 'GAS_REGULAR', #variables combustible
          'IPC', 'IMAE', 'TIPO_CAMBIO',  'IMPORTACION_VEHICULOS', #varibales economicas
         'GARITA_VIVIENDA_ACUM', 'SAN_JOSE_VIVIENDA_ACUM', 'SAN_ANTONIO_VIVIENDA_ACUM', 'TURRUCARES_VIVIENDA_ACUM',
          'SAN_JOSE_CONDOMINIO_ACUM', 'SAN_ANTONIO_CONDOMINIO_ACUM', 'GARITA_CONDOMINIO_ACUM', #variables vivienda
         'GARITA_INDUSTRIAL_ACUM', 'SAN_ANTONIO_INDUSTRIAL_ACUM', 'TURRUCARES_INDUSTRIAL_ACUM', #variables industriales
         'GARITA_COMERCIAL_ACUM', 'SAN_ANTONIO_COMERCIAL_ACUM', 'SAN_JOSE_COMERCIAL_ACUM',
          'TURRUCARES_COMERCIAL_ACUM', #variables comerciales
          'GARITA_SERVICIOS_ACUM', 'SAN_ANTONIO_SERVICIOS_ACUM', 'TURRUCARES_SERVICIOS_ACUM',
          'SAN JOSE_SERVICIOS_ACUM', #variables servicios
         'TPD_RN1_TOTAL' #varibales rutas primarias
        ]
```

In [15]:
```
#Definimos data set de variables independientes
X = data[lista_features]
print('Variables Independientes seleccionadas:'+str(X.columns))
```

```
Variables Independientes seleccionadas:Index(['DIESEL', 'GAS_SUPER', 'GAS_REGULAR', 'IPC', 'IMAE', 'TIPO_CAMBIO',
       'IMPORTACION_VEHICULOS', 'GARITA_VIVIENDA_ACUM',
```

```
       'SAN_JOSE_VIVIENDA_ACUM', 'SAN_ANTONIO_VIVIENDA_ACUM',
       'TURRUCARES_VIVIENDA_ACUM', 'SAN_JOSE_CONDOMINIO_ACUM',
       'SAN_ANTONIO_CONDOMINIO_ACUM', 'GARITA_CONDOMINIO_ACUM',
       'GARITA_INDUSTRIAL_ACUM', 'SAN_ANTONIO_INDUSTRIAL_ACUM',
       'TURRUCARES_INDUSTRIAL_ACUM', 'GARITA_COMERCIAL_ACUM',
       'SAN_ANTONIO_COMERCIAL_ACUM', 'SAN_JOSE_COMERCIAL_ACUM',
       'TURRUCARES_COMERCIAL_ACUM', 'GARITA_SERVICIOS_ACUM',
       'SAN_ANTONIO_SERVICIOS_ACUM', 'TURRUCARES_SERVICIOS_ACUM',
       'SAN JOSE_SERVICIOS_ACUM', 'TPD_RN1_TOTAL'],
      dtype='object')
```

In [16]:
```
#Tamaño de muestra inicial
print('Tamaño Muestra Inicial:'+str(X.shape))
```

```
Tamaño Muestra Inicial:(60, 26)
```

3.2. Revisión de Outliers en Variables Independientes

In [17]:
```
#Se crea bucle para revision de outliers en Variables Independientes
for i in lista_features:
    upper_lim = X[i].mean () + X[i].std () * Z
    lower_lim = X[i].mean () - X[i].std () * Z

    X = X[(X[i] < upper_lim) & (X[i] > lower_lim)]
```

In [18]:
```
#Tamaño de Muestra Final
print('Tamaño Muestra Final:'+str(X.shape))
```

```
Tamaño Muestra Final:(58, 26)
```

4. Ajustes finales del set de datos para modelo

4.1. Data Set Final del Modelo

In [19]:
```
# Se integran ambas Variables (Dependiente e Independientes) para generar el data set final del modelo
df_final = pd.merge(y, X, how='inner', on='ID')
```

In [20]:
```
#Revisión de tamaño final de muestra
print('Tamaño Muestra Final:'+str(df_final.shape))
```

```
df_final.shape
```

```
Tamaño Muestra Final:(58, 27)
```

Out[20]:
```
(58, 27)
```

In [21]:
```
#Definimos Data Final Variable Independiente
y =df_final[target]
```

In [22]:
```
#Definimos Data Final Variable Dependiente
X = df_final[lista_features]
```

In [23]:
```
#Split data set en set de entrenamiento (calibración del modelo) y set de prueba (evaluación del modelo)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.10, random_state=0)
```

4.2. Metodologia para la reduccón de dimensiones - Principal Component Analysis

In [24]:
```
#Definimos funcion para serapar features por categorías

def features(df):
    df_combustible = df[['DIESEL', 'GAS_SUPER', 'GAS_REGULAR']]
    df_economia = df[['IPC', 'IMAE', 'TIPO_CAMBIO']]
    df_vivienda = df[['GARITA_VIVIENDA_ACUM', 'SAN_JOSE_VIVIENDA_ACUM', 'SAN_ANTONIO_VIVIENDA_ACUM', 'TURRUCARES_VIVIENDA
    df_industrial = df[['GARITA_INDUSTRIAL_ACUM', 'SAN_ANTONIO_INDUSTRIAL_ACUM', 'TURRUCARES_INDUSTRIAL_ACUM']]
    df_comercial = df[['GARITA_COMERCIAL_ACUM', 'SAN_ANTONIO_COMERCIAL_ACUM', 'SAN_JOSE_COMERCIAL_ACUM','TURRUCARES_COMER
    df_servicio = df[['GARITA_SERVICIOS_ACUM', 'SAN_ANTONIO_SERVICIOS_ACUM', 'TURRUCARES_SERVICIOS_ACUM', 'SAN JOSE_SERVI
    df_lista = [df_combustible, df_economia, df_vivienda, df_industrial, df_comercial, df_servicio]

    return df_lista
```

In [25]:
```
#Definimos grupos de variables y numero de PC's
var = ['Combustible', 'Economia', 'Vivienda', 'Industrial', 'Comercial', 'Servicio']
n = [1, 2, 3, 1, 2, 2]
```

In [26]:
```
#Aplicamos la función para separar features por categorías
x_subset_train = features(X_train)
x_subset_test = features(X_test)
x_subset_deploy = features(datos)
```

In [27]:

```
# PCA_COMBUSTIBLE
l=0

#Standarizamos la data
scaler = StandardScaler()

# Fit on training set only.
scaler.fit(x_subset_train[l])

# Apply transform to both the training set and the test set.
x_subset_train[l] = scaler.transform(x_subset_train[l])
x_subset_test[l] = scaler.transform(x_subset_test[l])
x_subset_deploy[l] = scaler.transform(x_subset_deploy[l])


pca = PCA(n_components = n[l])
pca.fit(x_subset_train[l])

x_subset_train[l] = pca.transform(x_subset_train[l])
x_subset_test[l]  = pca.transform(x_subset_test[l])
x_subset_deploy[l]  = pca.transform(x_subset_deploy[l])

explained_variance = pca.explained_variance_ratio_

tx_train = pd.DataFrame(x_subset_train[l])
tx_train['ID'] = X_train.index
tx_train.columns = ['PC_Combustible_1', 'ID']

tx_test = pd.DataFrame(x_subset_test[l])
tx_test['ID'] = X_test.index
tx_test.columns = ['PC_Combustible_1', 'ID']

tx_deploy = pd.DataFrame(x_subset_deploy[l])
tx_deploy['ID'] = datos.index
tx_deploy.columns = ['PC_Combustible_1', 'ID']
```

In [28]:

```
# PCA_ECONOMIA
l=1

#Standarizamos la data
scaler = StandardScaler()
```

```
# Fit on training set only.
scaler.fit(x_subset_train[l])

# Apply transform to both the training set and the test set.
x_subset_train[l] = scaler.transform(x_subset_train[l])
x_subset_test[l] = scaler.transform(x_subset_test[l])
x_subset_deploy[l] = scaler.transform(x_subset_deploy[l])

pca = PCA(n_components = n[l])
pca.fit(x_subset_train[l])

x_subset_train[l] = pca.transform(x_subset_train[l])
x_subset_test[l]  = pca.transform(x_subset_test[l])
x_subset_deploy[l]  = pca.transform(x_subset_deploy[l])

explained_variance = pca.explained_variance_ratio_

tx_train_temp = pd.DataFrame(x_subset_train[l])
tx_train_temp['ID'] = X_train.index
tx_train_temp.columns = ['PC_Economia_1', 'PC_Economia_2', 'ID']
tx_train = pd.merge(tx_train, tx_train_temp, how='left', on='ID')

tx_test_temp = pd.DataFrame(x_subset_test[l])
tx_test_temp['ID'] = X_test.index
tx_test_temp.columns = ['PC_Economia_1', 'PC_Economia_2', 'ID']
tx_test = pd.merge(tx_test, tx_test_temp, how='left', on='ID')

tx_deploy_temp = pd.DataFrame(x_subset_deploy[l])
tx_deploy_temp['ID'] = datos.index
tx_deploy_temp.columns = ['PC_Economia_1', 'PC_Economia_2', 'ID']
tx_deploy = pd.merge(tx_deploy, tx_deploy_temp, how='left', on='ID')
```

In [29]:

```
# PCA_VIVIENDA
l=2

#Standarizamos la data
scaler = StandardScaler()

# Fit on training set only.
scaler.fit(x_subset_train[l])

# Apply transform to both the training set and the test set.
x_subset_train[l] = scaler.transform(x_subset_train[l])
x_subset_test[l] = scaler.transform(x_subset_test[l])
```

```
x_subset_deploy[l] = scaler.transform(x_subset_deploy[l])

pca = PCA(n_components = n[l])
pca.fit(x_subset_train[l])

x_subset_train[l] = pca.transform(x_subset_train[l])
x_subset_test[l]  = pca.transform(x_subset_test[l])
x_subset_deploy[l]  = pca.transform(x_subset_deploy[l])


explained_variance = pca.explained_variance_ratio_

tx_train_temp = pd.DataFrame(x_subset_train[l])
tx_train_temp['ID'] = X_train.index
tx_train_temp.columns = ['PC_vivienda_1', 'PC_Vivienda_2', 'PC_Vivienda_3', 'ID']
tx_train = pd.merge(tx_train, tx_train_temp, how='left', on='ID')

tx_test_temp = pd.DataFrame(x_subset_test[l])
tx_test_temp['ID'] = X_test.index
tx_test_temp.columns = ['PC_vivienda_1', 'PC_Vivienda_2', 'PC_Vivienda_3', 'ID']
tx_test = pd.merge(tx_test, tx_test_temp, how='left', on='ID')

tx_deploy_temp = pd.DataFrame(x_subset_deploy[l])
tx_deploy_temp['ID'] = datos.index
tx_deploy_temp.columns = ['PC_vivienda_1', 'PC_Vivienda_2', 'PC_Vivienda_3', 'ID']
tx_deploy = pd.merge(tx_deploy, tx_deploy_temp, how='left', on='ID')
```

In [30]:

```
# PCA_INDUSTRIAL
l=3

#Standarizamos la data
scaler = StandardScaler()

# Fit on training set only.
scaler.fit(x_subset_train[l])

# Apply transform to both the training set and the test set.
x_subset_train[l] = scaler.transform(x_subset_train[l])
x_subset_test[l] = scaler.transform(x_subset_test[l])
x_subset_deploy[l] = scaler.transform(x_subset_deploy[l])

pca = PCA(n_components = n[l])
pca.fit(x_subset_train[l])
```

```
x_subset_train[l] = pca.transform(x_subset_train[l])
x_subset_test[l]  = pca.transform(x_subset_test[l])
x_subset_deploy[l]  = pca.transform(x_subset_deploy[l])


explained_variance = pca.explained_variance_ratio_

tx_train_temp = pd.DataFrame(x_subset_train[l])
tx_train_temp['ID'] = X_train.index
tx_train_temp.columns = ['PC_industrial_1', 'ID']
tx_train = pd.merge(tx_train, tx_train_temp, how='left', on='ID')

tx_test_temp = pd.DataFrame(x_subset_test[l])
tx_test_temp['ID'] = X_test.index
tx_test_temp.columns = ['PC_industrial_1', 'ID']
tx_test = pd.merge(tx_test, tx_test_temp, how='left', on='ID')

tx_deploy_temp = pd.DataFrame(x_subset_deploy[l])
tx_deploy_temp['ID'] = datos.index
tx_deploy_temp.columns = ['PC_industrial_1', 'ID']
tx_deploy = pd.merge(tx_deploy, tx_deploy_temp, how='left', on='ID')
```

In [31]:

```
# PCA_COMERCIAL
l=4

#Standarizamos la data
scaler = StandardScaler()

# Fit on training set only.
scaler.fit(x_subset_train[l])

# Apply transform to both the training set and the test set.
x_subset_train[l] = scaler.transform(x_subset_train[l])
x_subset_test[l] = scaler.transform(x_subset_test[l])
x_subset_deploy[l] = scaler.transform(x_subset_deploy[l])

pca = PCA(n_components = n[l])
pca.fit(x_subset_train[l])

x_subset_train[l] = pca.transform(x_subset_train[l])
x_subset_test[l]  = pca.transform(x_subset_test[l])
x_subset_deploy[l]  = pca.transform(x_subset_deploy[l])
```

```
explained_variance = pca.explained_variance_ratio_

tx_train_temp = pd.DataFrame(x_subset_train[l])
tx_train_temp['ID'] = X_train.index
tx_train_temp.columns = ['PC_comercial_1', 'PC_comercial_2', 'ID']
tx_train = pd.merge(tx_train, tx_train_temp, how='left', on='ID')

tx_test_temp = pd.DataFrame(x_subset_test[l])
tx_test_temp['ID'] = X_test.index
tx_test_temp.columns = ['PC_comercial_1', 'PC_comercial_2', 'ID']
tx_test = pd.merge(tx_test, tx_test_temp, how='left', on='ID')

tx_deploy_temp = pd.DataFrame(x_subset_deploy[l])
tx_deploy_temp['ID'] = datos.index
tx_deploy_temp.columns = ['PC_comercial_1', 'PC_comercial_2', 'ID']
tx_deploy = pd.merge(tx_deploy, tx_deploy_temp, how='left', on='ID')
```

In [32]:

```
# PCA_SERVICIO
l=5

#Standarizamos la data
scaler = StandardScaler()

# Fit on training set only.
scaler.fit(x_subset_train[l])

# Apply transform to both the training set and the test set.
x_subset_train[l] = scaler.transform(x_subset_train[l])
x_subset_test[l] = scaler.transform(x_subset_test[l])
x_subset_deploy[l] = scaler.transform(x_subset_deploy[l])

pca = PCA(n_components = n[l])
pca.fit(x_subset_train[l])

x_subset_train[l] = pca.transform(x_subset_train[l])
x_subset_test[l]  = pca.transform(x_subset_test[l])
x_subset_deploy[l]  = pca.transform(x_subset_deploy[l])


explained_variance = pca.explained_variance_ratio_

tx_train_temp = pd.DataFrame(x_subset_train[l])
tx_train_temp['ID'] = X_train.index
tx_train_temp.columns = ['PC_servicio_1', 'PC_servicio_2', 'ID']
```

```
tx_train = pd.merge(tx_train, tx_train_temp, how='left', on='ID')

tx_test_temp = pd.DataFrame(x_subset_test[l])
tx_test_temp['ID'] = X_test.index
tx_test_temp.columns = ['PC_servicio_1', 'PC_servicio_2', 'ID']
tx_test = pd.merge(tx_test, tx_test_temp, how='left', on='ID')

tx_deploy_temp = pd.DataFrame(x_subset_deploy[l])
tx_deploy_temp['ID'] = datos.index
tx_deploy_temp.columns = ['PC_servicio_1', 'PC_servicio_2', 'ID']
tx_deploy = pd.merge(tx_deploy, tx_deploy_temp, how='left', on='ID')
```

5. Feauture Selection

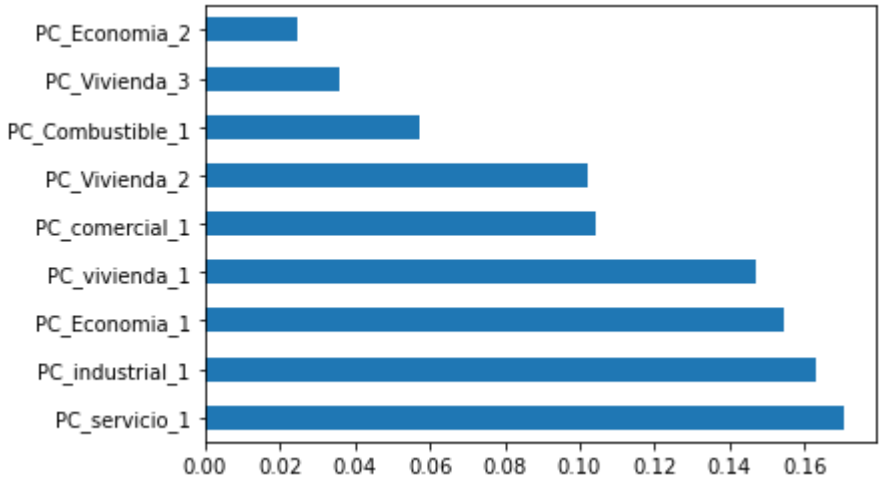
In [33]:
```
#Definition of predictors number to be used.   Aprox 10% of total instances.
n=9
```

In [34]:
```
X_train = tx_train.copy().set_index('ID')
X_test = tx_test.copy().set_index('ID')
```

In [35]:
```
from sklearn.ensemble import ExtraTreesRegressor

model = ExtraTreesRegressor()
model.fit(X_train,y_train)
print(model.feature_importances_) #use inbuilt class feature_importances of tree based classifiers
```

```
[0.05720374 0.1545225  0.02456126 0.14720246 0.10228431 0.03596992
 0.16284476 0.1039991  0.02223396 0.17031926 0.01885873]
```

In [36]:
```
#plot graph of feature importances for better visualization
import matplotlib.pyplot as plt
feat_importances = pd.Series(model.feature_importances_, index=X_train.columns)
feat_importances.nlargest(n).plot(kind='barh')
plt.show()
```



In [37]:
```
#Creamos Data Set final de Features
features = pd.Series(feat_importances.nlargest(n).index)

X_train = X_train[features]
X_test  = X_test[features]
```

5. Entrenamiento de los Modelos

- Regresión Lineal Simple
- Arbol de Regresión
- Regresión Ridge

5.1. Modelo de Regresión Lineal Simple

In [38]:
```
# Tunning de Hiperparámetros Regresión Lineal Simple
params_regresion = {
    'fit_intercept' : [True, False],
    'normalize' : [True,False],
    'n_jobs' :[1,-1]
}

regressor = linear_model.LinearRegression()

l_regressor = GridSearchCV(regressor, params_regresion, scoring='r2', cv=5, n_jobs=-1)
```

```
l_regressor.fit(X_train, y_train)
print(f'fit_intercept: {l_regressor.best_params_["fit_intercept"]}')
print(f'normalize: {l_regressor.best_params_["normalize"]}')
print(f'n_jobs: {l_regressor.best_params_["n_jobs"]}')
print(f'Best score: {l_regressor.best_score_}')
```

```
fit_intercept: True
normalize: True
n_jobs: 1
Best score: 0.6894569659910581
```

In [39]:

```
# Aplicación del Modelo de Regresión Lineal Simple selecionado
regresion_model = linear_model.LinearRegression(fit_intercept=l_regressor.best_params_["fit_intercept"],
                    normalize=l_regressor.best_params_["normalize"], n_jobs=l_regressor.best_params_["n_jobs"])
regresion_model.fit(X_train, y_train)
y_train_pred = regresion_model.predict(X_train)
y_pred = regresion_model.predict(X_test)
print('Train r2 score: ', r2_score(y_train_pred, y_train))
print('Test r2 score: ', r2_score(y_test, y_pred))
train_rmse = np.sqrt(mean_squared_error(y_train_pred, y_train))
test_rmse = np.sqrt(mean_squared_error(y_test, y_pred))
print(f'Train RMSE: {train_rmse:.4f}')
print(f'Test RMSE: {test_rmse:.4f}')
```

```
Train r2 score:  0.7358408522531141
Test r2 score:  0.8417760443963841
Train RMSE: 694.7038
Test RMSE: 574.2242
```

In [40]:

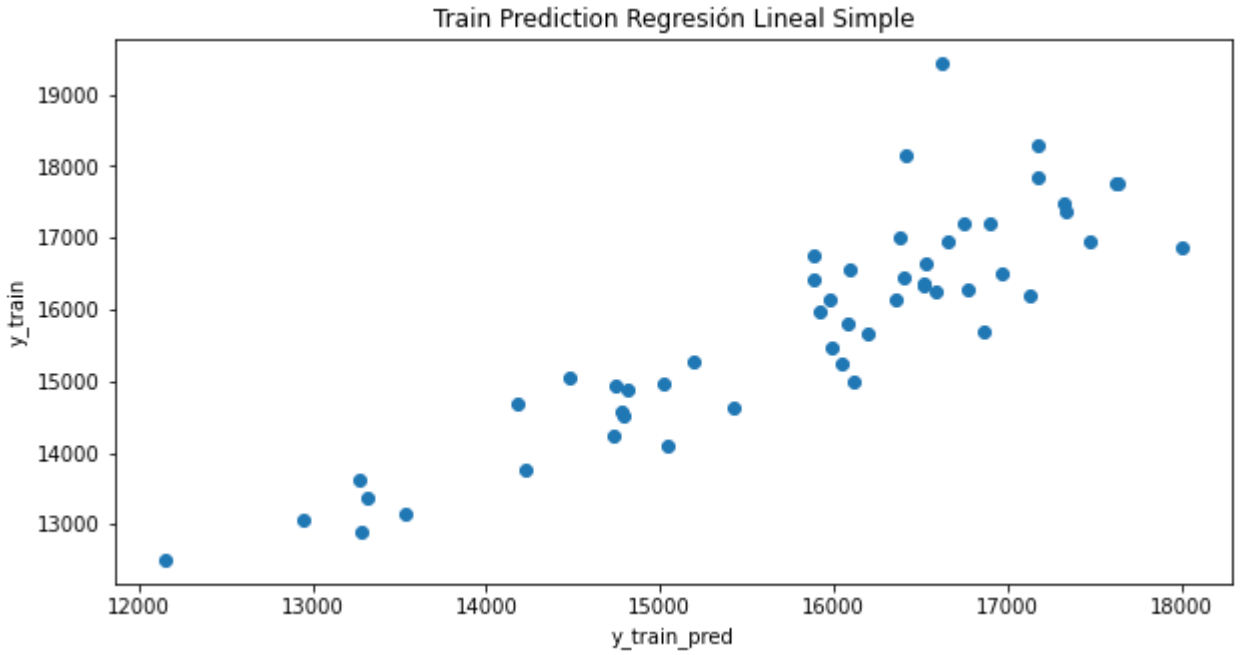
```
#TABLA DE REGRESION LINEAL SIMPLE SIN AJUSTAR
from regressors import stats

stats.summary(regresion_model, X_train, y_train)
```

```
Residuals:
       Min        1Q  Median      3Q        Max
-2799.0244 -318.3356 -45.728  471.07  1164.5898


Coefficients:
               Estimate  Std. Error   t value   p value
_intercept  15812.858910   96.338089  164.1392  0.000000
x1           -566.803777  346.183470   -1.6373  0.107726
x2          -1797.734500  463.216758   -3.8810  0.000300
x3           -924.426070  257.878962   -3.5847  0.000754
```

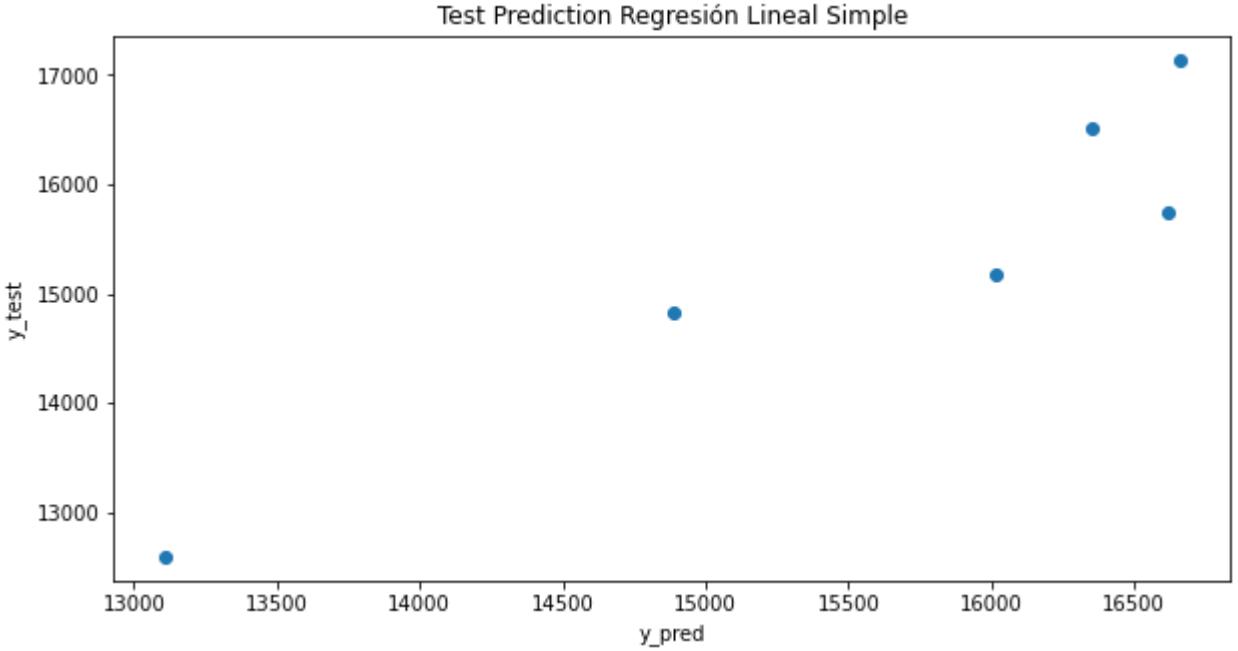
```
x4          -1024.845426  486.308594   -2.1074  0.040020
x5           -404.611228  357.128355   -1.1330  0.262530
x6             -8.274197  214.113863   -0.0386  0.969325
x7             17.075046   85.253620    0.2003  0.842054
x8            626.322537  372.932932    1.6795  0.099181
x9           -314.066604  228.641746   -1.3736  0.175568
---
R-squared:  0.79104,    Adjusted R-squared:  0.74626
F-statistic: 17.67 on 9 features
```

In [41]:

```
#Visualización Modelo de Regresión Lineal Simple

#Train
xv = y_train_pred
yv = y_train

plt.figure(figsize = (10,5))
plt.scatter(xv, yv, alpha=1)
plt.title('Train Prediction Regresión Lineal Simple')
plt.xlabel('y_train_pred')
plt.ylabel('y_train')
graf1 = plt.show()
print(graf1)

#Test
xv = y_pred
yv = y_test

plt.figure(figsize = (10,5))
plt.scatter(xv, yv, alpha=1)
plt.title('Test Prediction Regresión Lineal Simple')
plt.xlabel('y_pred')
plt.ylabel('y_test')
graf2 = plt.show()
print(graf2)
```


Train Prediction Regresión Lineal Simple
y_train
y_train_pred
19000
18000
17000
16000
15000
14000
13000
12000
13000
14000
15000
16000
17000
18000

None


Test Prediction Regresión Lineal Simple
y_test
y_pred
17000
16000
15000
14000
13000
13000
13500
14000
14500
15000
15500
16000
16500

None

5.2. Modelo Arbol de Regresión

In [42]:
```
# Tunning de Hiperparámetros Arbol de Regresión

params_tree = {
    'max_depth' : [2, 3, 4, 5],
    'min_samples_split' : [ 2, 3, 4],
    'min_samples_leaf' : [1, 2],
}

regressor = DecisionTreeRegressor(random_state = 0)

tree_regressor = GridSearchCV(regressor, params_tree, scoring='r2', cv=5, n_jobs=-1)
tree_regressor.fit(X_train, y_train)
print(f'Optimal max_depth: {tree_regressor.best_params_["max_depth"]:.2f}')
print(f'Optimal min_samples_split: {tree_regressor.best_params_["min_samples_split"]}')
print(f'Optimal min_samples_leaf: {tree_regressor.best_params_["min_samples_leaf"]}')
print(f'Best score: {tree_regressor.best_score_}')
```

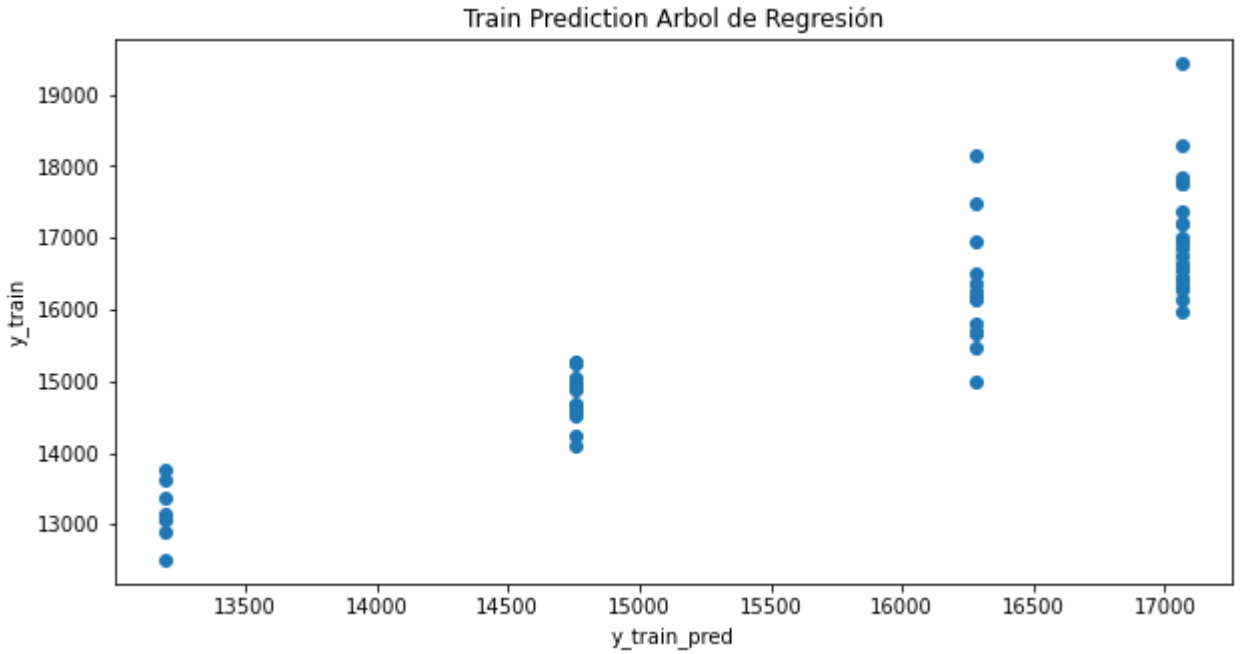
```
Optimal max_depth: 2.00
Optimal min_samples_split: 2
Optimal min_samples_leaf: 1
Best score: 0.6158117372984693
```

In [43]:
```
# Aplicación del Modelo de Arbol de Regresión selecionado

regressor = DecisionTreeRegressor(max_depth=tree_regressor.best_params_["max_depth"], min_samples_split=tree_regressor.be
                    min_samples_leaf=tree_regressor.best_params_["min_samples_leaf"])
regressor.fit(X_train, y_train)
y_train_pred = regressor.predict(X_train)
y_pred = regressor.predict(X_test)
print('Train r2 score: ', r2_score(y_train_pred, y_train))
print('Test r2 score: ', r2_score(y_test, y_pred))
train_rmse_tree = np.sqrt(mean_squared_error(y_train_pred, y_train))
test_rmse_tree = np.sqrt(mean_squared_error(y_test, y_pred))
print(f'Train RMSE: {train_rmse_tree:.4f}')
print(f'Test RMSE: {test_rmse_tree:.4f}')
```

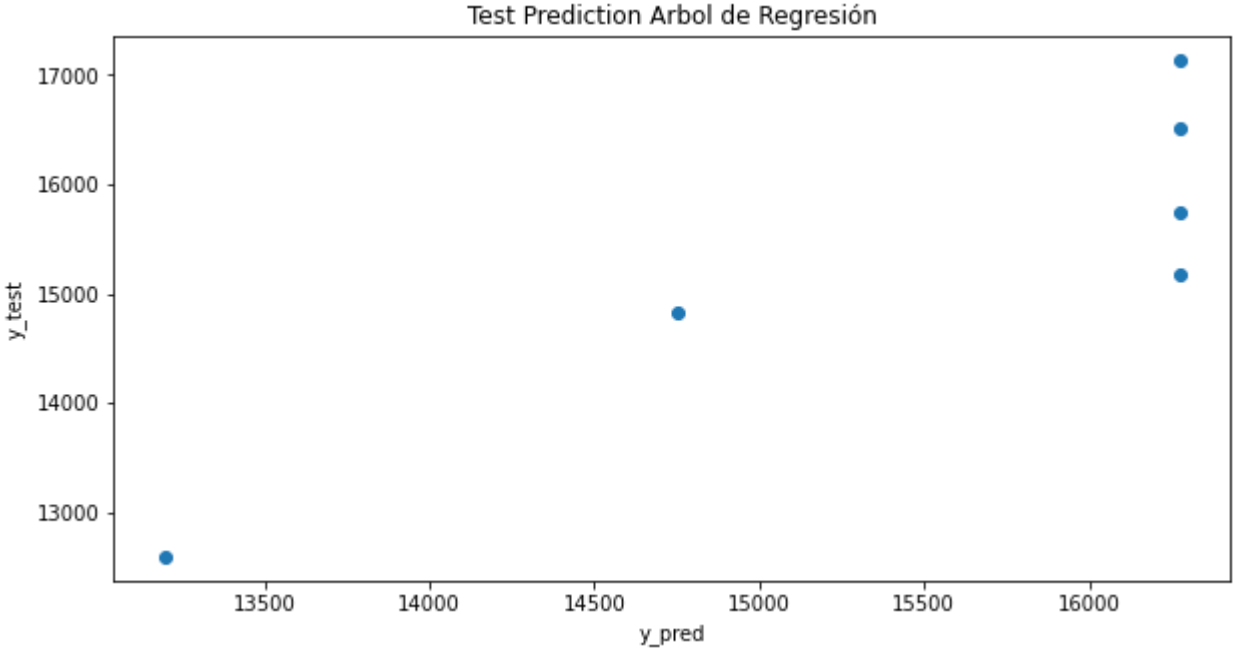
```
Train r2 score:  0.7399438574515272
Test r2 score:  0.7884071512670425
Train RMSE: 690.4089
Test RMSE: 664.0424
```

In [44]:
```
#Visualización Modelo Arbol de Regresión
```

```
#Train
xv = y_train_pred
yv = y_train

plt.figure(figsize = (10,5))
plt.scatter(xv, yv, alpha=1)
plt.title('Train Prediction Arbol de Regresión')
plt.xlabel('y_train_pred')
plt.ylabel('y_train')
graf1 = plt.show()
print(graf1)

#Test
xv = y_pred
yv = y_test

plt.figure(figsize = (10,5))
plt.scatter(xv, yv, alpha=1)
plt.title('Test Prediction Arbol de Regresión')
plt.xlabel('y_pred')
plt.ylabel('y_test')
graf2 = plt.show()
print(graf2)
```


Train Prediction Arbol de Regresión
y_train
y_train_pred
19000
18000
17000
16000
15000
14000
13000
13500
14000
14500
15000
15500
16000
16500
17000

None


Test Prediction Arbol de Regresión
y_test
y_pred
17000
16000
15000
14000
13000
13500
14000
14500
15000
15500
16000

None

5.3. Modelo Regresión Ridge

In [45]:
```
# Tunning de Hiperparámetros Regresión Ridge
from sklearn.linear_model import Ridge

params_ridge = {
    'alpha' : [0.1, 0.4, 0.5, .7, 0.75, 0.8, 0.85,.9, .99, 1, 5, 10, 20, 30],
    'fit_intercept' : [True, False],
    'normalize' : [True,False],
    'solver' : ['svd', 'cholesky', 'lsqr', 'sparse_cg', 'sag', 'saga']
}

ridge_model = Ridge()
ridge_regressor = GridSearchCV(ridge_model, params_ridge, scoring='r2', cv=5, n_jobs=-1)
ridge_regressor.fit(X_train, y_train)
print(f'Optimal alpha: {ridge_regressor.best_params_["alpha"]:.2f}')
print(f'Optimal fit_intercept: {ridge_regressor.best_params_["fit_intercept"]}')
print(f'Optimal normalize: {ridge_regressor.best_params_["normalize"]}')
print(f'Optimal solver: {ridge_regressor.best_params_["solver"]}')
print(f'Best score: {ridge_regressor.best_score_}')
```

```
Optimal alpha: 1.00
Optimal fit_intercept: True
Optimal normalize: False
Optimal solver: sparse_cg
Best score: 0.7124349775216232
```

In [46]:
```
# Aplicación del Modelo Regresión Ridge

ridge_model = Ridge(alpha=ridge_regressor.best_params_["alpha"], fit_intercept=ridge_regressor.best_params_["fit_intercep
                    normalize=ridge_regressor.best_params_["normalize"], solver=ridge_regressor.best_params_["solver"])
ridge_model.fit(X_train, y_train)
y_train_pred = ridge_model.predict(X_train)
y_pred = ridge_model.predict(X_test)
print('Train r2 score: ', r2_score(y_train_pred, y_train))
print('Test r2 score: ', r2_score(y_test, y_pred))
train_rmse = np.sqrt(mean_squared_error(y_train_pred, y_train))
test_rmse = np.sqrt(mean_squared_error(y_test, y_pred))
print(f'Train RMSE: {train_rmse:.4f}')
print(f'Test RMSE: {test_rmse:.4f}')
```

```
Train r2 score:  0.7045441213695998
Test r2 score:  0.8281702630405907
```

```
Train RMSE: 710.3899
Test RMSE: 598.4040
```

In [47]:

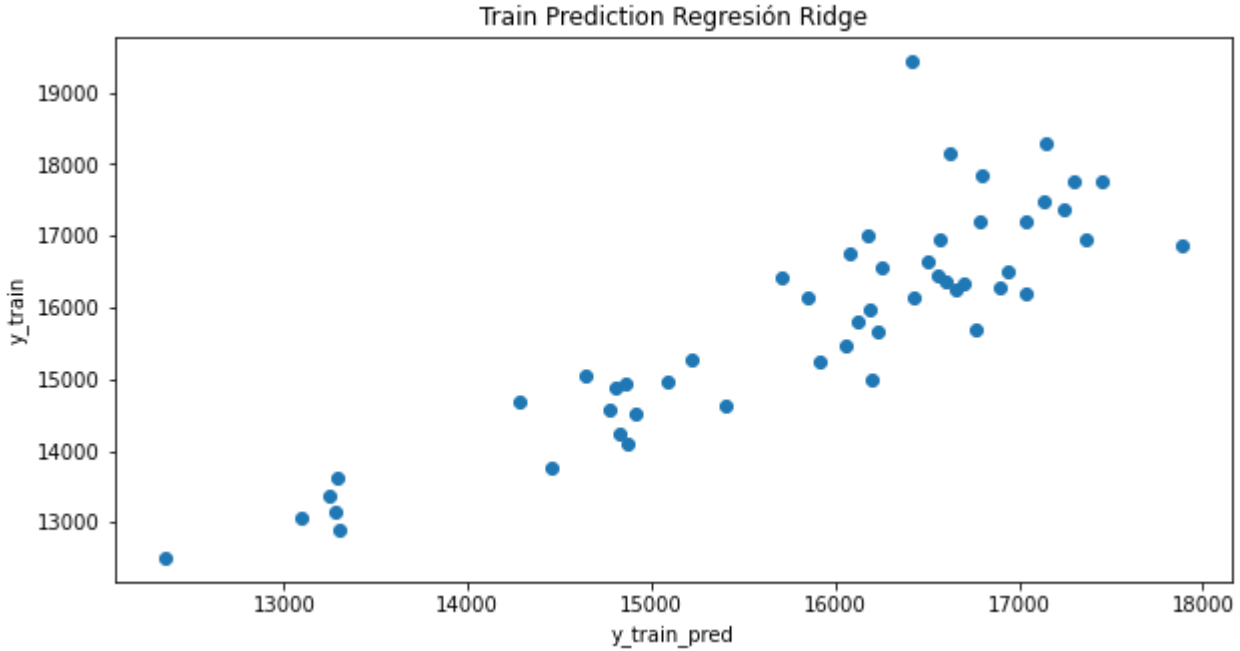
```
#TABLA DE REGRESION RIDGE SIN AJUSTAR
from regressors import stats

stats.summary(ridge_model, X_train, y_train)
```

```
Residuals:
       Min        1Q   Median       3Q        Max
-3007.3817 -341.9758  72.1045  423.428  1214.7774


Coefficients:
                Estimate  Std. Error   t value   p value
_intercept  15812.858910   98.513357  160.5149  0.000000
x1           -483.826871  354.000128   -1.3667  0.177701
x2          -1014.732067  473.675972   -2.1422  0.036970
x3           -911.813629  263.701746   -3.4577  0.001108
x4           -394.039021  497.289211   -0.7924  0.431813
x5           -214.152139  365.192143   -0.5864  0.560187
x6           -335.274333  218.948453   -1.5313  0.131877
x7            -37.622452   87.178606   -0.4316  0.667883
x8            291.408556  381.353581    0.7641  0.448303
x9           -155.059058  233.804368   -0.6632  0.510189
---
R-squared:  0.78150,    Adjusted R-squared:  0.73467
F-statistic: 16.69 on 9 features
```

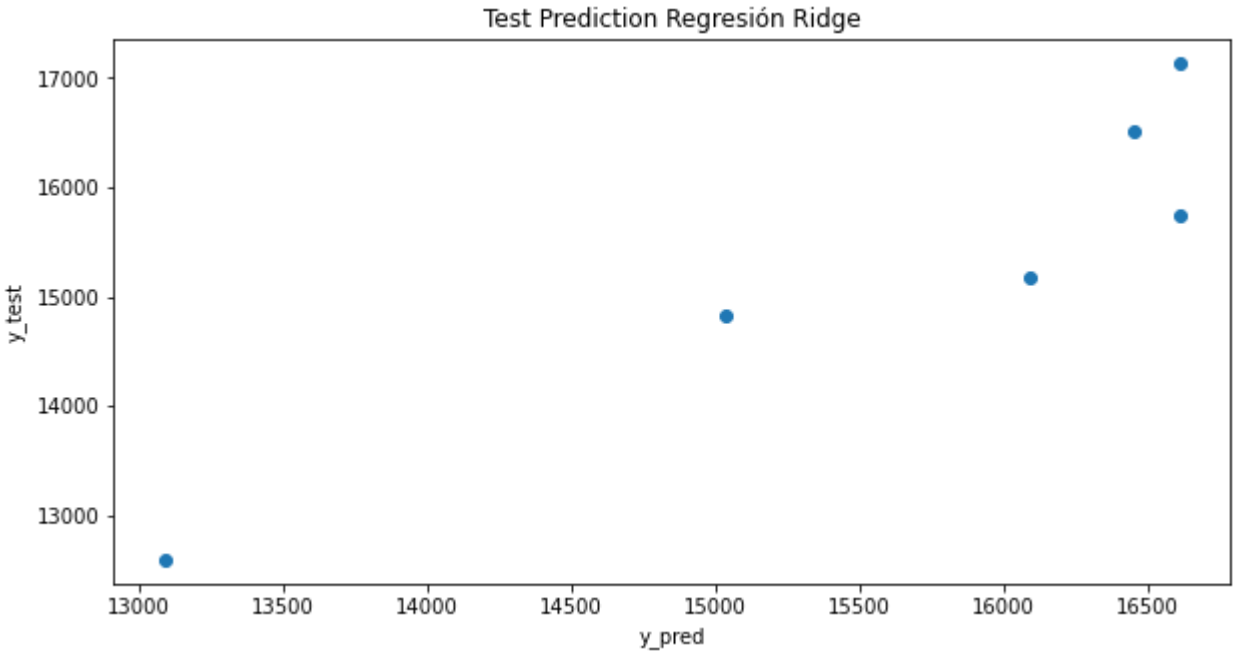
In [48]:

```
#Visualización Modelo Regresión Ridge

#Train
xv = y_train_pred
yv = y_train

plt.figure(figsize = (10,5))
plt.scatter(xv, yv, alpha=1)
plt.title('Train Prediction Regresión Ridge')
plt.xlabel('y_train_pred')
plt.ylabel('y_train')
graf1 = plt.show()
print(graf1)

#Test
xv = y_pred
yv = y_test
```

```
plt.figure(figsize = (10,5))
plt.scatter(xv, yv, alpha=1)
plt.title('Test Prediction Regresión Ridge')
plt.xlabel('y_pred')
plt.ylabel('y_test')
graf2 = plt.show()
print(graf2)
```



None



None

5.4. Coeficientes Regresiones

In [49]:
```
features = features.to_list()
```

In [50]:
```
regresion_model.coef_
```

Out[50]:
```
array([ -566.80377708, -1797.73450029,  -924.42607041, -1024.84542565,
        -404.61122793,    -8.27419691,    17.07504598,   626.32253696,
        -314.06660375])
```

In [51]:
```
ridge_model.coef_
```

Out[51]:
```
array([ -483.82687114, -1014.73206688,  -911.81362918,  -394.03902089,
        -214.15213895,  -335.2743335 ,   -37.6224522 ,   291.40855614,
        -155.05905764])
```

In [52]:
```
df_coeff = pd.DataFrame([features, regresion_model.coef_, ridge_model.coef_])
df_coeff = df_coeff.T
```

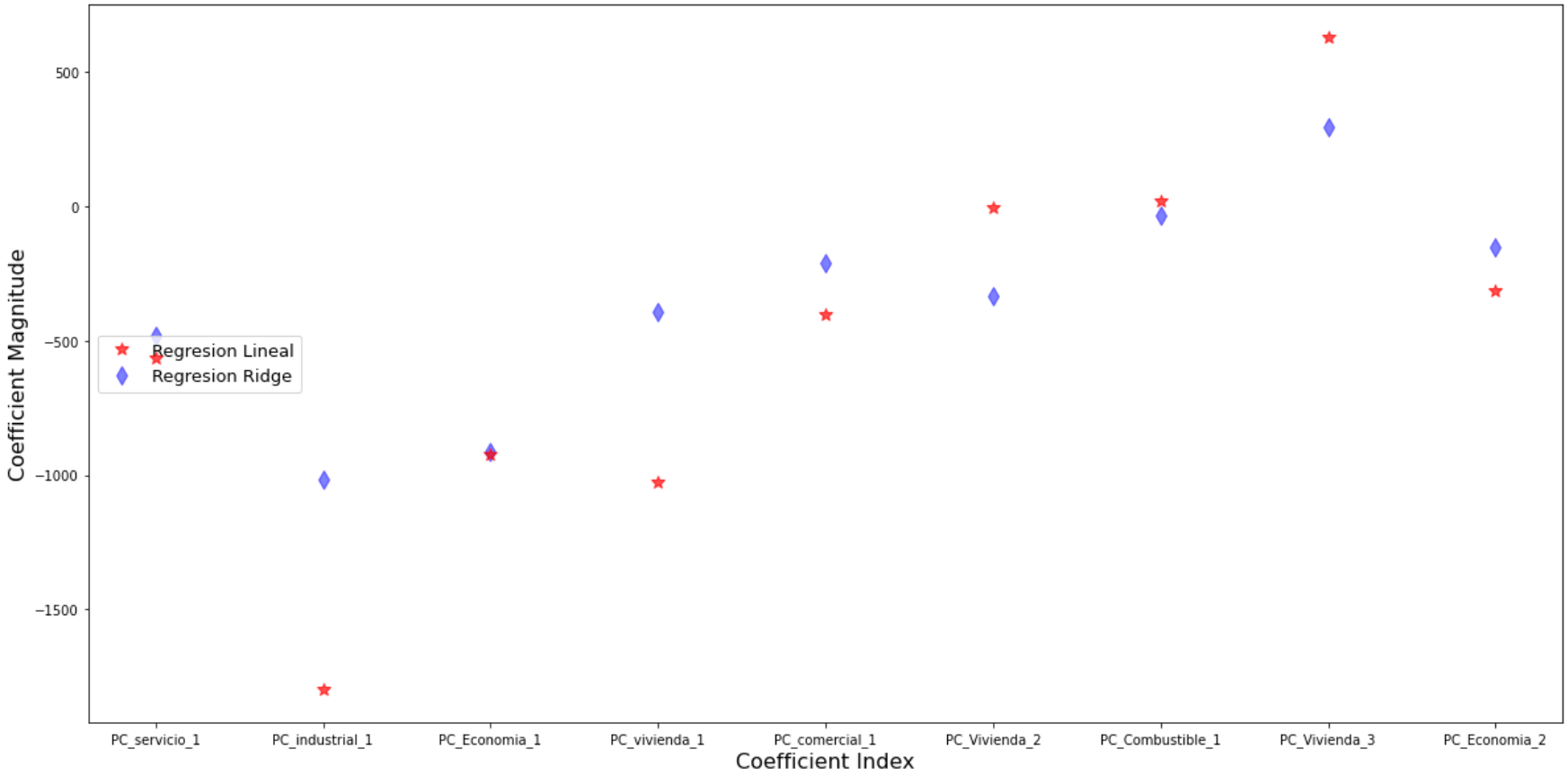
```
df_coeff.columns = ['Feature','Regresion_Lineal','Regresión_Ridge']
df_coeff
```

Out[52]:

	Feature	Regresion_Lineal	Regresión_Ridge
0	PC_servicio_1	-566.804	-483.827
1	PC_industrial_1	-1797.73	-1014.73
2	PC_Economia_1	-924.426	-911.814
3	PC_vivienda_1	-1024.85	-394.039
4	PC_comercial_1	-404.611	-214.152
5	PC_Vivienda_2	-8.2742	-335.274
6	PC_Combustible_1	17.075	-37.6225
7	PC_Vivienda_3	626.323	291.409
8	PC_Economia_2	-314.067	-155.059

In [53]:

```
plt.figure(figsize=(20,10))
plt.plot(features, regresion_model.coef_,alpha=0.7,linestyle='none',marker='*', markersize=10,color='red',label=r'Regresi
plt.plot(features, ridge_model.coef_,alpha=0.5,linestyle='none',marker='d',markersize=10,color='blue',label=r'Regresion R
plt.xlabel('Coefficient Index',fontsize=16)
plt.ylabel('Coefficient Magnitude',fontsize=16)
plt.legend(fontsize=13,loc='center left')
plt.show()
```



6. Deployment

In [54]:
```
tx_deploy = tx_deploy.set_index('ID')
```

In [55]:
```
tx_deploy = tx_deploy[features]
```

In [56]:
```
#Aplicamos el Modelo
TPD_Ridge=pd.Series(ridge_model.predict(tx_deploy))
TPD_Tree=pd.Series(regressor.predict(tx_deploy))
```

In [57]:
```
TPD=pd.DataFrame(TPD_Ridge)
TPD.columns =['Ridge']
```

```
TPD['Tree'] = TPD_Tree
```

In [58]:
```
TPD['Ridge_total']=TPD['Ridge']
TPD['Tree_total']=TPD['Tree']
```

In [59]:
```
TPD
```

Out[59]:

	Ridge	Tree	Ridge_total	Tree_total
0	38519.434517	17061.848433	38519.434517	17061.848433
1	38560.604588	17061.848433	38560.604588	17061.848433
2	38884.476441	17061.848433	38884.476441	17061.848433
3	38666.123573	17061.848433	38666.123573	17061.848433
4	38572.057468	17061.848433	38572.057468	17061.848433
5	38518.054969	17061.848433	38518.054969	17061.848433
6	38466.413566	17061.848433	38466.413566	17061.848433
7	38842.112881	17061.848433	38842.112881	17061.848433
8	38407.111396	17061.848433	38407.111396	17061.848433
9	38181.799919	17061.848433	38181.799919	17061.848433
10	38249.117527	17061.848433	38249.117527	17061.848433
11	38513.231222	17061.848433	38513.231222	17061.848433
12	38443.051926	17061.848433	38443.051926	17061.848433
13	38455.767297	17061.848433	38455.767297	17061.848433
14	38501.447994	17061.848433	38501.447994	17061.848433
15	38711.230603	17061.848433	38711.230603	17061.848433
16	40375.294565	17061.848433	40375.294565	17061.848433
17	39002.447329	17061.848433	39002.447329	17061.848433
18	38760.773404	17061.848433	38760.773404	17061.848433

	Ridge	Tree	Ridge_total	Tree_total
19	39352.025113	17061.848433	39352.025113	17061.848433
20	35482.107508	17061.848433	35482.107508	17061.848433
21	40032.410700	17061.848433	40032.410700	17061.848433
22	38565.610104	17061.848433	38565.610104	17061.848433
23	38552.910708	17061.848433	38552.910708	17061.848433
24	38538.402425	17061.848433	38538.402425	17061.848433

7. COMPARACION REAL- MODELO

In [60]:
```
dc = pd.DataFrame(y_test)
dc['TPD_Ridge'] = y_pred

dc['TPD_REAL']=dc['TOTAL_TPD']
dc['TPD_ESTIMADO']=dc['TPD_Ridge']
dc['ERROR']=dc['TPD_ESTIMADO']-dc['TPD_REAL']
dc['%_ERROR'] = (abs(dc['ERROR'])/dc['TPD_ESTIMADO'])*100
dc
```

Out[60]:

ID	TOTAL_TPD	TPD_Ridge	TPD_REAL	TPD_ESTIMADO	ERROR	%_ERROR
20182	16505.868940	16452.206809	16505.868940	16452.206809	-53.662131	0.326170
20181	15748.693347	16614.198513	15748.693347	16614.198513	865.505166	5.209431
20189	17128.190260	16610.390090	17128.190260	16610.390090	-517.800170	3.117327
20176	15168.318524	16092.582534	15168.318524	16092.582534	924.264010	5.743416
20162	14834.034483	15031.665090	14834.034483	15031.665090	197.630608	1.314762
20155	12602.322581	13087.200522	12602.322581	13087.200522	484.877942	3.704978

In [ ]:
```
```

Apéndice B: Memoria de cálculo Python para la Serie de Modelos B

SERIE B - MODELO RADIAL COYOL

1. CONFIGURACION GENERAL

1.1. Importamos las Librerias a utilizar

In [1]:

```
# Librerías básicas
import pandas as pd
import numpy as np
import warnings

# Librerias Machine Learnign & Estadísticas
from sklearn import linear_model
from sklearn.tree import DecisionTreeRegressor
from scipy import stats
import statsmodels
import statsmodels.api as sm
import statsmodels.formula.api as smf
import statsmodels.stats.api as sms
from statsmodels.compat import lzip
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import StandardScaler

# Visualización de Datos
import seaborn as sns
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import pylab

# Métricas de Evaluación
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error

# Otras
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
from sklearn.decomposition import PCA
import math
```

1.2. Cargamos el set de datos base

In [2]:
```
#Carga inicial del Data Set
data = pd.read_excel('C:/Users/Rodrigo Mata/Desktop/Proyecto Coyol/FINAL/DATASET_FINAL_NATALIA.xls', sheet_name='BASEN3')

#Seleccionados ID como index
data =data.set_index('ID')
```

1.3. Cargamos el set de deploy

In [3]:
```
#Cargamos base de referencia
datos = pd.read_excel('C:/Users/Rodrigo Mata/Desktop/Proyecto Coyol/FINAL/DATASET_FINAL_NATALIA.xls', sheet_name='APLICAC

#Seleccionados ID como index
datos =datos.set_index('ID')
```

1.4. Limpieza final de los datos

In [4]:
```
#Eliminamos columna YEAR
data=data.drop('YEAR', axis=1)
```

2. Definición de la Variable Dependiente (Target)

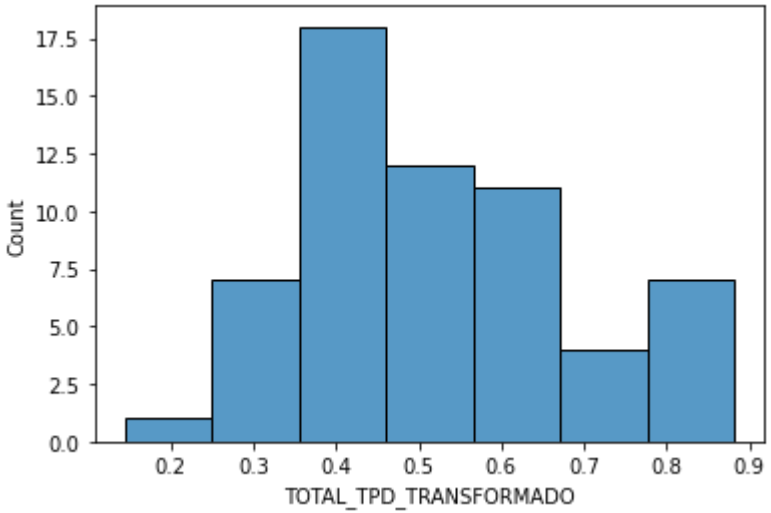
2.1. Seleccion de la Variable Independiente

In [5]:
```
#Definimos la variable independiente como el Total TPD Transformado
target = 'TOTAL_TPD_TRANSFORMADO'

#Generamos el vector final con la Variable Independiente
y = data[target]
```

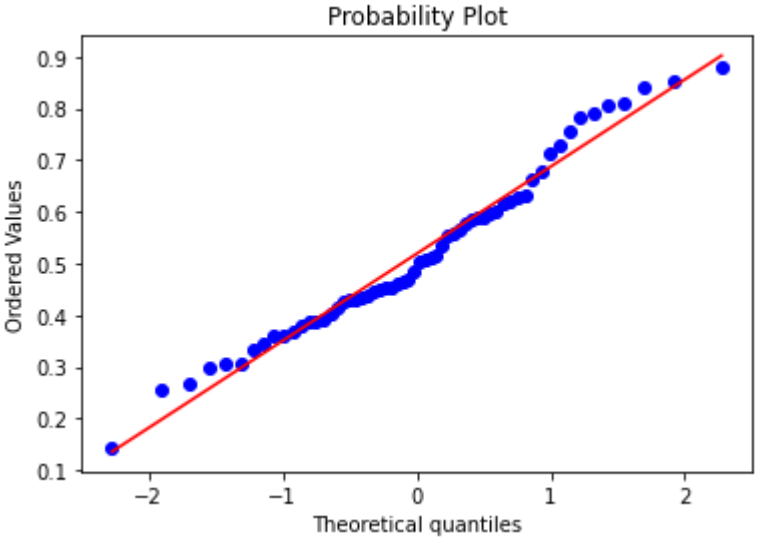
2.2. Normalidad de la Variable Independiente

In [6]:
```
#Metodo 1: Histograma
sns.histplot(data[target])
```

Out[6]: <AxesSubplot:xlabel='TOTAL_TPD_TRANSFORMADO', ylabel='Count'>



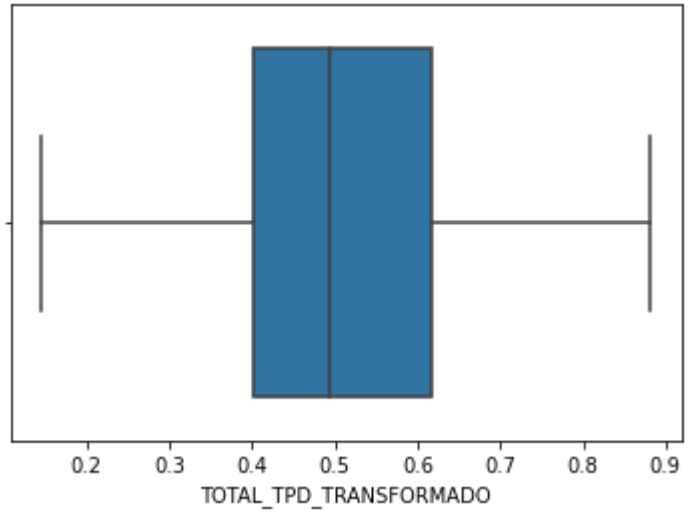
In [7]:
```
#metodo 2: Quantile-Quantile Plot
stats.probplot(data[target], dist='norm', plot=pylab)
pylab.show()
```



In [8]:
```
#Metodo 3: Box Plot
sns.boxplot(data[target])
```

```
C:\Users\Rodrigo Mata\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable a
s a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments wit
hout an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[8]: <AxesSubplot:xlabel='TOTAL_TPD_TRANSFORMADO'>



In [9]:

```
# Prueba de Normalidad de Variable Independiente
# Prueba Shapiro-Wilk
shapiro_test = stats.shapiro(data[target])
shapiro_test

if shapiro_test[1]> 0.05:
    print(shapiro_test)
    print('Normal')

else:
    print(shapiro_test)
    print('No Normal')
```

```
ShapiroResult(statistic=0.9739791750907898, pvalue=0.22742749750614166)
Normal
```

2.3. Filtros de Outliers en Variable Independiente

In [10]:

```
#Definimos el valor de Z para definición de valores extremos
Z = 3
```

In [11]:
```
#Tamaño de muestra inicial
print('Tamaño de Muestra Inicial:' + str(y.shape))
```

```
Tamaño de Muestra Inicial:(60,)
```

In [12]:
```
#Eliminamos valores extremos que estén fuera del rango de Z Desviaciones Estandar
upper_lim = y.mean () + y.std () * Z
lower_lim = y.mean () - y.std () * Z

y = y[(y < upper_lim) & (y > lower_lim)]
```

In [13]:
```
#Tamaño de muestra final
print('Tamaño de Muestra luego de eliminar oultiers:'+ str(y.shape))
```

```
Tamaño de Muestra luego de eliminar oultiers:(60,)
```

3. Definición de Variables Independientes (Features)

3.1. Selección de Variables Independientes

In [14]:
```
#Lista de Variables Independientes a revisar
lista_features = ['DIESEL', 'GAS_SUPER', 'GAS_REGULAR', #variables combustible
          'IPC', 'IMAE', 'TIPO_CAMBIO',  'IMPORTACION_VEHICULOS', #varibales economicas
         'GARITA_VIVIENDA_ACUM', 'SAN_JOSE_VIVIENDA_ACUM', 'SAN_ANTONIO_VIVIENDA_ACUM', 'TURRUCARES_VIVIENDA_ACUM',
          'SAN_JOSE_CONDOMINIO_ACUM', 'SAN_ANTONIO_CONDOMINIO_ACUM', 'GARITA_CONDOMINIO_ACUM', #variables vivienda
         'GARITA_INDUSTRIAL_ACUM', 'SAN_ANTONIO_INDUSTRIAL_ACUM', 'TURRUCARES_INDUSTRIAL_ACUM', #variables industriales
         'GARITA_COMERCIAL_ACUM', 'SAN_ANTONIO_COMERCIAL_ACUM', 'SAN_JOSE_COMERCIAL_ACUM',
          'TURRUCARES_COMERCIAL_ACUM', #variables comerciales
          'GARITA_SERVICIOS_ACUM', 'SAN_ANTONIO_SERVICIOS_ACUM', 'TURRUCARES_SERVICIOS_ACUM',
          'SAN JOSE_SERVICIOS_ACUM', #variables servicios
         'TPD_RN1_TOTAL' #varibales rutas primarias
        ]
```

In [15]:
```
#Definimos data set de variables independientes
X = data[lista_features]
print('Variables Independientes seleccionadas:'+str(X.columns))
```

```
Variables Independientes seleccionadas:Index(['DIESEL', 'GAS_SUPER', 'GAS_REGULAR', 'IPC', 'IMAE', 'TIPO_CAMBIO',
       'IMPORTACION_VEHICULOS', 'GARITA_VIVIENDA_ACUM',
```

```
       'SAN_JOSE_VIVIENDA_ACUM', 'SAN_ANTONIO_VIVIENDA_ACUM',
       'TURRUCARES_VIVIENDA_ACUM', 'SAN_JOSE_CONDOMINIO_ACUM',
       'SAN_ANTONIO_CONDOMINIO_ACUM', 'GARITA_CONDOMINIO_ACUM',
       'GARITA_INDUSTRIAL_ACUM', 'SAN_ANTONIO_INDUSTRIAL_ACUM',
       'TURRUCARES_INDUSTRIAL_ACUM', 'GARITA_COMERCIAL_ACUM',
       'SAN_ANTONIO_COMERCIAL_ACUM', 'SAN_JOSE_COMERCIAL_ACUM',
       'TURRUCARES_COMERCIAL_ACUM', 'GARITA_SERVICIOS_ACUM',
       'SAN_ANTONIO_SERVICIOS_ACUM', 'TURRUCARES_SERVICIOS_ACUM',
       'SAN JOSE_SERVICIOS_ACUM', 'TPD_RN1_TOTAL'],
      dtype='object')
```

In [16]:
```
#Tamaño de muestra inicial
print('Tamaño Muestra Inicial:'+str(X.shape))
```

```
Tamaño Muestra Inicial:(60, 26)
```

3.2. Revisión de Outliers en Variables Independientes

In [17]:
```
#Se crea bucle para revision de outliers en Variables Independientes
for i in lista_features:
    upper_lim = X[i].mean () + X[i].std () * Z
    lower_lim = X[i].mean () - X[i].std () * Z

    X = X[(X[i] < upper_lim) & (X[i] > lower_lim)]
```

In [18]:
```
#Tamaño de Muestra Final
print('Tamaño Muestra Final:'+str(X.shape))
```

```
Tamaño Muestra Final:(58, 26)
```

4. Ajustes finales del set de datos para modelo

4.1. Data Set Final del Modelo

In [19]:
```
# Se integran ambas Variables (Dependiente e Independientes) para generar el data set final del modelo
df_final = pd.merge(y, X, how='inner', on='ID')
```

In [20]:
```
#Revisión de tamaño final de muestra
print('Tamaño Muestra Final:'+str(df_final.shape))
```

```
df_final.shape
```

```
Tamaño Muestra Final:(58, 27)
```

Out[20]:
```
(58, 27)
```

In [21]:
```
#Definimos Data Final Variable Independiente
y =df_final[target]
```

In [22]:
```
#Definimos Data Final Variable Dependiente
X = df_final[lista_features]
```

In [23]:
```
#Split data set en set de entrenamiento (calibración del modelo) y set de prueba (evaluación del modelo)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.10, random_state=0)
```

4.2. Metodologia para la reduccón de dimensiones - Principal Component Analysis

In [24]:
```
#Definimos funcion para serapar features por categorías

def features(df):
    df_combustible = df[['DIESEL', 'GAS_SUPER', 'GAS_REGULAR']]
    df_economia = df[['IPC', 'IMAE', 'TIPO_CAMBIO']]
    df_vivienda = df[['GARITA_VIVIENDA_ACUM', 'SAN_JOSE_VIVIENDA_ACUM', 'SAN_ANTONIO_VIVIENDA_ACUM', 'TURRUCARES_VIVIENDA
    df_industrial = df[['GARITA_INDUSTRIAL_ACUM', 'SAN_ANTONIO_INDUSTRIAL_ACUM', 'TURRUCARES_INDUSTRIAL_ACUM']]
    df_comercial = df[['GARITA_COMERCIAL_ACUM', 'SAN_ANTONIO_COMERCIAL_ACUM', 'SAN_JOSE_COMERCIAL_ACUM','TURRUCARES_COMER
    df_servicio = df[['GARITA_SERVICIOS_ACUM', 'SAN_ANTONIO_SERVICIOS_ACUM', 'TURRUCARES_SERVICIOS_ACUM', 'SAN JOSE_SERVI
    df_lista = [df_combustible, df_economia, df_vivienda, df_industrial, df_comercial, df_servicio]

    return df_lista
```

In [25]:
```
#Definimos grupos de variables y numero de PC's
var = ['Combustible', 'Economia', 'Vivienda', 'Industrial', 'Comercial', 'Servicio']
n = [1, 2, 3, 1, 2, 2]
```

In [26]:
```
#Aplicamos la función para separar features por categorías
x_subset_train = features(X_train)
x_subset_test = features(X_test)
x_subset_deploy = features(datos)
```

In [27]:

```
# PCA_COMBUSTIBLE
l=0

#Standarizamos la data
scaler = StandardScaler()

# Fit on training set only.
scaler.fit(x_subset_train[l])

# Apply transform to both the training set and the test set.
x_subset_train[l] = scaler.transform(x_subset_train[l])
x_subset_test[l] = scaler.transform(x_subset_test[l])
x_subset_deploy[l] = scaler.transform(x_subset_deploy[l])


pca = PCA(n_components = n[l])
pca.fit(x_subset_train[l])

x_subset_train[l] = pca.transform(x_subset_train[l])
x_subset_test[l]  = pca.transform(x_subset_test[l])
x_subset_deploy[l]  = pca.transform(x_subset_deploy[l])

explained_variance = pca.explained_variance_ratio_

tx_train = pd.DataFrame(x_subset_train[l])
tx_train['ID'] = X_train.index
tx_train.columns = ['PC_Combustible_1', 'ID']

tx_test = pd.DataFrame(x_subset_test[l])
tx_test['ID'] = X_test.index
tx_test.columns = ['PC_Combustible_1', 'ID']

tx_deploy = pd.DataFrame(x_subset_deploy[l])
tx_deploy['ID'] = datos.index
tx_deploy.columns = ['PC_Combustible_1', 'ID']
```

In [28]:

```
# PCA_ECONOMIA
l=1

#Standarizamos la data
scaler = StandardScaler()
```

```
# Fit on training set only.
scaler.fit(x_subset_train[l])

# Apply transform to both the training set and the test set.
x_subset_train[l] = scaler.transform(x_subset_train[l])
x_subset_test[l] = scaler.transform(x_subset_test[l])
x_subset_deploy[l] = scaler.transform(x_subset_deploy[l])

pca = PCA(n_components = n[l])
pca.fit(x_subset_train[l])

x_subset_train[l] = pca.transform(x_subset_train[l])
x_subset_test[l]  = pca.transform(x_subset_test[l])
x_subset_deploy[l]  = pca.transform(x_subset_deploy[l])

explained_variance = pca.explained_variance_ratio_

tx_train_temp = pd.DataFrame(x_subset_train[l])
tx_train_temp['ID'] = X_train.index
tx_train_temp.columns = ['PC_Economia_1', 'PC_Economia_2', 'ID']
tx_train = pd.merge(tx_train, tx_train_temp, how='left', on='ID')

tx_test_temp = pd.DataFrame(x_subset_test[l])
tx_test_temp['ID'] = X_test.index
tx_test_temp.columns = ['PC_Economia_1', 'PC_Economia_2', 'ID']
tx_test = pd.merge(tx_test, tx_test_temp, how='left', on='ID')

tx_deploy_temp = pd.DataFrame(x_subset_deploy[l])
tx_deploy_temp['ID'] = datos.index
tx_deploy_temp.columns = ['PC_Economia_1', 'PC_Economia_2', 'ID']
tx_deploy = pd.merge(tx_deploy, tx_deploy_temp, how='left', on='ID')
```

In [29]:

```
# PCA_VIVIENDA
l=2

#Standarizamos la data
scaler = StandardScaler()

# Fit on training set only.
scaler.fit(x_subset_train[l])

# Apply transform to both the training set and the test set.
x_subset_train[l] = scaler.transform(x_subset_train[l])
x_subset_test[l] = scaler.transform(x_subset_test[l])
```

```
x_subset_deploy[l] = scaler.transform(x_subset_deploy[l])

pca = PCA(n_components = n[l])
pca.fit(x_subset_train[l])

x_subset_train[l] = pca.transform(x_subset_train[l])
x_subset_test[l]  = pca.transform(x_subset_test[l])
x_subset_deploy[l]  = pca.transform(x_subset_deploy[l])


explained_variance = pca.explained_variance_ratio_

tx_train_temp = pd.DataFrame(x_subset_train[l])
tx_train_temp['ID'] = X_train.index
tx_train_temp.columns = ['PC_vivienda_1', 'PC_Vivienda_2', 'PC_Vivienda_3', 'ID']
tx_train = pd.merge(tx_train, tx_train_temp, how='left', on='ID')

tx_test_temp = pd.DataFrame(x_subset_test[l])
tx_test_temp['ID'] = X_test.index
tx_test_temp.columns = ['PC_vivienda_1', 'PC_Vivienda_2', 'PC_Vivienda_3', 'ID']
tx_test = pd.merge(tx_test, tx_test_temp, how='left', on='ID')

tx_deploy_temp = pd.DataFrame(x_subset_deploy[l])
tx_deploy_temp['ID'] = datos.index
tx_deploy_temp.columns = ['PC_vivienda_1', 'PC_Vivienda_2', 'PC_Vivienda_3', 'ID']
tx_deploy = pd.merge(tx_deploy, tx_deploy_temp, how='left', on='ID')
```

In [30]:

```
# PCA_INDUSTRIAL
l=3

#Standarizamos la data
scaler = StandardScaler()

# Fit on training set only.
scaler.fit(x_subset_train[l])

# Apply transform to both the training set and the test set.
x_subset_train[l] = scaler.transform(x_subset_train[l])
x_subset_test[l] = scaler.transform(x_subset_test[l])
x_subset_deploy[l] = scaler.transform(x_subset_deploy[l])

pca = PCA(n_components = n[l])
pca.fit(x_subset_train[l])
```

```
x_subset_train[l] = pca.transform(x_subset_train[l])
x_subset_test[l]  = pca.transform(x_subset_test[l])
x_subset_deploy[l]  = pca.transform(x_subset_deploy[l])


explained_variance = pca.explained_variance_ratio_

tx_train_temp = pd.DataFrame(x_subset_train[l])
tx_train_temp['ID'] = X_train.index
tx_train_temp.columns = ['PC_industrial_1', 'ID']
tx_train = pd.merge(tx_train, tx_train_temp, how='left', on='ID')

tx_test_temp = pd.DataFrame(x_subset_test[l])
tx_test_temp['ID'] = X_test.index
tx_test_temp.columns = ['PC_industrial_1', 'ID']
tx_test = pd.merge(tx_test, tx_test_temp, how='left', on='ID')

tx_deploy_temp = pd.DataFrame(x_subset_deploy[l])
tx_deploy_temp['ID'] = datos.index
tx_deploy_temp.columns = ['PC_industrial_1', 'ID']
tx_deploy = pd.merge(tx_deploy, tx_deploy_temp, how='left', on='ID')
```

In [31]:

```
# PCA_COMERCIAL
l=4

#Standarizamos la data
scaler = StandardScaler()

# Fit on training set only.
scaler.fit(x_subset_train[l])

# Apply transform to both the training set and the test set.
x_subset_train[l] = scaler.transform(x_subset_train[l])
x_subset_test[l] = scaler.transform(x_subset_test[l])
x_subset_deploy[l] = scaler.transform(x_subset_deploy[l])

pca = PCA(n_components = n[l])
pca.fit(x_subset_train[l])

x_subset_train[l] = pca.transform(x_subset_train[l])
x_subset_test[l]  = pca.transform(x_subset_test[l])
x_subset_deploy[l]  = pca.transform(x_subset_deploy[l])
```

```
explained_variance = pca.explained_variance_ratio_

tx_train_temp = pd.DataFrame(x_subset_train[l])
tx_train_temp['ID'] = X_train.index
tx_train_temp.columns = ['PC_comercial_1', 'PC_comercial_2', 'ID']
tx_train = pd.merge(tx_train, tx_train_temp, how='left', on='ID')

tx_test_temp = pd.DataFrame(x_subset_test[l])
tx_test_temp['ID'] = X_test.index
tx_test_temp.columns = ['PC_comercial_1', 'PC_comercial_2', 'ID']
tx_test = pd.merge(tx_test, tx_test_temp, how='left', on='ID')

tx_deploy_temp = pd.DataFrame(x_subset_deploy[l])
tx_deploy_temp['ID'] = datos.index
tx_deploy_temp.columns = ['PC_comercial_1', 'PC_comercial_2', 'ID']
tx_deploy = pd.merge(tx_deploy, tx_deploy_temp, how='left', on='ID')
```

In [32]:

```
# PCA_SERVICIO
l=5

#Standarizamos la data
scaler = StandardScaler()

# Fit on training set only.
scaler.fit(x_subset_train[l])

# Apply transform to both the training set and the test set.
x_subset_train[l] = scaler.transform(x_subset_train[l])
x_subset_test[l] = scaler.transform(x_subset_test[l])
x_subset_deploy[l] = scaler.transform(x_subset_deploy[l])

pca = PCA(n_components = n[l])
pca.fit(x_subset_train[l])

x_subset_train[l] = pca.transform(x_subset_train[l])
x_subset_test[l]  = pca.transform(x_subset_test[l])
x_subset_deploy[l]  = pca.transform(x_subset_deploy[l])


explained_variance = pca.explained_variance_ratio_

tx_train_temp = pd.DataFrame(x_subset_train[l])
tx_train_temp['ID'] = X_train.index
tx_train_temp.columns = ['PC_servicio_1', 'PC_servicio_2', 'ID']
```

```
tx_train = pd.merge(tx_train, tx_train_temp, how='left', on='ID')

tx_test_temp = pd.DataFrame(x_subset_test[l])
tx_test_temp['ID'] = X_test.index
tx_test_temp.columns = ['PC_servicio_1', 'PC_servicio_2', 'ID']
tx_test = pd.merge(tx_test, tx_test_temp, how='left', on='ID')

tx_deploy_temp = pd.DataFrame(x_subset_deploy[l])
tx_deploy_temp['ID'] = datos.index
tx_deploy_temp.columns = ['PC_servicio_1', 'PC_servicio_2', 'ID']
tx_deploy = pd.merge(tx_deploy, tx_deploy_temp, how='left', on='ID')
```

5. Feauture Selection

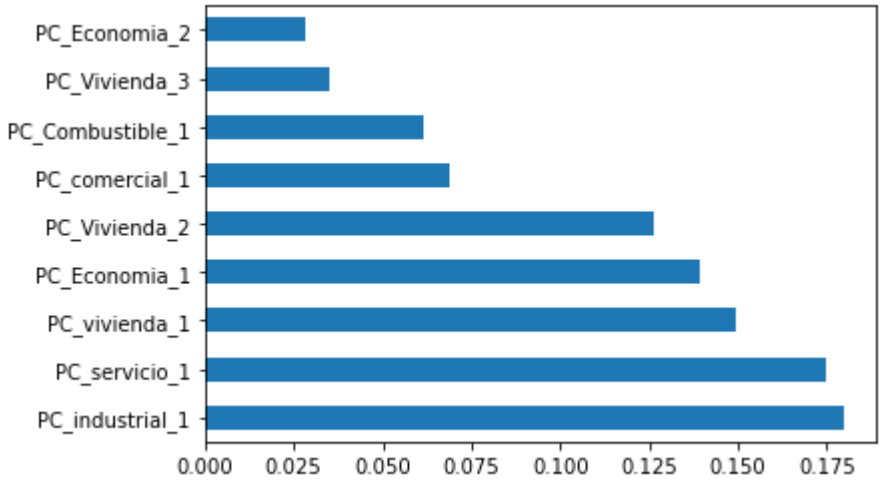
In [33]:
```
#Definition of predictors number to be used.   Aprox 10% of total instances.
n=9
```

In [34]:
```
X_train = tx_train.copy().set_index('ID')
X_test = tx_test.copy().set_index('ID')
```

In [35]:
```
from sklearn.ensemble import ExtraTreesRegressor

model = ExtraTreesRegressor()
model.fit(X_train,y_train)
print(model.feature_importances_) #use inbuilt class feature_importances of tree based classifiers
```

```
[0.06150567 0.13914238 0.02849585 0.14937687 0.12614568 0.03498167
 0.17966927 0.0688852  0.0217768  0.174611   0.01540961]
```

In [36]:
```
#plot graph of feature importances for better visualization
import matplotlib.pyplot as plt
feat_importances = pd.Series(model.feature_importances_, index=X_train.columns)
feat_importances.nlargest(n).plot(kind='barh')
plt.show()
```



In [37]:
```
#Creamos Data Set final de Features
features = pd.Series(feat_importances.nlargest(n).index)

X_train = X_train[features]
X_test  = X_test[features]
```

5. Entrenamiento de los Modelos

- Regresión Lineal Simple
- Arbol de Regresión
- Regresión Ridge

5.1. Modelo de Regresión Lineal Simple

In [38]:
```
# Tunning de Hiperparámetros Regresión Lineal Simple
params_regresion = {
    'fit_intercept' : [True, False],
    'normalize' : [True,False],
    'n_jobs' :[1,-1]
}

regressor = linear_model.LinearRegression()

l_regressor = GridSearchCV(regressor, params_regresion, scoring='r2', cv=5, n_jobs=-1)
```

```
l_regressor.fit(X_train, y_train)
print(f'fit_intercept: {l_regressor.best_params_["fit_intercept"]}')
print(f'normalize: {l_regressor.best_params_["normalize"]}')
print(f'n_jobs: {l_regressor.best_params_["n_jobs"]}')
print(f'Best score: {l_regressor.best_score_}')
```

```
fit_intercept: True
normalize: True
n_jobs: 1
Best score: 0.7027364765734959
```

In [39]:

```
# Aplicación del Modelo de Regresión Lineal Simple selecionado
regresion_model = linear_model.LinearRegression(fit_intercept=l_regressor.best_params_["fit_intercept"],
                    normalize=l_regressor.best_params_["normalize"], n_jobs=l_regressor.best_params_["n_jobs"])
regresion_model.fit(X_train, y_train)
y_train_pred = regresion_model.predict(X_train)
y_pred = regresion_model.predict(X_test)
print('Train r2 score: ', r2_score(y_train_pred, y_train))
print('Test r2 score: ', r2_score(y_test, y_pred))
train_rmse = np.sqrt(mean_squared_error(y_train_pred, y_train))
test_rmse = np.sqrt(mean_squared_error(y_test, y_pred))
print(f'Train RMSE: {train_rmse:.4f}')
print(f'Test RMSE: {test_rmse:.4f}')
```

```
Train r2 score:  0.7537786781714987
Test r2 score:  0.852404559094879
Train RMSE: 0.0693
Test RMSE: 0.0583
```

In [40]:

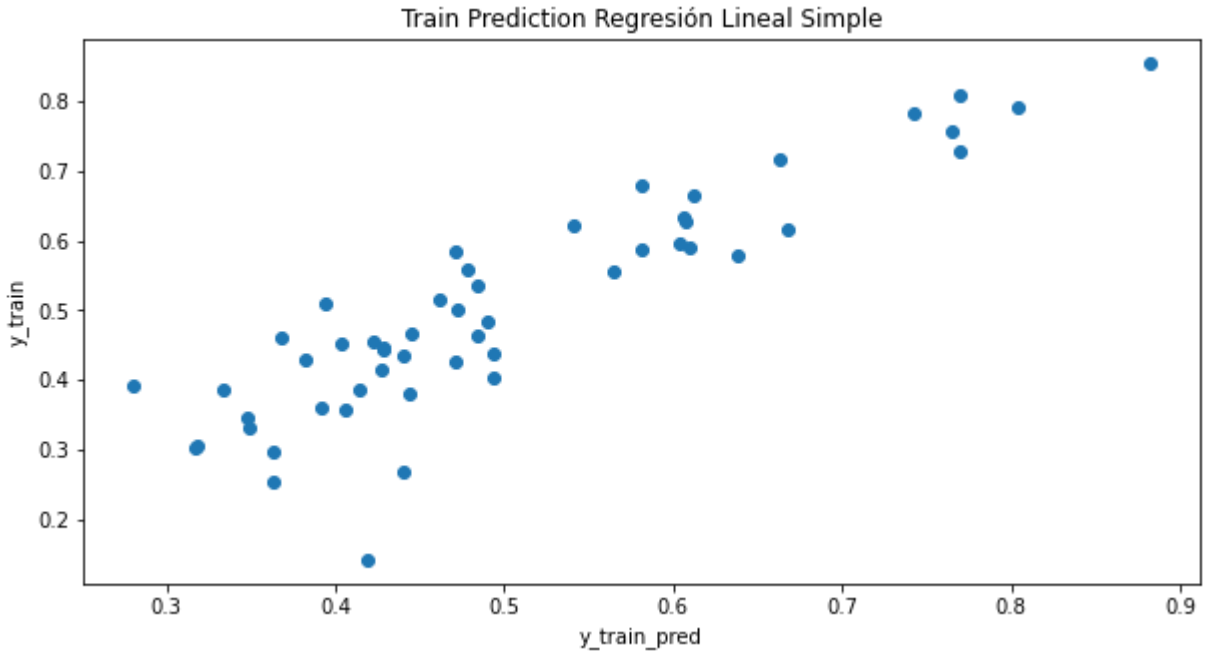
```
#TABLA DE REGRESION LINEAL SIMPLE SIN AJUSTAR
from regressors import stats

stats.summary(regresion_model, X_train, y_train)
```

```
Residuals:
    Min      1Q  Median      3Q     Max
-0.1177 -0.0481  0.0049  0.0299  0.2747


Coefficients:
            Estimate  Std. Error  t value   p value
_intercept  0.502538    0.009613  52.2783  0.000000
x1          0.181012    0.046220   3.9163  0.000268
x2          0.056339    0.034543   1.6310  0.109054
x3          0.105839    0.048525   2.1811  0.033808
```

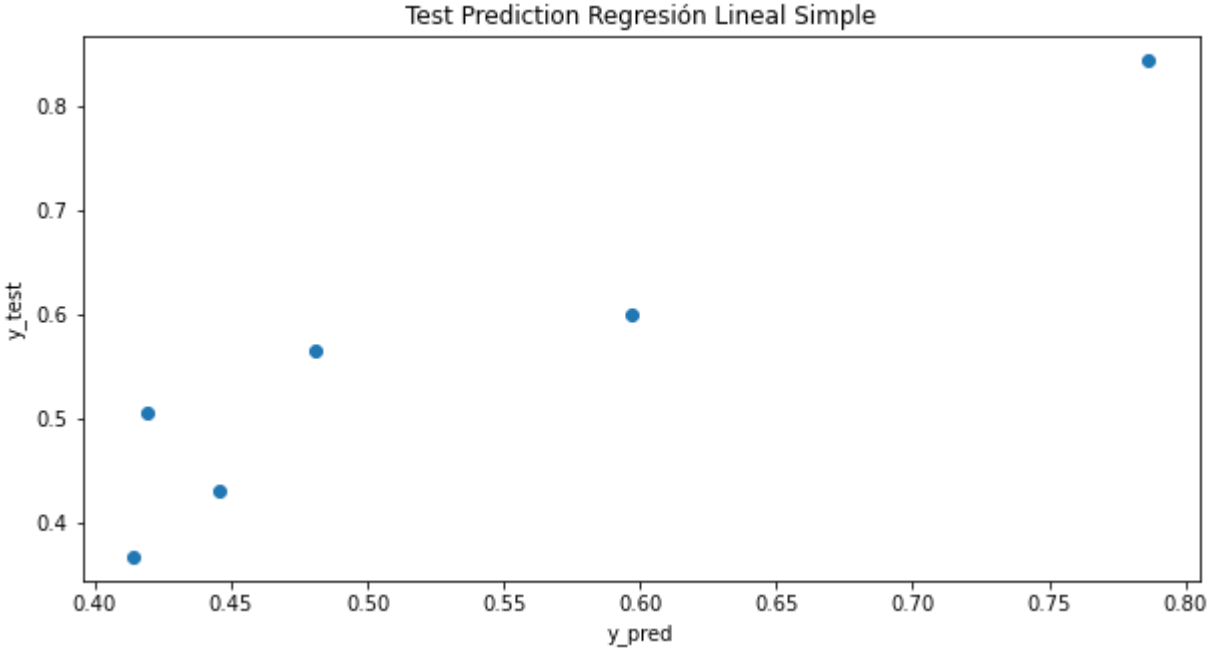
```
x4          0.092281    0.025732   3.5863  0.000751
x5          0.003879    0.021365   0.1816  0.856648
x6          0.041219    0.035635   1.1567  0.252778
x7         -0.001509    0.008507  -0.1774  0.859909
x8         -0.063640    0.037212  -1.7102  0.093303
x9          0.031600    0.022814   1.3851  0.172047
---
R-squared:  0.80243,    Adjusted R-squared:  0.76009
F-statistic: 18.95 on 9 features
```

In [41]:

```
#Visualización Modelo de Regresión Lineal Simple

#Train
xv = y_train_pred
yv = y_train

plt.figure(figsize = (10,5))
plt.scatter(xv, yv, alpha=1)
plt.title('Train Prediction Regresión Lineal Simple')
plt.xlabel('y_train_pred')
plt.ylabel('y_train')
graf1 = plt.show()
print(graf1)

#Test
xv = y_pred
yv = y_test

plt.figure(figsize = (10,5))
plt.scatter(xv, yv, alpha=1)
plt.title('Test Prediction Regresión Lineal Simple')
plt.xlabel('y_pred')
plt.ylabel('y_test')
graf2 = plt.show()
print(graf2)
```


Train Prediction Regresión Lineal Simple
y_train
y_train_pred
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.3
0.4
0.5
0.6
0.7
0.8
0.9

None


Test Prediction Regresión Lineal Simple
y_test
y_pred
0.4
0.5
0.6
0.7
0.8
0.40
0.45
0.50
0.55
0.60
0.65
0.70
0.75
0.80

None

5.2. Modelo Arbol de Regresión

In [42]:
```
# Tunning de Hiperparámetros Arbol de Regresión

params_tree = {
    'max_depth' : [2, 3, 4, 5],
    'min_samples_split' : [ 2, 3, 4],
    'min_samples_leaf' : [1, 2],
}

regressor = DecisionTreeRegressor(random_state = 0)

tree_regressor = GridSearchCV(regressor, params_tree, scoring='r2', cv=5, n_jobs=-1)
tree_regressor.fit(X_train, y_train)
print(f'Optimal max_depth: {tree_regressor.best_params_["max_depth"]:.2f}')
print(f'Optimal min_samples_split: {tree_regressor.best_params_["min_samples_split"]}')
print(f'Optimal min_samples_leaf: {tree_regressor.best_params_["min_samples_leaf"]}')
print(f'Best score: {tree_regressor.best_score_}')
```

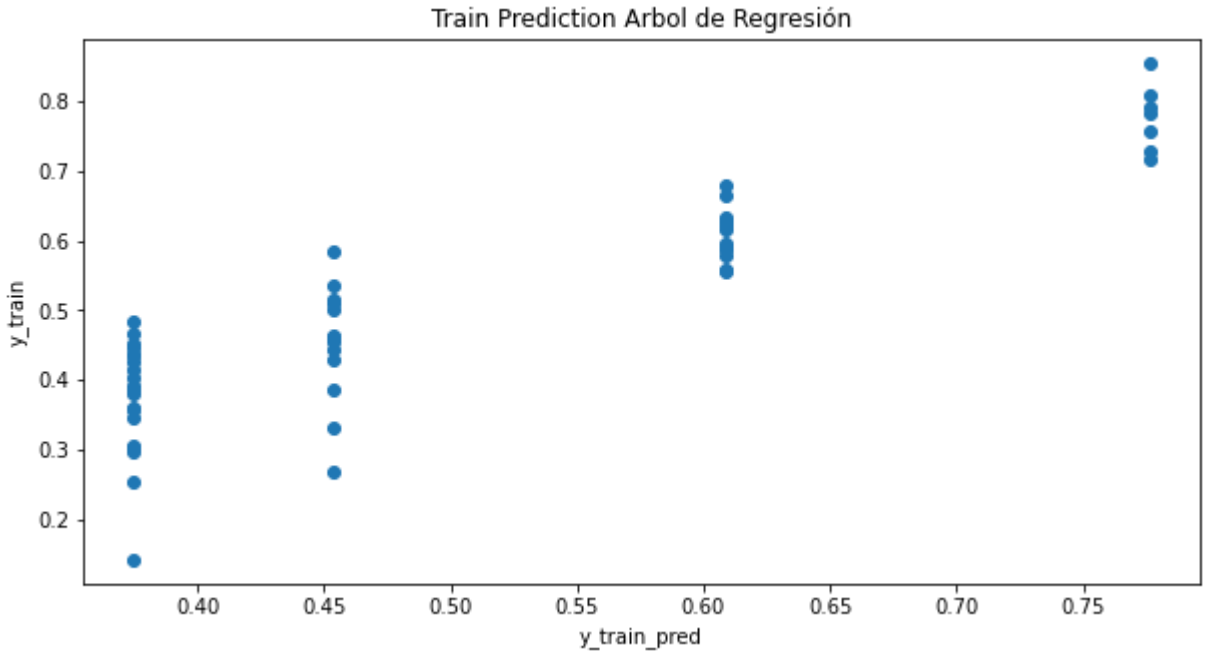
```
Optimal max_depth: 2.00
Optimal min_samples_split: 2
Optimal min_samples_leaf: 1
Best score: 0.6323651951248732
```

In [43]:
```
# Aplicación del Modelo de Arbol de Regresión selecionado

regressor = DecisionTreeRegressor(max_depth=tree_regressor.best_params_["max_depth"], min_samples_split=tree_regressor.be
                    min_samples_leaf=tree_regressor.best_params_["min_samples_leaf"])
regressor.fit(X_train, y_train)
y_train_pred = regressor.predict(X_train)
y_pred = regressor.predict(X_test)
print('Train r2 score: ', r2_score(y_train_pred, y_train))
print('Test r2 score: ', r2_score(y_test, y_pred))
train_rmse_tree = np.sqrt(mean_squared_error(y_train_pred, y_train))
test_rmse_tree = np.sqrt(mean_squared_error(y_test, y_pred))
print(f'Train RMSE: {train_rmse_tree:.4f}')
print(f'Test RMSE: {test_rmse_tree:.4f}')
```

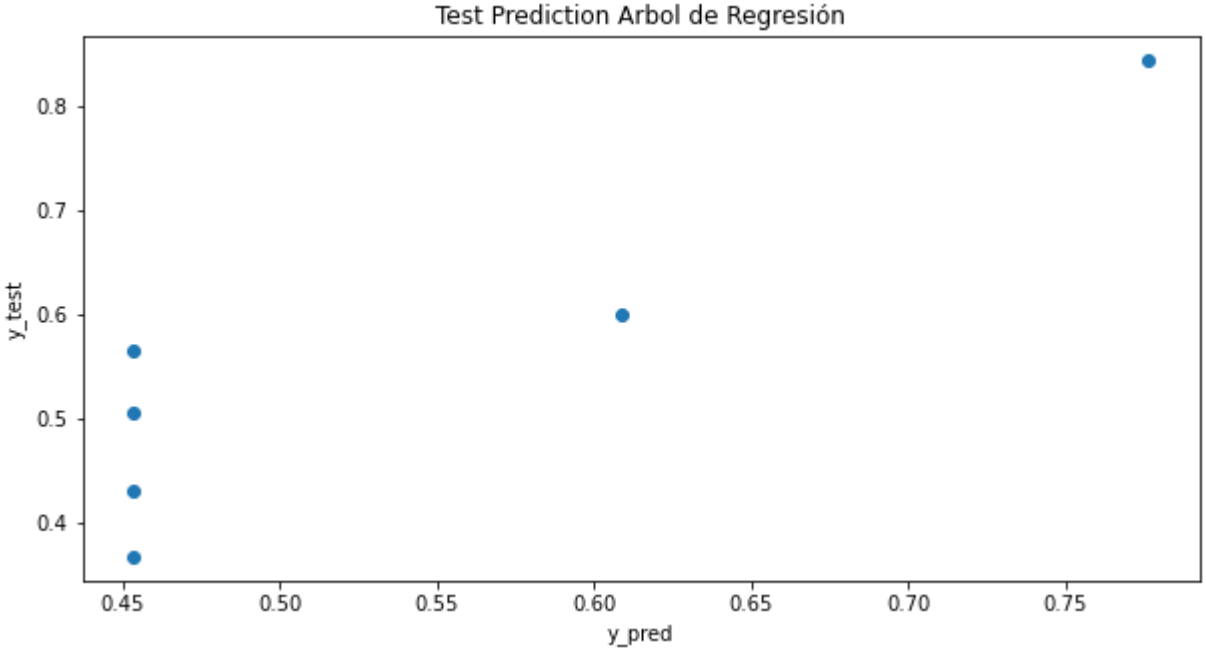
```
Train r2 score:  0.7567336898244291
Test r2 score:  0.7991195811497257
Train RMSE: 0.0690
Test RMSE: 0.0680
```

In [44]:
```
#Visualización Modelo Arbol de Regresión
```

```
#Train
xv = y_train_pred
yv = y_train

plt.figure(figsize = (10,5))
plt.scatter(xv, yv, alpha=1)
plt.title('Train Prediction Arbol de Regresión')
plt.xlabel('y_train_pred')
plt.ylabel('y_train')
graf1 = plt.show()
print(graf1)

#Test
xv = y_pred
yv = y_test

plt.figure(figsize = (10,5))
plt.scatter(xv, yv, alpha=1)
plt.title('Test Prediction Arbol de Regresión')
plt.xlabel('y_pred')
plt.ylabel('y_test')
graf2 = plt.show()
print(graf2)
```


Train Prediction Arbol de Regresión
y_train
y_train_pred
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.40
0.45
0.50
0.55
0.60
0.65
0.70
0.75

None


Test Prediction Arbol de Regresión
y_test
y_pred
0.8
0.7
0.6
0.5
0.4
0.45
0.50
0.55
0.60
0.65
0.70
0.75

None

5.3. Modelo Regresión Ridge

In [45]:
```
# Tunning de Hiperparámetros Regresión Ridge
from sklearn.linear_model import Ridge

params_ridge = {
    'alpha' : [0.1, 0.4, 0.5, .7, 0.75, 0.8, 0.85,.9, .99, 1, 5, 10, 20, 30],
    'fit_intercept' : [True, False],
    'normalize' : [True,False],
    'solver' : ['svd', 'cholesky', 'lsqr', 'sparse_cg', 'sag', 'saga']
}

ridge_model = Ridge()
ridge_regressor = GridSearchCV(ridge_model, params_ridge, scoring='r2', cv=5, n_jobs=-1)
ridge_regressor.fit(X_train, y_train)
print(f'Optimal alpha: {ridge_regressor.best_params_["alpha"]:.2f}')
print(f'Optimal fit_intercept: {ridge_regressor.best_params_["fit_intercept"]}')
print(f'Optimal normalize: {ridge_regressor.best_params_["normalize"]}')
print(f'Optimal solver: {ridge_regressor.best_params_["solver"]}')
print(f'Best score: {ridge_regressor.best_score_}')
```

```
Optimal alpha: 1.00
Optimal fit_intercept: True
Optimal normalize: False
Optimal solver: sparse_cg
Best score: 0.7247810491667256
```

In [46]:
```
# Aplicación del Modelo Regresión Ridge

ridge_model = Ridge(alpha=ridge_regressor.best_params_["alpha"], fit_intercept=ridge_regressor.best_params_["fit_intercep
                    normalize=ridge_regressor.best_params_["normalize"], solver=ridge_regressor.best_params_["solver"])
ridge_model.fit(X_train, y_train)
y_train_pred = ridge_model.predict(X_train)
y_pred = ridge_model.predict(X_test)
print('Train r2 score: ', r2_score(y_train_pred, y_train))
print('Test r2 score: ', r2_score(y_test, y_pred))
train_rmse = np.sqrt(mean_squared_error(y_train_pred, y_train))
test_rmse = np.sqrt(mean_squared_error(y_test, y_pred))
print(f'Train RMSE: {train_rmse:.4f}')
print(f'Test RMSE: {test_rmse:.4f}')
```

```
Train r2 score:  0.7251719494008566
Test r2 score:  0.8405302869671746
```

```
Train RMSE: 0.0709
Test RMSE: 0.0606
```

In [47]:

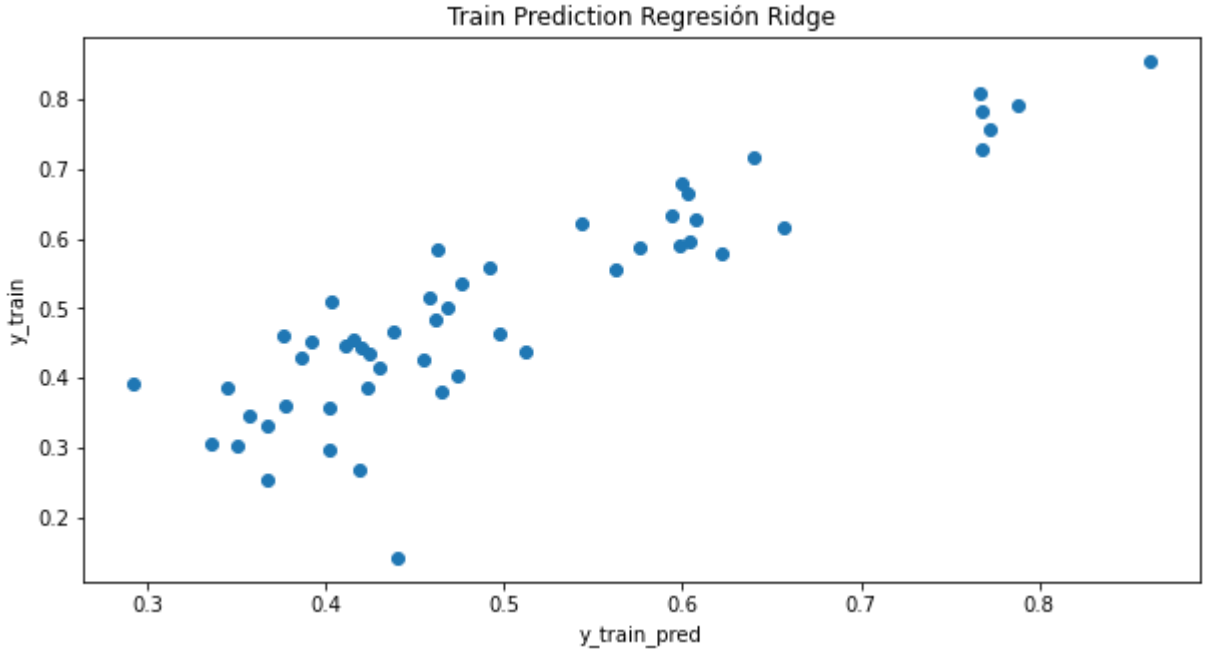
```
#TABLA DE REGRESION RIDGE SIN AJUSTAR
from regressors import stats

stats.summary(ridge_model, X_train, y_train)
```

```
Residuals:
    Min      1Q  Median      3Q     Max
-0.1225 -0.0422 -0.0068  0.0347  0.2957


Coefficients:
            Estimate  Std. Error  t value   p value
_intercept  0.502538    0.009834  51.1005  0.000000
x1          0.102026    0.047286   2.1577  0.035688
x2          0.048445    0.035339   1.3709  0.176417
x3          0.041579    0.049643   0.8376  0.406182
x4          0.091242    0.026325   3.4660  0.001080
x5          0.036863    0.021857   1.6865  0.097801
x6          0.021813    0.036456   0.5983  0.552262
x7          0.003988    0.008703   0.4583  0.648709
x8         -0.029544    0.038069  -0.7761  0.441295
x9          0.015605    0.023340   0.6686  0.506765
---
R-squared:  0.79321,    Adjusted R-squared:  0.74890
F-statistic: 17.90 on 9 features
```

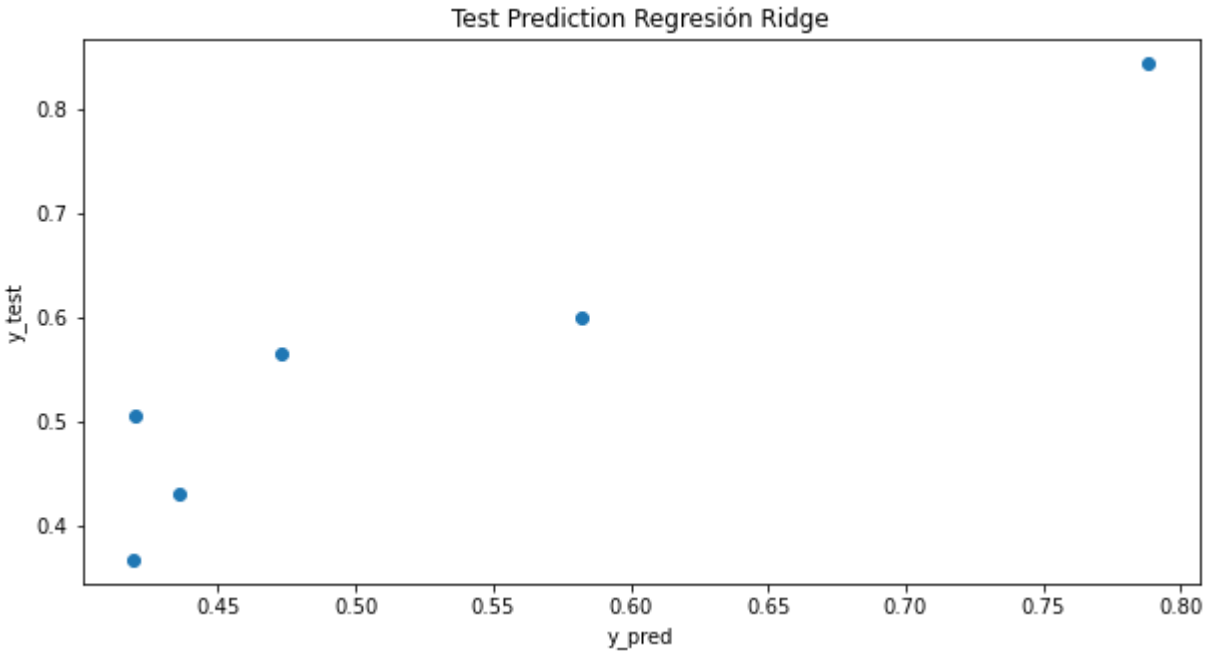
In [48]:

```
#Visualización Modelo Regresión Ridge

#Train
xv = y_train_pred
yv = y_train

plt.figure(figsize = (10,5))
plt.scatter(xv, yv, alpha=1)
plt.title('Train Prediction Regresión Ridge')
plt.xlabel('y_train_pred')
plt.ylabel('y_train')
graf1 = plt.show()
print(graf1)

#Test
xv = y_pred
yv = y_test
```

```
plt.figure(figsize = (10,5))
plt.scatter(xv, yv, alpha=1)
plt.title('Test Prediction Regresión Ridge')
plt.xlabel('y_pred')
plt.ylabel('y_test')
graf2 = plt.show()
print(graf2)
```



None



None

5.4. Coeficientes Regresiones

In [49]:
```
features = features.to_list()
```

In [50]:
```
regresion_model.coef_
```

Out[50]:
```
array([ 0.18101171,  0.05633863,  0.1058393 ,  0.09228085,  0.00387894,
        0.04121932, -0.00150896, -0.06364026,  0.03160024])
```

In [51]:
```
ridge_model.coef_
```

Out[51]:
```
array([ 0.10202613,  0.04844503,  0.04157927,  0.09124215,  0.03686264,
        0.02181311,  0.00398818, -0.02954431,  0.01560518])
```

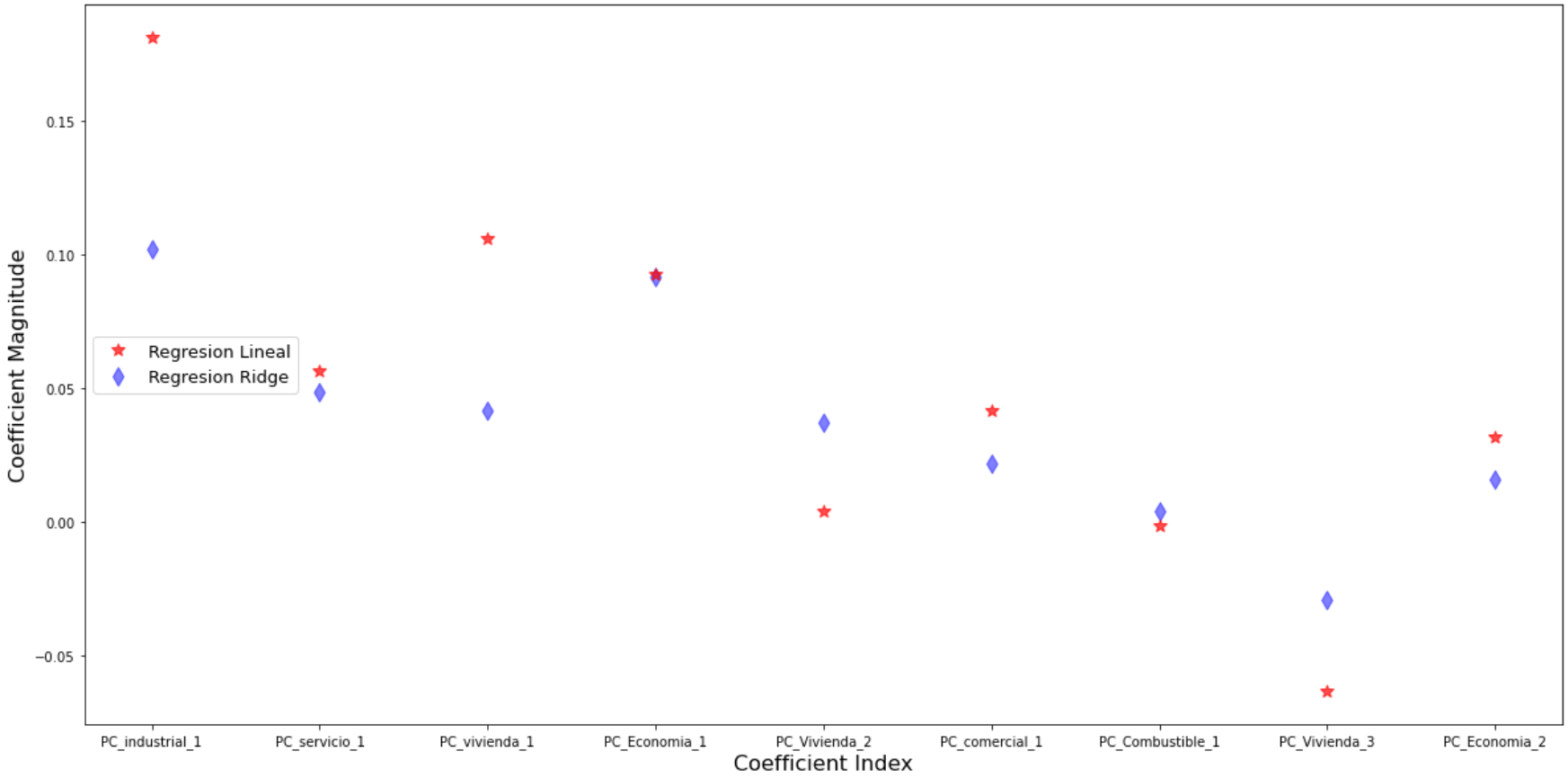
In [52]:
```
df_coeff = pd.DataFrame([features, regresion_model.coef_, ridge_model.coef_])
df_coeff = df_coeff.T
df_coeff.columns = ['Feature','Regresion_Lineal','Regresión_Ridge']
df_coeff
```

Out[52]:

	Feature	Regresion_Lineal	Regresión_Ridge
0	PC_industrial_1	0.181012	0.102026
1	PC_servicio_1	0.0563386	0.048445
2	PC_vivienda_1	0.105839	0.0415793
3	PC_Economia_1	0.0922809	0.0912421
4	PC_Vivienda_2	0.00387894	0.0368626
5	PC_comercial_1	0.0412193	0.0218131
6	PC_Combustible_1	-0.00150896	0.00398818
7	PC_Vivienda_3	-0.0636403	-0.0295443
8	PC_Economia_2	0.0316002	0.0156052

In [53]:

```
plt.figure(figsize=(20,10))
plt.plot(features, regresion_model.coef_,alpha=0.7,linestyle='none',marker='*', markersize=10,color='red',label=r'Regresi
plt.plot(features, ridge_model.coef_,alpha=0.5,linestyle='none',marker='d',markersize=10,color='blue',label=r'Regresion R
plt.xlabel('Coefficient Index',fontsize=16)
plt.ylabel('Coefficient Magnitude',fontsize=16)
plt.legend(fontsize=13,loc='center left')
plt.show()
```



6. Deployment

In [54]:
```
tx_deploy = tx_deploy.set_index('ID')
```

In [55]:
```
tx_deploy = tx_deploy[features]
```

In [56]:
```
#Aplicamos el Modelo
TPD_Ridge=pd.Series(ridge_model.predict(tx_deploy))
TPD_Tree=pd.Series(regressor.predict(tx_deploy))
```

In [57]:
```
TPD=pd.DataFrame(TPD_Ridge)
TPD.columns =['Ridge']
```

```
TPD['Tree'] = TPD_Tree
```

In [58]:

```
k=41871
e=math.e

TPD['Ridge_total']=k/(e**TPD['Ridge']+1)
TPD['Tree_total']=k/(e**TPD['Tree']+1)
```

In [59]:

```
TPD
```

Out[59]:

	Ridge	Tree	Ridge_total	Tree_total
0	-2.001119	0.374924	36884.773220	17056.224109
1	-2.005389	0.374924	36903.494778	17056.224109
2	-2.039358	0.374924	37050.302441	17056.224109
3	-2.016350	0.374924	36951.286197	17056.224109
4	-2.006571	0.374924	36908.671564	17056.224109
5	-2.000975	0.374924	36884.139081	17056.224109
6	-1.995622	0.374924	36860.577849	17056.224109
7	-2.037055	0.374924	37040.468441	17056.224109
8	-1.989826	0.374924	36834.953338	17056.224109
9	-1.967172	0.374924	36733.722406	17056.224109
10	-1.973940	0.374924	36764.149749	17056.224109
11	-2.000487	0.374924	36881.997166	17056.224109
12	-1.993339	0.374924	36850.497854	17056.224109
13	-1.994634	0.374924	36856.217742	17056.224109
14	-1.999287	0.374924	36876.720340	17056.224109
15	-2.020324	0.374924	36968.511917	17056.224109
16	-2.186945	0.374924	37645.001071	17056.224109
17	-2.049483	0.374924	37093.324427	17056.224109

	Ridge	Tree	Ridge_total	Tree_total
18	-2.025284	0.374924	36989.943289	17056.224109
19	-2.084435	0.374924	37239.277104	17056.224109
20	-1.696884	0.374924	35386.328530	17056.224109
21	-2.152286	0.374924	37511.484872	17056.224109
22	-2.006014	0.374924	36906.233543	17056.224109
23	-2.004668	0.374924	36900.339420	17056.224109
24	-2.003130	0.374924	36893.598337	17056.224109

7. COMPARACION REAL- MODELO

In [60]:
```
dc = pd.DataFrame(y_test)
dc['TPD_Ridge'] = y_pred

dc['TPD_REAL']=k/(e**dc['TOTAL_TPD_TRANSFORMADO']+1)
dc['TPD_ESTIMADO']=k/(e**dc['TPD_Ridge']+1)
dc['ERROR']=dc['TPD_ESTIMADO']-dc['TPD_REAL']
dc['%_ERROR'] = (abs(dc['ERROR'])/dc['TPD_ESTIMADO'])*100
dc
```

Out[60]:

ID	TOTAL_TPD_TRANSFORMADO	TPD_Ridge	TPD_REAL	TPD_ESTIMADO	ERROR	%_ERROR
20182	0.429659	0.435570	16505.868940	16446.808833	-59.060106	0.359098
20181	0.506032	0.419577	15748.693347	16606.789766	858.096419	5.167142
20189	0.367809	0.419066	17128.190260	16611.915262	-516.274998	3.107860
20176	0.565555	0.472833	15168.318524	16076.200936	907.882412	5.647369
20162	0.600281	0.582215	14834.034483	15007.533343	173.498860	1.156078
20155	0.842637	0.788369	12602.322581	13085.481938	483.159358	3.692331

Apéndice C: Memoria de cálculo Python para la Serie de Modelos C

SERIE C- MODELO RADIAL COYOL

1. CONFIGURACION GENERAL

1.1. Importamos las Librerias a utilizar

In [1]:

```
# Librerías básicas
import pandas as pd
import numpy as np
import warnings

# Librerias Machine Learnign & Estadísticas
from sklearn import linear_model
from sklearn.tree import DecisionTreeRegressor
from scipy import stats
import statsmodels
import statsmodels.api as sm
import statsmodels.formula.api as smf
import statsmodels.stats.api as sms
from statsmodels.compat import lzip
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import StandardScaler

# Visualización de Datos
import seaborn as sns
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import pylab

# Métricas de Evaluación
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error

# Otras
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
from sklearn.decomposition import PCA
import math
```

1.2. Cargamos el set de datos

In [2]:
```
#Carga inicial del Data Set
data = pd.read_excel('C:/Users/Rodrigo Mata/Desktop/Proyecto Coyol/FINAL/DATASET_FINAL_NATALIA.xls', sheet_name='BASE5')

#Seleccionados ID como index
data =data.set_index('ID')
```

1.3. Cargamos el set de deploy

In [3]:
```
#Cargamos base de referencia
#Load Data Set
datos = pd.read_excel('C:/Users/Rodrigo Mata/Desktop/Proyecto Coyol/FINAL/DATASET_FINAL_NATALIA.xls', sheet_name='APLICAC

#Seleccionados ID como index
datos =datos.set_index('ID')
```

1.3. Limpieza final de los datos

In [4]:
```
#Eliminamos co;lumna YEAR
data=data.drop('YEAR', axis=1)
```

2. Definición de la Variable Dependiente (Target)

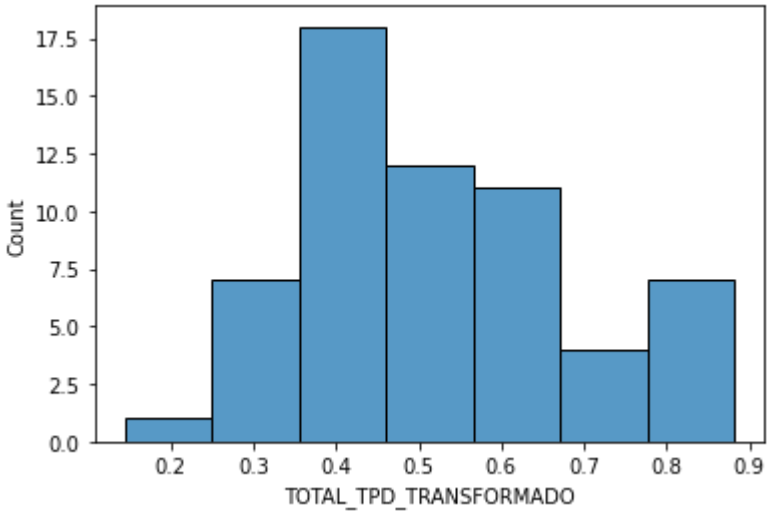
2.1. Seleccion de la Variable Independiente

In [5]:
```
#Definimos el noombre dela columna que contienen la variable independiente
target = 'TOTAL_TPD_TRANSFORMADO'

#Generamos el vector final con la Variable Independiente
y = data[target]
```

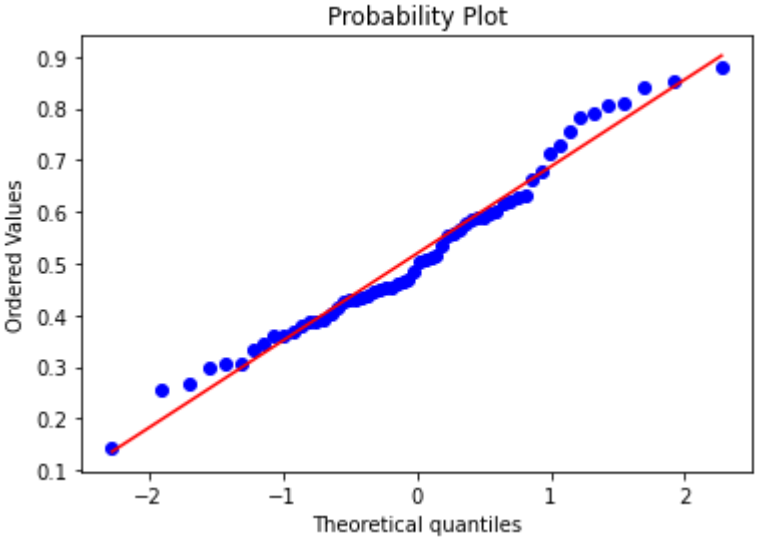
2.2. Normalidad de la Variable Independiente

In [6]:
```
#Metodo 1: Histograma
sns.histplot(data[target])
```

Out[6]: <AxesSubplot:xlabel='TOTAL_TPD_TRANSFORMADO', ylabel='Count'>



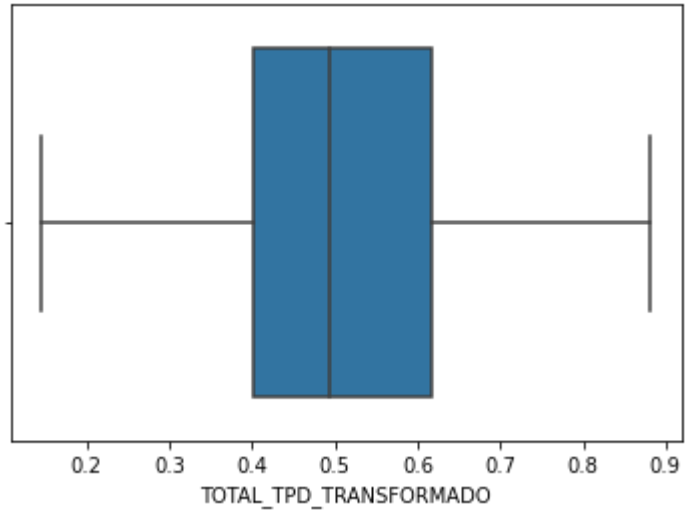
In [7]:
```
#metodo 2: Quantile-Quantile Plot
stats.probplot(data[target], dist='norm', plot=pylab)
pylab.show()
```



In [8]:
```
#Metodo 3: Box Plot
sns.boxplot(data[target])
```

```
C:\Users\Rodrigo Mata\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable a
s a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments wit
hout an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[8]: <AxesSubplot:xlabel='TOTAL_TPD_TRANSFORMADO'>



In [9]:

```
# Prueba de Normalidad de Variable Independiente
# Prueba Shapiro-Wilk
shapiro_test = stats.shapiro(data[target])
shapiro_test

if shapiro_test[1]> 0.05:
    print(shapiro_test)
    print('Normal')

else:
    print(shapiro_test)
    print('No Normal')
```

```
ShapiroResult(statistic=0.9739791750907898, pvalue=0.22742749750614166)
Normal
```

2.3. Filtros de Outliers en Variable Independiente

In [10]:

```
#Definimos el valor de Z para definición de valores extremos
```

```
Z = 3
```

In [11]:
```
#Tamaño de muestra inicial
y.shape
```

Out[11]: (60,)

In [12]:
```
#Eliminamos valores extremos que estén fuera del rango de Z Desviaciones Estandar
upper_lim = y.mean () + y.std () * Z
lower_lim = y.mean () - y.std () * Z

y = y[(y < upper_lim) & (y > lower_lim)]
```

In [13]:
```
#Tamaño de muestra final
y.shape
```

Out[13]: (60,)

3. Definición de Variables Independientes (Features)

3.1. Selección de Variables Independientes

In [14]:
```
#Lista de Variables Independientes a revisar
lista_features = ['DIESEL', 'GAS_SUPER', 'GAS_REGULAR', #variables combustible
          'IPC', 'IMAE', 'TIPO_CAMBIO',  'IMPORTACION_VEHICULOS', #varibales economicas
         'GARITA_VIVIENDA_ACUM', 'SAN_JOSE_VIVIENDA_ACUM', 'SAN_ANTONIO_VIVIENDA_ACUM', 'TURRUCARES_VIVIENDA_ACUM',
          'SAN_JOSE_CONDOMINIO_ACUM', 'SAN_ANTONIO_CONDOMINIO_ACUM', 'GARITA_CONDOMINIO_ACUM', #variables vivienda
         'GARITA_INDUSTRIAL_ACUM', 'SAN_ANTONIO_INDUSTRIAL_ACUM', 'TURRUCARES_INDUSTRIAL_ACUM', #variables industriales
         'GARITA_COMERCIAL_ACUM', 'SAN_ANTONIO_COMERCIAL_ACUM', 'SAN_JOSE_COMERCIAL_ACUM',
          'TURRUCARES_COMERCIAL_ACUM', #variables comerciales
          'GARITA_SERVICIOS_ACUM', 'SAN_ANTONIO_SERVICIOS_ACUM', 'TURRUCARES_SERVICIOS_ACUM',
          'SAN JOSE_SERVICIOS_ACUM', #variables servicios
         'TPD_RN1_TOTAL' #variabales rutas primarias
        ]
```

In [15]:
```
#Definimos data set de variables independientes
X = data[lista_features]
```

In [16]:
```
#Tamaño de muestra inicial
X.shape
```

Out[16]: (60, 26)

3.2. Revisión de Outliers en Variables Independientes

In [17]:
```
#Se crea bucle para revision de outliers en Variables Independientes
for i in lista_features:
    upper_lim = X[i].mean () + X[i].std () * Z
    lower_lim = X[i].mean () - X[i].std () * Z

    X = X[(X[i] < upper_lim) & (X[i] > lower_lim)]
```

In [18]:
```
#Tamaño de Muestra Final
X.shape
```

Out[18]: (58, 26)

4. Ajustes finales del set de datos para modelo

4.1. Data Set Final del Modelo

In [19]:
```
# Se integran ambas Variables (Dependiente e Independientes) para generar el data set final del modelo
df_final = pd.merge(y, X, how='inner', on='ID')
```

In [20]:
```
#Revisión de tamaño final de muestra
df_final.shape
```

Out[20]: (58, 27)

In [21]:
```
#Definimos Data Final Variable Independiente
y =df_final[target]
```

In [22]:
```
#Definimos Data Final Variable Dependiente
X = df_final[lista_features]
```

In [23]:
```
#Split data set en set de entrenamiento (calibración del modelo) y set de prueba (evaluación del modelo)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.10, random_state=0)
```

4.2. Metodologia para la reduccón de dimensiones - Principal Component Analysis

In [24]:
```
#Definimos funcion para serapar features por categorías

def feat(df):
    df_combustible = df[['DIESEL', 'GAS_SUPER', 'GAS_REGULAR']]
    df_economia = df[['IPC', 'IMAE', 'TIPO_CAMBIO']]
    df_vivienda = df[['GARITA_VIVIENDA_ACUM', 'SAN_JOSE_VIVIENDA_ACUM', 'SAN_ANTONIO_VIVIENDA_ACUM', 'TURRUCARES_VIVIENDA
    df_industrial = df[['GARITA_INDUSTRIAL_ACUM', 'SAN_ANTONIO_INDUSTRIAL_ACUM', 'TURRUCARES_INDUSTRIAL_ACUM']]
    df_comercial = df[['GARITA_COMERCIAL_ACUM', 'SAN_ANTONIO_COMERCIAL_ACUM', 'SAN_JOSE_COMERCIAL_ACUM','TURRUCARES_COMER
    df_servicio = df[['GARITA_SERVICIOS_ACUM', 'SAN_ANTONIO_SERVICIOS_ACUM', 'TURRUCARES_SERVICIOS_ACUM', 'SAN JOSE_SERVI
    df_lista = [df_combustible, df_economia, df_vivienda, df_industrial, df_comercial, df_servicio]

    return df_lista
```

In [25]:
```
#Definimos grupos de variables y numero de PC's
var = ['Combustible', 'Economia', 'Vivienda', 'Industrial', 'Comercial', 'Servicio']
n = [1, 2, 3, 1, 2, 2]
```

In [26]:
```
#Aplicamos la función para separar features por categorías
x_subset_train = feat(X_train)
x_subset_test = feat(X_test)
x_subset_deploy = feat(datos)
```

In [27]:
```
# PCA_COMBUSTIBLE
l=0

#Standarizamos la data
scaler = StandardScaler()

# Fit on training set only.
```

```
scaler.fit(x_subset_train[l])

# Apply transform to both the training set and the test set.
x_subset_train[l] = scaler.transform(x_subset_train[l])
x_subset_test[l] = scaler.transform(x_subset_test[l])
x_subset_deploy[l] = scaler.transform(x_subset_deploy[l])


pca = PCA(n_components = n[l])
pca.fit(x_subset_train[l])

x_subset_train[l] = pca.transform(x_subset_train[l])
x_subset_test[l]  = pca.transform(x_subset_test[l])
x_subset_deploy[l]  = pca.transform(x_subset_deploy[l])

explained_variance = pca.explained_variance_ratio_

tx_train = pd.DataFrame(x_subset_train[l])
tx_train['ID'] = X_train.index
tx_train.columns = ['PC_Combustible_1', 'ID']

tx_test = pd.DataFrame(x_subset_test[l])
tx_test['ID'] = X_test.index
tx_test.columns = ['PC_Combustible_1', 'ID']

tx_deploy = pd.DataFrame(x_subset_deploy[l])
tx_deploy['ID'] = datos.index
tx_deploy.columns = ['PC_Combustible_1', 'ID']
```

In [28]:

```
# PCA_ECONOMIA
l=1

#Standarizamos la data
scaler = StandardScaler()

# Fit on training set only.
scaler.fit(x_subset_train[l])

# Apply transform to both the training set and the test set.
x_subset_train[l] = scaler.transform(x_subset_train[l])
x_subset_test[l] = scaler.transform(x_subset_test[l])
x_subset_deploy[l] = scaler.transform(x_subset_deploy[l])

pca = PCA(n_components = n[l])
```

```
pca.fit(x_subset_train[l])

x_subset_train[l] = pca.transform(x_subset_train[l])
x_subset_test[l]  = pca.transform(x_subset_test[l])
x_subset_deploy[l]  = pca.transform(x_subset_deploy[l])

explained_variance = pca.explained_variance_ratio_

tx_train_temp = pd.DataFrame(x_subset_train[l])
tx_train_temp['ID'] = X_train.index
tx_train_temp.columns = ['PC_Economia_1', 'PC_Economia_2', 'ID']
tx_train = pd.merge(tx_train, tx_train_temp, how='left', on='ID')

tx_test_temp = pd.DataFrame(x_subset_test[l])
tx_test_temp['ID'] = X_test.index
tx_test_temp.columns = ['PC_Economia_1', 'PC_Economia_2', 'ID']
tx_test = pd.merge(tx_test, tx_test_temp, how='left', on='ID')

tx_deploy_temp = pd.DataFrame(x_subset_deploy[l])
tx_deploy_temp['ID'] = datos.index
tx_deploy_temp.columns = ['PC_Economia_1', 'PC_Economia_2', 'ID']
tx_deploy = pd.merge(tx_deploy, tx_deploy_temp, how='left', on='ID')
```

In [29]:

```
# PCA_VIVIENDA
l=2

#Standarizamos la data
scaler = StandardScaler()

# Fit on training set only.
scaler.fit(x_subset_train[l])

# Apply transform to both the training set and the test set.
x_subset_train[l] = scaler.transform(x_subset_train[l])
x_subset_test[l] = scaler.transform(x_subset_test[l])
x_subset_deploy[l] = scaler.transform(x_subset_deploy[l])

pca = PCA(n_components = n[l])
pca.fit(x_subset_train[l])

x_subset_train[l] = pca.transform(x_subset_train[l])
x_subset_test[l]  = pca.transform(x_subset_test[l])
x_subset_deploy[l]  = pca.transform(x_subset_deploy[l])
```

```
explained_variance = pca.explained_variance_ratio_

tx_train_temp = pd.DataFrame(x_subset_train[l])
tx_train_temp['ID'] = X_train.index
tx_train_temp.columns = ['PC_vivienda_1', 'PC_Vivienda_2', 'PC_Vivienda_3', 'ID']
tx_train = pd.merge(tx_train, tx_train_temp, how='left', on='ID')

tx_test_temp = pd.DataFrame(x_subset_test[l])
tx_test_temp['ID'] = X_test.index
tx_test_temp.columns = ['PC_vivienda_1', 'PC_Vivienda_2', 'PC_Vivienda_3', 'ID']
tx_test = pd.merge(tx_test, tx_test_temp, how='left', on='ID')

tx_deploy_temp = pd.DataFrame(x_subset_deploy[l])
tx_deploy_temp['ID'] = datos.index
tx_deploy_temp.columns = ['PC_vivienda_1', 'PC_Vivienda_2', 'PC_Vivienda_3', 'ID']
tx_deploy = pd.merge(tx_deploy, tx_deploy_temp, how='left', on='ID')
```

In [30]:

```
# PCA_INDUSTRIAL
l=3

#Standarizamos la data
scaler = StandardScaler()

# Fit on training set only.
scaler.fit(x_subset_train[l])

# Apply transform to both the training set and the test set.
x_subset_train[l] = scaler.transform(x_subset_train[l])
x_subset_test[l] = scaler.transform(x_subset_test[l])
x_subset_deploy[l] = scaler.transform(x_subset_deploy[l])

pca = PCA(n_components = n[l])
pca.fit(x_subset_train[l])

x_subset_train[l] = pca.transform(x_subset_train[l])
x_subset_test[l]  = pca.transform(x_subset_test[l])
x_subset_deploy[l]  = pca.transform(x_subset_deploy[l])


explained_variance = pca.explained_variance_ratio_

tx_train_temp = pd.DataFrame(x_subset_train[l])
tx_train_temp['ID'] = X_train.index
```

```
tx_train_temp.columns = ['PC_industrial_1', 'ID']
tx_train = pd.merge(tx_train, tx_train_temp, how='left', on='ID')

tx_test_temp = pd.DataFrame(x_subset_test[l])
tx_test_temp['ID'] = X_test.index
tx_test_temp.columns = ['PC_industrial_1', 'ID']
tx_test = pd.merge(tx_test, tx_test_temp, how='left', on='ID')

tx_deploy_temp = pd.DataFrame(x_subset_deploy[l])
tx_deploy_temp['ID'] = datos.index
tx_deploy_temp.columns = ['PC_industrial_1', 'ID']
tx_deploy = pd.merge(tx_deploy, tx_deploy_temp, how='left', on='ID')
```

In [31]:

```
# PCA_COMERCIAL
l=4

#Standarizamos la data
scaler = StandardScaler()

# Fit on training set only.
scaler.fit(x_subset_train[l])

# Apply transform to both the training set and the test set.
x_subset_train[l] = scaler.transform(x_subset_train[l])
x_subset_test[l] = scaler.transform(x_subset_test[l])
x_subset_deploy[l] = scaler.transform(x_subset_deploy[l])

pca = PCA(n_components = n[l])
pca.fit(x_subset_train[l])

x_subset_train[l] = pca.transform(x_subset_train[l])
x_subset_test[l]  = pca.transform(x_subset_test[l])
x_subset_deploy[l]  = pca.transform(x_subset_deploy[l])


explained_variance = pca.explained_variance_ratio_

tx_train_temp = pd.DataFrame(x_subset_train[l])
tx_train_temp['ID'] = X_train.index
tx_train_temp.columns = ['PC_comercial_1', 'PC_comercial_2', 'ID']
tx_train = pd.merge(tx_train, tx_train_temp, how='left', on='ID')

tx_test_temp = pd.DataFrame(x_subset_test[l])
tx_test_temp['ID'] = X_test.index
```

```
tx_test_temp.columns = ['PC_comercial_1', 'PC_comercial_2', 'ID']
tx_test = pd.merge(tx_test, tx_test_temp, how='left', on='ID')

tx_deploy_temp = pd.DataFrame(x_subset_deploy[l])
tx_deploy_temp['ID'] = datos.index
tx_deploy_temp.columns = ['PC_comercial_1', 'PC_comercial_2', 'ID']
tx_deploy = pd.merge(tx_deploy, tx_deploy_temp, how='left', on='ID')
```

In [32]:

```
# PCA_SERVICIO
l=5

#Standarizamos la data
scaler = StandardScaler()

# Fit on training set only.
scaler.fit(x_subset_train[l])

# Apply transform to both the training set and the test set.
x_subset_train[l] = scaler.transform(x_subset_train[l])
x_subset_test[l] = scaler.transform(x_subset_test[l])
x_subset_deploy[l] = scaler.transform(x_subset_deploy[l])

pca = PCA(n_components = n[l])
pca.fit(x_subset_train[l])

x_subset_train[l] = pca.transform(x_subset_train[l])
x_subset_test[l]  = pca.transform(x_subset_test[l])
x_subset_deploy[l]  = pca.transform(x_subset_deploy[l])


explained_variance = pca.explained_variance_ratio_

tx_train_temp = pd.DataFrame(x_subset_train[l])
tx_train_temp['ID'] = X_train.index
tx_train_temp.columns = ['PC_servicio_1', 'PC_servicio_2', 'ID']
tx_train = pd.merge(tx_train, tx_train_temp, how='left', on='ID')

tx_test_temp = pd.DataFrame(x_subset_test[l])
tx_test_temp['ID'] = X_test.index
tx_test_temp.columns = ['PC_servicio_1', 'PC_servicio_2', 'ID']
tx_test = pd.merge(tx_test, tx_test_temp, how='left', on='ID')

tx_deploy_temp = pd.DataFrame(x_subset_deploy[l])
tx_deploy_temp['ID'] = datos.index
```

```
tx_deploy_temp.columns = ['PC_servicio_1', 'PC_servicio_2', 'ID']
tx_deploy = pd.merge(tx_deploy, tx_deploy_temp, how='left', on='ID')
```

5. Feauture Selection

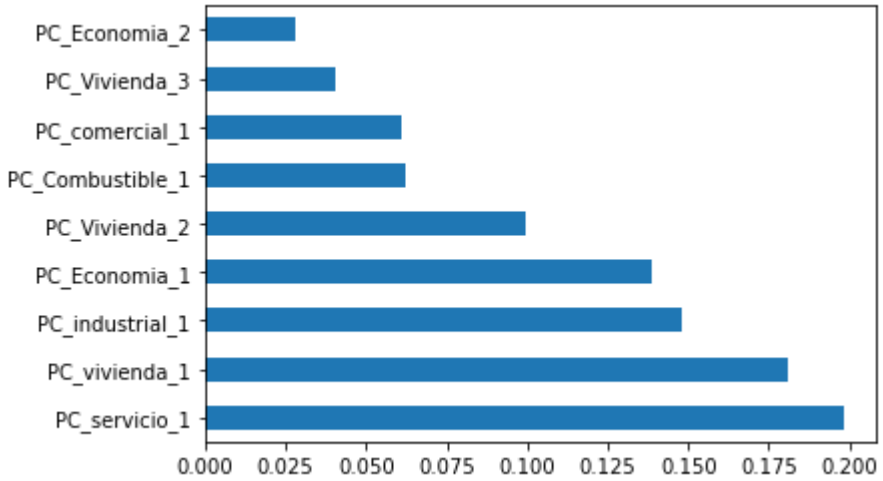
In [33]:
```
#Definition of predictors number to be used.   Aprox 10% of total instances.
n=9
```

In [34]:
```
X_train = tx_train.copy().set_index('ID')
X_test = tx_test.copy().set_index('ID')
```

In [35]:
```
from sklearn.ensemble import ExtraTreesRegressor

model = ExtraTreesRegressor()
model.fit(X_train,y_train)
print(model.feature_importances_) #use inbuilt class feature_importances of tree based classifiers
```

```
[0.0623607  0.13835892 0.02830701 0.18079953 0.09929822 0.04020697
 0.147905   0.06097618 0.02109296 0.19810917 0.02258533]
```

In [36]:
```
#plot graph of feature importances for better visualization
import matplotlib.pyplot as plt
feat_importances = pd.Series(model.feature_importances_, index=X_train.columns)
feat_importances.nlargest(n).plot(kind='barh')
plt.show()
```



In [37]:
```
#Creamos Data Set final de Features
features = pd.Series(feat_importances.nlargest(n).index)

X_train = X_train[features]
X_test  = X_test[features]
```

5. Entrenamiento de los Modelos

- Regresión Lineal Simple
- Arbol de Regresión
- Regresión Ridge

5.1. Modelo de Regresión Lineal Simple

In [38]:
```
# Tunning de Hiperparámetros Regresión Lineal Simple
params_regresion = {
    'fit_intercept' : [True, False],
    'normalize' : [True,False],
    'n_jobs' :[1,-1]
}

regressor = linear_model.LinearRegression()

l_regressor = GridSearchCV(regressor, params_regresion, scoring='r2', cv=5, n_jobs=-1)
```

```
l_regressor.fit(X_train, y_train)
print(f'fit_intercept: {l_regressor.best_params_["fit_intercept"]}')
print(f'normalize: {l_regressor.best_params_["normalize"]}')
print(f'n_jobs: {l_regressor.best_params_["n_jobs"]}')
print(f'Best score: {l_regressor.best_score_}')
```

```
fit_intercept: True
normalize: True
n_jobs: 1
Best score: 0.7032554522398817
```

In [39]:

```
# Aplicación del Modelo de Regresión Lineal Simple selecionado
regresion_model = linear_model.LinearRegression(fit_intercept=l_regressor.best_params_["fit_intercept"],
                    normalize=l_regressor.best_params_["normalize"], n_jobs=l_regressor.best_params_["n_jobs"])
regresion_model.fit(X_train, y_train)
y_train_pred = regresion_model.predict(X_train)
y_pred = regresion_model.predict(X_test)
print('Train r2 score: ', r2_score(y_train_pred, y_train))
print('Test r2 score: ', r2_score(y_test, y_pred))
train_rmse = np.sqrt(mean_squared_error(y_train_pred, y_train))
test_rmse = np.sqrt(mean_squared_error(y_test, y_pred))
print(f'Train RMSE: {train_rmse:.4f}')
print(f'Test RMSE: {test_rmse:.4f}')
```

```
Train r2 score:  0.7550322256349814
Test r2 score:  0.8558493242351031
Train RMSE: 0.0692
Test RMSE: 0.0576
```

In [40]:

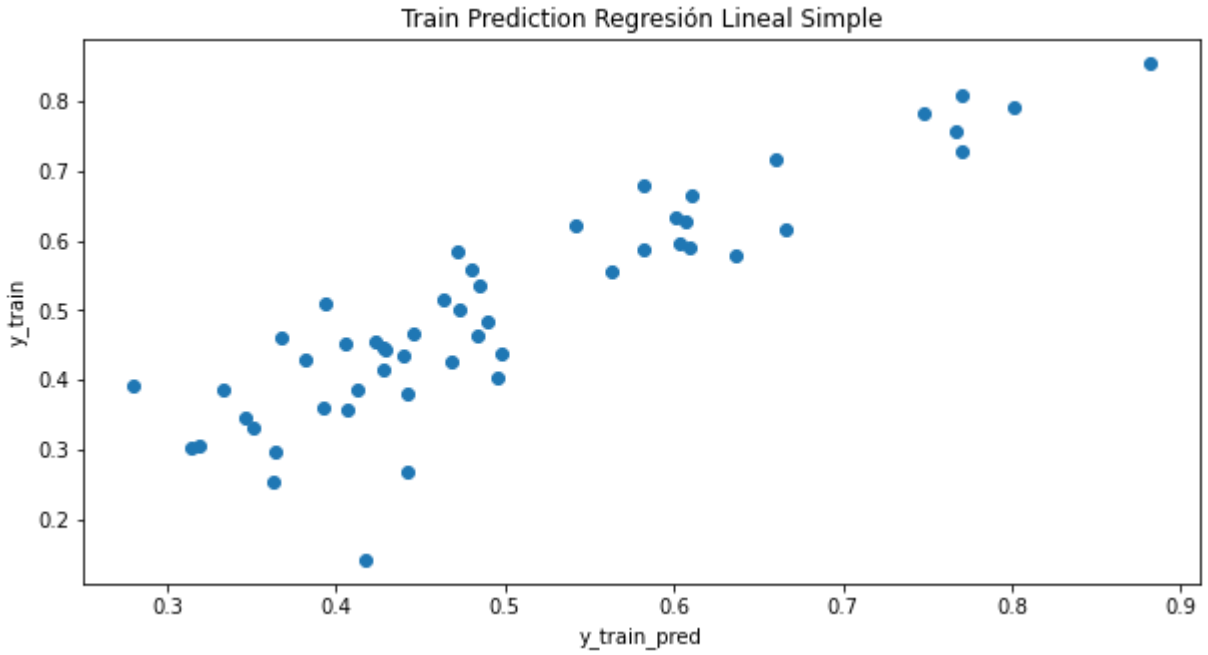
```
#TABLA RESUMEN DE LA REGRESION LINEAL BASE AJUSTADA
from regressors import stats

stats.summary(regresion_model, X_train, y_train)
```

```
Residuals:
    Min      1Q  Median      3Q     Max
-0.1172 -0.0479  0.0048  0.0298  0.2732


Coefficients:
            Estimate  Std. Error  t value   p value
_intercept  0.502538    0.009593  52.3855  0.000000
x1          0.059872    0.035023   1.7095  0.093435
x2          0.105067    0.048481   2.1672  0.034917
x3          0.183920    0.047237   3.8935  0.000288
```

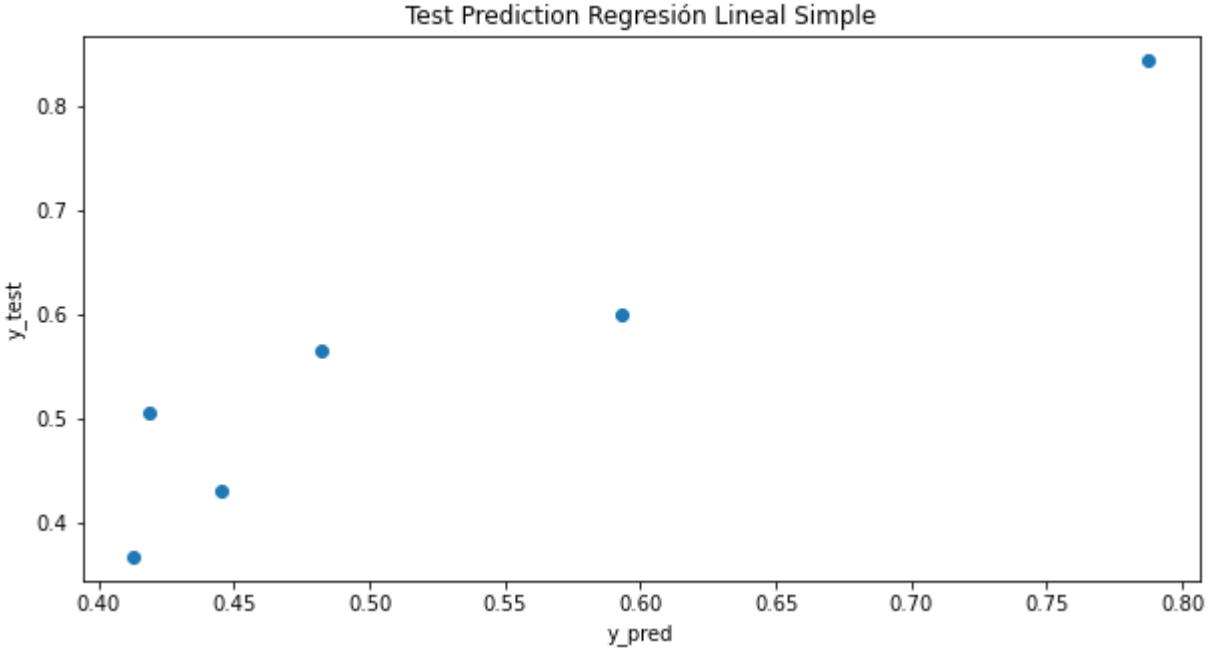
```
x4          0.091012    0.025463   3.5743  0.000778
x5          0.006138    0.022943   0.2675  0.790148
x6         -0.000624    0.008473  -0.0736  0.941596
x7          0.042009    0.036340   1.1560  0.253063
x8         -0.059540    0.039718  -1.4991  0.140018
x9          0.032531    0.022695   1.4334  0.157848
---
R-squared:  0.80323,    Adjusted R-squared:  0.76107
F-statistic: 19.05 on 9 features
```

In [41]:

```
#Visualización Modelo de Regresión Lineal Simple

#Train
xv = y_train_pred
yv = y_train

plt.figure(figsize = (10,5))
plt.scatter(xv, yv, alpha=1)
plt.title('Train Prediction Regresión Lineal Simple')
plt.xlabel('y_train_pred')
plt.ylabel('y_train')
graf1 = plt.show()
print(graf1)

#Test
xv = y_pred
yv = y_test

plt.figure(figsize = (10,5))
plt.scatter(xv, yv, alpha=1)
plt.title('Test Prediction Regresión Lineal Simple')
plt.xlabel('y_pred')
plt.ylabel('y_test')
graf2 = plt.show()
print(graf2)
```


Train Prediction Regresión Lineal Simple
y_train
y_train_pred
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.3
0.4
0.5
0.6
0.7
0.8
0.9

None


Test Prediction Regresión Lineal Simple
y_test
y_pred
0.4
0.5
0.6
0.7
0.8
0.40
0.45
0.50
0.55
0.60
0.65
0.70
0.75
0.80

None

5.2. Modelo Arbol de Regresión

In [42]:
```
# Tunning de Hiperparámetros Arbol de Regresión

params_tree = {
    'max_depth' : [2, 3, 4, 5],
    'min_samples_split' : [ 2, 3, 4],
    'min_samples_leaf' : [1, 2],
}

regressor = DecisionTreeRegressor(random_state = 0)

tree_regressor = GridSearchCV(regressor, params_tree, scoring='r2', cv=5, n_jobs=-1)
tree_regressor.fit(X_train, y_train)
print(f'Optimal max_depth: {tree_regressor.best_params_["max_depth"]:.2f}')
print(f'Optimal min_samples_split: {tree_regressor.best_params_["min_samples_split"]}')
print(f'Optimal min_samples_leaf: {tree_regressor.best_params_["min_samples_leaf"]}')
print(f'Best score: {tree_regressor.best_score_}')
```

```
Optimal max_depth: 5.00
Optimal min_samples_split: 2
Optimal min_samples_leaf: 1
Best score: 0.5758607904813013
```

In [43]:
```
# Aplicación del Modelo de Arbol de Regresión selecionado

regressor = DecisionTreeRegressor(max_depth=tree_regressor.best_params_["max_depth"], min_samples_split=tree_regressor.be
                    min_samples_leaf=tree_regressor.best_params_["min_samples_leaf"])
regressor.fit(X_train, y_train)
y_train_pred = regressor.predict(X_train)
y_pred = regressor.predict(X_test)
print('Train r2 score: ', r2_score(y_train_pred, y_train))
print('Test r2 score: ', r2_score(y_test, y_pred))
train_rmse_tree = np.sqrt(mean_squared_error(y_train_pred, y_train))
test_rmse_tree = np.sqrt(mean_squared_error(y_test, y_pred))
print(f'Train RMSE: {train_rmse_tree:.4f}')
print(f'Test RMSE: {test_rmse_tree:.4f}')
```

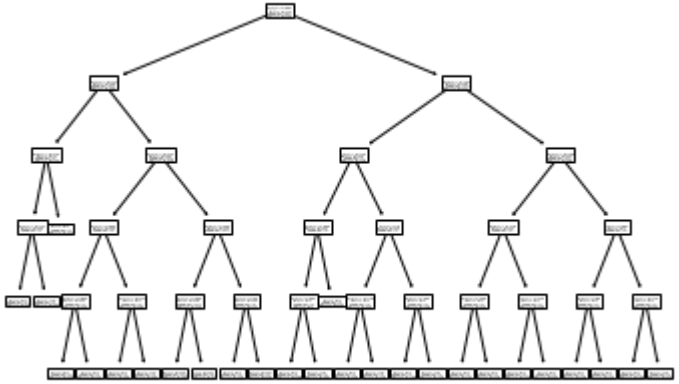
```
Train r2 score:  0.9567197318896029
Test r2 score:  0.4815738355262744
Train RMSE: 0.0318
Test RMSE: 0.1092
```

In [44]:
```
from sklearn import tree
```

```
tree.plot_tree(regressor)
```

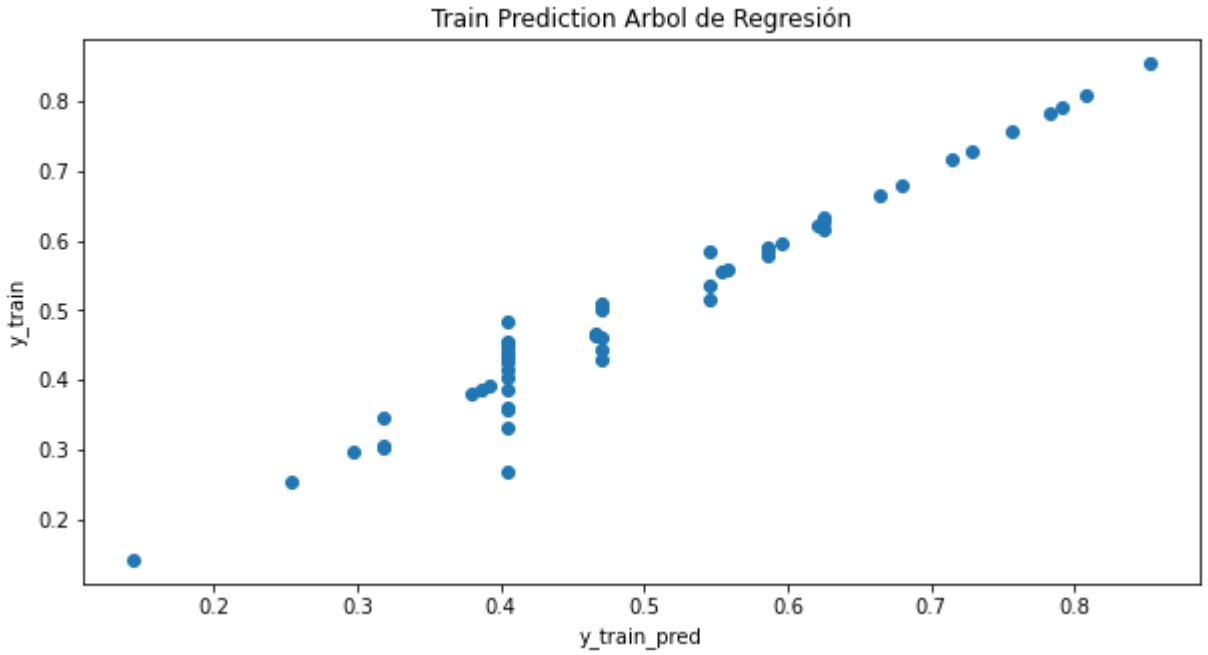
Out[44]:
```
[Text(138.0159574468085, 199.32, 'X[1] <= 0.991\nmse = 0.024\nsamples = 52\nvalue = 0.503'),
 Text(49.86382978723404, 163.07999999999998, 'X[4] <= -0.677\nmse = 0.008\nsamples = 33\nvalue = 0.406'),
 Text(21.370212765957447, 126.83999999999999, 'X[8] <= -0.055\nmse = 0.01\nsamples = 3\nvalue = 0.274'),
 Text(14.246808510638298, 90.6, 'X[4] <= -0.681\nmse = 0.002\nsamples = 2\nvalue = 0.339'),
 Text(7.123404255319149, 54.359999999999985, 'mse = 0.0\nsamples = 1\nvalue = 0.38'),
 Text(21.370212765957447, 54.359999999999985, 'mse = -0.0\nsamples = 1\nvalue = 0.297'),
 Text(28.493617021276595, 90.6, 'mse = 0.0\nsamples = 1\nvalue = 0.144'),
 Text(78.35744680851063, 126.83999999999999, 'X[7] <= -0.316\nmse = 0.006\nsamples = 30\nvalue = 0.419'),
 Text(49.86382978723404, 90.6, 'X[4] <= 0.559\nmse = 0.002\nsamples = 6\nvalue = 0.331'),
 Text(35.61702127659574, 54.359999999999985, 'X[8] <= 0.832\nmse = 0.001\nsamples = 4\nvalue = 0.302'),
 Text(28.493617021276595, 18.119999999999976, 'mse = 0.0\nsamples = 3\nvalue = 0.318'),
 Text(42.740425531914894, 18.119999999999976, 'mse = 0.0\nsamples = 1\nvalue = 0.254'),
 Text(64.11063829787234, 54.359999999999985, 'X[7] <= -0.501\nmse = 0.0\nsamples = 2\nvalue = 0.389'),
 Text(56.98723404255319, 18.119999999999976, 'mse = 0.0\nsamples = 1\nvalue = 0.393'),
 Text(71.23404255319149, 18.119999999999976, 'mse = -0.0\nsamples = 1\nvalue = 0.386'),
 Text(106.85106382978724, 90.6, 'X[8] <= 0.426\nmse = 0.004\nsamples = 24\nvalue = 0.441'),
 Text(92.60425531914893, 54.359999999999985, 'X[7] <= 0.596\nmse = 0.003\nsamples = 19\nvalue = 0.422'),
 Text(85.48085106382979, 18.119999999999976, 'mse = 0.003\nsamples = 14\nvalue = 0.405'),
 Text(99.72765957446808, 18.119999999999976, 'mse = 0.001\nsamples = 5\nvalue = 0.47'),
 Text(121.09787234042552, 54.359999999999985, 'X[7] <= 0.22\nmse = 0.002\nsamples = 5\nvalue = 0.514'),
 Text(113.97446808510638, 18.119999999999976, 'mse = 0.0\nsamples = 2\nvalue = 0.466'),
 Text(128.22127659574468, 18.119999999999976, 'mse = 0.001\nsamples = 3\nvalue = 0.545'),
 Text(226.16808510638296, 163.07999999999998, 'X[6] <= -2.494\nmse = 0.008\nsamples = 19\nvalue = 0.671'),
 Text(174.52340425531915, 126.83999999999999, 'X[6] <= -2.653\nmse = 0.002\nsamples = 7\nvalue = 0.776'),
 Text(156.71489361702126, 90.6, 'X[3] <= 2.356\nmse = 0.001\nsamples = 3\nvalue = 0.817'),
 Text(149.59148936170212, 54.359999999999985, 'X[5] <= 0.051\nmse = 0.0\nsamples = 2\nvalue = 0.8'),
 Text(142.46808510638297, 18.119999999999976, 'mse = 0.0\nsamples = 1\nvalue = 0.808'),
 Text(156.71489361702126, 18.119999999999976, 'mse = -0.0\nsamples = 1\nvalue = 0.791'),
 Text(163.83829787234043, 54.359999999999985, 'mse = 0.0\nsamples = 1\nvalue = 0.852'),
 Text(192.331914893617, 90.6, 'X[7] <= 0.3\nmse = 0.001\nsamples = 4\nvalue = 0.746'),
 Text(178.08510638297872, 54.359999999999985, 'X[6] <= -2.568\nmse = 0.0\nsamples = 2\nvalue = 0.722'),
 Text(170.96170212765958, 18.119999999999976, 'mse = 0.0\nsamples = 1\nvalue = 0.728'),
 Text(185.20851063829787, 18.119999999999976, 'mse = -0.0\nsamples = 1\nvalue = 0.715'),
 Text(206.57872340425533, 54.359999999999985, 'X[0] <= 2.497\nmse = 0.0\nsamples = 2\nvalue = 0.769'),
 Text(199.45531914893616, 18.119999999999976, 'mse = 0.0\nsamples = 1\nvalue = 0.783'),
 Text(213.70212765957447, 18.119999999999976, 'mse = -0.0\nsamples = 1\nvalue = 0.756'),
 Text(277.8127659574468, 126.83999999999999, 'X[4] <= -0.688\nmse = 0.001\nsamples = 12\nvalue = 0.609'),
 Text(249.3191489361702, 90.6, 'X[8] <= -0.418\nmse = 0.001\nsamples = 6\nvalue = 0.636'),
 Text(235.0723404255319, 54.359999999999985, 'X[5] <= 3.228\nmse = 0.0\nsamples = 2\nvalue = 0.672'),
 Text(227.94893617021276, 18.119999999999976, 'mse = 0.0\nsamples = 1\nvalue = 0.664'),
 Text(242.19574468085105, 18.119999999999976, 'mse = 0.0\nsamples = 1\nvalue = 0.679'),
 Text(263.5659574468085, 54.359999999999985, 'X[2] <= -1.603\nmse = 0.0\nsamples = 4\nvalue = 0.618'),
 Text(256.44255319148937, 18.119999999999976, 'mse = 0.0\nsamples = 3\nvalue = 0.625'),
 Text(270.68936170212766, 18.119999999999976, 'mse = 0.0\nsamples = 1\nvalue = 0.596'),
 Text(306.3063829787234, 90.6, 'X[5] <= 0.907\nmse = 0.0\nsamples = 6\nvalue = 0.582'),
 Text(292.0595744680851, 54.359999999999985, 'X[3] <= 0.89\nmse = 0.0\nsamples = 2\nvalue = 0.557'),
```

```
Text(284.93617021276594, 18.119999999999976, 'mse = 0.0\nsamples = 1\nvalue = 0.558'),
Text(299.18297872340423, 18.119999999999976, 'mse = 0.0\nsamples = 1\nvalue = 0.555'),
Text(320.5531914893617, 54.359999999999985, 'X[7] <= -0.266\nmse = 0.0\nsamples = 4\nvalue = 0.594'),
Text(313.4297872340425, 18.119999999999976, 'mse = 0.0\nsamples = 3\nvalue = 0.586'),
Text(327.67659574468087, 18.119999999999976, 'mse = -0.0\nsamples = 1\nvalue = 0.621')]
```

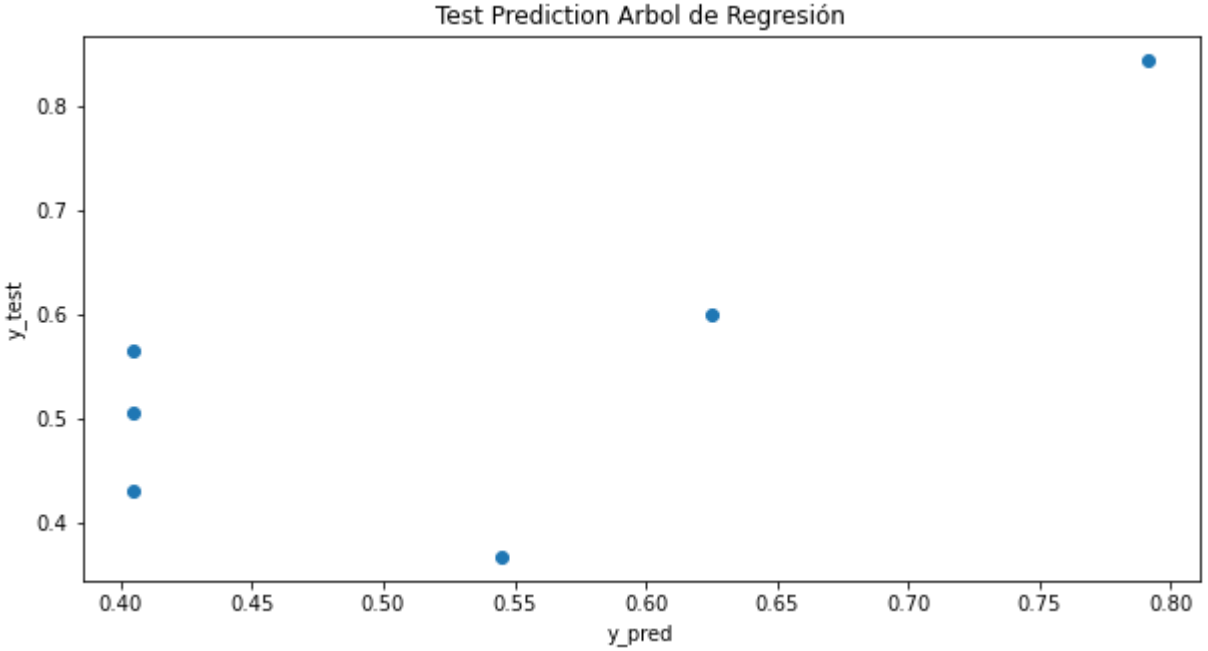


In [45]:

```
#Visualización Modelo Arbol de Regresión

#Train
xv = y_train_pred
yv = y_train

plt.figure(figsize = (10,5))
plt.scatter(xv, yv, alpha=1)
plt.title('Train Prediction Arbol de Regresión')
plt.xlabel('y_train_pred')
plt.ylabel('y_train')
graf1 = plt.show()
print(graf1)

#Test
xv = y_pred
yv = y_test

plt.figure(figsize = (10,5))
plt.scatter(xv, yv, alpha=1)
plt.title('Test Prediction Arbol de Regresión')
plt.xlabel('y_pred')
plt.ylabel('y_test')
graf2 = plt.show()
print(graf2)
```


Train Prediction Arbol de Regresión
y_train
y_train_pred
0.2
0.3
0.4
0.5
0.6
0.7
0.8

None


Test Prediction Arbol de Regresión
y_test
y_pred
0.40
0.45
0.50
0.55
0.60
0.65
0.70
0.75
0.80

None

5.3. Modelo Regresión Ridge

In [46]:
```
# Tunning de Hiperparámetros Regresión Ridge
from sklearn.linear_model import Ridge

params_ridge = {
    'alpha' : [0.1, 0.4, 0.5, .7, 0.75, 0.8, 0.85,.9, .99, 1, 5, 10, 20, 30],
    'fit_intercept' : [True, False],
    'normalize' : [True,False],
    'solver' : ['svd', 'cholesky', 'lsqr', 'sparse_cg', 'sag', 'saga']
}

ridge_model = Ridge()
ridge_regressor = GridSearchCV(ridge_model, params_ridge, scoring='r2', cv=5, n_jobs=-1)
ridge_regressor.fit(X_train, y_train)
print(f'Optimal alpha: {ridge_regressor.best_params_["alpha"]:.2f}')
print(f'Optimal fit_intercept: {ridge_regressor.best_params_["fit_intercept"]}')
print(f'Optimal normalize: {ridge_regressor.best_params_["normalize"]}')
print(f'Optimal solver: {ridge_regressor.best_params_["solver"]}')
print(f'Best score: {ridge_regressor.best_score_}')
```

```
Optimal alpha: 1.00
Optimal fit_intercept: True
Optimal normalize: False
Optimal solver: sparse_cg
Best score: 0.726207748424357
```

In [47]:
```
# Aplicación del Modelo Regresión Ridge

ridge_model = Ridge(alpha=ridge_regressor.best_params_["alpha"], fit_intercept=ridge_regressor.best_params_["fit_intercep
                    normalize=ridge_regressor.best_params_["normalize"], solver=ridge_regressor.best_params_["solver"])
ridge_model.fit(X_train, y_train)
y_train_pred = ridge_model.predict(X_train)
y_pred = ridge_model.predict(X_test)
print('Train r2 score: ', r2_score(y_train_pred, y_train))
print('Test r2 score: ', r2_score(y_test, y_pred))
train_rmse = np.sqrt(mean_squared_error(y_train_pred, y_train))
test_rmse = np.sqrt(mean_squared_error(y_test, y_pred))
print(f'Train RMSE: {train_rmse:.4f}')
print(f'Test RMSE: {test_rmse:.4f}')
```

```
Train r2 score:  0.7266348816094573
Test r2 score:  0.8395696389882017
```

```
Train RMSE: 0.0708
Test RMSE: 0.0607
```

In [48]:

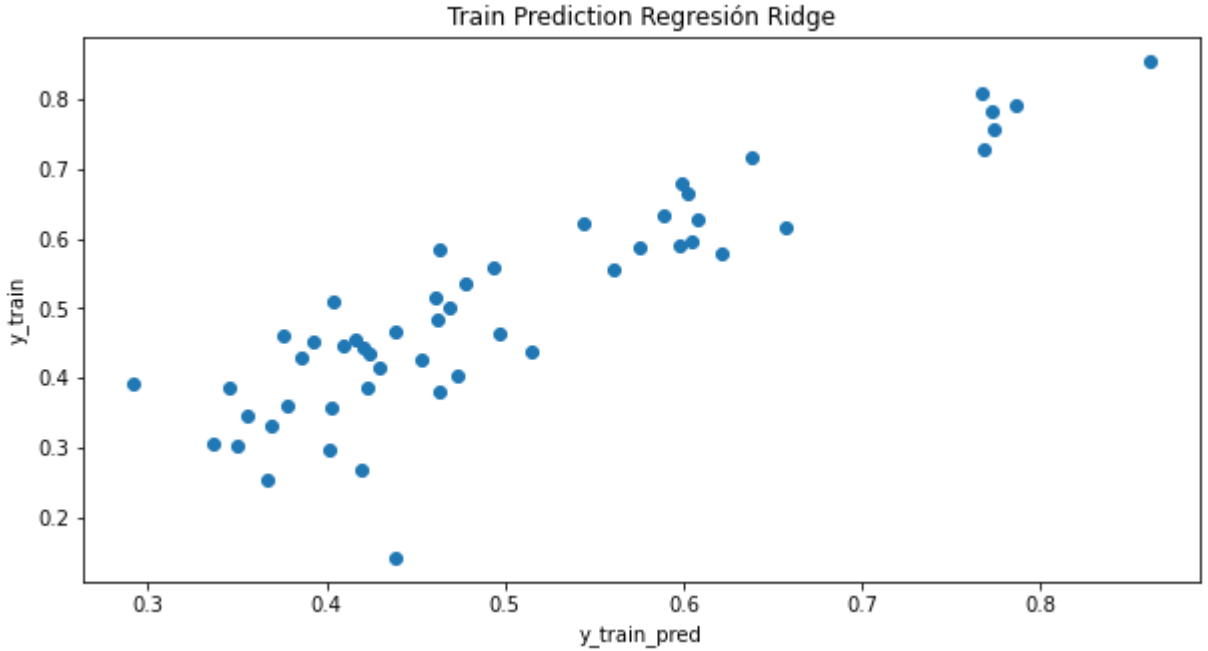
```
#TABLA DE REGRESION RIDGE AJUSTADA
from regressors import stats

stats.summary(ridge_model, X_train, y_train)
```

```
Residuals:
    Min      1Q  Median      3Q     Max
-0.1218 -0.0439 -0.0077  0.0362  0.294


Coefficients:
            Estimate  Std. Error  t value   p value
_intercept  0.502538    0.009813  51.2106  0.000000
x1          0.049840    0.035827   1.3912  0.170219
x2          0.040345    0.049594   0.8135  0.419709
x3          0.102675    0.048321   2.1248  0.038466
x4          0.090676    0.026047   3.4813  0.001032
x5          0.041279    0.023470   1.7588  0.084609
x6          0.004820    0.008668   0.5561  0.580574
x7          0.021909    0.037173   0.5894  0.558204
x8         -0.022641    0.040629  -0.5573  0.579782
x9          0.016332    0.023216   0.7035  0.484938
---
R-squared:  0.79410,    Adjusted R-squared:  0.74998
F-statistic: 18.00 on 9 features
```

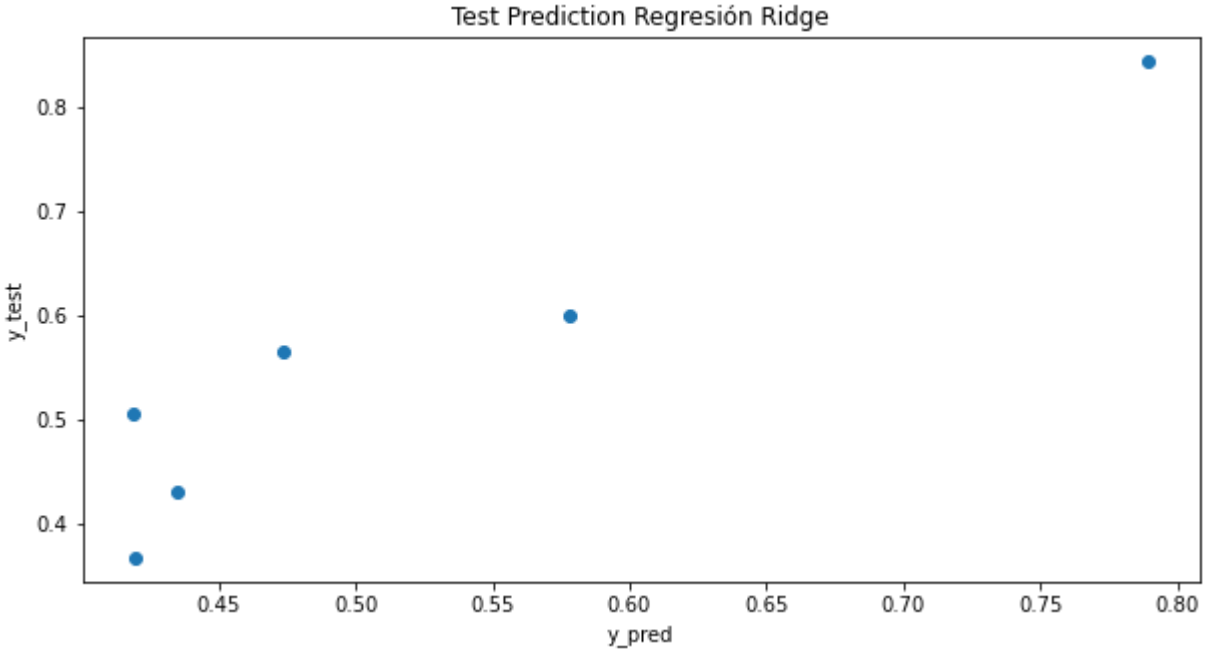
In [49]:

```
#Visualización Modelo Regresión Ridge

#Train
xv = y_train_pred
yv = y_train

plt.figure(figsize = (10,5))
plt.scatter(xv, yv, alpha=1)
plt.title('Train Prediction Regresión Ridge')
plt.xlabel('y_train_pred')
plt.ylabel('y_train')
graf1 = plt.show()
print(graf1)

#Test
xv = y_pred
yv = y_test
```

```
plt.figure(figsize = (10,5))
plt.scatter(xv, yv, alpha=1)
plt.title('Test Prediction Regresión Ridge')
plt.xlabel('y_pred')
plt.ylabel('y_test')
graf2 = plt.show()
print(graf2)
```



None



None

5.4. Coeficientes Regresiones

In [50]:
```
features = features.to_list()
```

In [51]:
```
features
```

Out[51]:
```
['PC_servicio_1',
 'PC_vivienda_1',
 'PC_industrial_1',
 'PC_Economia_1',
 'PC_Vivienda_2',
 'PC_Combustible_1',
 'PC_comercial_1',
 'PC_Vivienda_3',
 'PC_Economia_2']
```

In [52]:
```
regresion_model.coef_
```

Out[52]:
```
array([ 0.05987217,  0.10506664,  0.18392015,  0.09101237,  0.00613779,
```

```
       -0.00062387,  0.04200899, -0.05954009,  0.03253064])
```

In [53]:
```
ridge_model.coef_
```

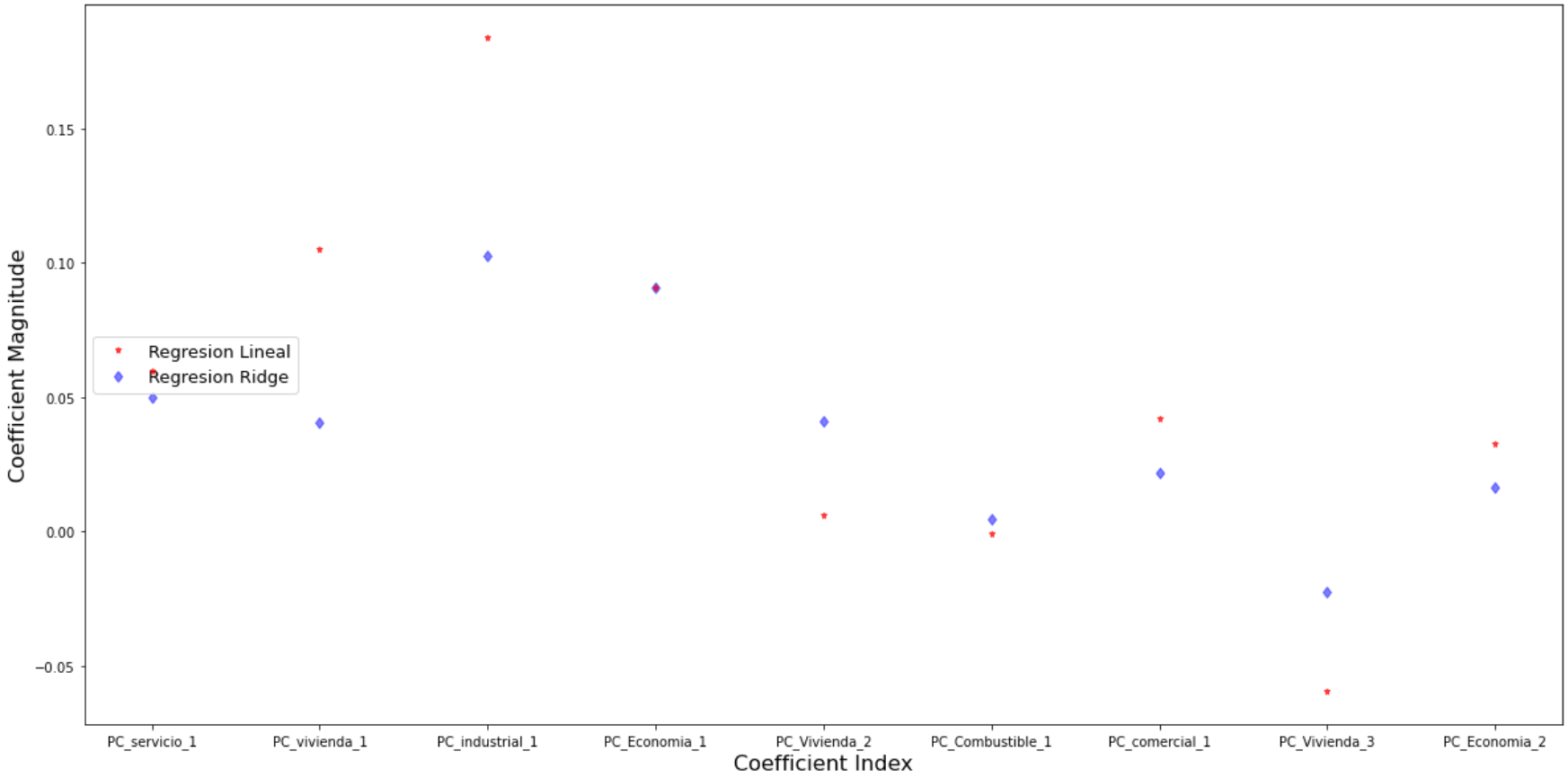
Out[53]:
```
array([ 0.04984009,  0.04034478,  0.10267465,  0.09067571,  0.04127881,
        0.0048202 ,  0.02190943, -0.02264124,  0.01633233])
```

In [54]:
```
df_coeff = pd.DataFrame([features, regresion_model.coef_, ridge_model.coef_])
df_coeff = df_coeff.T
df_coeff.columns = ['Feature','Regresion_Lineal','Regresión_Ridge']
df_coeff
```

Out[54]:

	Feature	Regresion_Lineal	Regresión_Ridge
0	PC_servicio_1	0.0598722	0.0498401
1	PC_vivienda_1	0.105067	0.0403448
2	PC_industrial_1	0.18392	0.102675
3	PC_Economia_1	0.0910124	0.0906757
4	PC_Vivienda_2	0.00613779	0.0412788
5	PC_Combustible_1	-0.000623871	0.0048202
6	PC_comercial_1	0.042009	0.0219094
7	PC_Vivienda_3	-0.0595401	-0.0226412
8	PC_Economia_2	0.0325306	0.0163323

In [55]:
```
plt.figure(figsize=(20,10))
plt.plot(features, regresion_model.coef_,alpha=0.7,linestyle='none',marker='*', markersize=5,color='red',label=r'Regresic
plt.plot(features, ridge_model.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,color='blue',label=r'Regresion Ri
plt.xlabel('Coefficient Index',fontsize=16)
plt.ylabel('Coefficient Magnitude',fontsize=16)
plt.legend(fontsize=13,loc='center left')
plt.show()
```



6. Deployment

In [56]:
```
tx_deploy = tx_deploy.set_index('ID')
```

In [57]:
```
tx_deploy = tx_deploy[features]
```

In [58]:
```
#Aplicamos el Modelo
TPD_Ridge=pd.Series(ridge_model.predict(tx_deploy))
TPD_Tree=pd.Series(regressor.predict(tx_deploy))
```

In [59]:
```
TPD=pd.DataFrame(TPD_Ridge)
TPD.columns =['Ridge']
```

```
TPD['Tree'] = TPD_Tree
```

In [60]:

```
k=41871
e=math.e

TPD['Ridge_total']=k/(e**TPD['Ridge']+1)
TPD['Tree_total']=k/(e**TPD['Tree']+1)
```

In [61]:

```
TPD
```

Out[61]:

	Ridge	Tree	Ridge_total	Tree_total
0	0.402779	0.404627	16775.398326	16756.823449
1	0.398601	0.404627	16817.427795	16756.823449
2	0.397344	0.404627	16830.074385	16756.823449
3	0.401349	0.404627	16789.779846	16756.823449
4	0.398577	0.404627	16817.668849	16756.823449
5	0.402919	0.404627	16773.994155	16756.823449
6	0.403586	0.404627	16767.287489	16756.823449
7	0.398411	0.404627	16819.331732	16756.823449
8	0.415537	0.404627	16647.292181	16756.823449
9	0.408147	0.404627	16721.452227	16756.823449
10	0.426414	0.404627	16538.333003	16756.823449
11	0.403490	0.404627	16768.255943	16756.823449
12	0.404016	0.404627	16762.963367	16756.823449
13	0.403474	0.404627	16768.416146	16756.823449
14	0.404359	0.404627	16759.512967	16756.823449
15	0.380474	0.404627	17000.157218	16756.823449
16	0.372477	0.404627	17080.965499	16756.823449
17	0.397448	0.404627	16829.025514	16756.823449

	Ridge	Tree	Ridge_total	Tree_total
18	0.381397	0.404627	16990.834968	16756.823449
19	0.319969	0.404627	17614.426554	16756.823449
20	0.707903	0.466237	13820.041652	16141.575139
21	0.254696	0.466237	18283.726932	16141.575139
22	0.396863	0.404627	16834.915344	16756.823449
23	0.398490	0.404627	16818.539816	16756.823449
24	0.400349	0.404627	16799.838189	16756.823449

In [62]:
```
# Aplicación del Modelo Regresión Ridge

ridge_model = Ridge(alpha=ridge_regressor.best_params_["alpha"], fit_intercept=ridge_regressor.best_params_["fit_intercep
                    normalize=ridge_regressor.best_params_["normalize"], solver=ridge_regressor.best_params_["solver"])
ridge_model.fit(X_train, y_train)
y_train_pred = ridge_model.predict(X_train)
y_pred = ridge_model.predict(X_test)
print('Train r2 score: ', r2_score(y_train_pred, y_train))
print('Test r2 score: ', r2_score(y_test, y_pred))
train_rmse = np.sqrt(mean_squared_error(y_train_pred, y_train))
test_rmse = np.sqrt(mean_squared_error(y_test, y_pred))
print(f'Train RMSE: {train_rmse:.4f}')
print(f'Test RMSE: {test_rmse:.4f}')
```

```
Train r2 score:  0.7266348816094573
Test r2 score:  0.8395696389882017
Train RMSE: 0.0708
Test RMSE: 0.0607
```

In [63]:
```
y_test
```

Out[63]:
```
ID
20182    0.429659
20181    0.506032
20189    0.367809
20176    0.565555
20162    0.600281
20155    0.842637
Name: TOTAL_TPD_TRANSFORMADO, dtype: float64
```

In [64]:
```
y_pred
```

Out[64]:
```
array([0.43474418, 0.41873499, 0.4197227 , 0.473483  , 0.57828197,
       0.78939141])
```

COMPARACION REAL_MODELO

In [65]:
```
dc = pd.DataFrame(y_test)
dc['TPD_Ridge'] = y_pred

dc['TPD_REAL']=k/(e**dc['TOTAL_TPD_TRANSFORMADO']+1)
dc['TPD_ESTIMADO']=k/(e**dc['TPD_Ridge']+1)
dc['ERROR']=dc['TPD_ESTIMADO']-dc['TPD_REAL']
dc['%_ERROR'] = (abs(dc['ERROR'])/dc['TPD_ESTIMADO'])*100
dc
```

Out[65]:

ID	TOTAL_TPD_TRANSFORMADO	TPD_Ridge	TPD_REAL	TPD_ESTIMADO	ERROR	%_ERROR
20182	0.429659	0.434744	16505.868940	16455.053257	-50.815683	0.308815
20181	0.506032	0.418735	15748.693347	16615.229116	866.535770	5.215310
20189	0.367809	0.419723	17128.190260	16605.331257	-522.859003	3.148742
20176	0.565555	0.473483	15168.318524	16069.763029	901.444505	5.609569
20162	0.600281	0.578282	14834.034483	15045.422979	211.388496	1.405002
20155	0.842637	0.789391	12602.322581	13076.286494	473.963913	3.624606

Apéndice D: Memoria de cálculo Python para la Serie de Modelos D

SERIE D - MODELO RADIAL COYOL

1. CONFIGURACION GENERAL

1.1. Importamos las Librerias a utilizar

In [1]:

```
# Librerías básicas
import pandas as pd
import numpy as np
import warnings

# Librerias Machine Learnign & Estadísticas
from sklearn import linear_model
from sklearn.tree import DecisionTreeRegressor
from scipy import stats
import statsmodels
import statsmodels.api as sm
import statsmodels.formula.api as smf
import statsmodels.stats.api as sms
from statsmodels.compat import lzip
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import StandardScaler

# Visualización de Datos
import seaborn as sns
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import pylab

# Métricas de Evaluación
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error

# Otras
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
from sklearn.decomposition import PCA
import math
```

In [2]:
```
ts = 0.1

features_simples = pd.Series(['PC_Economia_1','PC_servicio_1','PC_vivienda_1','PC_industrial_1'])
```

1.2. Cargamos el set de datos

In [3]:
```
#Carga inicial del Data Set
data = pd.read_excel('C:/Users/Rodrigo Mata/Desktop/Proyecto Coyol/FINAL/DATASET_FINAL_NATALIA.xls', sheet_name='BASE5')

#Seleccionados ID como index
data =data.set_index('ID')
```

1.3. Cargamos el set de deploy

In [4]:
```
#Cargamos base de referencia
#Load Data Set
datos = pd.read_excel('C:/Users/Rodrigo Mata/Desktop/Proyecto Coyol/FINAL/DATASET_FINAL_NATALIA.xls', sheet_name='APLICAC

#Seleccionados ID como index
datos =datos.set_index('ID')
```

1.3. Limpieza final de los datos

In [5]:
```
#Eliminamos co;lumna YEAR
data=data.drop('YEAR', axis=1)
```

2. Definición de la Variable Dependiente (Target)

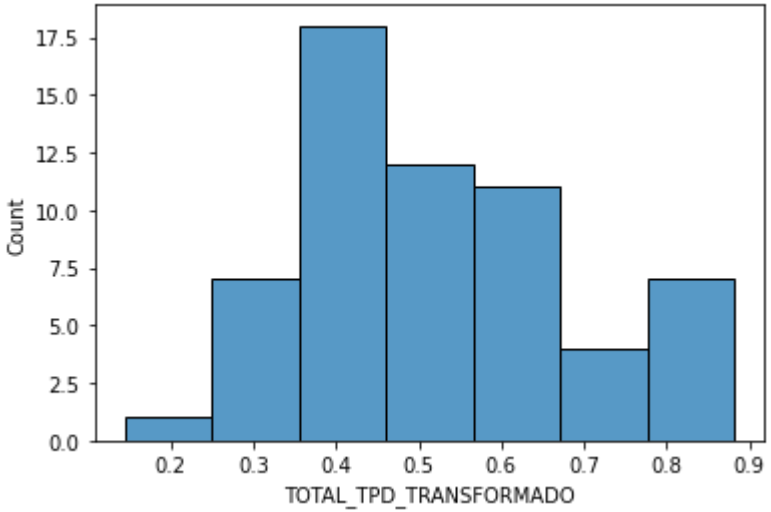
2.1. Seleccion de la Variable Independiente

In [6]:
```
#Definimos el noombre dela columna que contienen la variable independiente
target = 'TOTAL_TPD_TRANSFORMADO'

#Generamos el vector final con la Variable Independiente
y = data[target]
```

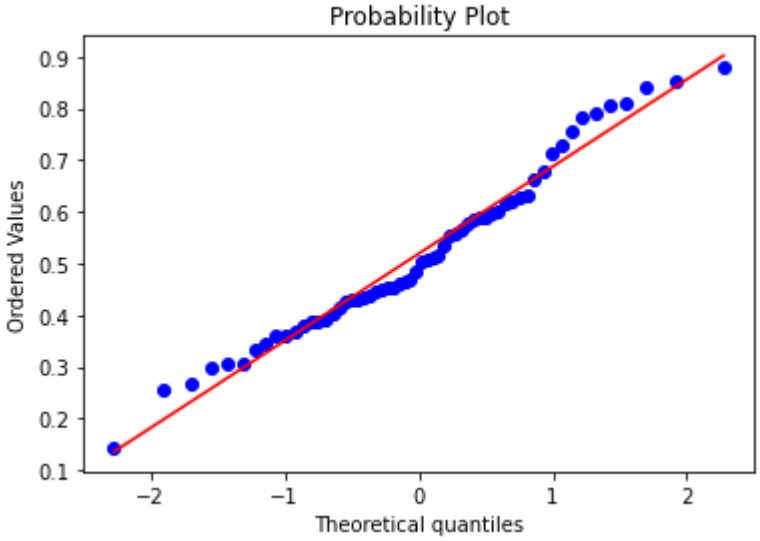
2.2. Normalidad de la Variable Independiente

In [7]:
```
#Metodo 1: Histograma
sns.histplot(data[target])
```

Out[7]: <AxesSubplot:xlabel='TOTAL_TPD_TRANSFORMADO', ylabel='Count'>



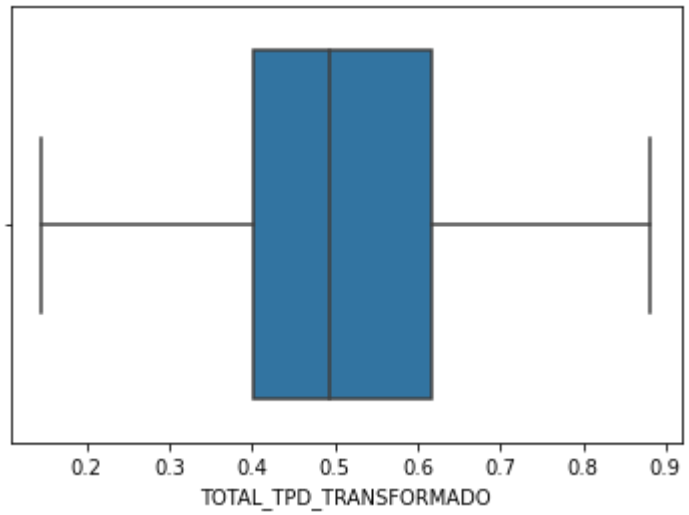
In [8]:
```
#metodo 2: Quantile-Quantile Plot
stats.probplot(data[target], dist='norm', plot=pylab)
pylab.show()
```



In [9]:
```
#Metodo 3: Box Plot
sns.boxplot(data[target])
```

```
C:\Users\Rodrigo Mata\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable a
s a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments wit
hout an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[9]: <AxesSubplot:xlabel='TOTAL_TPD_TRANSFORMADO'>



In [10]:
```
# Prueba de Normalidad de Variable Independiente
# Prueba Shapiro-Wilk
shapiro_test = stats.shapiro(data[target])
shapiro_test

if shapiro_test[1]> 0.05:
    print(shapiro_test)
    print('Normal')

else:
    print(shapiro_test)
    print('No Normal')
```

```
ShapiroResult(statistic=0.9739791750907898, pvalue=0.22742749750614166)
Normal
```

2.3. Filtros de Outliers en Variable Independiente

In [11]:
```
#Definimos el valor de Z para definición de valores extremos
Z = 3
```

In [12]:
```
#Tamaño de muestra inicial
y.shape
```

Out[12]: (60,)

In [13]:
```
#Eliminamos valores extremos que estén fuera del rango de Z Desviaciones Estandar
upper_lim = y.mean () + y.std () * Z
lower_lim = y.mean () - y.std () * Z

y = y[(y < upper_lim) & (y > lower_lim)]
```

In [14]:
```
#Tamaño de muestra final
y.shape
```

Out[14]: (60,)

3. Definición de Variables Independientes (Features)

3.1. Selección de Variables Independientes

In [15]:
```
#Lista de Variables Independientes a revisar
lista_features = ['DIESEL', 'GAS_SUPER', 'GAS_REGULAR', #variables combustible
          'IPC', 'IMAE', 'TIPO_CAMBIO',  'IMPORTACION_VEHICULOS', #varibales economicas
         'GARITA_VIVIENDA_ACUM', 'SAN_JOSE_VIVIENDA_ACUM', 'SAN_ANTONIO_VIVIENDA_ACUM', 'TURRUCARES_VIVIENDA_ACUM',
          'SAN_JOSE_CONDOMINIO_ACUM', 'SAN_ANTONIO_CONDOMINIO_ACUM', 'GARITA_CONDOMINIO_ACUM', #variables vivienda
         'GARITA_INDUSTRIAL_ACUM', 'SAN_ANTONIO_INDUSTRIAL_ACUM', 'TURRUCARES_INDUSTRIAL_ACUM', #variables industriales
         'GARITA_COMERCIAL_ACUM', 'SAN_ANTONIO_COMERCIAL_ACUM', 'SAN_JOSE_COMERCIAL_ACUM',
          'TURRUCARES_COMERCIAL_ACUM', #variables comerciales
          'GARITA_SERVICIOS_ACUM', 'SAN_ANTONIO_SERVICIOS_ACUM', 'TURRUCARES_SERVICIOS_ACUM',
          'SAN JOSE_SERVICIOS_ACUM', #variables servicios
         'TPD_RN1_TOTAL' #variabales rutas primarias
        ]
```

In [16]:
```
#Definimos data set de variables independientes
X = data[lista_features]
```

In [17]:
```
#Tamaño de muestra inicial
X.shape
```

Out[17]: (60, 26)

3.2. Revisión de Outliers en Variables Independientes

In [18]:
```
#Se crea bucle para revision de outliers en Variables Independientes
for i in lista_features:
    upper_lim = X[i].mean () + X[i].std () * Z
    lower_lim = X[i].mean () - X[i].std () * Z

    X = X[(X[i] < upper_lim) & (X[i] > lower_lim)]
```

In [19]:
```
#Tamaño de Muestra Final
X.shape
```

Out[19]: (58, 26)

4. Ajustes finales del set de datos para modelo

4.1. Data Set Final del Modelo

In [20]:
```
# Se integran ambas Variables (Dependiente e Independientes) para generar el data set final del modelo
df_final = pd.merge(y, X, how='inner', on='ID')
```

In [21]:
```
#Revisión de tamaño final de muestra
df_final.shape
```

Out[21]: (58, 27)

In [22]:
```
#Definimos Data Final Variable Independiente
y =df_final[target]
```

In [23]:
```
#Definimos Data Final Variable Dependiente
X = df_final[lista_features]
```

In [24]:
```
#Split data set en set de entrenamiento (calibración del modelo) y set de prueba (evaluación del modelo)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=ts, random_state=0)
```

4.2. Metodologia para la reduccón de dimensiones - Principal Component Analysis

In [25]:
```
#Definimos funcion para serapar features por categorías

def feat(df):
    df_combustible = df[['DIESEL', 'GAS_SUPER', 'GAS_REGULAR']]
    df_economia = df[['IPC', 'IMAE', 'TIPO_CAMBIO']]
    df_vivienda = df[['GARITA_VIVIENDA_ACUM', 'SAN_JOSE_VIVIENDA_ACUM', 'SAN_ANTONIO_VIVIENDA_ACUM', 'TURRUCARES_VIVIENDA
    df_industrial = df[['GARITA_INDUSTRIAL_ACUM', 'SAN_ANTONIO_INDUSTRIAL_ACUM', 'TURRUCARES_INDUSTRIAL_ACUM']]
    df_comercial = df[['GARITA_COMERCIAL_ACUM', 'SAN_ANTONIO_COMERCIAL_ACUM', 'SAN_JOSE_COMERCIAL_ACUM','TURRUCARES_COMER
    df_servicio = df[['GARITA_SERVICIOS_ACUM', 'SAN_ANTONIO_SERVICIOS_ACUM', 'TURRUCARES_SERVICIOS_ACUM', 'SAN JOSE_SERVI
    df_lista = [df_combustible, df_economia, df_vivienda, df_industrial, df_comercial, df_servicio]

    return df_lista
```

In [26]:
```
#Definimos grupos de variables y numero de PC's
var = ['Combustible', 'Economia', 'Vivienda', 'Industrial', 'Comercial', 'Servicio']
n = [1, 2, 3, 1, 2, 2]
```

In [27]:
```
#Aplicamos la función para separar features por categorías
x_subset_train = feat(X_train)
x_subset_test = feat(X_test)
x_subset_deploy = feat(datos)
```

In [28]:
```
# PCA_COMBUSTIBLE
l=0

#Standarizamos la data
scaler = StandardScaler()

# Fit on training set only.
scaler.fit(x_subset_train[l])

# Apply transform to both the training set and the test set.
x_subset_train[l] = scaler.transform(x_subset_train[l])
x_subset_test[l] = scaler.transform(x_subset_test[l])
x_subset_deploy[l] = scaler.transform(x_subset_deploy[l])


pca = PCA(n_components = n[l])
pca.fit(x_subset_train[l])

x_subset_train[l] = pca.transform(x_subset_train[l])
x_subset_test[l]  = pca.transform(x_subset_test[l])
x_subset_deploy[l]  = pca.transform(x_subset_deploy[l])

explained_variance = pca.explained_variance_ratio_

tx_train = pd.DataFrame(x_subset_train[l])
tx_train['ID'] = X_train.index
tx_train.columns = ['PC_Combustible_1', 'ID']

tx_test = pd.DataFrame(x_subset_test[l])
tx_test['ID'] = X_test.index
tx_test.columns = ['PC_Combustible_1', 'ID']

tx_deploy = pd.DataFrame(x_subset_deploy[l])
```

```
tx_deploy['ID'] = datos.index
tx_deploy.columns = ['PC_Combustible_1', 'ID']
```

In [29]:
```
explained_variance
```

Out[29]:
```
array([0.9460952])
```

In [30]:
```
# PCA_ECONOMIA
l=1

#Standarizamos la data
scaler = StandardScaler()

# Fit on training set only.
scaler.fit(x_subset_train[l])

# Apply transform to both the training set and the test set.
x_subset_train[l] = scaler.transform(x_subset_train[l])
x_subset_test[l] = scaler.transform(x_subset_test[l])
x_subset_deploy[l] = scaler.transform(x_subset_deploy[l])

pca = PCA(n_components = n[l])
pca.fit(x_subset_train[l])

x_subset_train[l] = pca.transform(x_subset_train[l])
x_subset_test[l]  = pca.transform(x_subset_test[l])
x_subset_deploy[l]  = pca.transform(x_subset_deploy[l])

explained_variance = pca.explained_variance_ratio_

tx_train_temp = pd.DataFrame(x_subset_train[l])
tx_train_temp['ID'] = X_train.index
tx_train_temp.columns = ['PC_Economia_1', 'PC_Economia_2', 'ID']
tx_train = pd.merge(tx_train, tx_train_temp, how='left', on='ID')

tx_test_temp = pd.DataFrame(x_subset_test[l])
tx_test_temp['ID'] = X_test.index
tx_test_temp.columns = ['PC_Economia_1', 'PC_Economia_2', 'ID']
tx_test = pd.merge(tx_test, tx_test_temp, how='left', on='ID')

tx_deploy_temp = pd.DataFrame(x_subset_deploy[l])
tx_deploy_temp['ID'] = datos.index
```

```
tx_deploy_temp.columns = ['PC_Economia_1', 'PC_Economia_2', 'ID']
tx_deploy = pd.merge(tx_deploy, tx_deploy_temp, how='left', on='ID')
```

In [31]:
```
explained_variance
```

Out[31]:
```
array([0.85569097, 0.07941958])
```

In [32]:
```
# PCA_VIVIENDA
l=2

#Standarizamos la data
scaler = StandardScaler()

# Fit on training set only.
scaler.fit(x_subset_train[l])

# Apply transform to both the training set and the test set.
x_subset_train[l] = scaler.transform(x_subset_train[l])
x_subset_test[l] = scaler.transform(x_subset_test[l])
x_subset_deploy[l] = scaler.transform(x_subset_deploy[l])

pca = PCA(n_components = n[l])
pca.fit(x_subset_train[l])

x_subset_train[l] = pca.transform(x_subset_train[l])
x_subset_test[l]  = pca.transform(x_subset_test[l])
x_subset_deploy[l]  = pca.transform(x_subset_deploy[l])


explained_variance = pca.explained_variance_ratio_

tx_train_temp = pd.DataFrame(x_subset_train[l])
tx_train_temp['ID'] = X_train.index
tx_train_temp.columns = ['PC_vivienda_1', 'PC_Vivienda_2', 'PC_Vivienda_3', 'ID']
tx_train = pd.merge(tx_train, tx_train_temp, how='left', on='ID')

tx_test_temp = pd.DataFrame(x_subset_test[l])
tx_test_temp['ID'] = X_test.index
tx_test_temp.columns = ['PC_vivienda_1', 'PC_Vivienda_2', 'PC_Vivienda_3', 'ID']
tx_test = pd.merge(tx_test, tx_test_temp, how='left', on='ID')

tx_deploy_temp = pd.DataFrame(x_subset_deploy[l])
tx_deploy_temp['ID'] = datos.index
```

```
tx_deploy_temp.columns = ['PC_vivienda_1', 'PC_Vivienda_2', 'PC_Vivienda_3', 'ID']
tx_deploy = pd.merge(tx_deploy, tx_deploy_temp, how='left', on='ID')
```

In [33]:

```
explained_variance
```

Out[33]:

```
array([0.87808187, 0.08942417, 0.02234737])
```

In [34]:

```
# PCA_INDUSTRIAL
l=3

#Standarizamos la data
scaler = StandardScaler()

# Fit on training set only.
scaler.fit(x_subset_train[l])

# Apply transform to both the training set and the test set.
x_subset_train[l] = scaler.transform(x_subset_train[l])
x_subset_test[l] = scaler.transform(x_subset_test[l])
x_subset_deploy[l] = scaler.transform(x_subset_deploy[l])

pca = PCA(n_components = n[l])
pca.fit(x_subset_train[l])

x_subset_train[l] = pca.transform(x_subset_train[l])
x_subset_test[l]  = pca.transform(x_subset_test[l])
x_subset_deploy[l]  = pca.transform(x_subset_deploy[l])


explained_variance = pca.explained_variance_ratio_

tx_train_temp = pd.DataFrame(x_subset_train[l])
tx_train_temp['ID'] = X_train.index
tx_train_temp.columns = ['PC_industrial_1', 'ID']
tx_train = pd.merge(tx_train, tx_train_temp, how='left', on='ID')

tx_test_temp = pd.DataFrame(x_subset_test[l])
tx_test_temp['ID'] = X_test.index
tx_test_temp.columns = ['PC_industrial_1', 'ID']
tx_test = pd.merge(tx_test, tx_test_temp, how='left', on='ID')

tx_deploy_temp = pd.DataFrame(x_subset_deploy[l])
tx_deploy_temp['ID'] = datos.index
```

```
tx_deploy_temp.columns = ['PC_industrial_1', 'ID']
tx_deploy = pd.merge(tx_deploy, tx_deploy_temp, how='left', on='ID')
```

In [35]:
```
explained_variance
```

Out[35]:
```
array([0.96127922])
```

In [36]:
```
# PCA_COMERCIAL
l=4

#Standarizamos la data
scaler = StandardScaler()

# Fit on training set only.
scaler.fit(x_subset_train[l])

# Apply transform to both the training set and the test set.
x_subset_train[l] = scaler.transform(x_subset_train[l])
x_subset_test[l] = scaler.transform(x_subset_test[l])
x_subset_deploy[l] = scaler.transform(x_subset_deploy[l])

pca = PCA(n_components = n[l])
pca.fit(x_subset_train[l])

x_subset_train[l] = pca.transform(x_subset_train[l])
x_subset_test[l]  = pca.transform(x_subset_test[l])
x_subset_deploy[l]  = pca.transform(x_subset_deploy[l])


explained_variance = pca.explained_variance_ratio_

tx_train_temp = pd.DataFrame(x_subset_train[l])
tx_train_temp['ID'] = X_train.index
tx_train_temp.columns = ['PC_comercial_1', 'PC_comercial_2', 'ID']
tx_train = pd.merge(tx_train, tx_train_temp, how='left', on='ID')

tx_test_temp = pd.DataFrame(x_subset_test[l])
tx_test_temp['ID'] = X_test.index
tx_test_temp.columns = ['PC_comercial_1', 'PC_comercial_2', 'ID']
tx_test = pd.merge(tx_test, tx_test_temp, how='left', on='ID')

tx_deploy_temp = pd.DataFrame(x_subset_deploy[l])
tx_deploy_temp['ID'] = datos.index
```

```
tx_deploy_temp.columns = ['PC_comercial_1', 'PC_comercial_2', 'ID']
tx_deploy = pd.merge(tx_deploy, tx_deploy_temp, how='left', on='ID')
```

In [37]:
```
explained_variance
```

Out[37]:
```
array([0.91093752, 0.0638554 ])
```

In [38]:
```
# PCA_SERVICIO
l=5

#Standarizamos la data
scaler = StandardScaler()

# Fit on training set only.
scaler.fit(x_subset_train[l])

# Apply transform to both the training set and the test set.
x_subset_train[l] = scaler.transform(x_subset_train[l])
x_subset_test[l] = scaler.transform(x_subset_test[l])
x_subset_deploy[l] = scaler.transform(x_subset_deploy[l])

pca = PCA(n_components = n[l])
pca.fit(x_subset_train[l])

x_subset_train[l] = pca.transform(x_subset_train[l])
x_subset_test[l]  = pca.transform(x_subset_test[l])
x_subset_deploy[l]  = pca.transform(x_subset_deploy[l])


explained_variance = pca.explained_variance_ratio_

tx_train_temp = pd.DataFrame(x_subset_train[l])
tx_train_temp['ID'] = X_train.index
tx_train_temp.columns = ['PC_servicio_1', 'PC_servicio_2', 'ID']
tx_train = pd.merge(tx_train, tx_train_temp, how='left', on='ID')

tx_test_temp = pd.DataFrame(x_subset_test[l])
tx_test_temp['ID'] = X_test.index
tx_test_temp.columns = ['PC_servicio_1', 'PC_servicio_2', 'ID']
tx_test = pd.merge(tx_test, tx_test_temp, how='left', on='ID')

tx_deploy_temp = pd.DataFrame(x_subset_deploy[l])
tx_deploy_temp['ID'] = datos.index
```

```
tx_deploy_temp.columns = ['PC_servicio_1', 'PC_servicio_2', 'ID']
tx_deploy = pd.merge(tx_deploy, tx_deploy_temp, how='left', on='ID')
```

In [39]:
```
explained_variance
```

Out[39]:
```
array([0.78162536, 0.12170168])
```

5. Feauture Selection

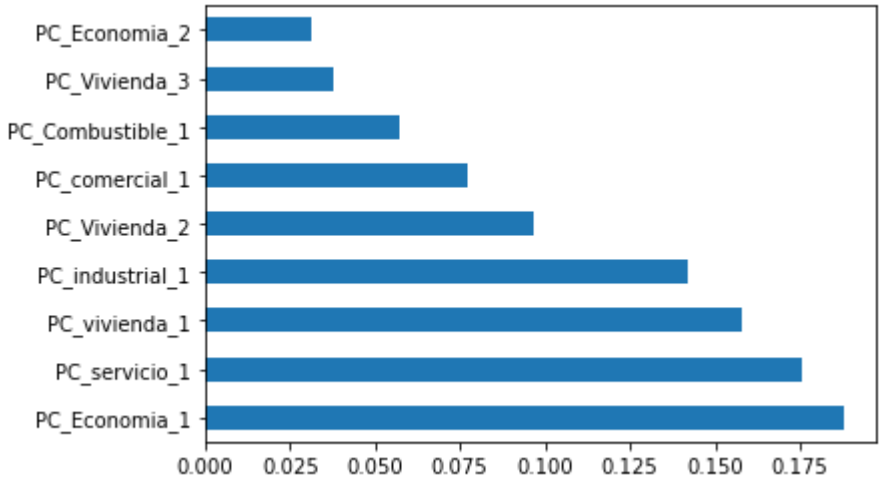
In [40]:
```
#Definition of predictors number to be used.   Aprox 10% of total instances.
n=9
```

In [41]:
```
X_train = tx_train.copy().set_index('ID')
X_test = tx_test.copy().set_index('ID')
```

In [42]:
```
from sklearn.ensemble import ExtraTreesRegressor

model = ExtraTreesRegressor()
model.fit(X_train,y_train)
print(model.feature_importances_) #use inbuilt class feature_importances of tree based classifiers
```

```
[0.05723656 0.18776721 0.03114698 0.15762934 0.09670906 0.03756114
 0.14201865 0.07714755 0.01841067 0.17570457 0.01866827]
```

In [43]:
```
#plot graph of feature importances for better visualization
import matplotlib.pyplot as plt
feat_importances = pd.Series(model.feature_importances_, index=X_train.columns)
feat_importances.nlargest(n).plot(kind='barh')
plt.show()
```



In [44]:
```
#Creamos Data Set final de Features
X_train = X_train[features_simples]
X_test  = X_test[features_simples]
```

5. Entrenamiento del Modelo

- Regresión Ridge

5.1. Rigde

In [45]:
```
# Tunning de Hiperparámetros Regresión Ridge
from sklearn.linear_model import Ridge

params_ridge = {
    'alpha' : [0.1, 0.4, 0.5, .7, 0.75, 0.8, 0.85,.9, .99, 1, 5, 10, 20, 30],
    'fit_intercept' : [True, False],
    'normalize' : [True,False],
    'solver' : ['svd', 'cholesky', 'lsqr', 'sparse_cg', 'sag', 'saga']
}

ridge_model = Ridge()
ridge_regressor = GridSearchCV(ridge_model, params_ridge, scoring='r2', cv=5, n_jobs=-1)
ridge_regressor.fit(X_train, y_train)
print(f'Optimal alpha: {ridge_regressor.best_params_["alpha"]:.2f}')
print(f'Optimal fit_intercept: {ridge_regressor.best_params_["fit_intercept"]}')
```

```
print(f'Optimal normalize: {ridge_regressor.best_params_["normalize"]}')
print(f'Optimal solver: {ridge_regressor.best_params_["solver"]}')
print(f'Best score: {ridge_regressor.best_score_}')
```

```
Optimal alpha: 0.99
Optimal fit_intercept: True
Optimal normalize: False
Optimal solver: svd
Best score: 0.7036697132925436
```

In [46]:

```
# Aplicación del Modelo Regresión Ridge

ridge_model = Ridge(alpha=ridge_regressor.best_params_["alpha"], fit_intercept=ridge_regressor.best_params_["fit_intercep
                    normalize=ridge_regressor.best_params_["normalize"], solver=ridge_regressor.best_params_["solver"])
ridge_model.fit(X_train, y_train)
y_train_pred = ridge_model.predict(X_train)
y_pred = ridge_model.predict(X_test)
print('Train r2 score: ', r2_score(y_train_pred, y_train))
print('Test r2 score: ', r2_score(y_test, y_pred))
train_rmse = np.sqrt(mean_squared_error(y_train_pred, y_train))
test_rmse = np.sqrt(mean_squared_error(y_test, y_pred))
print(f'Train RMSE: {train_rmse:.4f}')
print(f'Test RMSE: {test_rmse:.4f}')
```

```
Train r2 score:  0.6477306996620347
Test r2 score:  0.8653160457094522
Train RMSE: 0.0773
Test RMSE: 0.0557
```

In [47]:

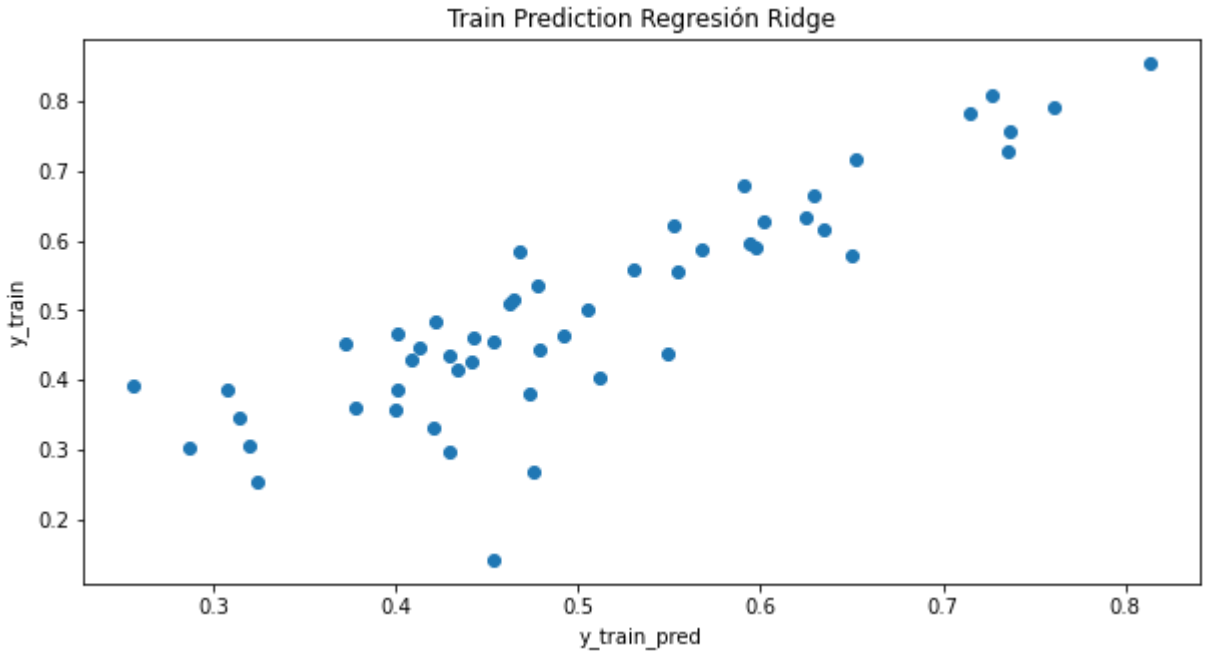
```
#TABLA DE REGRESION RIDGE AJUSTADA
from regressors import stats

stats.summary(ridge_model, X_train, y_train)
```

```
Residuals:
   Min      1Q  Median      3Q     Max
-0.136 -0.0496 -0.0132  0.0187  0.3086


Coefficients:
            Estimate  Std. Error  t value   p value
_intercept  0.502538    0.010716  46.8960  0.000000
x1          0.073003    0.024850   2.9377  0.004954
x2          0.082292    0.028891   2.8484  0.006320
x3          0.045766    0.023304   1.9638  0.055010
x4          0.150061    0.031324   4.7906  0.000015
```

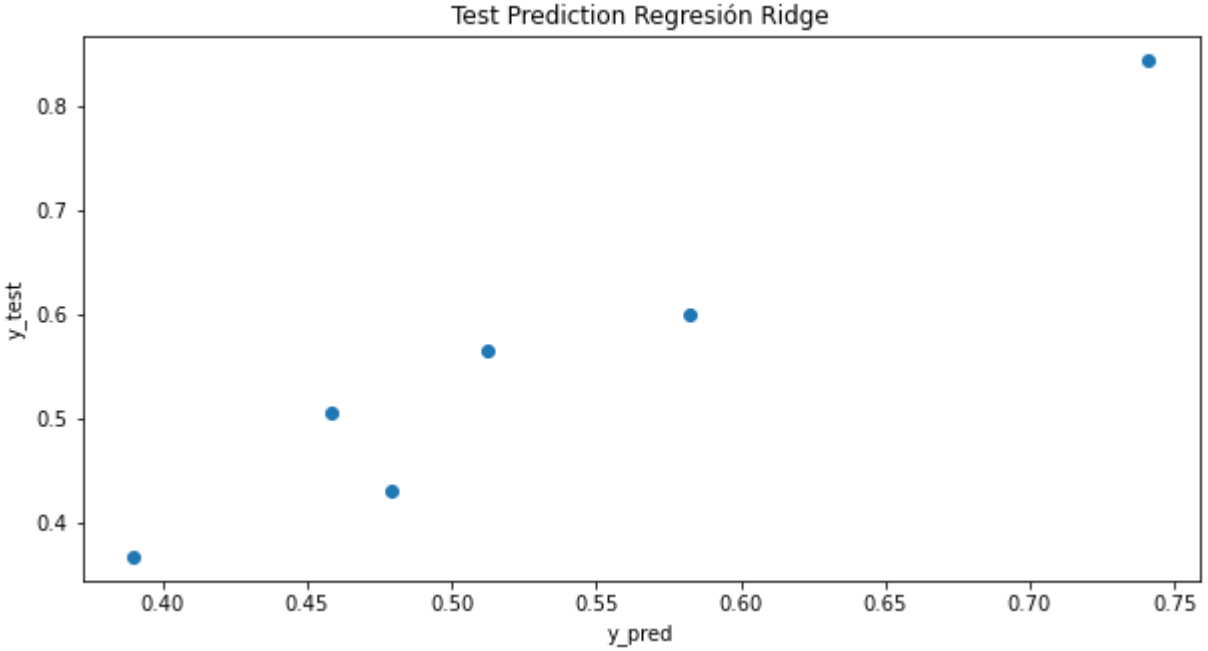
```
---
R-squared:  0.75447,    Adjusted R-squared:  0.73358
F-statistic: 36.11 on 4 features
```

In [48]:

```
#Visualización Modelo Regresión Ridge

#Train
xv = y_train_pred
yv = y_train

plt.figure(figsize = (10,5))
plt.scatter(xv, yv, alpha=1)
plt.title('Train Prediction Regresión Ridge')
plt.xlabel('y_train_pred')
plt.ylabel('y_train')
graf1 = plt.show()
print(graf1)

#Test
xv = y_pred
yv = y_test

plt.figure(figsize = (10,5))
plt.scatter(xv, yv, alpha=1)
plt.title('Test Prediction Regresión Ridge')
plt.xlabel('y_pred')
plt.ylabel('y_test')
graf2 = plt.show()
print(graf2)
```


Train Prediction Regresión Ridge
y_train
y_train_pred
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.3
0.4
0.5
0.6
0.7
0.8

None


Test Prediction Regresión Ridge
y_test
y_pred
0.4
0.5
0.6
0.7
0.8
0.40
0.45
0.50
0.55
0.60
0.65
0.70
0.75

None

6. Deployment

In [49]:
```
tx_deploy = tx_deploy.set_index('ID')
```

In [50]:
```
tx_deploy = tx_deploy[features_simples]
```

In [51]:
```
#Aplicamos el Modelo
TPD_Ridge=pd.Series(ridge_model.predict(tx_deploy))
```

In [52]:
```
TPD=pd.DataFrame(TPD_Ridge)
TPD.columns =['Ridge']
```

In [53]:
```
k=41871
e=math.e

TPD['Ridge_total']=k/(e**TPD['Ridge']+1)
```

In [54]:
```
TPD
```

Out[54]:

	Ridge	Ridge_total
0	0.399678	16806.592063
1	0.395692	16846.704135
2	0.396387	16839.704702
3	0.398393	16819.516919
4	0.395904	16844.566418
5	0.398727	16816.160700
6	0.398783	16815.592644
7	0.398211	16821.345790
8	0.418323	16619.356732

	Ridge	Ridge_total
9	0.407524	16727.719360
10	0.434220	16460.286754
11	0.399678	16806.592063
12	0.399678	16806.592063
13	0.399678	16806.592063
14	0.399678	16806.592063
15	0.362848	17178.427065
16	0.349645	17312.341636
17	0.390876	16895.221562
18	0.364373	17162.985666
19	0.333187	17479.692388
20	0.601010	14827.049284
21	0.246226	18371.015672
22	0.399678	16806.592063
23	0.399678	16806.592063
24	0.399678	16806.592063

In [55]:

```
# Aplicación del Modelo Regresión Ridge

ridge_model = Ridge(alpha=ridge_regressor.best_params_["alpha"], fit_intercept=ridge_regressor.best_params_["fit_intercep
                    normalize=ridge_regressor.best_params_["normalize"], solver=ridge_regressor.best_params_["solver"])
ridge_model.fit(X_train, y_train)
y_train_pred = ridge_model.predict(X_train)
y_pred = ridge_model.predict(X_test)
print('Train r2 score: ', r2_score(y_train_pred, y_train))
print('Test r2 score: ', r2_score(y_test, y_pred))
train_rmse = np.sqrt(mean_squared_error(y_train_pred, y_train))
test_rmse = np.sqrt(mean_squared_error(y_test, y_pred))
print(f'Train RMSE: {train_rmse:.4f}')
print(f'Test RMSE: {test_rmse:.4f}')
```

```
Train r2 score:  0.6477306996620347
Test r2 score:  0.8653160457094522
Train RMSE: 0.0773
Test RMSE: 0.0557
```

In [56]:
```
y_test
```

Out[56]:
```
ID
20182    0.429659
20181    0.506032
20189    0.367809
20176    0.565555
20162    0.600281
20155    0.842637
Name: TOTAL_TPD_TRANSFORMADO, dtype: float64
```

In [57]:
```
y_pred
```

Out[57]:
```
array([0.47882296, 0.45863712, 0.38972284, 0.51264424, 0.58203438,
       0.74110288])
```

COMPARACION REAL_MODELO

In [58]:
```
dc = pd.DataFrame(y_test)
dc['TPD_Ridge'] = y_pred

dc['TPD_REAL']=k/(e**dc['TOTAL_TPD_TRANSFORMADO']+1)
dc['TPD_ESTIMADO']=k/(e**dc['TPD_Ridge']+1)
dc['ERROR']=dc['TPD_ESTIMADO']-dc['TPD_REAL']
dc['%_ERROR'] = (abs(dc['ERROR'])/dc['TPD_ESTIMADO'])*100
dc
```

Out[58]:

ID	TOTAL_TPD_TRANSFORMADO	TPD_Ridge	TPD_REAL	TPD_ESTIMADO	ERROR	%_ERROR
20182	0.429659	0.478823	16505.868940	16016.917967	-488.950973	3.052716
20181	0.506032	0.458637	15748.693347	16217.022963	468.329617	2.887889
20189	0.367809	0.389723	17128.190260	16906.840840	-221.349420	1.309230
20176	0.565555	0.512644	15168.318524	15683.782012	515.463488	3.286602

ID	TOTAL_TPD_TRANSFORMADO	TPD_Ridge	TPD_REAL	TPD_ESTIMADO	ERROR	%_ERROR
20162	0.600281	0.582034	14834.034483	15009.271991	175.237508	1.167528
20155	0.842637	0.741103	12602.322581	13514.410471	912.087891	6.749002