Tecnológico de Estudios Superiores
de Cuautitlán Izcalli
Organismo Público Descentralizado del Estado de México

MAESTRÍA EN TECNOLOGÍAS DE LA INFORMACIÓN

Machine Learning como herramienta de Análisis Predictivo en el área Financiera de la Industria Automotriz

TESIS

QUE PARA OBTENER EL GRADO DE:

MAESTRO EN TECNOLOGÍAS DE LA INFORMACIÓN

PRESENTA:

Lic. Israel Alejandro Avila Velazquez

DIRECTOR DE TESIS:

M. en C. MARÍA DEL CONSUELO MACIAS GONZÁLEZ

CUAUTITLÁN IZCALLI, EDO. DE MÉXICO 2 DICIEMBRE 2022.

EDUCACIÓN | TECNOLÓGICO NACIONAL DE MÉXICO | GOBIERNO DEL ESTADO DE MÉXICO | TECNOLÓGICO DE ESTUDIOS SUPERIORES DE CUAUTITLÁN IZCALLI | EDOMÉX

"2022. Año del Quincentenario de Toluca, Capital del Estado de México".

Cuautitlán Izcalli, Estado de México a 24 de noviembre de 2022
TESCI/DIDT/208/XI/22

DIRECCIÓN ACADÉMICA
DEPARTAMENTO DE INVESTIGACIÓN Y DESARROLLO TECNOLÓGICO
COORDINACIÓN DE POSGRADO

LICENCIADO
ISRAEL ALEJANDRO AVILA VELAZQUEZ
P R E S E N T E

Por este conducto me permito informarle que puede proceder a la digitalización del Trabajo de Tesis titulado:

"MACHINE LEARNING COMO HERRAMIENTA DE ANÁLISIS PREDICTIVO EN EL ÁREA FINANCIERA DE LA INDUSTRIA AUTOMOTRIZ"

Ya que la comisión encargada de revisar el trabajo que se presenta para efectos de titulación, han dado su autorización conforme a lo estipulado en el Lineamiento para la operación de los Estudios de Posgrado en el Sistema Nacional de Institutos Tecnológicos.

Sin nada más que agregar, quedo a sus órdenes para cualquier aclaración.

A T E N T A M E N T E

TECNOLÓGICO DE ESTUDIOS SUPERIORES DE CUAUTITLÁN IZCALLI
UNIDAD DE POSGRADO

MTRA. ROCIO ORTEGA JIMÉNEZ
DEPARTAMENTO DE
INVESTIGACIÓN Y DESARROLLO TECNOLÓGICO
COORDINACIÓN DE POSGRADO

c.c.p. Archivo
Departamento de Titulación
Expediente del alumno

2022

Índice

Capítulo I

Problemática

Introducción

Actualmente la Industria automotriz, sufre de varios factores que han complicado la operación de ventas y producción de unidades en las armadoras.

Una de ellas fue la pandemia, que ha provocado cierre de Distribuidoras y una caída en ventas contra años anteriores.

Adicional existe un problema de escases de microchips, por una alta demanda de aparatos tecnológicos y una poca producción, que no está dando abasto a nivel mundial y que está provocando paro de producción en las plantas armadoras de las marcas Automotrices lo que representa pérdidas millonarias.

Por otro lado, el área financiera juega un papel muy importante, ya que tiene como objetivo custodiar e invertir los valores y recursos de la empresa, controlar activos y operación diaria, así como de proteger el capital invertido por los socios.

Por lo cual en este proyecto de investigación pretende desarrollar una herramienta de Análisis predictivo, su objetivo es ayudar a identificar y pronosticar situaciones, como las solicitudes de crédito que se generar por medio de un crédito automotriz, tomando en consideración datos demográficos, versiones de autos, perfiles de cliente, historial de solicitudes, estatus socioeconómico, entre otros datos.

Cabe mencionar que lo importante es conocer mejor el perfil de cliente en las diferentes marcas Automotrices y poder ofrecer mejores productos y servicios que permitirá tener ahorros costos de almacenaje de unidades y tiempos de entrega.

Esto permitirá generar una mejor confianza y lealtad de los clientes lo que se transformará en mejores resultados para la compañía y los distribuidores, mediante los resultados obtenidos se podrá mejorar volumen de producción y venta, incluyendo variables fuera de lo normal como la pandemia y actualmente la falta de microchips a nivel mundial, entre otras.

Problemática

Un pronóstico consiste en predecir el comportamiento futuro de un producto o servicio específico. De esta forma se logra estimar cuánto producto se va a vender o va a demandar el mercado. El pronóstico se predice con base en los volúmenes, en órdenes de pedido, facturación o ventas durante un período de tiempo.

Un buen pronóstico puede afectar de manera positiva los niveles de capacidad, inventario, abastecimiento, las necesidades financieras y la estructura general del negocio.

El principal problema o reto al pronosticar la demanda, es la baja asertividad que puede tener el pronóstico que realicemos.

Existen varios desafíos o problemas comunes a la hora de pronosticar la demanda, lo que puede generar un alto error en su predicción.

Algunos de estos problemas pueden ser cuando existen nuevos productos, debido a que son difíciles de predecir por la falta de datos históricos que permitan encontrar patrones. También cuando hay una gran variedad de productos, datos geográficos amplios y cuando existen datos con variaciones aleatorias en los patrones históricos.

Aunque el panorama al realizar tus pronósticos no parezca siempre tan alentador, tenemos actualmente algunas herramientas tecnológicas como la BigData, Inteligencia Artificial y los modelos de Machine Learning.

En este proyecto veremos algunas de estas técnicas y revisaremos algunos casos revisando los diferentes resultados y la efectividad que puede generar estos modelos de aprendizaje en los pronósticos.

Objetivo General

- Diseñar y desarrollar un modelo Machine Learning que impacte en el análisis predictivo de la generación de solicitudes de créditos financieros cuyos resultados permita tomar decisiones a corto y mediano plazo en el área automotriz de Servicios Financieros.

Objetivos Particulares

- Investigar y conocer las reglas de negocio, así como los datos actuales para el cálculo del pronóstico de contratos.
- Evaluar la utilización de tecnologías asociadas a Machine Learning, para la manipulación de datos.
- Clasificar y organizar los datos para alimentar el algoritmo de aprendizaje.
- Proponer varios modelos predictivos de acuerdo con las herramientas tecnológicas elegidas.
- Validar, interpretar y evaluar los modelos predictivos propuestos.
- Toma de decisiones.

Justificación

La finalidad del modelo predictivo tendrá un impacto muy importante en el área financiera por que ayudará a cubrir las siguientes necesidades:

Optimizar recursos. Trabajando con el análisis de datos podremos tener una mejor optimización de recursos como la distribución de formatos de solicitudes de crédito, personal, folletos, marketing, y se podrá canalizar los recursos adecuadamente, además de poder utilizar mejor los recursos económicos que fondearan los créditos automotrices.

Fidelización del cliente. Se podrá personalizar descuentos ofertas, servicios de acuerdo con el vehículo del cliente, región, edad, género, entre otros.

Cumplimiento de Objetivos. Permitirá tener pronósticos financieros que ayudará a alcanzar los objetivos de ventas.

Sistema de recomendación. Que nos ayudara a identificar las necesidades de los clientes, y hacer mejoras para alcanzar una mejor satisfacción de los clientes.

Este desarrollo estará basado en la ciencia de Datos, La ciencia de Datos está compuesta por tres elementos que ayudará al desarrollo: Estadística, Programación y Experiencia del tema.

La máquina de aprendizaje tendrá la función de aprender patrones históricos del negocio y de los perfiles de los clientes y prospectos, la cual se podría trabajar en una Big Data, y en herramientas BI para su interpretación, mismas que forman parte de las TI actualmente.

Para esto se requiere trabajar en tres fases importantes: el entendimiento (Entender correctamente las reglas de negocio), análisis y modelado (depurar y preparar los datos para hacer un análisis de estos de forma que se elijan las metodologías correctas) y acción (evaluar los resultados con respecto a la meta que se identificó, y repetir hasta conseguir algo satisfactorio que se permita tener el mejor diseño).

Los principales conceptos que son esenciales para el proyecto: es el aprendizaje (machine learning) que el sistema realice, en base a la información recibida de las diferentes bases de datos(bigdata), y con esta información el sistema tendrá que realizar cálculos estadísticos que permitirá la predicción de datos, generando un ciclo de aprendizaje nuevo cada que exista una nueva variable.

Lo prioritario es el utilizar ciertas herramientas como Algoritmos utilizados en máquinas de aprendizaje por ejemplo KNIME o MLflow, así como lenguajes de programación con un enfoque estadístico como codigo R y Herramientas de Bigdata como Hadoop para el almacenaje de toda la información.

En resumen, lo importante será analizar los diferentes algoritmos que se pueden utilizar en Python y revisar los resultados en cada caso, así como sus desviaciones y con esto determinar qué tan efectivo son estas herramientas en el sector Financiero Automotriz.

Capítulo II

Marco teórico

Marco Teórico.

La industria automotriz es un conjunto de compañías y organizaciones relacionadas en las áreas de diseño, desarrollo, manufactura, marketing y ventas de automóviles. Es uno de los sectores económicos más importantes en el mundo por sus ingresos.

El potencial de la industria automotriz mexicana es tal que representa el segundo sector económico más importante del país, además de que significa el elemento primordial de la modernización y estrategias de globalización de este.

A nivel global, la importancia de la industria automotriz en las economías nacionales y su papel como propulsor para el desarrollo de otros sectores de alto valor agregado han provocado que diversos países tengan como uno de sus principales objetivos el desarrollo y/o fortalecimiento de esta industria. México no es la excepción, pues la industria automotriz en nuestro país ha representado un sector estratégico para el desarrollo de nuestro país. Su participación en las exportaciones la coloca como la industria más importante, superando incluso al sector petrolero.

Por otro lado, esta industria genera una gran cantidad de datos, tipos de unidades, marcas, clientes, proveedores, contratos de financiamiento, datos geográficos y demográficos, entre los que podemos mencionar, esta información en conjunto se le denomina BigData o los macrodatos, también llamados datos masivos, inteligencia de datos, datos a gran escala, son términos que hace referencia a conjuntos de datos tan grandes y complejos que precisan de aplicaciones informáticas no tradicionales de procesamiento de datos para tratarlos adecuadamente. (Macrodatos e inteligencia de datos, alternativas a big data, 2021).

Big data representa los activos de información caracterizados por un volumen, velocidad y variedad tan altos que requieren una tecnología específica y métodos analíticos para su transformación en valor. (Andrea De Mauro, 2020).

Otro de los conceptos que veremos en este proyecto es la Ciencia de Datos que es un campo interdisciplinario que involucra métodos científicos, procesos y sistemas para extraer conocimiento o un mejor entendimiento de datos en sus diferentes formas, ya sea estructurados o no estructurados, lo cual es una continuación de algunos campos de análisis de datos como la estadística, la minería de datos, el aprendizaje automático, y la analítica predictiva. (Dr. Alex Liu, 2021).

Algunas de las técnicas que posibilitan la explotación de los datos son la Minería de datos, extrayendo información que no es detectada a simple vista. La cual combina técnicas semiautomáticas de inteligencia artificial, análisis estadístico, bases de datos y visualización gráfica, para la obtención de información que no esté representada explícitamente en los datos. (Martínez, 2021)

Una de las herramientas que utilizaremos en este proyecto para la predicciones de datos es el aprendizaje automático, aprendizaje automatizado o aprendizaje de máquinas (del inglés, Machine Learning) es el subcampo de las ciencias de la computación y una rama de la inteligencia artificial, cuyo objetivo es desarrollar técnicas que permitan que las computadoras aprendan. Se dice que un agente aprende cuando su desempeño mejora con la experiencia y mediante el uso de datos; es decir, cuando la habilidad no estaba presente en su genotipo o rasgos de nacimiento. (Norvig, 2021)

Los sistemas o tipos de Machine Learning pueden ser clasificados de acuerdo con la cantidad y tipo de supervisión que tienen durante su entrenamiento (Brian, 2020), existen 4 tipos de categorías principales:

- Aprendizaje Supervisado
- Aprendizaje No Supervisado
- Aprendizaje Semi-Supervisado
- Aprendizaje De Reforzamiento

En este proyecto conceptualizaremos a los dos más importantes:

El aprendizaje o algoritmo supervisado es una técnica usada en minería de datos, en la que se genera una función de pronóstico a partir del entrenamiento previo sobre datos etiquetados. Es decir, aprendemos a partir de casos reales y extrapolamos el resultado a los casos futuros. (Villalba, 2018)

Los Algoritmos Supervisados más Importantes:

- K-nearest neighbors (KNN, vecinos más cercanos K)
- Red Neural
- Máquinas de soporte de vectores
- Regresión Lineal
- Arboles de decisiones y bosques aleatorios

El aprendizaje no supervisado son un conjunto de técnicas que permiten inferir modelos para extraer conocimiento de conjuntos de datos donde a priori se desconoce. Las técnicas de aprendizaje no supervisado se pueden aplicar sin necesidad de tener los datos etiquetados para el entrenamiento.

Los Algoritmos No Supervisados más importantes:

- Agrupamiento (Clustering): Medios k, análisis de agrupamiento jerárquico
- Machine Learning de Asociación de Regla: Eclat y A priori
- Visualización y Reducción de Dimensionalidad: Núcleo PCA, distribuido de t PC10A

No podemos hablar de Machine Learning sin mencionar la inteligencia artificial (IA) es la habilidad de los ordenadores para hacer actividades que normalmente requieren inteligencia humana, también se puede definir como la capacidad de las máquinas para usar algoritmos, aprender de los datos y utilizar lo aprendido en la toma de decisiones tal y como lo haría un ser humano. (ROUHIAINEN, 2018)

El análisis predictivo es un área de la minería de datos que consiste en la extracción de información existente en los datos y su utilización para predecir tendencias y patrones de comportamiento, pudiendo aplicarse sobre cualquier evento desconocido, ya sea en el pasado, presente o futuro. El análisis predictivo se fundamenta en la identificación de relaciones entre variables en eventos pasados, para luego explotar dichas relaciones y predecir posibles resultados en futuras situaciones. (Timón, 2021)

Para complementar los resultados de un análisis predictivo es importante utilizar interfaces o herramientas como Business Intelligent que permite para explorar y analizar información estructurada sobre un área (normalmente almacenada en un datawarehouse), para descubrir tendencias o patrones, a partir de los cuales derivar ideas y extraer conclusiones. El proceso de Business Intelligent incluye la comunicación de los descubrimientos y efectuar los cambios. Las áreas incluyen clientes, proveedores, productos, servicios y competidores. (Gartner, 2021)

Por último, al finalizar estas proyecciones de datos o análisis predictivos, es importante generar Indicadores Clave de Desempeño o KPI (Key Performance Indicator) es un valor medible de forma cualitativa o cuantitativa, generalmente expresado como un porcentaje o ratio, permite evaluar el progreso hacia la consecución de objetivos planteados en una empresa. La consecución de estos objetivos se revisa a intervalos regulares. El reto es encontrar el indicador más idóneo que esté ligado a lo que se está monitorizando. (Parmenter, 2021)

Capítulo III

Metodología

3.1 Metodología CRISP-DM

Este trabajo de investigación se desarrollará con la metodología CRISP-DM, que son las siglas de Cross-Industry Standard Process for Data Mining, es un método probado y más utilizado para orientar trabajos de minería de datos y machine learning.

CRISP-DM está compuesta por seis fases, las cuales dependen entre sí tanto en forma secuencial como cíclica, pudiendo encontrarse interacciones que permitan mejorar la aproximación obtenida en otras fases anteriores (Gil, 2021).

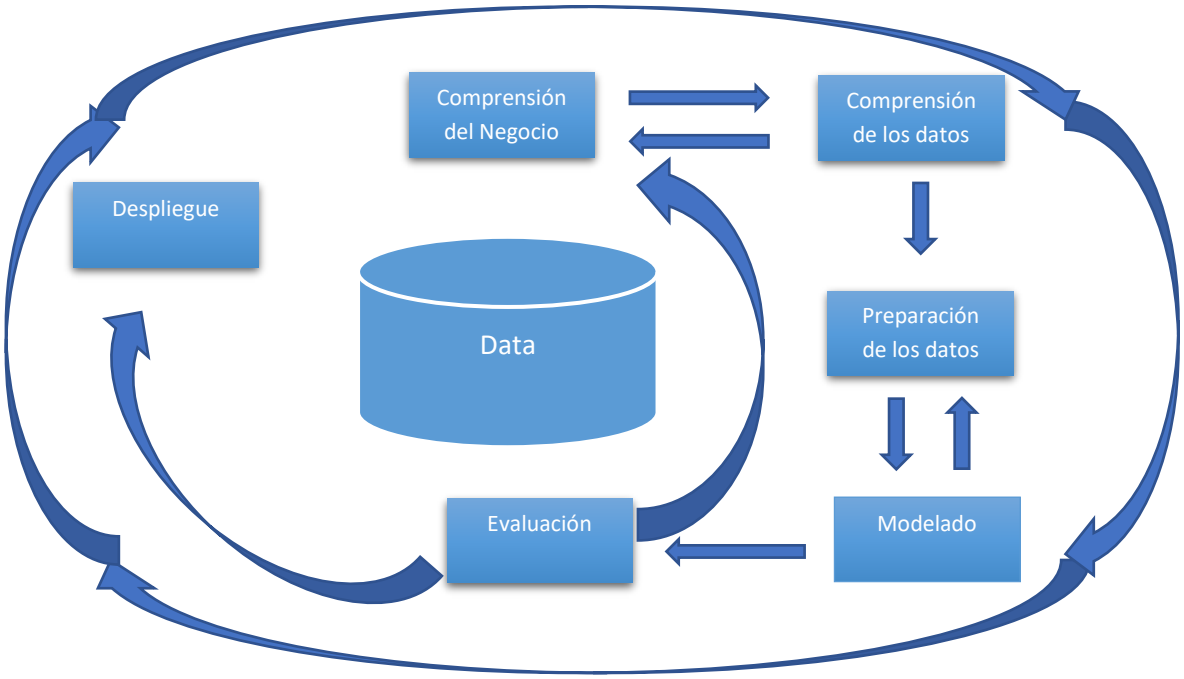


Fig1. Fases de Metodología CRISP-DM.

3.2 Comprensión del negocio.

Dentro de la primera fase de nuestra metodología se encuentra la comprensión del negocio, es probablemente la más importante y agrupa las tareas de comprensión de los objetivos y requisitos del proyecto desde una perspectiva empresarial o institucional, con el fin de convertirlos en objetivos técnicos y en un plan de proyecto, en esta fase se da la importancia al objetivo principal del proyecto.

Como sabemos nuestro proyecto está involucrado en la industria automotriz que se encarga del diseño, desarrollo, fabricación, ensamblaje, comercialización y venta de automóviles. Es una industria que requiere de la exactitud de tiempos y movimientos para lograr las entregas en tiempo y forma.

Uno de sus objetivos es administrar correctamente los recursos, estimar los resultados de ciertas inversiones, estudiar el lanzamiento de nuevos productos, evaluar tendencias del mercado y, en general, elaborar proyectos de largo alcance.

El objetivo de nuestro proyecto es “Diseñar y desarrollar un modelo Machine Learning que impacte en el análisis predictivo de la contratación de créditos financieros cuyos resultados permita tomar decisiones a corto y mediano plazo en el área automotriz de Servicios Financieros”.

Para esto será necesario conocer las diferentes herramientas tecnológicas para desarrollar los algoritmos de machine learning, mismas, que diseñaremos y definiremos en las siguientes fases de la metodología.

Fig2.Fase de compresión del Negocio.

3.3 Comprensión de los datos.

En esta fase de la metodología se realiza la recolección inicial de los datos, se familiariza con los datos y se averiguar su calidad, así como se identificar las relaciones más evidentes para formular las primeras hipótesis.

En nuestro proyecto contamos con información e históricos, que se alimenta de diferentes sistemas y nos permitirá trabajar con nuestra metodología. Contamos con Históricos de Ventas, Solicitudes de Créditos Automotrices y Contratos cerrados, así como los diferentes catálogos que complementa la información.

Una vez identificados las tablas con información necesitamos describir detalladamente el tipo de información, así como explorar e identificar posibles errores de información o valores fuera de rango, para asegurar que nuestra información es confiable y no presente problemas al ejecutar nuestros algoritmos y ejecute los resultados esperados.

Fig3. Fase de compresión de los datos.

3.4 Preparación de los datos.

Después de la recolección inicial de datos, el siguiente paso de nuestra metodología es preparar los datos para adaptarlos al proceso de Machine Learning, en esta preparación delimitaremos el uso de los parámetros o campos que necesitaremos para nuestro análisis predictivo

Esta fase se encuentra relacionada con la fase de modelado, puesto que, en función de la técnica de modelado elegida, los datos requieren ser procesados de diferentes formas.

Dentro de la preparación de los datos es necesario primero realizar la selección de datos, donde vamos delimitando y agrupando los datos de acuerdo con los criterios previamente establecidos en las fases anteriores.

Posteriormente sigue la limpieza de los datos, esta tarea complementa a la anterior, y es una de las que más tiempo y esfuerzo consume, ya que es necesario llevar a cabo diversas técnicas que pueden aplicarse para optimizar la calidad de los datos y prepararlos para la fase de modelación.

Después continuamos con la estructuración de los datos, que incluye las operaciones de preparación de los datos tales como la generación de nuevos atributos a partir de atributos ya existentes, integración de nuevos registros o transformación de valores para atributos existentes, como pueden ser conversiones de datos o equivalencias.

La integración de los datos involucra la creación de nuevas estructuras, fusiones, nuevas tablas de procesamiento, o creación de nuevos campos, todo esto para facilitar la comprensión de los datos.

Por último, el formateo de los datos que consiste principalmente, en la realización de transformaciones sintácticas de los datos sin modificar su significado, esto, con la idea de permitir o facilitar el empleo de algún algoritmo de machine learning.

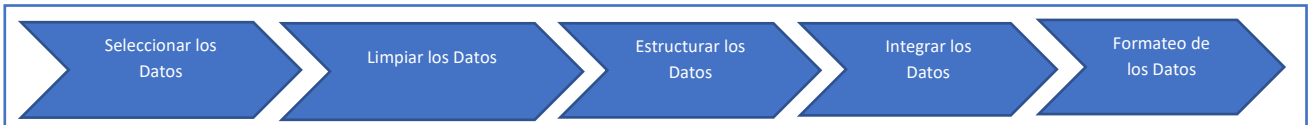


Fig4. Fase de preparación de los datos.

3.5 Modelado.

En esta fase, se seleccionan las técnicas de modelado más apropiadas para el proyecto de Machine Learning. Las técnicas por utilizar en esta fase se eligen en función de los siguientes criterios:

- Ser apropiada al problema.
- Disponer de datos adecuados.
- Cumplir los requisitos del problema.
- Tiempo adecuado para obtener un modelo.
- Conocimiento de la técnica.

Primero se debe selección de la técnica de modelado que consiste en la selección de la técnica de Machine Learning más apropiada al tipo de problema a resolver, después debemos generar un plan de prueba para probar la calidad y validez de este. Seguimos con la construcción del modelo, donde se ejecuta sobre los datos previamente preparados para generar uno o más modelos. Por último, sigue la evaluación del modelo, donde se interpretan los datos y se comparan contra datos reales, en esta fase es importante el conocimiento y experiencia del negocio, para determinar si los datos hacen sentido o no.

En nuestro proyecto hemos decidido usar el tipo de algoritmos supervisados ya que tenemos datos plenamente identificados, además que nuestros datos de entrada deberán predecir el mismo tipo de dato.

Otra parte fundamental es que utilizaremos Python, que actualmente es uno de los lenguajes de programación que se adecuaran al machine learnign, y cabe mencionar que tiene una herramienta, scikit-learn, que es una de las más completas, ya que integra muchos algoritmos como: K-nearest neighbors (KNN, vecinos más cercanos K), Red Neural, Máquinas de soporte de vectores, Regresión lineal, Arboles de decisiones y bosques aleatorios.

Nosotros estaremos utilizando Regresión lineal, por tener una mayor exactitud, además que utiliza series de tiempo y nos permite resolver nuestro planteamiento del problema, además permite comprender la relación de nuestras variables que deseamos buscar.

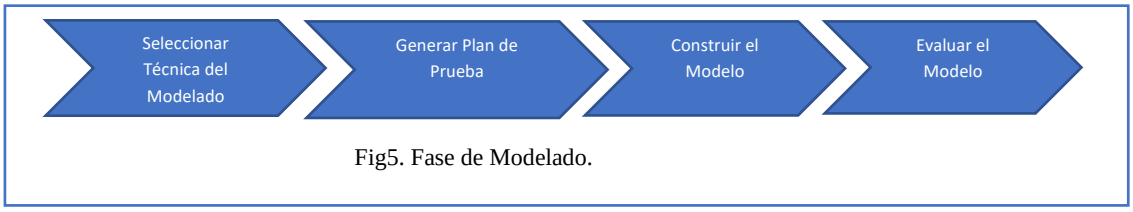


Fig5. Fase de Modelado.

3.6 Evaluación.

Esta fase se divide en tres procesos el primero es la evaluación de los resultados. En donde se evalúa el modelo en relación con los objetivos del negocio y busca determinar si hay alguna razón de negocio para la cual, el modelo sea deficiente, o si es aconsejable probar el modelo, en un problema real si el tiempo y restricciones lo permiten, después viene el proceso de revisión, donde se califica al proceso entero de Machine Learning, con el objeto de identificar elementos que pudieran ser mejorados.

Por último, la determinación de futuras fases. Si se ha determinado que las fases hasta este momento han generado resultados satisfactorios, podría pasarse a la fase siguiente, en caso contrario podría decidirse por otra iteración desde la fase de preparación de datos o de modelación con otros parámetros. Podría ser incluso que en esta fase se decida partir desde cero con un nuevo proyecto de Machine Learning.

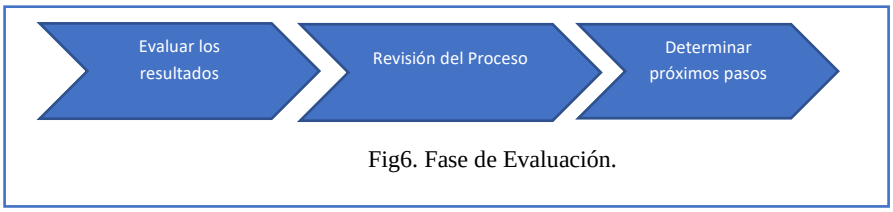


Fig6. Fase de Evaluación.

3.7 Despliegue o Implantación.

Este se divide en cuatro procesos el primero de ellos es el plan de implementación. Donde el resultado del análisis de Machine Learning debe implementarse en alguna organización, esta tarea toma los resultados de la evaluación y concluye una estrategia para su implementación.

La monitorización y mantenimiento aplica Si los modelos resultantes del proceso del análisis son implementados en el dominio del problema como parte de la rutina diaria o periódica, es aconsejable preparar estrategias de monitorización y mantenimiento para ser aplicadas sobre los algoritmos a fin de detectar posibles desviaciones futuras.

Por último, el informe final es la conclusión del proyecto de Machine Learning realizado, dependiendo del plan de implementación, este informe puede ser sólo un resumen de los puntos importantes del proyecto y la experiencia lograda o puede ser una presentación final que incluya y explique los resultados logrados con el proyecto.

Revisión del proyecto: En este punto se evalúa qué fue lo correcto y qué lo incorrecto, qué es lo que se hizo bien y qué es lo que se requiere mejorar a futuro.

Fig7. Fase de despliegue o implantación.

Capítulo IV

Diseño del Modelo de predicción

Machine Learning

4.1 Como opera un Pronóstico.

Para comprender como se realizan los pronósticos financieros, debemos saber que uno de los más usados en el área financiera son los pronósticos financieros cuantitativos que están basados en datos previos o históricos: los estados financieros, análisis de series temporales, el balance general, los indicadores económicos y los modelos de regresión son algunos de los métodos cuantitativos más empleados.

Para nuestros proyectos trabajaremos en un pronóstico financiero cuantitativo:

- Pronóstico de Solicitudes de Crédito para Financiamiento de autos, utilizando el histórico desde el año 2017 a septiembre 2022

Las solicitudes de Crédito son solicitudes que hacen los clientes finales a una financiera o banco para poder obtener un crédito automotriz, este se basa en un plan de financiamiento, que contempla un enganche y un periodo de mensualidades a pagar, con un interés que puede ser fijo o variable.

Histórico de número de Solicitudes de Crédito generadas por mes del año 2016 a septiembre 2022.

Año	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec	Grand Total
2016					5,260	7,372	7,467	8,243	8,106	7,953	9,638	12,442	66,481
2017	10,790	7,770	8,315	6,657	7,432	8,649	9,008	8,736	8,510	9,125	8,587	7,785	101,364
2018	6,892	5,546	5,782	6,483	6,990	6,252	7,011	7,233	5,664	5,466	6,433	5,919	75,671
2019	5,413	3,492	4,027	3,457	3,388	3,548	4,155	3,803	3,633	3,433	4,075	3,599	46,023
2020	3,592	3,086	2,260	1,043	1,675	2,366	2,535	2,521	2,912	2,917	3,179	3,281	31,367
2021	2,603	2,585	3,313	2,953	3,216	3,670	3,087	2,971	3,161	2,711	2,875	2,765	35,910
2022	2,350	2,973	3,527	2,693	3,475	3,543	3,109	3,695	3,200				28,565

Tabla1. Histórico de Solicitudes de Crédito.

4.2 Descubriendo los datos.

Los datos utilizados para nuestro proyecto serán tomados de diferentes bases de datos y tablas de información:

- SFFXVHC. Catálogo de los diferentes vehículos y marcas que se comercializan.
- SFFXCUS. Catálogo de Clientes. Catálogo de clientes finales.
- SFFXAPP. Tabla con el Histórico Solicitudes de Crédito Automotriz desde el año 2016. Base de datos que contiene todas las solicitudes de crédito que han realizado los clientes que quieren adquirir un vehículo por medio de un crédito automotriz.

Los datos que utilizaremos están almacenados en diferentes lugares, ya que son parte de diferentes procesos, en la fase de preparación de datos, evaluaremos como consolidar toda la información para poderla trabajar en nuestros algoritmos de aprendizaje.

A continuación, presentaremos una descripción detallada de las tablas a utilizar:

SFFXVHC

Campo	Llave	Tipo	Longitud	Nulls	Descripcion
C_FIN_CO	PK1,	CHAR(08)	8	NOT NULL	COMPANY ID
I_CTRCT_VHCL_VIN	PK2	CHAR(17)	17	NOT NULL	Numero de Serie Vehicular VIN
I_CTRCT_VHCL_MY		CHAR(04)	4	NOT NULL WITH DEFAULT	Año modelo del vehiculo: 2015, 2016, ...
I_CTRCT_VHCL_BRND		CHAR(30)	30	NOT NULL WITH DEFAULT	Marca de Vehiculo CHRYSLER ALFA ROMEO MERCEDES BENZ FORD MOTORS COMPANY LAND ROVER …
I_CTRCT_BDY_MOD		CHAR(06)	6	NOT NULL WITH DEFAULT	Codigo del tipo de Vehiculo JC.. FF…
I_CTRCT_CPOS		CHAR(03)	3	NOT NULL WITH DEFAULT	Version del Vehiculo
C_CTRCT_DLR		CHAR(07)	7	NOT NULL WITH DEFAULT	Codigo de Distribuidor: 01M0235 01Q0512 ….
I_UPD		CHAR(07)	7	NOT NULL WITH DEFAULT	ID usuario del ultimo cambio
N_CMPTRPGM_UPD		CHAR(08)	8	NOT NULL WITH DEFAULT	Computadora u programa donde se registre el ultimo cambio
T_STMP_UPD		TIMESTAMP	26	NOT NULL WITH DEFAULT	Fecha de ultimo cambio

Tabla2. Estructura de tabla de Vehículos.

SFFXCUS

Campo	Llave	Tipo	Longitud	Nulls	Descripcion
C_FIN_CO	PK	CHAR(08)	8	NOT NULL	COMPANY ID
C_CUST_NPPI		CHAR(01)	1	NOT NULL WITH DEFAULT	Cliente comparte datos: Space or Y = YES & N = No
N_CUST_FIRST		CHAR(80)	80	NOT NULL WITH DEFAULT	Nombre del Cliente
N_CUST_LAST		CHAR(60)	60	NOT NULL WITH DEFAULT	Apellido Materno Cliente Not apply PM
N_CUST_LAST_2		CHAR(60)	60	NOT NULL WITH DEFAULT	Apellido Materno Cliente Not Apply PM
D_BIRTH		DATE	10	NOT NULL WITH DEFAULT	Fecha de cumpleaños
C_RFC		CHAR(13)	13	NOT NULL WITH DEFAULT	RFC
C_CUST_SEX		CHAR(01)	1	NOT NULL WITH DEFAULT	Genero, Female or Male.
C_FISCAL	FK	CHAR(08)	8	NOT NULL WITH DEFAULT	Fiscal regimen catalog
I_CUST_EMAIL		CHAR(60)	60	NOT NULL WITH DEFAULT	email cliente
C_OCCUPATION	FK	CHAR(08)	8	NOT NULL WITH DEFAULT	Ocupacion Cliente
N_CUST_EMPLMT		CHAR(80)	80	NOT NULL WITH DEFAULT	Empresa donde labora cliente
C_MARITAL	FK	CHAR(08)	8	NOT NULL WITH DEFAULT	Estado Civil
A_CUST_INC		DEC(10,2)	6	NOT NULL WITH DEFAULT	Ingresos Mensuales
I_UPD		CHAR(07)	7	NOT NULL WITH DEFAULT	ID Usuario ultimo cambios
N_CMPTRPGM_UPD		CHAR(08)	8	NOT NULL WITH DEFAULT	Computadora u programa donde se registre el ultimo cambio
T_STMP_UPD		TIMESTAMP	26	NOT NULL WITH DEFAULT	Fecha de ultimo cambio

Tabla3. Estructura de tabla de Clientes.

SFFXAPP

Campo	Llave	Tipo	Longitud	Nulls	Descripcion
C_FIN_CO	PK	CHAR(08)	8	NOT NULL	COMPANY ID
I_CRDAP	PK	CHAR(15)	15	NOT NULL	Numero de Solicitud de Credito
I_CR_CTRCT	FK	CHAR(10)	10	NOT NULL WITH DEFAULT	Numero de contrato
C_ERR		CHAR(03)	3	NOT NULL WITH DEFAULT	Error
C_CUST_NPPI		CHAR(01)	1	NOT NULL WITH DEFAULT	Cliente permite compartir sus datos: **Y = YES & N = No**
D_CRDAP		DATE	10	NOT NULL WITH DEFAULT	Fecha solicitud Credito - **YYYY-MM-DD**
C_CRDAP_STATUS		CHAR(08)	8	NOT NULL WITH DEFAULT	Status de solicitud de credito
I_CRDAP_VHCL_MY		CHAR(04)	4	NOT NULL WITH DEFAULT	Año Modelo Vehiculo
X_CRDAP_VHCL_BRND		CHAR(08)	8	NOT NULL WITH DEFAULT	Marca del vehiculo
I_CRDAP_BDY_MOD		CHAR(06)	6	NOT NULL WITH DEFAULT	Codigo vehiculo
I_CRDAP_CPOS		CHAR(03)	3	NOT NULL WITH DEFAULT	Version Vehiculo
I_CRDAP_SALES_CNCPT		CHAR(03)	3	NOT NULL WITH DEFAULT	Tipo Venta
I_CRDAP_DLR_CODE		CHAR(07)	7	NOT NULL WITH DEFAULT	Codigo dealer
I_UPD		CHAR(07)	7	NOT NULL WITH DEFAULT	ID Usuario ultimo cambios
N_CMPTRPGM_UPD		CHAR(08)	8	NOT NULL WITH DEFAULT	Computadora u programa donde se registre el ultimo cambio
T_STMP_UPD		TIMESTAMP	26	NOT NULL WITH DEFAULT	Fecha de ultimo cambio
C_FLEET		CHAR(01)	1	NOT NULL WITH DEFAULT	Venta Flotilla
P_CR_CAT		DEC(4,2)	3	NOT NULL WITH DEFAULT	CAT Anual
I_CR_XTRA_PAYMT		CHAR(02)	2	NOT NULL WITH DEFAULT	Pagos adicionales
I_CR_CTRCT_TERM		CHAR(02)	2	NOT NULL WITH DEFAULT	Plazo del Credito
C_CR_CTRCT_PLAN		CHAR(03)	3	NOT NULL WITH DEFAULT	Plan de Financiamiento
P_CR_CTRCT_RATE		DEC(4,2)	3	NOT NULL WITH DEFAULT	Tasa de Interes

Tabla4. Estructura de tabla de Solicitudes de Crédito.

Una vez identificado las fuentes de información, vamos a revisar lo que contienen nuestras tablas y visualizar de manera que se pueda identificar si hay algunas inconsistencias en la información.

- Datos de Solicitudes de Crédito (SFFXAPP) Periodo 2016-2022

	Totals	ALFA ROMERO	CHRYSLER	DODGE	FIAT	JEEP	MITSUBISHI	RAM
2016	66481	213	905	30663	7617	5788	9265	9326
2017	101364	316	564	43013	16183	8920	14336	14450
2018	75671	203	322	27175	11306	8611	11129	14031
2019	46023	93	301	16296	5391	6465	667	11724
2020	31367	51	112	8483	5289	5856	4	10681
2021	35910	72	128	8706	6665	6814	-	11939
2022	28582	45	105	8478	3580	5869	-	9304

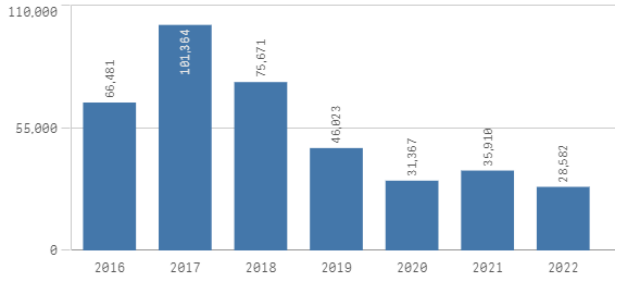


Fig8. Histórico de Solicitudes donde observar en los datos han que se ha tenido una caída en las solicitudes de crédito a partir del 2020, esto sugiere a la pandemia del Covid-19, así mismo no se ven solicitudes de crédito para la marca de Mitsubishi a partir del 2021, esto debido a que dejo de operar. Por otro lado, se ven íntegros los datos.

- SFFXVHC. Catálogo de los diferentes vehículos y marcas que se comercializan.

VIN	Codigo Vehiculo	Version	Marca
MMBNL45K0HH082334	XR471C	Y82	MI
MMBNL45K0HH082379	XR471C	Y82	MI
MMBNL45K0HH082561	XR471C	Y82	MI
MMBNL45K0HH082608	XR471C	Y82	MI
MMBNL45K0HH082611	XR471C	Y82	MI
MMBNL45K0HH082690	XR471C	Y82	MI
MMBNL45K0HH082852	XR471C	Y82	MI
MMBNL45K0HH082947	XR471C	Y82	MI
MMBNL45K0HH082995	XR471C	Y82	MI

Fig9. Tabla Vehículos.

- Catálogo de Clientes. Catálogo de clientes finales.

Nombre	Apellido Paterno	Apellido Materno	Genero	Email	RFC
TRANSPORTES ZAPOTILTENSES SA DE CV	N/A	N/A	N	ZZPOTILTENSES@GMAIL.COM	TZA030331SZ5
BEATRIZ	LUGO	PEREZ	F	zzay2000@hotmail.com	LUPY730423MW5
VICTOR HUGO	ZERMEÃ O	GODINEZ	M	zvico@gmail.com	ZEGV810929001
ELDA JIRED	VALENZUELA	LUQUIN	M	zvanegas65@hotmail.com	VALE841021
HECTOR MIGUEL	ROSADO	MENDOZA	M	zvanegas65@hotmail.com	ROMH520929
MARTHA ANGELICA	DORANTES	LINO	F	zvanegas65@hotmail.com	DOLM730410
ZURA GUADALUPE	LARA	BALLADO	F	ZUURA.LARA@GMAIL.COM	LABZ781021N25
PATRICIA	ARENAL	DELGADO	F	zusetpatricia@outlook.com	AEDP811213
BENITO	ZURITA	DOMINGUEZ	M	ZURYTAA@HOTMAIL.COM	ZUDB720321
JUANA YOSHIRA	FLORES	QUEZADA	M	zurizaday26@hotmail.con	FOQJ8501022P0
ZURISADAI	RIVAS	ORTIZ	F	ZURIVAS18@HOTMAIL.COM	RIOZ930213
JOSE MANUEL	ZURITA	SEGURA	M	ZURITA018@HOTMAIL.COM	ZUSM870307
LUNMO SA DE CV	N/A	N/A	N	zurigr_24@hotmail.com	LUN100215U68
RUBEN ZURIEL	CASTRO	MOLINA	M	zuriel.castro@hotmail.com	CAMR820120SXA

Fig10. Tabla Clientes.

Una vez revisada y explorada la información de los datos, se puede afirmar que estos son completos. Los datos cubren los casos requeridos para la obtención de los resultados necesarios para poder cumplir los objetivos del proyecto. Los datos no contienen errores, ya que son datos generados automáticamente y tienen validaciones de entrada en los sistemas que permite tener una calidad en la información.

Para su manejo los datos serán convertidos a archivos csv. También estandarizaremos los periodos a trabajar, y delimitaremos la información a 2020 hasta septiembre 2022, esto porque estos 3 años, la generación de solicitudes ha tenido factores que han afectado por variables externas como la pandemia y problemas de suministro.

Además, integraremos un consolidado de solicitudes por fecha que nos ayudara a que nuestro modelo sea eficiente y dinámico para así obtener los resultados esperados.

4.3 Escogiendo las mejores herramientas para nuestro proyecto.

Una vez identificado nuestros datos, procedemos a identificar y elegir nuestras herramientas que nos ayudaran a generar los algoritmos de machine learning, este punto es igual de importante que los anteriores porque de esta decisión dependerá los tiempos, costos y esfuerzos.

Una de las decisiones que se tomo es utilizar Python, por sus diversas ventajas, entre las que podemos mencionar son:

Es de licenciamiento de software libre, aprobada por OSI, por lo que se puede usar y distribuir libremente, incluso para uso comercial. La licencia de Python es administrada por Python Software Foundation., esto nos ayuda a tener bajos costos en el desarrollo de nuestro proyecto.

Otro beneficio es que utiliza paquetes integrados para el uso de algoritmos machine learning. Tal es el caso de Scikit-learn este es el paquete de Machine Learning más popular de Python debido a su sencillez y a su variedad de posibilidades de uso. Cuenta con algoritmos de clasificación, regresión, clustering y reducción de dimensionalidad, además existe compatibilidad con otras librerías de Python como NumPy, SciPy.

Se pueden combinar y depuran con otras estructuras de datos y aplicaciones externas como Pandas o PyBrain, y es compatible con otras herramientas de análisis como Codigo R, y de visualización como Qlik Sense.

Una vez tomada la decisión procedemos a su instalación, primero deberemos instalar el proyecto Jupyter y que tiene agregado Python.

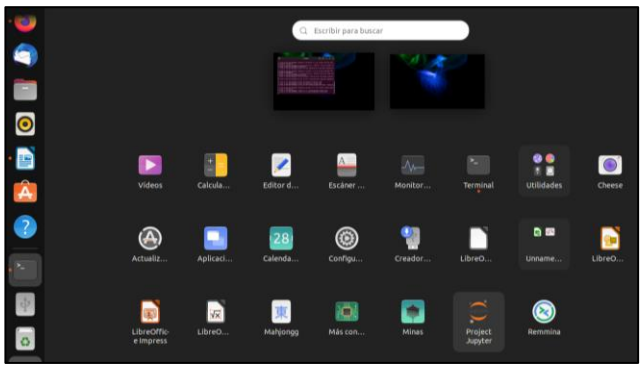


Fig11. Instalación de Project Jupyter.

Una vez instalado, podemos entrar al área de trabajo, donde se guardarán todos nuestros proyectos.

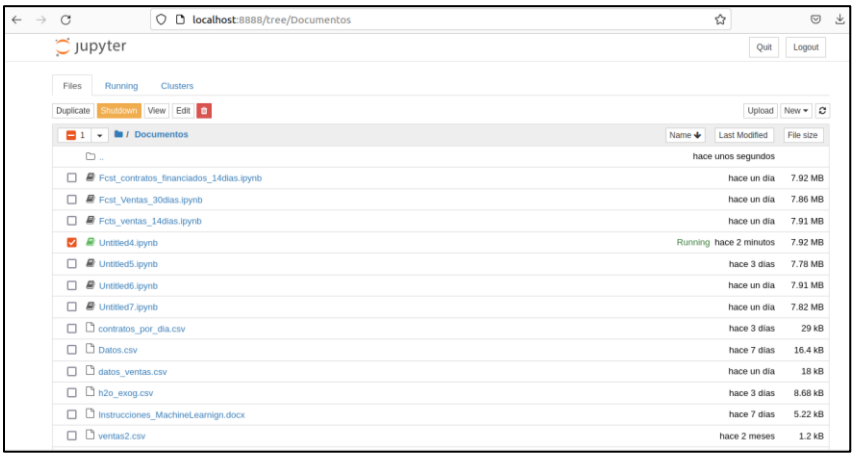


Fig12. Área de trabajo de Júpiter.

Posteriormente debemos instalar todas las librerías necesarias para trabajar en el desarrollo del modelo

pip install pandas
pip install --upgrade pip
pip install pandas==0.17.0 -i https://pypi.tuna.tsinghua.edu.cn/simple
pip install Prophet
pip install --upgrade plotly
pip install pystan==2.19.1.1 prophet
pip install localpip
localpip install fbprophet
pip install fbprophet
pip install gcc
pip install seaborn
pip install -U scikit-learn scipy matplotlib

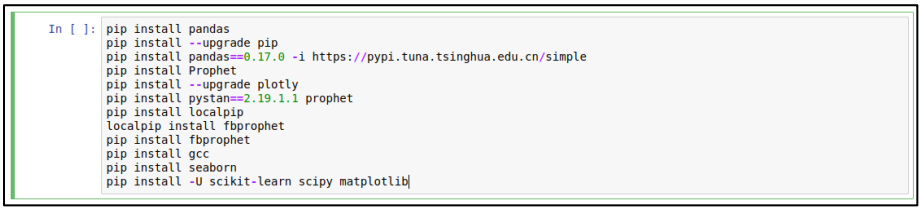


Fig13. Instalación de paquetes y librerías..

4.4 Manos a la obra.

El tipo de aprendizaje de machine Learning que utilizaremos es el supervisado por que tenemos identificado el resultado que deseamos obtener y el modelo que utilizaremos es el de regresión lineal, existen diferentes tipos de regresión lineal, que nos ayuda a comprender y predecir el comportamiento de sistemas complejos o analizar datos financieros o de ventas.

Vamos a iniciar con dos modelos, el modelo autorregresivo y el modelo estadístico (ARIMA), para predecir el número de solicitudes de crédito diarias. Para esto, usaremos el uso de skforecast, una sencilla librería de Python que permite dar solución a los problemas de pronósticos.

Junto con estos modelos y nuestro historial de solicitudes de crédito, vamos a generar un pronóstico capaz de predecir el número de solicitudes que generara la Financiera.

Con el fin de evaluar el modelo, simularemos el proceso completo, evaluando los periodos anteriores y reales simulando la predicción con estos mismos periodos

Ahora mostraremos como se ejecuta nuestro modelo, junto con las imágenes, mostraremos y detallaremos cada uno de los procesos, desde el llamado de las librerías hasta los resultados generados

4.4.1. Primeros pasos del modelo

1. Iniciamos con la importación de nuestras librerías numpy (a soporte para crear vectores y matrices grandes multidimensionales) y pandas(especializada en la manipulación y el análisis de datos, además ofrece estructuras de datos y operaciones para manipular tablas numéricas y series temporales) para nuestro pronostico, también agregamos de librerías para gráficos, pronósticos y aprendizaje.

```
# Iniciamos con la importacion de nuestras librerias numpy(a soporte para crear
# vectores y matrices grandes multidimensionales)
# y pandas(especializada en la manipulación y el análisis de datos ademas
# ofrece estructuras de datos y operaciones para manipular tablas numéricas
# y series temporales) para nuestro pronostico
# ==============================================================================
import numpy as np
import pandas as pd

# Importamos algunas librerias que nos ayudaran con los Gráficos
# ==============================================================================
import matplotlib.pyplot as plt
import seaborn as sns
import hvplot.pandas
%matplotlib inline
from statsmodels.graphics.tsaplots import plot_acf
from statsmodels.graphics.tsaplots import plot_pacf
plt.style.use('fivethirtyeight')

# Importamos librerias para crear los Modelados de Forecasting y aprendizaje
# ==============================================================================
from skforecast.ForecasterAutoreg import ForecasterAutoreg
from skforecast.model_selection import grid_search_forecaster
from skforecast.model_selection import backtesting_forecaster
from skforecast.model_selection_statsmodels import backtesting_sarimax
from skforecast.model_selection_statsmodels import grid_search_sarimax
from sklearn.linear_model import Ridge
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import make_pipeline
from sklearn.metrics import mean_absolute_error

# Por ultimo agregamos la libreria warnings para que nos envie informacion en caso de algun error
# ==============================================================================
import warnings
warnings.filterwarnings('ignore')
```

Fig14. Importación de librerías.

2. Importamos nuestros datos, considerando fecha y numero de solicitudes de crédito generadas, este archivo lo trabajamos en formato *.csv.

```
# Importamos nuestra base de datos que ya trabajamos, utilizando solo fecha y
# acumulado de numero de solicitudes de credito, este archivo lo trabajamos en formato *.csv
# ======================================================================
datos = pd.read_csv('datos.csv', sep =',')
datos.info()
datos.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1004 entries, 0 to 1003
Data columns (total 2 columns):
 #   Column       Non-Null Count  Dtype 
---  ------       --------------  ----- 
 0   date         1004 non-null   object
 1   solicitudes  1004 non-null   int64 
dtypes: int64(1), object(1)
memory usage: 15.8+ KB
```

	date	solicitudes
0	01/01/20	0
1	02/01/20	132
2	03/01/20	98
3	04/01/20	66
4	05/01/20	1

Fig15. Importación de la base de datos de solicitudes de crédito.

3. Ahora convertimos la columna date (fecha de solicitud), por si existieran datos como texto, también validamos nuestro índice que almacenara los datos que este correcto, y validamos que todos los registros se hayan agregado. Verificamos que un índice temporal está completo para determinar mínimos y máximos.

```
# En caso de que alguna fecha este con formato de texto, hacemos una conversión del formato fecha, ademas se indica
# que la serie de datos es de 1dia, y se genera un indice en pandas para hacer uso de las funcionalidades de fechas
# ======================================================================
datos['date'] = pd.to_datetime(datos['date'], format='%d/%m/%y')
datos = datos.set_index('date')
datos = datos.asfreq('1D')
datos = datos.sort_index()
```

```
# Verificamos que un índice temporal está completo para determinar minimos maximos
# ======================================================================
(datos.index == pd.date_range(
                    start = datos.index.min(),
                    end   = datos.index.max(),
                    freq  = datos.index.freq)
                ).all()
```

```
True
```

```
print(f"Valores missing: {datos.isnull().any(axis=1).sum()}")
```

```
Valores missing: 0
```

Fig16. Validación de campo Date.

4. Posteriormente, lo que haremos es separar los periodos por rangos de fecha, que se delimitaran en periodo de entrenamiento, validación y pruebas

```
# Un punto importante es separar los rangos de fechas para identificar los periodos de entrenamiento y de pruebas.
# en este caso delimitaremos las fechas:
# Entrenamiento
# Validacion
# Prueba
# ======================================================================
fin_train = '2022-03-30 23:59:00'
fin_validacion = '2022-06-30 23:59:00'

datos_train = datos.loc[: fin_train, :]
datos_val   = datos.loc[fin_train:fin_validacion, :]
datos_test  = datos.loc[fin_validacion:, :]

print(f"Fechas train      : {datos_train.index.min()} --- {datos_train.index.max()}  (n={len(datos_train)})")
print(f"Fechas validación : {datos_val.index.min()} --- {datos_val.index.max()}  (n={len(datos_val)})")
print(f"Fechas test       : {datos_test.index.min()} --- {datos_test.index.max()}  (n={len(datos_test)})")
```

```
Fechas train      : 2020-01-01 00:00:00 --- 2022-03-30 00:00:00  (n=820)
Fechas validación : 2022-03-31 00:00:00 --- 2022-06-30 00:00:00  (n=92)
Fechas test       : 2022-07-01 00:00:00 --- 2022-09-30 00:00:00  (n=92)
```

Fig17. Validación de campo Date.

5. Ahora mostraremos nuestros datos en una gráfica donde se mostrará la agrupación antes mencionada.

```
# Posteriormente creamos un gráfico boxplot para revisar la estacionalidad mensual y determinar
# que la informacion sea congruente
# ========================================================================
datos['year'] = datos.index.year
boxplot  = datos.sort_values('year').hvplot.box(
               y                = 'solicitudes',
               by               = 'year',
               legend           = False,
               box_fill_color   = None,
               outlier_fill_color = None
           )
lineplot = datos.groupby('year')['solicitudes'].median().hvplot.line(legend=False)
scatterplot = datos.groupby('year')['solicitudes'].median().hvplot.scatter(legend=False)
layout = boxplot * lineplot * scatterplot
layout = layout.opts(title='Distribución de Solicitudes de Credito por año', ylabel='Solicitudes', width=600)
layout
```

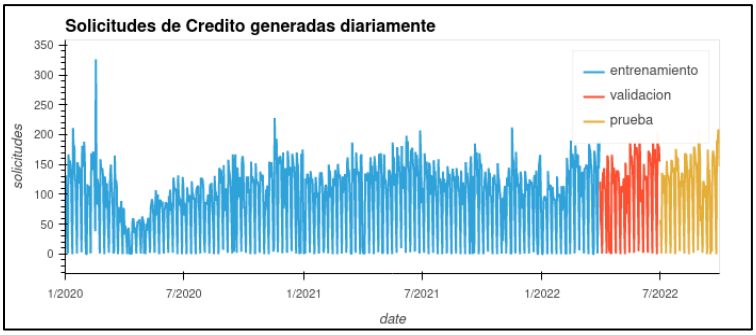


Fig18. Generación de grafica con histórico de solicitudes.

6. Ahora haremos una validación de los datos y revisaremos si hay alguna estacionalidad o un periodo fuera de lo normal, iniciaremos con la revisión de datos por año, misma que se puede observar que ya hay una recuperación del promedio diario de solicitudes de crédito por año.

```
# Posteriormente creamos un gráfico boxplot para revisar la estacionalidad mensual y determinar
# que la informacion sea congruente
# ========================================================================
datos['year'] = datos.index.year
boxplot  = datos.sort_values('year').hvplot.box(
               y                = 'solicitudes',
               by               = 'year',
               legend           = False,
               box_fill_color   = None,
               outlier_fill_color = None
           )
lineplot = datos.groupby('year')['solicitudes'].median().hvplot.line(legend=False)
scatterplot = datos.groupby('year')['solicitudes'].median().hvplot.scatter(legend=False)
layout = boxplot * lineplot * scatterplot
layout = layout.opts(title='Distribución de Solicitudes de Credito por año', ylabel='Solicitudes', width=600)
layout
```

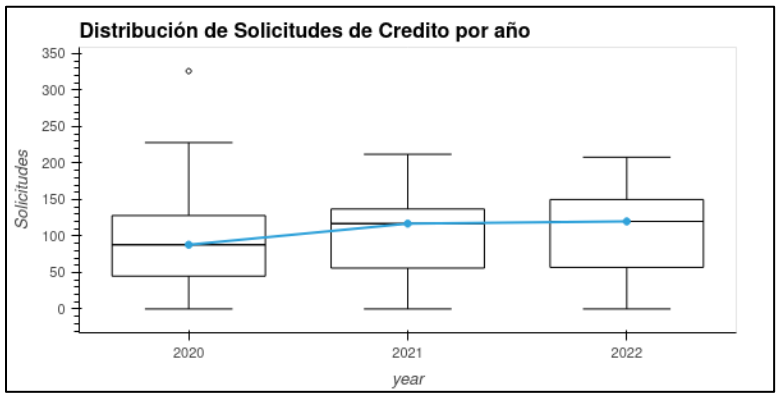


Fig19. Generación de grafica con histórico de solicitudes promedio diario por año.

7. Ahora revisaremos los datos por mes, aquí identificamos que solo abril es un mes de baja generacion del promedio diario de solicitudes de credito, en general todo el año es constante.

```
# Posteriormente creamos un gráfico boxplot para revisar la estacionalidad mensual y determinar
# que la informacion sea congruente
# ================================================================================
datos['mes'] = datos.index.month
boxplot  = datos.sort_values('mes').hvplot.box(
               y                = 'solicitudes',
               by               = 'mes',
               legend           = False,
               box_fill_color   = None,
               outlier_fill_color = None
           )
lineplot = datos.groupby('mes')['solicitudes'].median().hvplot.line(legend=False)
scatterplot = datos.groupby('mes')['solicitudes'].median().hvplot.scatter(legend=False)
layout = boxplot * lineplot * scatterplot
layout = layout.opts(title='Distribución de Solicitudes de Credito por mes', ylabel='Solicitudes', width=600)
layout
```

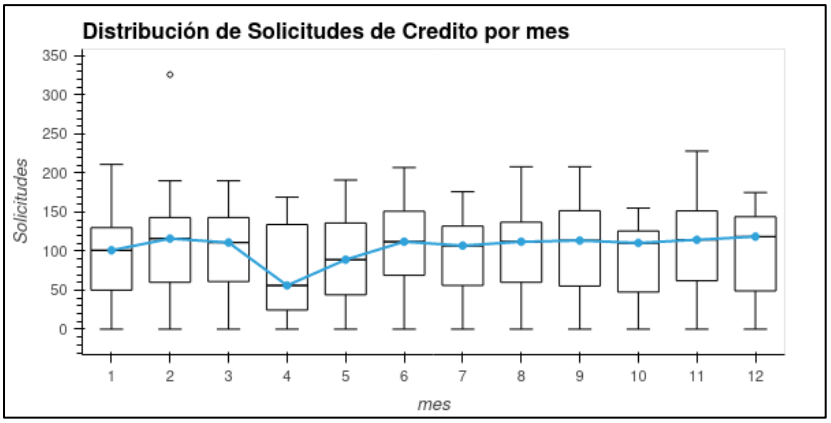


Fig20. Generación de grafica con histórico de solicitudes promedio diario por mes.

8. Ahora por dias del mes aquí podemos observar que inicia el mes con una moderada generacion de solicitudes y hasta la segunda quincena del mes aumentan ligeramente el promedio diario de las solicitudes de credito y tiende a dar una baja hacia el cierre de mes.

```
# Ahora creamos un gráfico boxplot para revisar la estacionalidad diaria por mes y determinar
# que la informacion haga sentido
# ================================================================================

datos['dia_mes'] = pd.Series(datos.index).dt.day.values
boxplot = datos.hvplot.box(
            y                = 'solicitudes',
            by               = 'dia_mes',
            legend           = False,
            box_fill_color   = None,
            outlier_fill_color = None
          )
lineplot = datos.groupby('dia_mes')['solicitudes'].median().hvplot.line(legend=False)
scatterplot = datos.groupby('dia_mes')['solicitudes'].median().hvplot.scatter(legend=False)
layout = boxplot * lineplot * scatterplot
layout = layout.opts(title='Distribución de Solicitudes de Credito por día del mes', ylabel='solicitudes', width=600)
layout
```

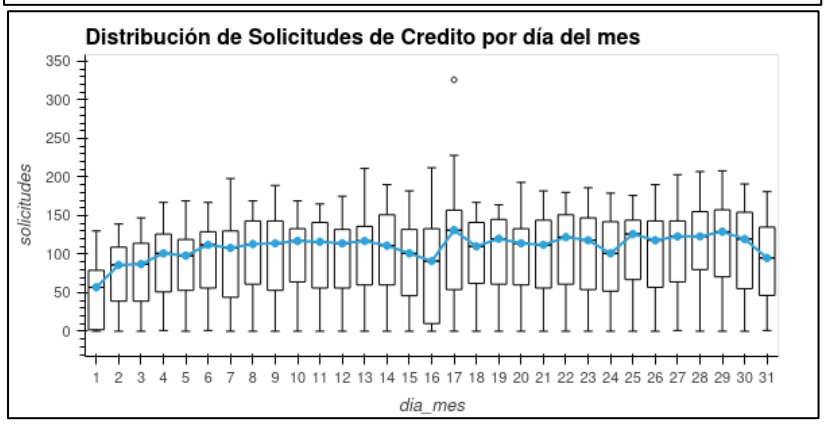


Fig21. Generación de grafica con histórico de solicitudes promedio diario por días del mes.

9. Ahora por dias de la semana, podemos observar que los dias con un mayor promedio de solicitudes diarias son los lunes, y claramente se observa que el fin de semana es muy baja la generacion de solicitudes, y de acuerdo a la experiencia es cuando mas clientes hay en las distribuidoras automotrices, hace suponer que la mayoria de las solicitudes del sabado y domingo las reportan los dias lunes.

```
# Ahora creamos un gráfico boxplot para revisar la estacionalidad por dia de la semana y determinar
# que la informacion haga sentido
# ==============================================================================

datos['dia_semana'] = datos.index.day_of_week + 1
boxplot = datos.sort_values('dia_semana').hvplot.box(
            y                = 'solicitudes',
            by               = 'dia_semana',
            legend           = False,
            box_fill_color   = None,
            outlier_fill_color = None
          )
lineplot = datos.groupby('dia_semana')['solicitudes'].median().hvplot.line(legend=False)
scatterplot = datos.groupby('dia_semana')['solicitudes'].median().hvplot.scatter(legend=False)
layout = boxplot * lineplot * scatterplot
layout = layout.opts(
            title  = 'Distribución de Solicitudes de Credito por día de la semana',
            ylabel = 'solicitudes',
            width  = 600
          )
layout
```

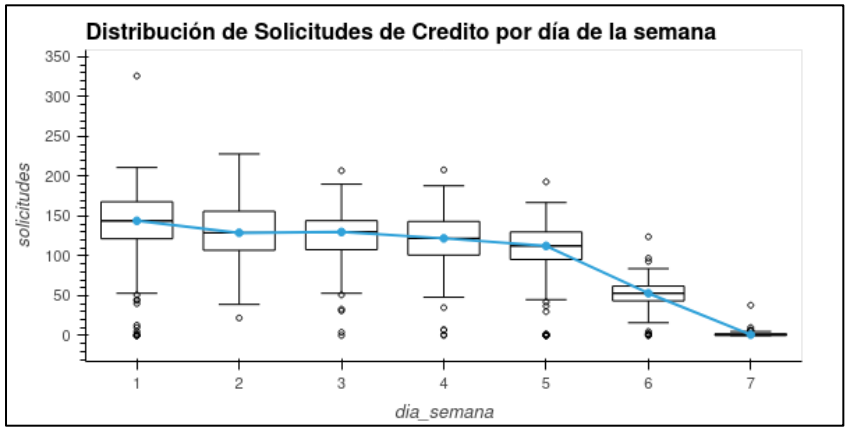


Fig22. Generación de grafica con histórico de solicitudes promedio diario por día de la semana.

4.4.2. Modelo Autorregresivo.

10. Una vez revisado los datos, para identificar posibles desviaciones y estacionalidades, trabajaremos con nuestro primer modelo Autorregresivo. Llamaremos la funcion forecaster para entrenar nuestro modelo, para esto por cada prediccion utilizara los ultimos 14 dias

```
# Ahora se crea un modelo a entrenar autorregresivo
# recursivo a partir de un modelo de regresión lineal con penalización Ridge
# este proceso utiliza 14 Lags, o los ultimos 14 dias en donde se generaron solicitudes de credito
# ==============================================================================
forecaster = ForecasterAutoreg(
                regressor = make_pipeline(StandardScaler(), Ridge(random_state=123)),
                lags      = 14
             )

forecaster.fit(y=datos_train.solicitudes)
forecaster
```

```
================
ForecasterAutoreg
================
Regressor: Pipeline(steps=[('standardscaler', StandardScaler()),
                ('ridge', Ridge(random_state=123))])
Lags: [ 1  2  3  4  5  6  7  8  9 10 11 12 13 14]
Window size: 14
Included exogenous: False
Type of exogenous variable: None
Exogenous variables names: None
Training range: [Timestamp('2020-01-01 00:00:00'), Timestamp('2022-03-30 00:00:00')]
Training index type: DatetimeIndex
Training index frequency: D
Regressor parameters: {'standardscaler__copy': True, 'standardscaler__with_mean': True, 'standardscaler__with_std': True, 'ridg
e__alpha': 1.0, 'ridge__copy_X': True, 'ridge__fit_intercept': True, 'ridge__max_iter': None, 'ridge__normalize': 'deprecated',
'ridge__positive': False, 'ridge__random_state': 123, 'ridge__solver': 'auto', 'ridge__tol': 0.001}
Creation date: 2022-11-10 22:40:47
Last fit date: 2022-11-10 22:40:47
Skforecast version: 0.4.2
```

Fig23. Generacion de pronostico y entrenamiento del modelo.

11. Despues ejecutaremos un proceso donde empieza a realizar las predicciones por semana, donde nos mostrara una muestra de 7 dias que predice el modelo, ademas este proceso nos devuelve una metrica de error (promedio de los errores elevados al cuadrado, entre mas se acerca al cero es menor el error y mas acertada la prediccion) de nuestro modelo contra lo real, a este proceso se le llama backtesting

```
# Despues se evalúa el comportamiento que habría tenido el modelo si se hubiese entrenado
# con los datos desde 2020-07-01 al 2021-06-30 y, después, se realizazan predicciones de 7 en 7 días
# sin reentrenar el modelo. A este tipo de evaluación se le conoce como backtesting,
# Esta función devuelve, además de las predicciones, una métrica de error
# ==============================================================================
metrica, predicciones = backtesting_forecaster(
                            forecaster = forecaster,
                            y          = datos.solicitudes,
                            initial_train_size = len(datos.loc[:fin_validacion]),
                            steps      = 7,
                            refit      = False,
                            metric     = 'mean_absolute_error',
                            verbose    = True
                        )

print(f'Error backtest: {metrica}')
predicciones.head(5)
```

```
Information of backtesting process
----------------------------------
Number of observations used for initial training or as initial window: 912
Number of observations used for backtesting: 92
    Number of folds: 14
    Number of steps per fold: 7
    Last fold only includes 1 observations

Data partition in fold: 0
    Training:   2020-01-01 00:00:00 -- 2022-06-30 00:00:00
    Validation: 2022-07-01 00:00:00 -- 2022-07-07 00:00:00
Data partition in fold: 1
    Training:   2020-01-01 00:00:00 -- 2022-06-30 00:00:00
    Validation: 2022-07-08 00:00:00 -- 2022-07-14 00:00:00
Data partition in fold: 2
    Training:   2020-01-01 00:00:00 -- 2022-06-30 00:00:00
    Validation: 2022-07-15 00:00:00 -- 2022-07-21 00:00:00
Data partition in fold: 3
    Training:   2020-01-01 00:00:00 -- 2022-06-30 00:00:00
    Validation: 2022-07-22 00:00:00 -- 2022-07-28 00:00:00
Data partition in fold: 4
    Training:   2020-01-01 00:00:00 -- 2022-06-30 00:00:00
    Validation: 2022-07-29 00:00:00 -- 2022-08-04 00:00:00
Data partition in fold: 5
    Training:   2020-01-01 00:00:00 -- 2022-06-30 00:00:00
    Validation: 2022-08-05 00:00:00 -- 2022-08-11 00:00:00
Data partition in fold: 6
    Training:   2020-01-01 00:00:00 -- 2022-06-30 00:00:00
    Validation: 2022-08-12 00:00:00 -- 2022-08-18 00:00:00
Data partition in fold: 7
    Training:   2020-01-01 00:00:00 -- 2022-06-30 00:00:00
    Validation: 2022-08-19 00:00:00 -- 2022-08-25 00:00:00
Data partition in fold: 8
    Training:   2020-01-01 00:00:00 -- 2022-06-30 00:00:00
    Validation: 2022-08-26 00:00:00 -- 2022-09-01 00:00:00
Data partition in fold: 9
    Training:   2020-01-01 00:00:00 -- 2022-06-30 00:00:00
    Validation: 2022-09-02 00:00:00 -- 2022-09-08 00:00:00
Data partition in fold: 10
    Training:   2020-01-01 00:00:00 -- 2022-06-30 00:00:00
    Validation: 2022-09-09 00:00:00 -- 2022-09-15 00:00:00
Data partition in fold: 11
    Training:   2020-01-01 00:00:00 -- 2022-06-30 00:00:00
    Validation: 2022-09-16 00:00:00 -- 2022-09-22 00:00:00
Data partition in fold: 12
    Training:   2020-01-01 00:00:00 -- 2022-06-30 00:00:00
    Validation: 2022-09-23 00:00:00 -- 2022-09-29 00:00:00
Data partition in fold: 13
    Training:   2020-01-01 00:00:00 -- 2022-06-30 00:00:00
    Validation: 2022-09-30 00:00:00 -- 2022-09-30 00:00:00

Error backtest: [25.51916069]
```

Fig24. Entrenamiento de nuestro modelo

	pred
2022-07-01	131.541064
2022-07-02	64.262067
2022-07-03	11.401287
2022-07-04	175.208264
2022-07-05	162.972093

Fig25. Muestra de la prediccion.

12. Una vez ejecutado el proceso, procedemos a visualizarlo en una grafica, realizando una comprobacion entre el periodo de prueba y la prediccion

```
# Ahora generamos un gráfico comparando la prediccion  de nuestro modelo autoregresivo contra los datos reales
# ==============================================================================

plot_test = datos_test.solicitudes.hvplot.line(label='test')
plot_predict = predicciones.hvplot.line(label='predicción')
layout = plot_test * plot_predict
layout = layout.opts(
            title = 'Predicción (Autoregresivo) vs vistas reales',
            ylabel = 'solicitudes',
            legend_position = 'bottom_left'
        )
layout
```

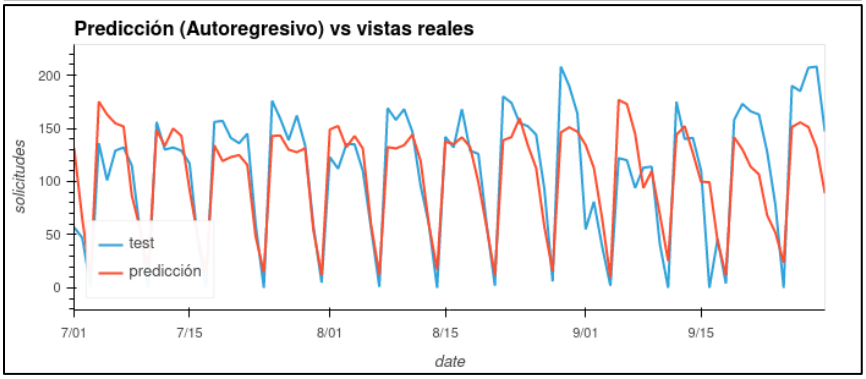


Fig26. Comportamiento de la prediccion vs datos reales.

13. Otra manera de poder mejorar este modelo, es optimizar, combinando los hipermapametros o lags que se generaron y repetir el proceso de backtesting

```
# Optimización de hiperparámetros (tuning)
# Una manera de mejorar nuestro modelo es entrenar un modelo con cada combinación
# de hiperparámetros y lags, y evaluar su capacidad predictiva mediante backtesting.
# a este proceso se le llama Grid search de lags e hiperparámetros
# ==============================================================================
forecaster = ForecasterAutoreg(
                regressor = make_pipeline(StandardScaler(), Ridge(random_state=123)),
                lags      = 14
             )

# Hiperparámetros del regresor: al utilizar un pipeline, el nombre del hiperparámetro
# va precedido por el nombre del regressor.
param_grid = {'ridge__alpha': np.logspace(-3, 3, 10)}

# Lags utilizados como predictores
lags_grid = [7, 14, 21, [7, 14, 21]]

resultados_grid = grid_search_forecaster(
                        forecaster  = forecaster,
                        y           = datos.loc[:fin_validacion, 'solicitudes'],
                        param_grid  = param_grid,
                        lags_grid   = lags_grid,
                        steps       = 7,
                        metric      = 'mean_absolute_error',
                        refit       = False,
                        initial_train_size = len(datos_train),
                        return_best = True,
                        verbose     = False
                    )
```

Fig27. Optimizacion del modelo

Fig28. Ejecucion del proceso de optimizacion del modelo de prediccion.

14. Una vez ejecutado el proceso, ejecutamos los resultados obtenidos

```
# Aqui se muestran los mejores Rresultados del Grid Search, y con estos reentrenamos el modelo con la instruccion
# forecaster
# ==============================================================================
resultados_grid
```

	lags	params	metric	ridge__alpha
20	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14...	{'ridge__alpha': 0.001}	26.147657	0.001000
21	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14...	{'ridge__alpha': 0.004641588833612777}	26.147700	0.004642
22	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14...	{'ridge__alpha': 0.021544346900318832}	26.147896	0.021544
23	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14...	{'ridge__alpha': 0.1}	26.148821	0.100000
24	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14...	{'ridge__alpha': 0.46415888336127775}	26.153102	0.464159
25	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14...	{'ridge__alpha': 2.154434690031882}	26.172896	2.154435
26	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14...	{'ridge__alpha': 10.0}	26.263522	10.000000
30	[7, 14, 21]	{'ridge__alpha': 0.001}	26.544269	0.001000
31	[7, 14, 21]	{'ridge__alpha': 0.004641588833612777}	26.544304	0.004642
32	[7, 14, 21]	{'ridge__alpha': 0.021544346900318832}	26.544468	0.021544
33	[7, 14, 21]	{'ridge__alpha': 0.1}	26.545229	0.100000
34	[7, 14, 21]	{'ridge__alpha': 0.46415888336127775}	26.548759	0.464159
35	[7, 14, 21]	{'ridge__alpha': 2.154434690031882}	26.565092	2.154435
36	[7, 14, 21]	{'ridge__alpha': 10.0}	26.639871	10.000000
27	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14...	{'ridge__alpha': 46.41588833612773}	26.668771	46.415888
37	[7, 14, 21]	{'ridge__alpha': 46.41588833612773}	26.967805	46.415888
10	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]	{'ridge__alpha': 0.001}	27.467387	0.001000
11	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]	{'ridge__alpha': 0.004641588833612777}	27.467443	0.004642
12	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]	{'ridge__alpha': 0.021544346900318832}	27.467702	0.021544
13	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]	{'ridge__alpha': 0.1}	27.468904	0.100000
14	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]	{'ridge__alpha': 0.46415888336127775}	27.474476	0.464159
15	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]	{'ridge__alpha': 2.154434690031882}	27.500215	2.154435
16	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]	{'ridge__alpha': 10.0}	27.617119	10.000000
17	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]	{'ridge__alpha': 46.41588833612773}	28.146916	46.415888
28	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14...	{'ridge__alpha': 215.44346900318823}	28.236276	215.443469
38	[7, 14, 21]	{'ridge__alpha': 215.44346900318823}	28.384875	215.443469
0	[1, 2, 3, 4, 5, 6, 7]	{'ridge__alpha': 0.001}	29.512636	0.001000
1	[1, 2, 3, 4, 5, 6, 7]	{'ridge__alpha': 0.004641588833612777}	29.512680	0.004642
2	[1, 2, 3, 4, 5, 6, 7]	{'ridge__alpha': 0.021544346900318832}	29.512886	0.021544
3	[1, 2, 3, 4, 5, 6, 7]	{'ridge__alpha': 0.1}	29.513842	0.100000
4	[1, 2, 3, 4, 5, 6, 7]	{'ridge__alpha': 0.46415888336127775}	29.518276	0.464159
5	[1, 2, 3, 4, 5, 6, 7]	{'ridge__alpha': 2.154434690031882}	29.538793	2.154435
6	[1, 2, 3, 4, 5, 6, 7]	{'ridge__alpha': 10.0}	29.632703	10.000000
18	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]	{'ridge__alpha': 215.44346900318823}	29.976733	215.443469
7	[1, 2, 3, 4, 5, 6, 7]	{'ridge__alpha': 46.41588833612773}	30.042293	46.415888
8	[1, 2, 3, 4, 5, 6, 7]	{'ridge__alpha': 215.44346900318823}	32.141019	215.443469
29	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14...	{'ridge__alpha': 1000.0}	32.901869	1000.000000
39	[7, 14, 21]	{'ridge__alpha': 1000.0}	33.543268	1000.000000
19	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]	{'ridge__alpha': 1000.0}	35.246118	1000.000000
9	[1, 2, 3, 4, 5, 6, 7]	{'ridge__alpha': 1000.0}	39.458890	1000.000000

Fig29. Resultados del proceso de optimizacion del modelo de prediccion.

15. Volvemos a ejecutar el forecaster para obtener los nuevos resultados

```
forecaster
```

```
=================
ForecasterAutoreg
=================
Regressor: Pipeline(steps=[('standardscaler', StandardScaler()),
                ('ridge', Ridge(alpha=0.001, random_state=123))])
Lags: [ 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21]
Window size: 21
Included exogenous: False
Type of exogenous variable: None
Exogenous variables names: None
Training range: [Timestamp('2020-01-01 00:00:00'), Timestamp('2022-06-30 00:00:00')]
Training index type: DatetimeIndex
Training index frequency: D
Regressor parameters: {'standardscaler__copy': True, 'standardscaler__with_mean': True, 'standardscaler__with_std': True, 'ridg
e__alpha': 0.001, 'ridge__copy_X': True, 'ridge__fit_intercept': True, 'ridge__max_iter': None, 'ridge__normalize': 'deprecate
d', 'ridge__positive': False, 'ridge__random_state': 123, 'ridge__solver': 'auto', 'ridge__tol': 0.001}
Creation date: 2022-11-10 22:41:27
Last fit date: 2022-11-10 22:41:35
Skforecast version: 0.4.2
```

Fig30. Ejecucion del proceso para generar el pronostico.

16. Realizado el forecaster, volvemos a ejecutar el proceso de backtesting, para que nos presente la nueva prediccion y la nueva metrica de error

```
# Una vez identificado el mejor modelo, se entrena utilizando tanto el
# conjunto de entrenamiento como el de validación y se calcula su error con el conjunto de test.
# este proceso lo que intenta es reducir el error de la predicccion(primera comparacion de error
#Autoregresivo-ridge variables exogenas false)
# ==============================================================================
metrica, predicciones = backtesting_forecaster(
                            forecaster = forecaster,
                            y          = datos.solicitudes,
                            initial_train_size = len(datos.loc[:fin_validacion, :]),
                            steps      = 7,
                            metric     = 'mean_absolute_error',
                            verbose    = True
                        )

print(f'Error backtest: {metrica}')
```

```
Information of backtesting process
----------------------------------
Number of observations used for initial training or as initial window: 912
Number of observations used for backtesting: 92
    Number of folds: 14
    Number of steps per fold: 7
    Last fold only includes 1 observations

Data partition in fold: 0
    Training:   2020-01-01 00:00:00 -- 2022-06-30 00:00:00
    Validation: 2022-07-01 00:00:00 -- 2022-07-07 00:00:00
Data partition in fold: 1
    Training:   2020-01-01 00:00:00 -- 2022-06-30 00:00:00
    Validation: 2022-07-08 00:00:00 -- 2022-07-14 00:00:00
Data partition in fold: 2
    Training:   2020-01-01 00:00:00 -- 2022-06-30 00:00:00
    Validation: 2022-07-15 00:00:00 -- 2022-07-21 00:00:00
Data partition in fold: 3
    Training:   2020-01-01 00:00:00 -- 2022-06-30 00:00:00
    Validation: 2022-07-22 00:00:00 -- 2022-07-28 00:00:00
Data partition in fold: 4
    Training:   2020-01-01 00:00:00 -- 2022-06-30 00:00:00
    Validation: 2022-07-29 00:00:00 -- 2022-08-04 00:00:00
Data partition in fold: 5
    Training:   2020-01-01 00:00:00 -- 2022-06-30 00:00:00
    Validation: 2022-08-05 00:00:00 -- 2022-08-11 00:00:00
Data partition in fold: 6
    Training:   2020-01-01 00:00:00 -- 2022-06-30 00:00:00
    Validation: 2022-08-12 00:00:00 -- 2022-08-18 00:00:00
Data partition in fold: 7
    Training:   2020-01-01 00:00:00 -- 2022-06-30 00:00:00
    Validation: 2022-08-19 00:00:00 -- 2022-08-25 00:00:00
Data partition in fold: 8
    Training:   2020-01-01 00:00:00 -- 2022-06-30 00:00:00
    Validation: 2022-08-26 00:00:00 -- 2022-09-01 00:00:00
Data partition in fold: 9
    Training:   2020-01-01 00:00:00 -- 2022-06-30 00:00:00
    Validation: 2022-09-02 00:00:00 -- 2022-09-08 00:00:00
Data partition in fold: 10
    Training:   2020-01-01 00:00:00 -- 2022-06-30 00:00:00
    Validation: 2022-09-09 00:00:00 -- 2022-09-15 00:00:00
Data partition in fold: 11
    Training:   2020-01-01 00:00:00 -- 2022-06-30 00:00:00
    Validation: 2022-09-16 00:00:00 -- 2022-09-22 00:00:00
Data partition in fold: 12
    Training:   2020-01-01 00:00:00 -- 2022-06-30 00:00:00
    Validation: 2022-09-23 00:00:00 -- 2022-09-29 00:00:00
Data partition in fold: 13
    Training:   2020-01-01 00:00:00 -- 2022-06-30 00:00:00
    Validation: 2022-09-30 00:00:00 -- 2022-09-30 00:00:00

Error backtest: [23.80363818]
```

Fig31. Resultados del modelo de prediccion de autorregresion con su metrica de error.

17. Utilizando el mismo modelo y tratando de reducir el error, podemos agregar una varible exógena(son variables no controladas), como analizamos en las graficas anteriores, los fines de semana no tienen el mismo comportamiento que los dias de la semana, asi que podemos excluir el sabado y domingo de nuestras predicciones, mejorando nuestro modelo.

```
# Otra prueba que realizarmos es agregar una variable exogena para nuestro modelo autoregresivo ederivado de que
# en el análisis gráfico mostraba evidencias de que,
# los fines de semana, el número de solicitudes de credito disminuian. El día de la semana
# al que corresponde cada fecha puede saberse a futuro, por lo que se puede emplear como variable exógena.
# ==============================================================================
datos=datos.drop(columns=['mes', 'dia_mes'])
# One hot encoding del día de la semana y la hora del día
datos=pd.get_dummies(datos, columns=['dia_semana'])
datos.head(3)
```

date	solicitudes	year	dia_semana_1	dia_semana_2	dia_semana_3	dia_semana_4	dia_semana_5	dia_semana_6	dia_semana_7
2020-01-01	0	2020	0	0	1	0	0	0	0
2020-01-02	132	2020	0	0	0	1	0	0	0
2020-01-03	98	2020	0	0	0	0	1	0	0

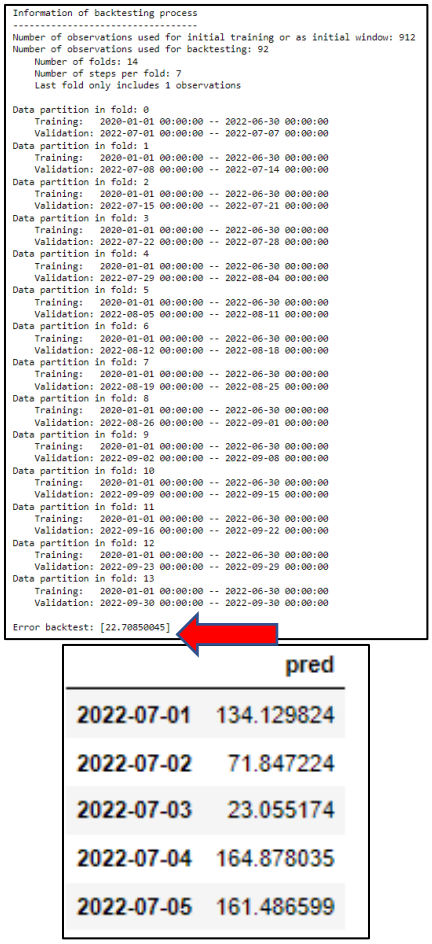
Fig32. Utilizando variables exógenas.

18. Nuevamente hacemos la agrupacion de los periodos, pero ahora excluyendo los fines de semana, y como observaran reduje el margen de error, y solo tomo los datos de lunes a viernes.

```
# realizamos nuestra separacion de periodos, de entrenamiento, validacion y pruebas.
# ==============================================================================
datos_train = datos.loc[: fin_train, :]
datos_val   = datos.loc[fin_train:fin_validacion, :]
datos_test  = datos.loc[fin_validacion:, :]
```

```
# Con los pasos anteriores procedemos a crear el modelo y entrenarlo. Revisamos la
# prediccion y valor del error
# ==============================================================================
forecaster = ForecasterAutoreg(
                regressor = make_pipeline(StandardScaler(), Ridge(alpha=10.0)),
                lags      = 21,
             )

col_exog = [column for column in datos.columns if column.startswith(('dia'))]
forecaster.fit(y=datos_train.solicitudes, exog=datos_train[col_exog])

metrica, predicciones = backtesting_forecaster(
                            forecaster = forecaster,
                            y          = datos.solicitudes,
                            exog       = datos[col_exog],
                            initial_train_size = len(datos.loc[:fin_validacion, :]),
                            steps      = 7,
                            metric     = 'mean_absolute_error',
                            refit      = False,
                            verbose    = True
                        )

print(f'Error backtest: {metrica}')
predicciones.head(5)
```

Fig33. Agrupacion de periodos y ejecucion del pronostico con variables exógenas



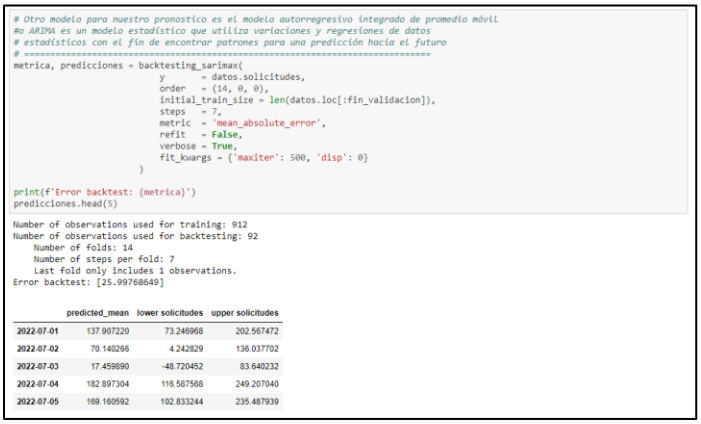
```
Information of backtesting process
----------------------------------
Number of observations used for initial training or as initial window: 912
Number of observations used for backtesting: 92
    Number of folds: 14
    Number of steps per fold: 7
    Last fold only includes 1 observations

Data partition in fold: 0
    Training:   2020-01-01 00:00:00 -- 2022-06-30 00:00:00
    Validation: 2022-07-01 00:00:00 -- 2022-07-07 00:00:00
Data partition in fold: 1
    Training:   2020-01-01 00:00:00 -- 2022-06-30 00:00:00
    Validation: 2022-07-08 00:00:00 -- 2022-07-14 00:00:00
Data partition in fold: 2
    Training:   2020-01-01 00:00:00 -- 2022-06-30 00:00:00
    Validation: 2022-07-15 00:00:00 -- 2022-07-21 00:00:00
Data partition in fold: 3
    Training:   2020-01-01 00:00:00 -- 2022-06-30 00:00:00
    Validation: 2022-07-22 00:00:00 -- 2022-07-28 00:00:00
Data partition in fold: 4
    Training:   2020-01-01 00:00:00 -- 2022-06-30 00:00:00
    Validation: 2022-07-29 00:00:00 -- 2022-08-04 00:00:00
Data partition in fold: 5
    Training:   2020-01-01 00:00:00 -- 2022-06-30 00:00:00
    Validation: 2022-08-05 00:00:00 -- 2022-08-11 00:00:00
Data partition in fold: 6
    Training:   2020-01-01 00:00:00 -- 2022-06-30 00:00:00
    Validation: 2022-08-12 00:00:00 -- 2022-08-18 00:00:00
Data partition in fold: 7
    Training:   2020-01-01 00:00:00 -- 2022-06-30 00:00:00
    Validation: 2022-08-19 00:00:00 -- 2022-08-25 00:00:00
Data partition in fold: 8
    Training:   2020-01-01 00:00:00 -- 2022-06-30 00:00:00
    Validation: 2022-08-26 00:00:00 -- 2022-09-01 00:00:00
Data partition in fold: 9
    Training:   2020-01-01 00:00:00 -- 2022-06-30 00:00:00
    Validation: 2022-09-02 00:00:00 -- 2022-09-08 00:00:00
Data partition in fold: 10
    Training:   2020-01-01 00:00:00 -- 2022-06-30 00:00:00
    Validation: 2022-09-09 00:00:00 -- 2022-09-15 00:00:00
Data partition in fold: 11
    Training:   2020-01-01 00:00:00 -- 2022-06-30 00:00:00
    Validation: 2022-09-16 00:00:00 -- 2022-09-22 00:00:00
Data partition in fold: 12
    Training:   2020-01-01 00:00:00 -- 2022-06-30 00:00:00
    Validation: 2022-09-23 00:00:00 -- 2022-09-29 00:00:00
Data partition in fold: 13
    Training:   2020-01-01 00:00:00 -- 2022-06-30 00:00:00
    Validation: 2022-09-30 00:00:00 -- 2022-09-30 00:00:00

Error backtest: [22.70850045]
```

	pred
2022-07-01	134.129824
2022-07-02	71.847224
2022-07-03	23.055174
2022-07-04	164.878035
2022-07-05	161.486599

Fig34. Muestra de la prediccion con variables exógenas.

4.4.3. Modelo ARIMA

19. Otro modelo que vamos a revisar es el ARIMA, que tambien es un modelo regresivo, y es muy usado en series temporales de tiempo. Para este modelo omitiremos los pasos iniciales de revision de los datos para encontrar estacionalidades y la agrupacion y divicion de periodos, ya que estos pasos ya los realizamos en el modelo anterior. Aquí simplemente ejecutamos el backtesting_sarimax(), para ejecutar nuestra pedicccion.



```
# Otro modelo para nuestro pronostico es el modelo autorregresivo integrado de promedio móvil
#o ARIMA es un modelo estadístico que utiliza variaciones y regresiones de datos
# estadísticos con el fin de encontrar patrones para una predicción hacia el futuro
# ==============================================================================
metrica, predicciones = backtesting_sarimax(
                            y       = datos.solicitudes,
                            order   = (14, 0, 0),
                            initial_train_size = len(datos.loc[:fin_validacion]),
                            steps   = 7,
                            metric  = 'mean_absolute_error',
                            refit   = False,
                            verbose = True,
                            fit_kwargs = {'maxiter': 500, 'disp': 0}
                        )

print(f'Error backtest: {metrica}')
predicciones.head(5)
```

```
Number of observations used for training: 912
Number of observations used for backtesting: 92
    Number of folds: 14
    Number of steps per fold: 7
    Last fold only includes 1 observations.
Error backtest: [25.99768649]
```

	predicted_mean	lower solicitudes	upper solicitudes
2022-07-01	137.907220	73.246968	202.567472
2022-07-02	70.140266	4.242829	136.037702
2022-07-03	17.459890	-48.720452	83.640232
2022-07-04	182.897304	116.587568	249.207040
2022-07-05	169.160592	102.833244	235.487939

Fig35.Ejecucion del Modelo ARIMA

20. Este modelo, tambien nos envia una metrica de error entre la prediccion y lo real. Posteriormente enviamos a graficar el modelo para ver visiblemente las variaciones.

```
# Ahora mostramos grafica el modelo ARIMA vs los datos reales

plot_test = datos_test.solicitudes.hvplot.line(label='test')
plot_predict = predicciones['predicted_mean'].hvplot.line(label='predicción')
layout = plot_test * plot_predict
layout = layout.opts(
            title = 'Predicción (ARIMA) vs vistas reales',
            ylabel = 'solicitudes',
            legend_position = 'bottom_left'
          )
layout
```

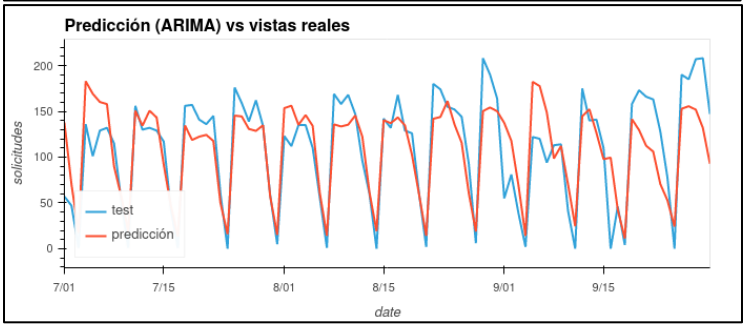


Fig36. Compracion entre real y prediccion en el modelo ARIMA

21. Asi como lo hicimos con el modelo de autorregresion, en este modelo podemos reutilizar los lags combinando los hipermapametros para optimizar los resultados de nuestro modelo.

```
# Los modelos ARIMA tienen una serie de hiperparámetros que controlan su comportamiento:
# p: número de lags incluidos como predictores en el modelo autoregresivo.
# d: número de veces que se diferencian los datos, esto es el número de veces que a cada valor se le
# resta el valor anterior.
# q: tamaño de ventana para la media móvil.


# ==============================================================================
param_grid = {'order': [(14, 0, 0), (14, 2, 0), (14, 1, 0), (14, 1, 1), (14, 1, 4),
                        (21, 0, 0), (21, 0, 0), (21, 1, 0), (21, 1, 1), (21, 1, 4)]}

results = grid_search_sarimax(
            y          = datos.solicitudes,
            param_grid = param_grid,
            initial_train_size = len(datos.loc[:fin_validacion]),
            steps      = 7,
            metric     = 'mean_absolute_error',
            refit      = False,
            verbose    = False,
            fit_kwargs = {'maxiter': 200, 'disp': 0}
          )

results
```

```
root       INFO  Number of models compared: 10
100%|██████████| 10/10 [05:01<00:00, 30.13s/it]
```

	params	metric	order	seasonal_order	trend
9	{'order': (21, 1, 4), 'seasonal_order': (0, 0,...	[23.45706227425837]	(21, 1, 4)	(0, 0, 0, 0)	None
7	{'order': (21, 1, 0), 'seasonal_order': (0, 0,...	[24.216994487027183]	(21, 1, 0)	(0, 0, 0, 0)	None
8	{'order': (21, 1, 1), 'seasonal_order': (0, 0,...	[24.22193393139303]	(21, 1, 1)	(0, 0, 0, 0)	None
5	{'order': (21, 0, 0), 'seasonal_order': (0, 0,...	[24.301097930137438]	(21, 0, 0)	(0, 0, 0, 0)	None
6	{'order': (21, 0, 0), 'seasonal_order': (0, 0,...	[24.301097930137438]	(21, 0, 0)	(0, 0, 0, 0)	None
4	{'order': (14, 1, 4), 'seasonal_order': (0, 0,...	[24.994198236456942]	(14, 1, 4)	(0, 0, 0, 0)	None
3	{'order': (14, 1, 1), 'seasonal_order': (0, 0,...	[25.638250697055405]	(14, 1, 1)	(0, 0, 0, 0)	None
2	{'order': (14, 1, 0), 'seasonal_order': (0, 0,...	[25.750366421648312]	(14, 1, 0)	(0, 0, 0, 0)	None
0	{'order': (14, 0, 0), 'seasonal_order': (0, 0,...	[25.99768649355545]	(14, 0, 0)	(0, 0, 0, 0)	None
1	{'order': (14, 2, 0), 'seasonal_order': (0, 0,...	[30.31832400383429]	(14, 2, 0)	(0, 0, 0, 0)	None

Fig37. Mejorando el modelo ARIMA

22. Una vez optimizados los resultados, nuevamente ejecutamos el backtesting, y revisamos la informacion que predijo, asi como la metrica de error.

```
# Ahora ejecutamos nuestro modelo y revisamos el error(este se compara)
# ==============================================================================
metrica, predicciones = backtesting_sarimax(
                            y       = datos.solicitudes,
                            order   = (21, 1, 1),
                            seasonal_order = (0, 0, 0, 0),
                            initial_train_size = len(datos.loc[:fin_validacion]),
                            steps   = 7,
                            metric  = 'mean_absolute_error',
                            refit   = False,
                            verbose = True,
                            fit_kwargs = {'maxiter': 250, 'disp': 0}
                        )

print(f'Error backtest: {metrica}')
```

```
Number of observations used for training: 912
Number of observations used for backtesting: 92
    Number of folds: 14
    Number of steps per fold: 7
    Last fold only includes 1 observations.
Error backtest: [24.22193393]
```

Fig38. Ejecutando el modelo ARIMA con metrica de error

23. Repitiendo el mismo procedimiento de nuestro modelo anterior,ahora agregaremos las variables exogenas, exluyendo los fines de semana, y ejectuamos nuestro proceso de backtesting.

```
# Ahora hacemos lo mismo pero para el modelo ARIMA, incluimos las variables exógenas y ejecutamos
#la prediccion y el valor del error
# ==============================================================================
metrica, predicciones = backtesting_sarimax(
                            y          = datos.solicitudes,
                            exog       = datos[col_exog],
                            order      = (21, 1, 1),
                            seasonal_order = (0, 0, 0, 0),
                            initial_train_size = len(datos.loc[:fin_validacion]),
                            steps      = 7,
                            metric     = 'mean_absolute_error',
                            refit      = False,
                            verbose    = True,
                            fit_kwargs = {'maxiter': 250, 'disp': 0},
                        )

print(f'Error backtest: {metrica}')
predicciones.head(5)
```

```
Number of observations used for training: 912
Number of observations used for backtesting: 92
    Number of folds: 14
    Number of steps per fold: 7
    Last fold only includes 1 observations.
Error backtest: [24.18110445]
```

	predicted_mean	lower solicitudes	upper solicitudes
2022-07-01	137.769678	76.095293	199.444063
2022-07-02	66.983675	3.642282	130.325069
2022-07-03	13.273259	-50.510982	77.057500
2022-07-04	183.206571	119.251846	247.161295
2022-07-05	171.006619	107.039283	234.973954

Fig39. Ejecutando el modelo ARIMA con variables ex[ogenas asi como su metrica de error

24. Por ultimo ejecutamos nuestra grafica para ver los resultados de nuestro modelo ARIMA y los datos reales, pero adicional nuestro modelo , nos permite agregar resultados o intervalos de la prediccion, que nos permite estar dentro de los rangos reales.

```
# Por ultimo nuestro modelo ARIMA, tambien nos calcula los parametros de intervales como maximos y minimos,
# mismo que podemos observar en nuestra grafica

plot_test = datos_test.solicitudes.hvplot.line(label='test')
plot_predict = predicciones['predicted_mean'].hvplot.line(label='predicción')
plot_intervalo = predicciones.hvplot.area(
                    y     = 'lower solicitudes',
                    y2    = 'upper solicitudes',
                    color = 'red',
                    alpha = 0.2,
                    label = 'intervalo predicción'
                )
layout = plot_intervalo * plot_test * plot_predict
layout = layout.opts(
            title = 'Predicción (ARIMA) vs vistas reales',
            ylabel = 'Solicitudes de Credito',
            legend_position = 'bottom_left'
        )
layout
```

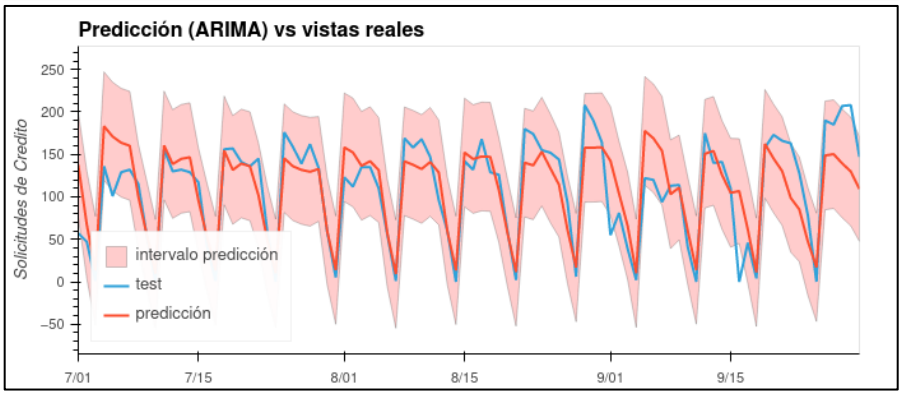


Fig40. Grafica comparando real contra predicccion e Intervalos

Como se pudo observar, la generacion y ejecucion del modelo es rapida, gracias a la ayuda de Scikit-learn porque evita que la programación sea muy compleja gracias a su variedad de módulos y algoritmos que facilitan el aprendizaje y trabajo. En nuestro siguiente capitulo detallaremos la evaluación de los modelos los dos modelos y su aplicación.

Capítulo V

Análisis y Evaluación del

Modelo de Predicción

5.1 Tomando decisiones.

Una vez que ejecutamos los modelos de aprendizaje, podemos evaluar y comparar cual fue el modelo que obtuvo mejores resultados, y para esto tenemos que revisar las metricas de error que arrojo cada proceso. En este caso el modelo Autorregresivo, que utilizo variables exogenas, fue el mejor porque la metrica de error fue tan solo de 22.7085, un mejor numero comparado contra los otros modelos, entre la metrica de error se acerque mas al cero, es un indicativo que nuestro modelo es el mejor.

Modelo	Variables exógenas	Métrica
Autorregresivo	NO	23.8036
Autorregresivo	SI	22.7085
ARIMA	NO	24.2219
ARIMA	SI	24.1811

Tabla5. Comparación de Modelos

Algunos puntos que nos ayudaria a mejorar el modelo son incluir como predictores si el día es festivo y excluirlos, asi como lo hicimos con los fines de semana. Entre mas variables externas quitemos y que podamos controlar la que existan sera mas optimo nuestro modelo.

5.2 Implementando nuestra herramienta.

Una vez definido nuestro mejor modelo, podemos ponerlo a prueba en un ejercicio real.

Este proceso conmumente en una empresa, requiere de mas pruebas, identificando mas periodos, combinaciones de datos, filtrado por marcas, por regiones , entre otros. Ademas de documentacion del proceso, definir periodos y politicas de ejecucion, y sobre todo autorizaciones en diferentes areas y niveles.

Simulando que esta implementacion fuera un éxito, un ejercicio comun seria, predecir el ultimo cuarto del año, cuantas solicitudes de credito estarian generandose.

Esto nos ayuda a presupuestar la inversion de dinero que necesita la financiera para hacerle frente a los compromisos con los clientes, para nuestro ejemplo, usaremos solamente los años 2021 y 2022

Solicitudes de Credito												
	Ene	Feb	Mar	Abr	May	Jun	Jul	Ago	Sep	Oct	Nov	Dic
2021	2,603	2,585	3,313	2,953	3,216	3,670	3,087	2,971	3,161	2,711	2,875	2,765
2022	2,350	2,973	3,527	2,693	3,475	3,543	3,109	3,695	3,200			

Tabla5. Historial de solicitudes del año 2021 y 2022

Ejecutaremos nuestro modelo asignado y veremos los resultados que obtenemos

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import linear_model
from sklearn.metrics import mean_squared_error, r2_score


x = np.array([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21])
y = np.array([2603,2585,3313,2953,3216,3670,3087,2971,3161,2711,2875,2765,2350,2973,3527,2693,3475,3543,3109,3695,3200])
z = np.array([22,23,24]) #Meses que se desean pronosticar

plt.scatter(x,y,label='data', color='blue')
plt.title('Distribución entre meses y solicitudes de credito generadas en los primeros 21 meses');

regresion_lineal = linear_model.LinearRegression()
regresion_lineal.fit(x.reshape(-1,1), y)
print('\nParámetros del modelo de regresión')
print('b (Pendiente) = ' + str(regresion_lineal.coef_) + ', a (Punto de corte) = ' + str(regresion_lineal.intercept_))

pronostico = regresion_lineal.predict(z.reshape(-1,1))

print('\nPronostico:')
print('\n')
for i in range(len(z)):
    print('Pronóstico para el mes {0} = {1} '.format(z[i], pronostico[i]))

pronostico_entrenamiento = regresion_lineal.predict(x.reshape(-1,1))
mse = mean_squared_error(y_true = y, y_pred = pronostico_entrenamiento)
rmse = np.sqrt(mse)
print('\nEvaluación de calidad de la regresión:')
print('\n')
print('Error Cuadrático Medio (MSE) = ' + str(mse))


plt.plot(x,pronostico_entrenamiento,label='data', color='red')
plt.xlabel('Meses')
plt.ylabel('Solicitudes de Credito')
```

Fig41. Ejecucion de pronostico mes 22-24

```
Pronostico:


Pronóstico para el mes 22 = 3289.166666666667
Pronóstico para el mes 23 = 3309.069264069264
Pronóstico para el mes 24 = 3328.9718614718613
```

```
Error Cuadrático Medio (MSE) = 121726.11925376211
```

```
Text(0, 0.5, 'Solicitudes de Credito')
```

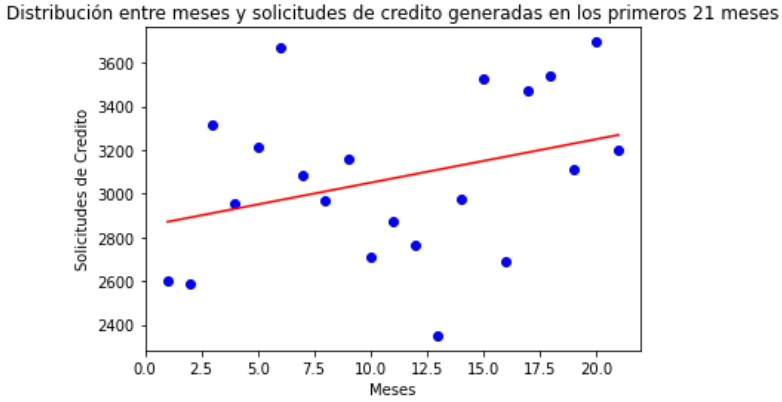


Fig42. Grafica Historial de Solicitudes de Credito primeros 21 meses

En el ejercicio nos hizo una predicción para el mes 22,23 y 24, que equivalen al ultimo cuarto del 2022. También nos generó un error alto, por el numero pequeño de la muestra, sin embargo, el numero pronosticado está dentro de la tendencia real de la generación de las solicitudes de crédito como vemos en la siguiente grafica.

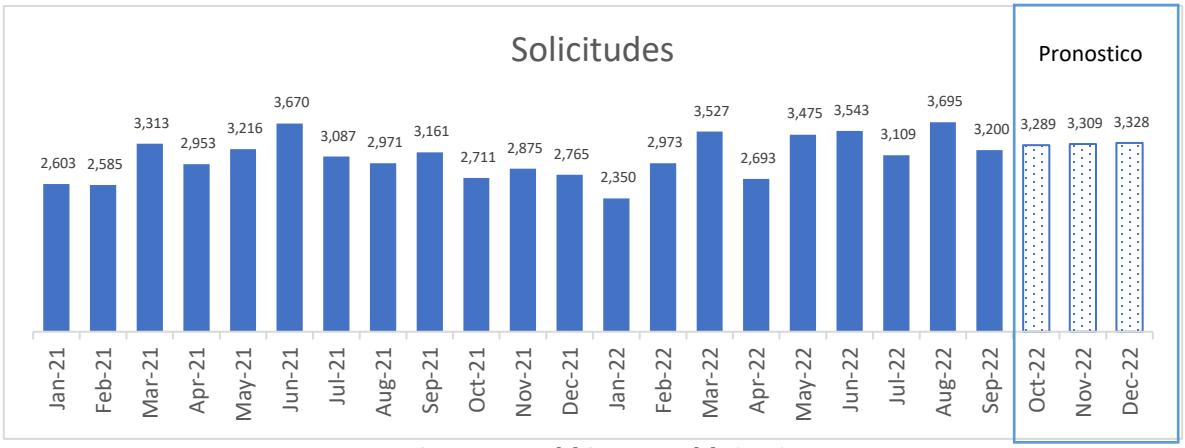


Fig43. Pronostico del 3er cuarto del año 2022

Este tipo de ejercicios, como se mencionó anteriormente es muy importante definir los tiempos de ejecución, entre un pronóstico y el siguiente, lo ideal es generar un presupuesto anual con estas predicciones y cada tres meses hacer pronósticos para identificar posibles factores externos que afecten al pronostico inicial. También cuando se implementa por primera vez es recomendable ejecutarlo cada mes para identificar posibles omisiones o nuevas variantes que modifique nuestro modelo o inclusive cambie por completo el modelo generado. También es recomendable realizar cada 6 meses la prueba con diferentes modelos para que nos permita siempre tener los resultados óptimos.

También es importante y eso dependerá de hasta donde se quiere llegar con el nivel de información y análisis, pero con estos modelos, podemos agregar más variables que nos ayuden a ser más precisos y tener un nivel de detalle más amplio, por ejemplo, agrupar las solicitudes por Marca (Chrysler, Jeep Fiat, otras), detalles por zonas (Norte, sur, etc.), por ciudades, por nivel de ingresos de los clientes, por rango de edades o género, entre otras.

Una ventaja de los modelos de aprendizaje es que estas decisiones se pueden tomar desde el inicio de cada proyecto, o al final realizar los cambios pertinentes, porque así la información lo amerita, como se menciono al inicio el proyecto, nadie estaba preparado para una pandemia o problemas de logística, y se tuvo que hacer toda una reestructura de los análisis de información.

CONCLUSIONES

Conclusión general

La exorbitante cantidad de información que se procesa diariamente en el mundo de la industria Automotriz es interminable, y herramientas como Machine Learning permiten, tener avances tecnológicos y estrategias que mejoren la productividad, las ventas, y sus finanzas.

Hay infinidad de aplicaciones que se pueden desarrollar, desde un pronóstico financiero como el que vimos, hasta un pronóstico de gastos de producción de unidades sin que existan aún, pronosticar anomalías de sensores en las unidades, pronosticar rotaciones de inventario, pronosticar tipos de unidades que los clientes quieren comprar considerando edad, estatus social, región, ingresos etc.

Como efecto del presente trabajo de investigación, se presentó el diseño de un método predictivo para datos contenidos en los registros de solicitudes de financiamiento basado en series de tiempo. El uso de esta técnica nos permitió trabajar con datos del mundo real y simulados, en el que se demuestra que el modelo predictivo es muy efectivo para el caso de estudio, que depende del número de muestras y características, lo cual influye en la predicción.

Pero también es cierto y es importante mencionar que este tipo de técnicas son muy complejas y que, a pesar de usar software de licenciamiento gratuito, el equipo tecnológico donde deben ejecutarse estos modelos son de alta eficiencia, rápido procesamiento y alto costo, sin mencionar que las personas que estén involucras en estos proyectos deben tener amplios conocimientos, tanto en el negocio, como en programación y estadística, lo que hace que los proyectos puedan tener un impacto económico muy alto. En el actual proyecto de hecho se tuvo que delimitar tanto los registros como periodos y ciertas características de los datos, por tener un problema de capacidad del equipo en donde se corrieron los modelos.

El proyecto nos ayudó a prevenir gastos innecesarios, a tener un mejor control de los recursos, y sobre todo a tener mejor panorama del negocio, porque permite tomar decisiones de forma oportuna, en los canales adecuados, permite hacer correcciones o mejoras a los procesos, permite conocer los riesgos y ayuda a tener una mejor productividad, mejores ganancias y reducción gastos, lo que representa una mejor planeación financiera y cumplimiento de los objetivos.

Podemos concluir que Machine Learning ayuda a la Industria Automotriz para seguir creciendo en términos de innovación, eficiencia, movilidad, sustentabilidad, seguridad, calidad y sobre todo entregar un producto adecuado a las preferencias de los consumidores finales, pero por el momento no todos los negocios que involucran esta industria están preparados económica y tecnológicamente para desarrollar estas herramientas.

Recomendaciones futuras.

Se considera investigar más variables que guardan relación directa e indirectamente con el perfil del cliente, así como, ingresos, genero, región, entre otras y se propone optimizar mejoras en la predicción

Se propone analizar con mayor detenimiento a las desviaciones y la temporalidad que dará origen de otras investigaciones como la satisfacción del cliente y sobre todo el entender sus necesidades.

Ampliando las variables y características, con más casos adicionales y atributos contenidos en los registros ayudaría a optimizar las predicciones mitigando aún más los errores al predecir.

Referencias:

Andrea De Mauro, M. G. (10 de 7 de 2020). *Emerald logo*. Obtenido de A formal definition of Big Data based on its essential features: https://www.emerald.com/insight/content/doi/10.1108/LR-06-2015-0061/full/html

Bancomex. (7 de 10 de 2021). *https://www.bancomext.com/.* Obtenido de https://www.bancomext.com/: https://www.bancomext.com/wp-content/uploads/2018/11/Libro-Fintech.pdf

Brian, R. R. (26 de 10 de 2020). *Una introducción a los modelos de Machine Learning.* Obtenido de BUAP: https://repositorioinstitucional.buap.mx/handle/20.500.12371/10527

Caparros, 1. E. (7 de 10 de 2021). *Sociedad Española de Medicina de Laboratorio.* Obtenido de Sociedad Española de Medicina de Laboratorio: https://www.seqc.es/download/revista/388/1224/772147163/1024/cms/Qu%C3%ADmica%20Cl%C3%ADnica%201994;13%20(5)%20221-228.pdf/

Dr. Alex Liu, P. D. (7 de 10 de 2021). *researchmethods.* Obtenido de Data Science and Data Scientist: http://www.researchmethods.org/DataScienceDataScientists.pdf

Gartner. (7 de 10 de 2021). *Glosario de Gartner.* Obtenido de Glosario de Gartner: www.gartner.com

Gil, D. Á. (21 de 01 de 2021). *Metodología CRISP-DM.* Obtenido de Adictos al trabajo: https://www.adictosaltrabajo.com/2021/01/14/metodologia-crisp-dm/

Macrodatos e inteligencia de datos, alternativas a big data. (7 de 10 de 2021). Obtenido de Fundeu RAE: https://www.fundeu.es/recomendacion/macrodatosalternativa-abig-data-1582/

Martínez, B. B. (2021). *MINERÍA DE DATOS.* Puebla: http://bbeltran.cs.buap.mx/NotasMD.pdf.

Norvig, S. J. (2021). *Inteligencia Artificial: Un Enfoque Moderno.* Mexico: 1995.

Parmenter, D. (2021). *Key Performance Indicators: Developing, Implementing, and Using Winning KPIs.* John Wiley & Sons.

ROUHIAINEN, L. (2018). *Inteligencia artificial - 101 cosas que debes saber hoy sobre nuestro futuro.* Barcelona: https://www.planetadelibros.com/libros_contenido_extra/40/39307_Inteligencia_artificial.pdf.

Timón, C. E. (7 de 10 de 2021). *Análisis predictivo: técnicas y modelos utilizados y aplicaciones del mismo.* Obtenido de openaccess: http://openaccess.uoc.edu/webapps/o2/bitstream/10609/59565/6/caresptimTFG0117memm%C3%B2ria.pdf

Villalba, F. (1 de 10 de 2018). *Aprendizaje supervisado en R.* Obtenido de fervilber: https://fervilber.github.io/Aprendizaje-supervisado-en-R/