

## Tabla de Contenidos

1.1. Diseño Metodológico .....	5
1.2. Requerimientos .....	6
1. TRATAMIENTO DE LOS DATOS.....	8
2.1. Procesamiento .....	9
2.1.1. Construcción de programas Python para validación datos.....	9
2.1.2. Técnicas de limpieza de datos (Valores atípicos).....	10
2.1.4. Extracción de variables explicativas.....	14
2.1.5. Preselección de variables explicativas.....	15
2. INTRODUCCIÓN A LOS MODELOS .....	17
3.1. Definición de modelos de pronóstico de serie de tiempo .....	19
3.2. Características de modelos de serie de tiempo .....	19
3.3. Tipos de Modelos de pronósticos de Series de Tiempo.....	20
3.3.1. Modelo Arima .....	21
3.3.2. Modelo LSTM .....	21
3.3.3. Modelo Random Forest.....	22
3.3.4. Modelo Prophet .....	23
3.3.5. Modelo SARIMAX.....	24
3.3.6. Modelo Teoría de Colas .....	25
3.3.7. Modelo de Bosque aleatorio .....	26
3.4. Definición de métricas adecuadas de desempeño.....	26
8. PROCESOS DE CALIBRACIÓN O ENTRENAMIENTO DE CADA MODELO .....	30
4.1. Modelos .....	30
4.1.1. Modelo Prophet : .....	33
4.1.2. Modelo Teoría de Colas:.....	35
4.1.3. Modelo Bosque Aleatorio:.....	36
4.1.4. Modelo Arima:.....	37
4.1.5. Modelo LSTM_Predictions .....	39
9. RESULTADOS – DISCUSIONES .....	41
10. IMPLEMENTACIÓN DEL MODELO SELECCIONADO .....	43

<b>6.1. Requerimiento 1: Herramienta descriptiva (Tablero de Mando de Históricos para tiempos de Espera).....</b>	<b>43</b>
<b>6.2. Requerimiento 2: Herramienta de predicción de tiempos de atención (Aplicativo web) .....</b>	<b>44</b>

### Lista de tablas

Tabla 1 + Nombre de tu tabla. ....**¡Error! Marcador no definido.**

### Lista de figuras

Figura 1. Si pusiste alguna forma en tu texto.....**¡Error! Marcador no definido.**

### 1.1. Diseño Metodológico

Para el presente trabajo se aplica un enfoque de Big Data; Extracción, transformación y carga (ETL), Se recopilaron datos históricos de pacientes atendidos durante un periodo de tiempo con manejo de datos cualitativos y cuantitativos que den respuesta a los objetivos planteados y que permitan abordar el problema descrito.

Con lo anterior se estableció un orden de actividades en primer lugar la recopilación de los datos como son datos de tiempo de espera de atención de urgencias, clasificación del Triage, régimen, entidad, edad, sexo. con la información obtenida de la atención de urgencias se estableció las variables de cada una de las interdependencias más relevantes para el estudio del caso, seguido se recopiló los tiempos de cada una de las operaciones realizadas en la atención de los pacientes, desde el ingreso al servicio de urgencias hasta su atención médica por el profesional de la salud.

Para tener claro y como guía de observación fue necesario conocer el procedimiento establecido por la Subred en la atención de urgencias para determinar las variables para la realización de la medición de los tiempos en cada una de las operaciones realizadas en la atención, posteriormente las cargamos la información a la herramienta Power BI y repositorio de datos y de los fuentes (Python) en la ruta : <https://github.com/Vitotoju/Compensar>; con el fin de poder visualizar de una manera más ágil los datos y que nos permitirán tener un contexto global de la información en el área de urgencias y así mismo determinar las variables más

relevantes que influyen en los tiempos de espera para ser utilizadas en el modelo de machine learning.

## 1.2. Requerimientos

El proyecto pretende entregar una herramienta que, al aplicarla sobre las atenciones de los pacientes de urgencias, permita identificar alertas sobre los tiempos de atención históricos y posibles problemas en sus características actuales y predicciones sobre si terminarían con dificultades en su cierre, conllevando a generar mejoras en la atención de los tiempos de pacientes. Estas alertas se generarán a partir del entendimiento de los datos históricos, donde se realizarán comparaciones entre lo realmente ejecutado y la capacidad instalada en esas áreas de urgencias; para así identificar las características entre los datos y poder detectar el motivo porque están aumentando los tiempos de atención de pacientes en el área de urgencias

A continuación, se definen los requerimientos del negocio:

**Tabla 1: Requerimientos de negocio**

Aspecto	Nombre	Requerimiento
R1	Herramienta descriptiva	Desarrollar una herramienta que proporcione a los usuarios información de los tiempos de espera históricos en Triage por meses, días, semanas y turnos el cual se podría utilizar para mejorar la atención de los pacientes en minutos.

R2	Herramienta de predicción de tiempos de atención	Desarrollar una Herramienta que permita predecir los tiempos de espera mínimo a 1 mes, el cual se podría utilizar como alerta para tomar decisiones para mejorar la eficiencia de las filas y tiempos de espera.
----	--	--

## 1. TRATAMIENTO DE LOS DATOS

A partir de los requerimientos planteados y validando las fuentes de datos (Servidor de Dinámica Gerencial – Software institucional ERP), se decide tomar como fuente de información la base de datos entregada por Subred Sur ESE.

A partir de esta base de datos se crean varias sentencias SQL en el motor de Base de Datos y se realizan varios procesos ETL que se detallan a continuación:

- Se crea sentencia SQL inicial llamada “Tiempos de Triage en Área de Urgencias” en el servidor de pruebas con las siguientes variables (19 campos

*("TIEMPO\_LLEGADA\_A\_TRIAGE", "TIEMPO\_TRIAGE\_A\_INGRESO", "TIEMPO\_INGRESO\_A\_FOLIO", "TIEMPO\_TOTAL", "FECHA\_LLEGADA", "FECHA\_TRIAGE", "FECHA\_INGRESO", "FECHA\_CONSULTA", "CENTRO\_ATENCION", "CLASIFICACION\_TRIAGE", "PACIENTE\_#\_DOCUMENTO", "PACIENTE\_EDAD", "EDAD\_RANGO", "SEXO", "RÉGIMEN PACIENTE", "NOMBRE\_ENTIDAD", "NOM\_TIPO\_HISTORIA", "DIAGNOSTICO", "NOMBRE DX" , con el objetivo de extraer los datos principales y usar de la base de datos y así poder analizar las posibles variables*

- Se crea esta sentencia SQL final llamada : “Tiempos Total de tiempos de Triage ”, el cual se decidió debido a la necesidad del proyecto y así usar solamente las requeridas para el caso que estamos llevando (10 variables) :

*"FECHA\_LLEGADA", "TIEMPO\_TOTAL\_FINAL", "CENTRO\_ATENCION", "CLASIFICACION\_TRIAGE", "PACIENTE\_EDAD", "PACIENTE\_#\_DOCUMENTO", "EDAD\_RANGO", "NOMBRE\_ENTIDAD", "SEXO", "Month"*



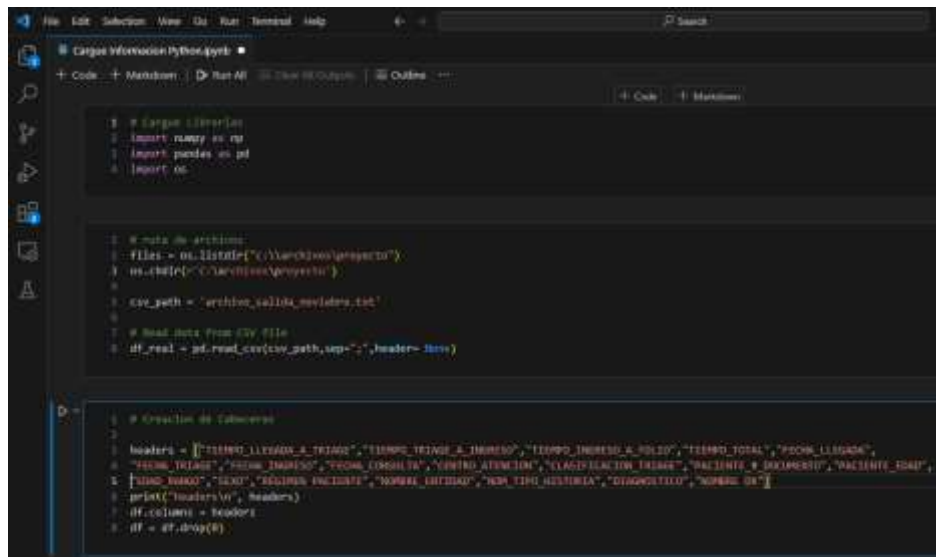
## 2.1. Procesamiento

Para realizar todo el procesamiento de la información se utilizan varias herramientas como son: “Visual Studio Code”, Power BI, Streamlit. GitHub; para así realizar todos los temas de cargue, validación, transformación, creación de pantallas, visualización de datos, creación de informes y gráficas, creación de tableros y creación de un aplicativo online para ser visualizado en la intranet.

### 2.1.1. Construcción de programas Python para validación datos

Inicialmente se decide crear varios programas en Python para poder validar los datos y su relación con las demás variables como son:

- Programa cargue y validación de la información (Cargue Información Python.ipynb):



```

1 # Cargue Librerías
2 import numpy as np
3 import pandas as pd
4 import os

5
6 # Ruta de archivos:
7 files = os.listdir("C:\\archivos\\proyecto")
8 os.chdir("C:\\archivos\\proyecto")
9
10 csv_path = 'archivos_salida_noviembre.csv'
11
12 # Read data from CSV file
13 df_real = pd.read_csv(csv_path, sep=";", header= None)
14
15
16 # Creación de Columnas
17
18 headers = ["TIEMPO_LLAMADA_A_TRIAGE", "TIEMPO_TRIAGE_A_INGRESO", "TIEMPO_INGRESO_A_PULSO", "TIEMPO_TOTAL", "PROM_LLAMADA",
19 "PROM_TRIAGE", "PROM_INGRESO", "PROM_CONSULTA", "PROM_ATENCION", "CLASIFICACION TRIAGE", "PACIENTE + DOCUMENTO", "PACIENTE REAL",
20 "PROM_PULSO", "PROM", "RESUMEN ENFERMEDAD", "TEMPERATURA", "PROM TIPO ALERGIAS", "DIAGNOSTICO", "NOMBRE DO"]
21
22 print(headers)
23 df.columns = headers
24 df = df.drop(h)
  
```

Figura 1 – Programa Python – Cargue de Información

- EjerciciosPracticos\_Limpieza.ipynb
- Modelo de teoría de colas - 1 oct.ipynb
- Modelos\_de\_Regresión.ipynb
- Varios\_Modelos\_de\_series\_de\_tiempo. ipynb
- Varios\_Modelos\_Otros.ipynb
- Modelos \_time\_series\_forecasting\_arima\_lstm\_Random\_forest\_prophet. Ipynb
- Version\_final\_modelos\_prediccion. ipynb

Con los programas anteriores (Python) se realizaron diferentes pruebas de procesamiento:

- Técnicas de limpieza de datos (Valores atípicos, imputación de valores nulos)
- Extracción de variables explicativas

### **2.1.2. Técnicas de limpieza de datos (Valores atípicos)**

Producto del procesamiento de los datos realizado por medio de los programas de python y como resultado se evidencia franjas de datos o valores atípicos que afectan el resultado de los modelos a predecir ver (*Figura 2 – Gráficas de densidad variables (Tiempo total, Hora, Día, Mes)*) a los archivos planos generados de las consultas de la base de datos, teniendo en cuenta el gran volumen del dataset y con la intención de enfocar el esfuerzo del trabajo para que los resultados puedan ser interpretables, se filtran y seleccionan los tiempos en minutos mayores a 0 y menores a 420 eliminando del análisis otros tiempos en minutos como los minutos menores a 0( son registros que no contienen fechas,) y mayores a 420 minutos (ósea a 7 horas de espera) que

creemos que se ajusta a los modelos que queremos predecir, estos datos se deben que no fueron ingresadas en el sistema en mismo momento de atención del proceso.

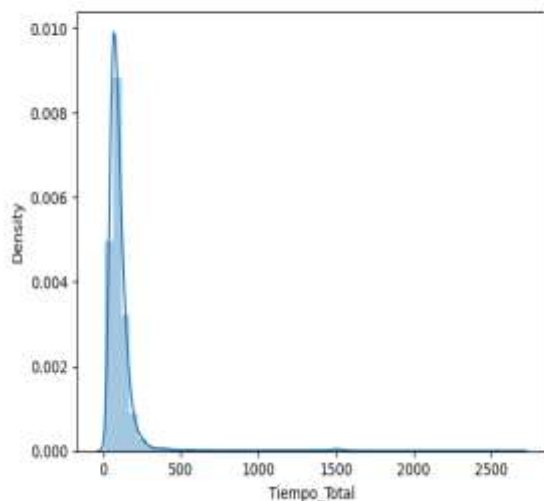
Este proceso reduce los registros a 82.416 de las atenciones realizadas en el año 2023 de la SUBRED SUR ESE.

Se buscan valores atípicos en las diferentes variables seleccionadas usando graficas de densidad y cajas de valor:

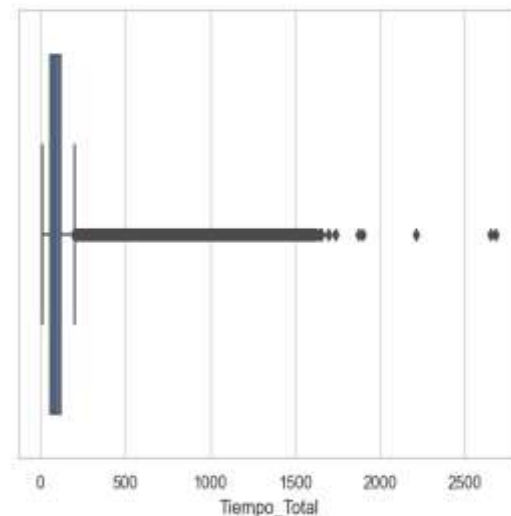
- Tiempo Total
- Horas
- Día
- Mes

### Gráficas: Valores atípicos en columnas numéricas

```
sns.distplot(df["Tiempo_Total"])
<Axes: xlabel='Tiempo_Total', ylabel='Density'>
```



```
<Axes: xlabel='Tiempo_Total'>
```



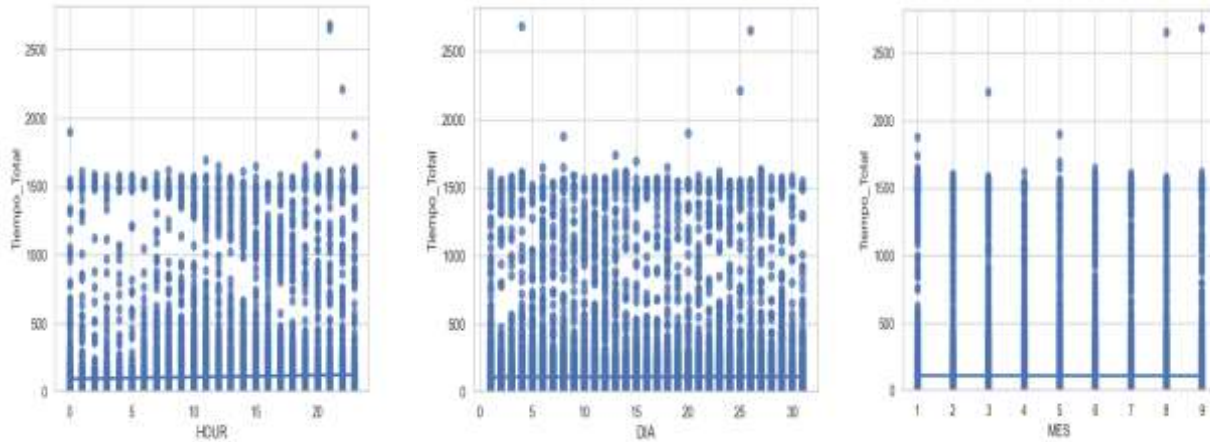


Figura 2 – Graficas de densidad variables (Tiempo total, Hora, Día, Mes)

### 2.1.3. Técnicas de limpieza de datos (Imputación valores nulos)

Producto del procesamiento de los datos realizado por medio de los programas de python y como resultado de las técnicas realizadas se evidencia los siguientes registros con imputaciones para valores nulos (ver Tabla 3).

Algunas de las técnicas realizadas para detectar campos null en el dataframe creado en python es:

- `print(df.isnull().sum())`
- `# Para ver todas las filas que tienen valores faltantes`  
`is_NaN = df.isnull()`  
`row_has_NaN = is_NaN.any(axis=1)`  
`rows_with_NaN = df[row_has_NaN]`  
`rows_with_NaN.head(100)`

En la tabla siguiente se presenta el proceso de identificación de valores nulos en el dataframe y además se presenta el tratamiento realizado a los valores nulos encontrados.

**Tabla 3 : Imputación de valores nulos**

Columna	Non-Null	Tratamiento
TIEMPO_TOTAL	5200	Se reemplaza por la media de la misma columna
FECHA_LLEGADA	7830	Se reemplaza con los datos FECHA_TRIAGE
FECHA_TRIAGE	57320	Se reemplaza con los datos FECHA_INGRESO
FECHA_INGRESO	777	Se reemplaza con los datos FECHA_ATENCION
FECHA_ATENCION	15	Se reemplaza con los datos FECHA_INGRESO
MEDICO	15	Se reemplaza por la media de la misma columna
CENTRO_ATENCION	35	Se reemplaza por el centro de atención del MEDICO tratante
NOMBRE_ENTIDAD	7830	Se reemplaza por el valor más común de la sede encontrada.

Todos los campos que se encontró registros con valores NULL en este caso la variable NOMBRE\_ENTIDAD se reemplaza por el dato más común según la sede que fue atendido el paciente, de esta manera no hay forma de identificar o sustituir los datos por los reales y este

reemplazo del dato no afecta la información que están aportando para el correcto desarrollo de este proyecto.

Las fechas de variables faltantes se reemplaza por la fecha siguiente encontrada en el proceso así:

- FECHA LLEGADA, si tiene campo null se reemplaza por la siguiente fecha existente que es la FECHA TRIAGE
- FECHA TRIAGE, si tiene campo null se reemplaza por la siguiente fecha existente que es la FECHA INGRESO
- FECHA INGRESO, si tiene campo null se reemplaza por la siguiente fecha existente que es la FECHA ATENCION
- FECHA\_ATENCION se reemplaza por la fecha anterior encontrada (ósea FECHA INGRESO), se logra evidenciar que estas fechas normalmente son las mismas o tiene una varianza entre 10 y 15 minutos que no afectan su información.

#### **2.1.4. Extracción de variables explicativas**

Producto del procesamiento realizado se obtienen las siguientes variables explicativas, después de eliminar las variables que contenían un gran porcentaje de nulos y las que no aportaban información para la pregunta de negocio:

**Tabla 4: Variables explicativas**

#	Columna	Import a	#	Columna	Import a
1	FECHA_TOTAL	A	5	DIA	B
2	MEDICO	A	6	TURNOS	B
3	AÑO	B	7	HORA	B
4	MES	B	8	DIA_SEMANA	B

### 2.1.5. Preselección de variables explicativas

Para seleccionar las variables más importantes para el proyecto de análisis de eficiencia en la contratación y desarrollar una herramienta que permita realizar las predicciones planteadas con los datos de La SUBRED SUR ESE, específicamente se seleccionaron los años del 2022 al 2023 debido a que estos años contaban con información típica. El año 2020 fue descartado debido a que por motivos del COVID19 el comportamiento de los contratos no tuvo un comportamiento normal, lo que podría afectar la precisión de los modelos al incluir datos atípicos.

Inicialmente, sin los filtros, la base de datos contenía **1.375.710** filas y 19 columnas o variables como se mencionó en el capítulo 6, después del tratamiento o procesamiento de datos se eliminaron columnas duplicadas que podrían generar datos erróneos al correr los diferentes modelos o que no aportaban información a la pregunta de investigación.

Después de las validaciones y pruebas de los datos y de correlacionar todas las variables para determinar si aportaba información importante no redundante con respecto a otras variables existentes. Esto llevó a la selección de las siguientes variables en un principio para así aplicarlas en la siguiente etapa que es crear algoritmos de predicción para calcular el mejor tiempo de atención de urgencias:

- "FECHA\_LLEGADA", "TIEMPO\_TOTAL\_FINAL", "CENTRO\_ATENCION", "CLASIFICACION\_TRIAGE", "PACIENTE\_EDAD", "PACIENTE\_#\_DOCUMENTO", "EDAD\_RANGO", "NOMBRE\_ENTIDAD", "SEXO", "Month"

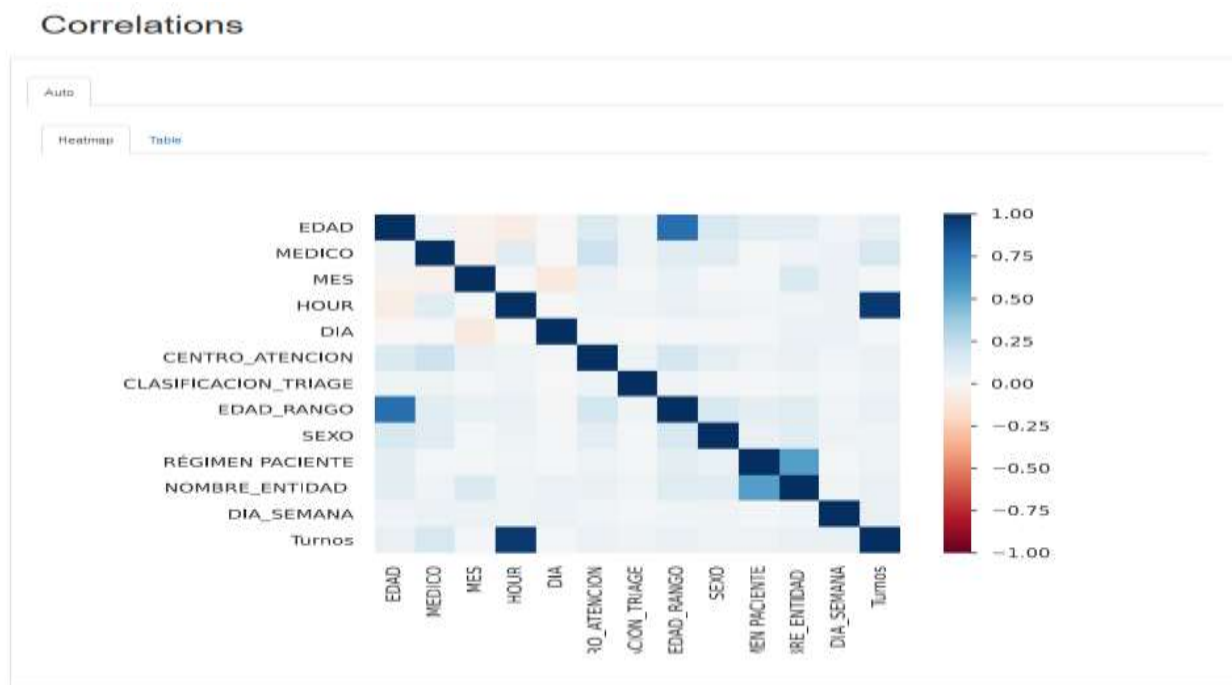


Figura 3 – Grafica Correlación de variables

Al estimar la matriz de correlación, se puede observar que no se evidencia correlación lineal fuerte entre las variables seleccionadas dando como resultado la independencia de variables y así se puede realizar una interpretación más adecuada del modelo de predicción al momento de analizar los resultados.



## 2. INTRODUCCIÓN A LOS MODELOS

Machine Learning (ML) es un campo de estudio que involucra el desarrollo de algoritmos y modelos que permiten a las computadoras aprender de los datos para mejorar su rendimiento y hacer predicciones que permitirán tomar decisiones basadas en ese aprendizaje. Dentro del campo de machine learning hay diferentes áreas de especialidad entre las que se encuentran el aprendizaje supervisado y el aprendizaje no supervisado (Kuhn & Johnson, 2013, 41).

Grafica que explica los diferentes modelos de machine learning :

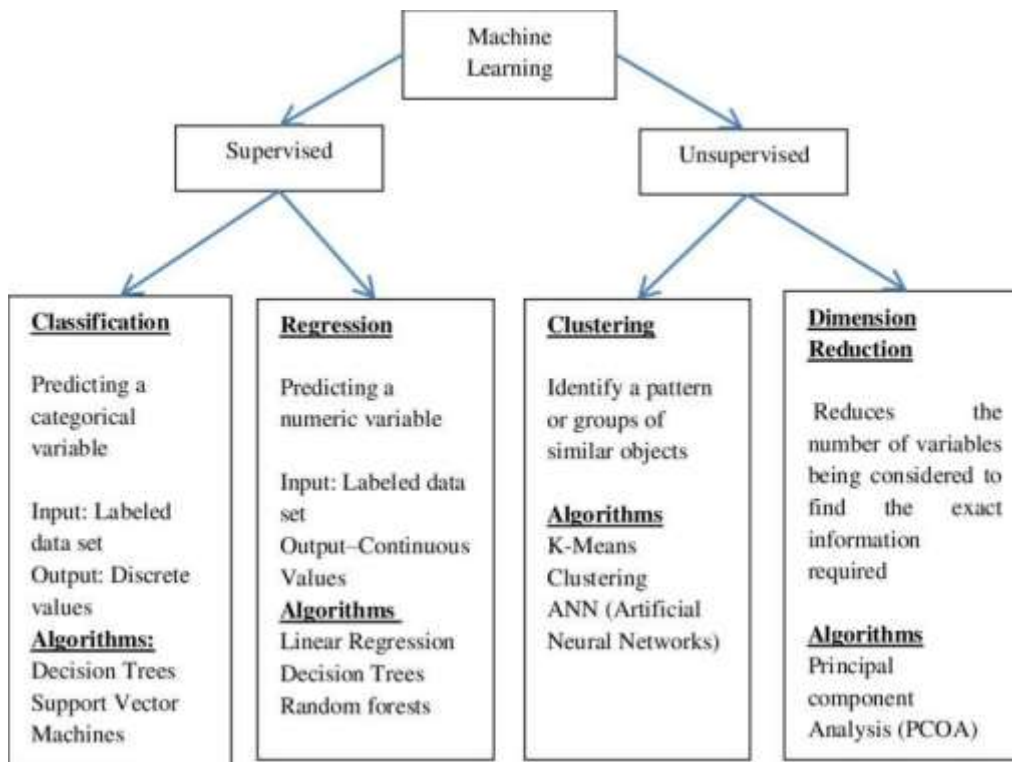


Figura 4 – Grafica Clasificación amplia de técnicas de aprendizaje automático (Suryakanthi, 2020, #)

El aprendizaje supervisado implica entrenar un modelo en un conjunto de datos etiquetados, donde se conocen las etiquetas. El objetivo es aprender una función que pueda predecir las etiquetas para datos nuevos de los que no se tiene información. El aprendizaje supervisado se basa en encontrar una función que pueda asignar características de entrada a etiquetas de salida, dado un conjunto de ejemplos de entrenamiento. En este caso el algoritmo aprende a predecir una variable objetivo (también conocida como variable dependiente) en función de los datos de entrada (también conocidos como características o variables independientes) tomando como base un conjunto de ejemplos para definir una función o modelo que permita completar los datos faltantes en la relación entre la variable dependiente y las variables independientes permitiendo realizar predicciones (James et al., 2013, 26).

En el aprendizaje no supervisado el algoritmo intenta identificar patrones en los datos sin etiquetas ni resultados predefinidos en los que no hay una variable dependiente de la que se tengan ejemplos conocidos. El algoritmo se entrena en un conjunto de datos sin etiquetar sin ningún conocimiento previo de la estructura de datos. El objetivo del aprendizaje no supervisado es identificar patrones o agrupaciones en los datos que se pueden usar para descubrir información y tomar decisiones informadas. A diferencia del aprendizaje supervisado, donde los datos de entrenamiento están etiquetados, los algoritmos de aprendizaje no supervisado tienen que encontrar patrones y agrupaciones ocultos en los datos por sí mismos.

### 3.1. Definición de modelos de pronóstico de serie de tiempo

Una serie de tiempo es una secuencia de datos u observaciones, medidos en determinados momentos y ordenados cronológicamente. Visualmente, es una curva que evoluciona en el tiempo. Una serie de tiempo es un conjunto de observaciones sobre los valores que toma una variable (cuantitativa) a través del tiempo.

#### **Ejemplo de algunas aplicaciones de datos de serie temporal**

1. Anual: - PIB, serie macroeconómica
2. Trimestral:- Ingresos de una empresa.
3. Mensual: - Ventas, gastos, salario
4. Semanal:- Demanda, Precio de Gasolina y Diesel
5. Diariamente:- Precio de cierre de acciones, valor sensex, transacción diaria de cajero automático
6. Por hora: - AAQI

### 3.2. Características de modelos de serie de tiempo

Una serie temporal es una colección de puntos de datos que se almacenan con respecto a su tiempo. El análisis matemático y estadístico realizado en este tipo de datos para encontrar patrones ocultos e información significativa se denomina análisis de series temporales. Las técnicas de modelado de series temporales se utilizan para comprender patrones pasados a partir

de los datos y tratar de pronosticar horizontes futuros. Estas técnicas y metodologías han ido evolucionando durante décadas.

Las observaciones con marcas de tiempo continuas y variables objetivo a veces se enmarcan como problemas de regresión sencillos al descomponer las fechas en minutos, horas, días, semanas, meses, años, etc., que no es la forma correcta de manejar dichos datos porque los resultados obtenidos son deficientes. En este capítulo, aprenderá el enfoque correcto para manejar datos de series de tiempo.

Hay diferentes tipos de datos, como estructurados, semiestructurados y no estructurados, y cada tipo debe manejarse a su manera para obtener el máximo conocimiento. Existen diferentes tipos de modelos de pronósticos de series de tiempo y otros tipos de modelos que veremos a continuación.

### **3.3. Tipos de Modelos de pronósticos de Series de Tiempo**

Los problemas de predicción de series temporales son complejos para el modelado predictivo. A diferencia de modelos predictivos de regresión, las series temporales también añaden la complejidad de una dependencia de secuencia entre las variables de entrada. Un poderoso tipo de red neuronal diseñada para manejar secuencias la dependencia se llaman Redes Neuronales Recurrentes.

En el presente proyecto se utilizarán varios modelos de pronóstico de series de Tiempo:

### 3.3.1. Modelo Arima

ARIMA (AutoRegressive Integrated Moving Average) y SARIMAX (Seasonal AutoRegressive Integrated Moving Average with eXogenous regressors) son modelos estadísticos ampliamente reconocidos y utilizados para la predicción de series temporales (forecasting). Este modelo consta de tres componentes. El elemento autorregresivo (AR) relaciona el valor actual con valores pasados (lags). El elemento de media móvil (MA) asume que el error de predicción es una combinación lineal de los errores de predicción pasados. Por último, el componente integrado (I) indica que los valores de la serie original han sido reemplazados por la diferencia entre valores consecutivos (y este proceso de diferencia puede haberse realizado más de una vez). Si bien los modelos ARIMA son ampliamente conocidos, los modelos SARIMAX extienden el marco de ARIMA al incorporar patrones estacionales y variables exógenas.

En la notación del modelo ARIMA-SARIMAX, los parámetros  $p$ ,  $d$ , y  $q$  representan las componentes autor regresivas, de diferenciación y de media móvil, respectivamente.  $P$ ,  $D$ , y  $Q$  son las mismas componentes para la parte estacional del modelo y  $m$  el número de períodos en cada temporada.

### 3.3.2. Modelo LSTM

Las redes de memoria larga-corto plazo por sus siglas LSTM es un tipo de red neuronal recurrente que se utiliza en el aprendizaje profundo debido a sus grandes arquitecturas las cuales pueden ser entrenadas con éxito.

Recordemos que una Red LSTM es un tipo de Red Neuronal Recurrente que permite analizar secuencias (como texto, conversaciones, o precisamente el comportamiento histórico de una acción en la bolsa) y que además tiene una memoria de largo plazo.

Las LSTMs son sensibles a la escala de los datos de entrada, especialmente cuando se utilizan las funciones de activación sigmoide (por defecto) o tanh. Puede ser una buena práctica reescalar los datos al rango de 0 a 1, también llamado normalización. Podemos normalizar fácilmente el conjunto de datos utilizando la clase de pre procesamiento MinMaxScaler de la biblioteca scikit-learn.

### 3.3.3. Modelo Random Forest

Un modelo Random Forest está formado por múltiples árboles de decisión individuales. Cada uno de estos árboles es entrenado con una muestra ligeramente diferente de los datos de entrenamiento, generada mediante una técnica conocida como bootstrapping. Para realizar predicciones sobre nuevas observaciones, se combinan las predicciones de todos los árboles que conforman el modelo.

Muchos métodos predictivos generan modelos globales en los que una única ecuación se aplica a todo el espacio muestral. Cuando el caso de uso implica múltiples predictores, que interaccionan entre ellos de forma compleja y no lineal, es muy difícil encontrar un único modelo global que sea capaz de reflejar la relación entre las variables. Los métodos estadísticos y de *machine learning* basados en árboles engloban a un conjunto de técnicas supervisadas no paramétricas que consiguen segmentar el espacio de los predictores en regiones simples, dentro

de las cuales es más sencillo manejar las interacciones. Es esta característica la que les proporciona gran parte de su potencial.

Los métodos basados en árboles han ganado reconocimiento como una referencia en el ámbito de la predicción debido a los excelentes resultados que ofrecen en una amplia gama de problemas. En este documento se explora cómo se construyen y utilizan los modelos Random Forest.

### **3.3.4. Modelo Prophet**

FB Prophet es un modelo muy extendido para la predicción de series temporales desarrollado por Facebook. El paquete FB Prophet ha sido desarrollado tanto para R como para Python. Para su instalación pincha aquí. El objetivo de este paquete es dar a los usuarios una herramienta potente y de fácil uso para predecir resultados de negocio. Está muy bien documentado en su web, por lo que es fácil aprender a usarlo y de ajustarlo.

El algoritmo subyacente es una generalización de un modelo de regresión aditivo, que se descompone en tres componentes principales (tendencia, estacionalidad y festivos), más los regresores adicionales. La tendencia y la estacionalidad son dos componentes de las series temporales muy importantes pero difíciles de cuantificar. Prophet hace un gran trabajo capturando ambos. Además, la relación precisión – velocidad de este algoritmo es muy alta.

Es un modelo que tiene fácil descomposición, por lo que la interpretabilidad es alta y es fácil extraer los coeficientes de cada una de las componentes del modelo. Así se puede estudiar el impacto de la estacionalidad, tendencia, festivos y otras variables regresoras.

Prophet alcanza sus mejores resultados con series estacionarias o series con una tendencia y estacionalidad marcadas. Suele ser peor cuando aumenta la aleatoria de la muestra. Como veremos más adelante en la práctica, la serie de datos que tenemos tiene una estacionalidad bastante clara, por lo que este modelo se ajustaba muy bien al problema entre manos

### 3.3.5. Modelo SARIMAX

SARIMAX (Promedio móvil integrado autorregresivo estacional con factores exógenos) es una versión actualizada del modelo ARIMA. podemos decir que SARIMAX es un modelo equivalente estacional como SARIMA y Auto ARIMA. también puede hacer frente a efectos externos. Esta característica del modelo se diferencia de otros modelos.

Concretamente estudian la relación entre una variable de interés y una serie de variables explicativas (que influyen en la variable de interés). En el marco multivariante se considera el pasado tanto de la variable que se quiere explicar, como el de las variables que están relacionadas con dicha variable.

Los modelos multivariantes de series temporales son una generalización de los modelos univariantes, la diferencia está en que en vez de una sola variable hay  $n$  variables (en vez de una serie, hay varias series).



### 3.3.6. Modelo Teoría de Colas

La teoría de colas es una rama de las matemáticas aplicadas que se ocupa del estudio de las colas o filas de espera en diversos contextos, desde sistemas de servicio hasta procesos de producción y logística. Su objetivo principal es analizar y optimizar la eficiencia de sistemas en los que hay demanda y asignación de recursos. La teoría de colas proporciona un marco para comprender y modelar situaciones en las que individuos, objetos o eventos llegan a un sistema y esperan ser atendidos por servidores.

La notación de Kendall es una forma de describir un modelo de colas con tres parámetros:  $A/B/c$ , donde  $A$  representa el proceso de llegada,  $B$  representa el proceso de servicio y  $c$  es el número de servidores. Por ejemplo, un modelo  $M/M/1$  es un sistema de colas con un proceso de llegada de Poisson ( $M$ ), un proceso de servicio exponencial ( $M$ ) y un solo servidor (1).

- Modelo  $M/M/1$ : Proceso de llegada de Poisson, proceso de servicio exponencial y un solo servidor.
- Modelo  $M/M/c$ : Proceso de llegada de Poisson, proceso de servicio exponencial y  $c$  servidores.
- Modelo  $M/D/1$ : Proceso de llegada de Poisson, tiempo de servicio constante y un solo servidor.
- Modelo  $M/G/1$ : Proceso de llegada de Poisson y un solo servidor, pero con una distribución de servicio general ( $G$ ).

### 3.3.7. Modelo de Bosque aleatorio

Los algoritmos basados en árboles son métodos populares de aprendizaje automático que se utilizan para resolver problemas de aprendizaje supervisado. Estos algoritmos son flexibles y pueden resolver cualquier tipo de problema (clasificación o regresión).

Los algoritmos basados en árboles tienden a usar la media para características (features) continuas o el modo para características categóricas cuando hacen predicciones sobre muestras de entrenamiento en las regiones a las que pertenecen. También producen predicciones con alta precisión, estabilidad y facilidad de interpretación.

## 3.4. Definición de métricas adecuadas de desempeño

Al desarrollar modelos de aprendizaje automático, generalmente comparamos varios modelos durante la fase de construcción. Luego, estimamos los rendimientos de esos modelos y se seleccionamos el modelo que considera que tiene mayor probabilidades de funcionar bien. Necesitamos medidas objetivas de desempeño para poder decidir qué pronóstico conservar como su pronóstico real. A lo largo de este cuaderno, vamos a usar numerosas herramientas para la evaluación de modelos. Veremos diferentes estrategias para evaluar modelos de aprendizaje automático en general y adaptaciones y consideraciones específicas a tener en cuenta para la previsión. También verá diferentes métricas para calificar el rendimiento del modelo.

- **Métrica 1: MSE**

El error cuadrático medio (MSE) es una de las métricas más utilizadas en el aprendizaje automático. Se calcula como el promedio de los errores al cuadrado. Para calcular el MSE, toma los errores por fila de datos, eleva al cuadrado esos errores y luego toma el promedio de ellos.

La métrica de error MSE es excelente para comparar diferentes modelos en el mismo conjunto de datos. La escala del MSE será la misma para cada modelo aplicado al mismo conjunto de datos. Sin embargo, la escala de la métrica no es muy intuitiva, lo que dificulta su interpretación fuera de la evaluación comparativa de múltiples modelos.

- **Métrica 2: RMSE**

El RMSE, o raíz del error cuadrático medio, es la raíz cuadrada del error cuadrático medio. Como puede comprender, sacar la raíz cuadrada del MSE no hace ninguna diferencia cuando desea utilizar las métricas de error para clasificar los rendimientos en orden.

Sin embargo, hay una ventaja en usar el RMSE en lugar del MSE. La razón para sacar la raíz cuadrada del MSE es que la escala del RMSE es la misma que la escala de la variable original. En la fórmula MSE, se toma el promedio de los errores al cuadrado. Esto hace que el valor sea difícil de interpretar. El uso de la raíz cuadrada hará que la escala de la métrica de error vuelva a la escala de sus valores reales.

- **Métrica 3: MAE**

El error absoluto medio (MAE) se calcula tomando las diferencias absolutas entre los valores pronosticados y reales por fila. El promedio de esos errores absolutos es el error absoluto medio.

El MAE toma los valores absolutos de los errores antes de promediarlos. Tomar el promedio de los errores absolutos es una forma de asegurarse de que la suma de los errores no haga que se cancelen entre sí.

Has visto que el MSE usa el cuadrado de los errores para evitar esto, y el MAE es una alternativa a esto. El MAE tiene una fórmula más intuitiva: es la métrica de error que la mayoría de la gente encuentra intuitivamente. Sin embargo, el RMSE es generalmente favorecido sobre el MAE.

- **Métrica 4: MAPE**

El MAPE, abreviatura de Error porcentual absoluto medio, se calcula tomando el error de cada predicción, dividido por el valor real. Esto se hace para obtener los errores relativos a los valores reales. Esto hará que la medida del error sea un porcentaje y, por lo tanto, está estandarizado.

Como hemos entendido de las medidas de error anteriores, no se estandarizaron en una escala entre cero y uno. Sin embargo, esta estandarización es muy útil. Esto facilita la comunicación de los resultados de rendimiento.

Para calcular el MAPE, toma los valores absolutos de esos porcentajes por fila y calcula su promedio.

El MAPE mide un porcentaje de error. Es una medida de error, por lo que los valores más bajos para el MAPE son mejores. Sin embargo, puede convertir fácilmente el MAPE en una medida de bondad de ajuste calculando  $1 - \text{MAPE}$ . En muchos casos, es más fácil comunicar el desempeño en términos de un resultado positivo que negativo.

- **Métrica 5: R<sup>2</sup>**

La métrica  $R^2$  (R cuadrado) es una métrica que está muy cerca de la métrica  $1 - \text{MAPE}$ .

Es una métrica de rendimiento en lugar de una métrica de error, lo que la hace ideal para estimar el rendimiento de nuestro modelo.

El  $R^2$  es un valor que tiende a estar entre 0 y 1, siendo 0 malo y 1 perfecto. Por lo tanto, se puede usar fácilmente como un porcentaje multiplicándolo por 100. El único caso en el que el  $R^2$  puede ser negativo es si su pronóstico es más del 100 % incorrecto.

La fórmula hace un cálculo interesante. Calcula una relación entre la suma de los errores al cuadrado y la suma de las desviaciones entre el pronóstico y el promedio. Esto se reduce a un porcentaje de aumento de su modelo sobre el uso del promedio como modelo. Si su modelo es una predicción tan mala como usar el promedio, entonces el  $R^2$  será cero. Como el promedio se usa a menudo como modelo de referencia, esta es una métrica de rendimiento muy práctica.

## 8. PROCESOS DE CALIBRACIÓN O ENTRENAMIENTO DE CADA MODELO

En este capítulo se pretende dar respuesta al requerimiento R2 “Herramienta de predicción de tiempos de atención” (Desarrollar una herramienta que proporcione a los usuarios información de los tiempos de espera históricos en Triague por meses, días, semanas y turnos el cual se podría utilizar para mejorar la atención de los pacientes en minutos..), por lo que se busca, por medio de un modelo entender los posibles tiempos de atención futuros, filtrados por días, meses, sedes, de tal manera que se puedan presentar de una forma de fácil interpretación para el usuario final.

### 4.1. Modelos

A partir de programas realizados en Python (desarrollos de programas que se informó en el capítulo “6. TRATAMIENTO DE LOS DATOS” y después de depurar la información se definieron las siguientes variables :

- "FECHA\_LLEGADA", "TIEMPO\_TOTAL\_FINAL", "CENTRO\_ATENCION", "CLASIFICACION\_TRIAGE", "PACIENTE\_EDAD", "PACIENTE\_#\_DOCUMENTO", "EDAD\_RANGO", "NOMBRE\_ENTIDAD", "SEXO", "Month"

Con los campos anteriores se crearon varias funciones en programa python que sirvieron como iniciar los entrenamientos o aprendizajes de los algoritmos para cada uno de los modelos a probar.

Se detallan las siguientes funciones generales

- Función para evaluar Métricas

```

1  # Funcion para evaluar Metricas
2
3  def evaluacion_metrica(y_true, y_pred, titulo, metrics_df):
4
5      def mean_absolute_percentage_error(y_true, y_pred):
6          y_true, y_pred = np.array(y_true), np.array(y_pred)
7          return np.mean(np.abs((y_true - y_pred) / y_true)) * 100
8
9      mse = metrics.mean_squared_error(y_true, y_pred)
10     mae = metrics.mean_absolute_error(y_true, y_pred)
11     rmse = np.sqrt(mse)
12     mape = mean_absolute_percentage_error(y_true, y_pred)
13     r2 = metrics.r2_score(y_true, y_pred)
14
15     # Crear un nuevo DataFrame con los resultados
16     new_metrics_df = pd.DataFrame({
17         'TimeSeries': [titulo],
18         'MSE': [mse],
19         'MAE': [mae],
20         'RMSE': [rmse],
21         'MAPE': [mape],
22         'R2': [r2]
23     })
24
25     # Concatenar el nuevo DataFrame con el existente
26     metrics_df = pd.concat([metrics_df, new_metrics_df], ignore_index=True)
27
28     print('Evaluation metric results:-')
29     print(f'MSE is : {metrics.mean_squared_error(y_true, y_pred)}')
30     print(f'MAE is : {metrics.mean_absolute_error(y_true, y_pred)}')
31     print(f'RMSE is : {np.sqrt(metrics.mean_squared_error(y_true, y_pred))}')
32     print(f'MAPE is : {mean_absolute_percentage_error(y_true, y_pred)}')
33     print(f'R2 is : {metrics.r2_score(y_true, y_pred)}',end='\n\n')
34
35     return metrics_df # Devolver el DataFrame actualizado

```

Figura 5 – Función en código python para evaluar métricas

- Función para División de para entrenamiento y prueba

```

1  # División de para entrenamiento y prueba
2  train_data = df6[:len(df6)-24]
3  test_data = df6[len(df6)-24:]
4  test=test_data.copy()
5
6  df6=df6.reset_index()
7
8  df_fb=df6.rename(columns={"Month":"ds", "TIEMPO_TOTAL_FINAL":"y"})
9
10 train_data_pr = df_fb.iloc[:len(df6)-24]
11 test_data_pr = df_fb.iloc[len(df6)-24:]

```

Figura 6 – Función en código python para entrenamiento y pruebas de modelos de predicción

- Función para agrupamiento de variables de tiempo (hora, día de la semana, semana, turno, mes)

```

1 # Agrupar de acuerdo a la grafica
2
3 # agrupamiento de tiempo (dia, hora, mes, turno)
4 nuevo_df = df.groupby('Month')['TIEMPO_TOTAL_FINAL'].mean().reset_index()
5 nuevo_df = df.groupby('Dia')['TIEMPO_TOTAL_FINAL'].mean().reset_index()
6 nuevo_df = df.groupby('Turno')['TIEMPO_TOTAL_FINAL'].mean().reset_index()
7 nuevo_df = df.groupby('Hour')['TIEMPO_TOTAL_FINAL'].mean().reset_index()
8
9 df6 = nuevo_df.copy()
10
11 # Convierte el índice Period a cadena
12 #df6.index = df6.index.astype(str)
13 df6['Month1'] = df6['Month'].astype(str)
14
15 # Crea el gráfico de líneas
16 fig = px.line(df6, x='Month1', y='TIEMPO_TOTAL_FINAL', template='plotly_dark', title='Total minutos')
17 |
18 # Muestra el gráfico
19 fig.show()

```

Figura 7 – Función en código python para agrupamiento de tiempo (hora, día, mes, etc.)

Con las funciones anteriores queremos aplicar la misma metodología en cada de las modelos a probar, en específico la función de métricas nos va ayudar a comparar la mayoría de los modelos ya que define cual se acerca más a la tendencia en tiempo de los datos a probar en este caso, vamos a entrenar los algoritmos con datos desde el año 2017 que representan 1.375.710 registros.

A continuación, se muestra la gráfica de Meses vs Total Minutos

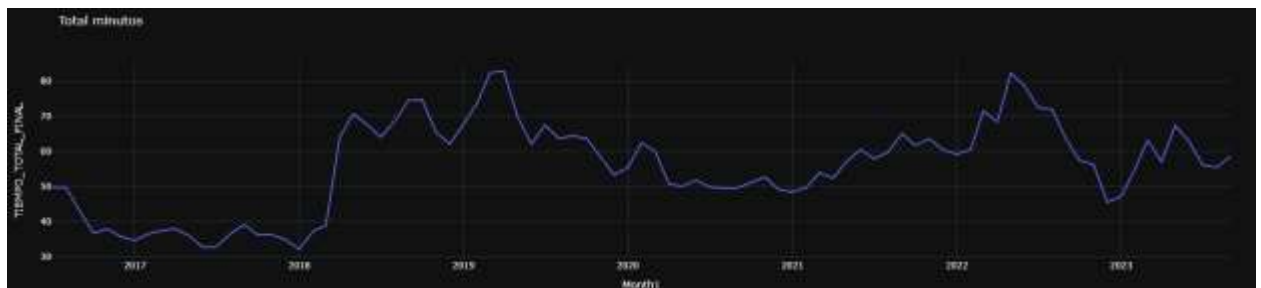


Figura 8 – Grafica Total minutos desde el año 2017



Para el procesamiento de los datos, se realizaron varias pruebas de parametrización, se llevaron a cabo varios procedimientos adicionales debido que al correr los modelos de predicción nos daba datos errados o que se salían o eran mínimos los cálculos, porque no llevaba la tendencia de la gráfica (mostrada anteriormente, figura 8) en el cual reflejaba que algo estaba mal, por lo que se decidió además de la selección de las variables más significativas y la limpieza de los datos más rigurosa, de igual manera de tomar los datos solo a partir del año 2022 y también agrupándolos antes de entrar al modelo y así los resultados fueron más limpios y cercanos a los datos históricos.

Las pruebas se inician con las variables Month (mes) y TOTAL TIEMPO, que usando la función agrupamiento por Month nos da 21 registros a modelar con los diferentes algoritmos de predicción.

#### 4.1.1. Modelo Prophet :

Se adapta el código python para ejecutar el modelo:

```
# 1. Modelo : Prophet Forecast
from prophet import Prophet

train_data_pr['ds'] = train_data_pr['ds'].dt.to_timestamp()
m = Prophet()
m.fit(train_data_pr)

future = m.make_future_dataframe(periods=24, freq='MS')
prophet_pred = m.predict(future)
# prophet_pred.tail() --- ver detalle

# asignar a prophet_pred
prophet_pred = pd.DataFrame({"Date" : prophet_pred[-24:]['ds'], "Pred" : prophet_pred[-24:]['yhat']})
prophet_pred = prophet_pred.set_index("Date")
prophet_pred.index.freq = "MS"
prophet_pred
test_data["Prophet_Predictions"] = prophet_pred['Pred'].values
test_data.head()

# Grafica test_data
a=test_data[["TIEMPO_TOTAL_FINAL","Prophet_Predictions"]]
fig = px.line(a, x=test_data.index, y=a.columns, template = "plotly_dark",
              title="Predicción con Modelo Prophet")
fig.show()

# evaluación de Prophet_Predictions
metrics_df = evaluacion_métrica(test_data["TIEMPO_TOTAL_FINAL"], test_data["Prophet_Predictions"], "Prophet", metrics_df)
```

Figura 9 – Modelo Prophet

Y da como resultado :

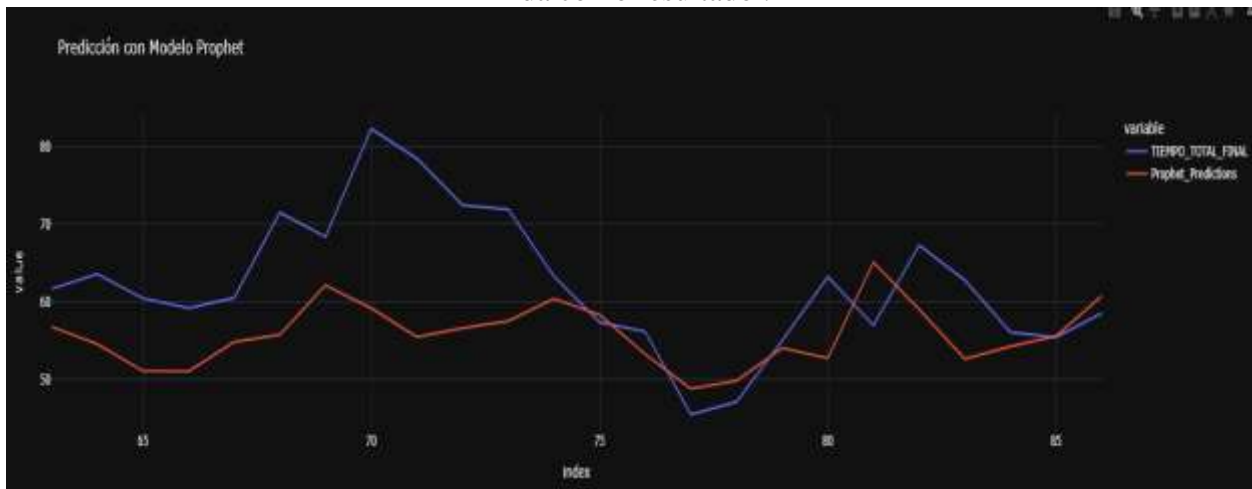


Figura 10 – Resultado Modelo Prophet

Y al ejecutar el código de Métricas:

```
Evaluation metric results:-
MSE is : 104.1237577310501
MAE is : 7.949467745982609
RMSE is : 10.204104945121356
MAPE is : 11.858375958335932
R2 is : -0.39785311209883023
```

Análisis del resultado: se acerca mucho al dato histórico, y además el R2 es muy cerca a 0 da la

#### 4.1.2. Modelo Teoría de Colas:

Se adapta el código python para ejecutar el modelo:

```
# prueba 4

# Tasas de llegada y servicio iniciales
tasa_llegada_inicial = 5 # Tasa de llegada promedio de pacientes por hora ( $\lambda$ )
tasa_servicio_inicial = 3 # Tasa de servicio promedio de pacientes por hora ( $\mu$ )
tasa_llegada = 5 # Tasa de llegada promedio de pacientes por hora ( $\lambda$ )
tasa_servicio = 3 # Tasa de servicio promedio de pacientes por hora ( $\mu$ )
num_medicos = 4 # Puedes ajustar esto según tus necesidades
tiempo_espera_total = 0
clientes_atendidos = 0
tiempo_simulacion = 24 # Tiempo de simulación en horas

# Parámetros para la proyección
num_simulaciones = 2 # Número de simulaciones a realizar
tasas_llegada_proyectadas = [7, 9, 11] # Tasas de llegada a probar
tasas_servicio_proyectadas = [5, 7, 9] # Tasas de servicio a probar

# Listas para almacenar resultados de las simulaciones
resultados_tiempo_espera_promedio = []

# Diccionarios para almacenar métricas por turno y día de la semana
turnos = ['MADRUGADA', 'MAÑANA', 'TARDE', 'NOCHE']
meses = ['01', '02', '03', '04', '05', '06', '07', '08', '09', '10', '11', '12']
dias_semana = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
metricas_por_turno = {turno: {'espera': 0, 'atendidos': 0} for turno in turnos}
metricas_por_dia_semana = {dias: {'espera': 0, 'atendidos': 0} for dias in dias_semana}
metricas_por_mes = {mes: {'espera': 0, 'atendidos': 0} for mes in meses}

def llegada_paciente(env, medico, turno, dia_semana, mes, centro, clasificacion):
    global tiempo_espera_total, clientes_atendidos, tiempo_real_total
    llegada = np.random.exponential(1 / tasa_llegada)
    yield env.timeout(llegada)
    # print(f"Llegó un paciente a las {env.now:.2f} horas.")
    with medico.request() as req:
        yield req
        tiempo_espera = env.now - llegada
        tiempo_espera_total += tiempo_espera
        # print(f"Paciente atendido a las {env.now:.2f} horas después de esperar {tiempo_espera:.2f} horas.")
        servicio = np.random.exponential(1 / tasa_servicio)
        yield env.timeout(servicio)
        clientes_atendidos += 1
```

Figura 11 – Código python Modelo Teoría de colas

Y al ejecutar el código de Métricas:

```
# Análisis de los resultados
mejor_proyeccion = min(resultados_tiempo_espera_promedio_extendido, key=lambda x: x[2])
print("Mejor proyección:")
print(f"Tasa de Llegada Proyectada: {mejor_proyeccion[0]} pacientes por hora")
print(f"Tasa de Servicio Proyectada: {mejor_proyeccion[1]} pacientes por hora")
print(f"Tiempo de Espera Promedio: {mejor_proyeccion[2]:.2f} horas")

# Análisis de los resultados real
mejor_proyeccion2 = min(resultados_tiempo_espera_promedio, key=lambda x: x[2])
print("Mejor proyección 2:")
print(f"Tasa de Llegada : {mejor_proyeccion2[0]} pacientes por hora")
print(f"Tasa de Servicio: {mejor_proyeccion2[1]} pacientes por hora")
print(f"Tiempo de Espera Promedio: {mejor_proyeccion2[2]:.2f} horas")

# Puedes tomar decisiones basadas en estos resultados, como ajustar las tasas de llegada y servicio para reducir los tiempos de espera.
```

Mejor proyección:  
Tasa de Llegada Proyectada: 11 pacientes por hora  
Tasa de Servicio Proyectada: 9 pacientes por hora  
Tiempo de Espera Promedio: 23.00 horas  
Mejor proyección 2:  
Tasa de Llegada : 11 pacientes por hora  
Tasa de Servicio: 9 pacientes por hora  
Tiempo de Espera Promedio: 22.51 horas

Figura 12 – Resultado Modelo Teoría de colas

### 4.1.3. Modelo Bosque Aleatorio:

Se adapta el código python para ejecutar el modelo:

```
# version 2 - tiempo promedio
# Preprocesar los datos
# Preprocesar los datos
data['FECHA_LLEGADA'] = pd.to_datetime(data['FECHA_LLEGADA'])

#data.loc[:, 'FECHA_LLEGADA'] = pd.to_datetime(data['FECHA_LLEGADA'])
data.loc[:, 'Mes'] = data['FECHA_LLEGADA'].dt.month
data.loc[:, 'Día'] = data['FECHA_LLEGADA'].dt.day
data.loc[:, 'DIA_SEMANA'] = data['FECHA_LLEGADA'].dt.dayofweek
data.loc[:, 'HOUR'] = data['FECHA_LLEGADA'].dt.hour

# quitar registros malos

# Calcular la mediana
median = data['Tiempo_Minutos_Total'].median()

# Corregir valores atípicos
data.loc[data['Tiempo_Minutos_Total'] > 420, 'Tiempo_Minutos_Total'] = median
data.loc[data['Tiempo_Minutos_Total'] < 0, 'Tiempo_Minutos_Total'] = median

# Calcular el total medicos
medico_contador = data.groupby(['Mes', 'Día', 'DIA_SEMANA', 'HOUR'])['MEDICO'].count().reset_index()

# Calcular el medico promedio por grupo
medico_promedio = data.groupby(['Mes', 'Día', 'DIA_SEMANA', 'HOUR'])['MEDICO'].nunique().reset_index()
medico_promedio['Promedio_Medicos'] = medico_contador['MEDICO'] / medico_promedio['MEDICO']

# Crear un modelo de regresión (Random Forest)
X = tiempo_promedio[['Mes', 'Día', 'DIA_SEMANA', 'HOUR']]
y = tiempo_promedio['Tiempo_Minutos_Total']
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X, y)

# Realizar predicciones para diferentes valores
nuevos_datos = pd.DataFrame({
    'Mes': [1,2,3,4,5,6,7],
    'Día': [10,11,12,13,14,15,16],
    'DIA_SEMANA': [1,2,3,4,5,6,7],
    'HOUR': [7,8,9,10,11,12,13]
})
```

Figura 13 – Código python Modelo Bosque Aleatorio

Y al ejecutar el código de Métricas:

```
print(f"Tiempo de espera predicho: {tiempo_espera_predicho[0]} minutos")
print(f"Tiempo de espera predicho: {tiempo_espera_predicho[1]} minutos")
print(f"Tiempo de espera predicho: {tiempo_espera_predicho[2]} minutos")
print(f"Tiempo de espera predicho: {tiempo_espera_predicho[3]} minutos")
print(f"Tiempo de espera predicho: {tiempo_espera_predicho[4]} minutos")

Tiempo de espera predicho: 74.59235000000004 minutos
Tiempo de espera predicho: 79.17985000000002 minutos
Tiempo de espera predicho: 85.67735000000002 minutos
Tiempo de espera predicho: 82.24480000000005 minutos
Tiempo de espera predicho: 66.15245000000002 minutos
```

Figura 14 – Resultado del Modelo Bosque Aleatorio

#### 4.1.4. Modelo Arima:

Se adapta el código python para ejecutar el modelo:

```
# 2. Modelo : ARIMA

#train_data['Month'] = train_data['Month'].dt.to_timestamp()
train_data['Month_numeric'] = train_data['Month'].dt.year * 12 + train_data['Month'].dt.month

# Luego, utiliza 'Month_numeric' como la serie temporal en auto_arima
modelo_auto = auto_arima(train_data['Month_numeric'], start_p=0, d=1, start_q=0,
                        max_p=4, max_d=2, max_q=4, start_P=0,
                        D=1, start_Q=0, max_P=2, max_D=1,
                        max_Q=2, m=24, seasonal=True,
                        suppress_warnings=True, stepwise=True,
                        random_state=20, n_fits=50)

print(modelo_auto)

arima_model = SARIMAX(train_data["TIEMPO_TOTAL_FINAL"], order = (0,1,0), seasonal_order = (0,1,0,24))
arima_result = arima_model.fit()
arima_result.summary()

# line plot of residual errors
residuals = pd.DataFrame(arima_result.resid)
residuals.plot(figsize = (16,5));
plt.show();

# kernel density plot of residual errors
residuals.plot(kind='kde', figsize = (16,5))
plt.show()
print(residuals.describe())

modelo_auto.plot_diagnostics(figsize=(20,8))
plt.show()

print(modelo_auto.summary())

arima_pred = arima_result.predict(start = len(train_data), end = len(df6)-1, typ="levels").rename("ARIMA Predictions")
test_data['ARIMA_Predictions'] = arima_pred

# Grafica test_data
a=test_data[["TIEMPO_TOTAL_FINAL","ARIMA_Predictions"]]
fig = px.line(a, x=test_data.index, y=a.columns,template = "plotly_dark",
            title="Predicción con Modelo ARIMA")
fig.show()

# evaluacion de Prophet_Predictions
metrics_df = evaluacion_metrice(test_data["TIEMPO_TOTAL_FINAL"], test_data["ARIMA_Predictions"], "ARIMA", metrics_df)
```

Figura 15 – Código python Modelo Arima

Y da como resultado:

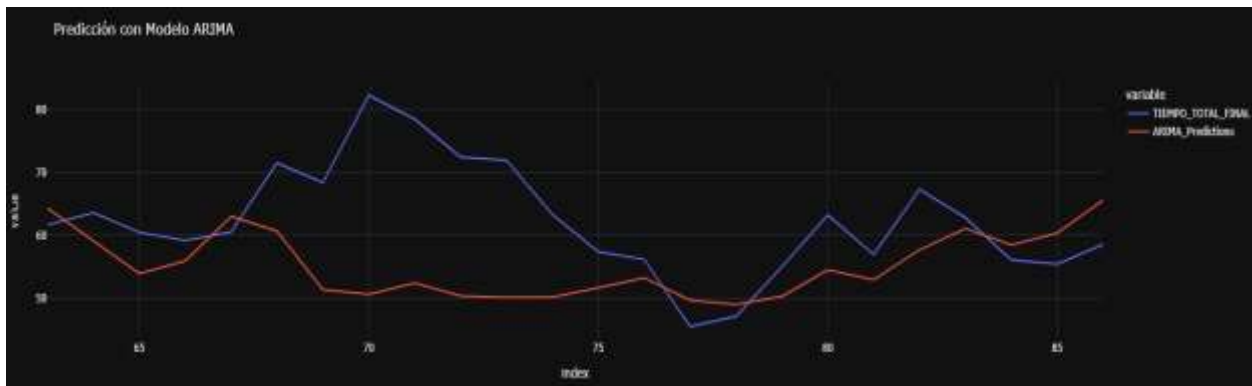


Figura 16 – Resultado Modelo Arima

Y al ejecutar el código de Métricas:

```
Evaluation metric results:-
MSE is : 153.19750148927574
MAE is : 9.182391845997557
RMSE is : 12.377297826637111
MAPE is : 13.57487471800206
R2 is : -1.0566641935425425
```

Figura 17 – Resultado Métricas Modelo Arima

Análisis del resultado: se acerca mucho al dato histórico en los últimos meses, pero cuando se cambia la variable por más tiempo el modelo difiere mucho del histórico y además el R2 esta más lejos del 0.



#### 4.1.5. Modelo LSTM\_Predictions

Se adapta el código python para ejecutar el modelo:

```

1  # 3. Modelo : LSTM_Predictions
2
3  # Seleccionar solo columnas numéricas
4  numeric_columns = train_data.select_dtypes(include=['float64']).columns
5
6  # Aplicar MinMaxScaler solo a las columnas numéricas
7  scaler = MinMaxScaler()
8  scaler.fit(train_data[numeric_columns])
9
10 # Transformar el conjunto de entrenamiento y prueba solo en las columnas numéricas
11 scaled_train_data = scaler.transform(train_data[numeric_columns])
12 scaled_test_data = scaler.transform(test_data[numeric_columns])
13
14 # Definir parámetros
15 n_input = 24
16 n_features = 1
17
18 # Crear el generador
19 generator = TimeseriesGenerator(scaled_train_data, scaled_train_data, length=n_input, batch_size=1)
20
21 # Crear el modelo LSTM
22 lstm_model = Sequential()
23 lstm_model.add(LSTM(200, activation='relu', input_shape=(n_input, n_features)))
24 lstm_model.add(Dense(1))
25 lstm_model.compile(optimizer='adam', loss='mse')
26
27 # Resumen del modelo
28 lstm_model.summary()
29
30 # Entrenar el modelo utilizando el generador
31 lstm_model.fit(generator, epochs=10) # Ajusta el número de épocas según sea necesario
32
33 # Obtener el historial de entrenamiento
34 history = lstm_model.history
35
36 # Verificar si hay algún error durante el entrenamiento
37 if history is None:
38     print("Error: El objeto History no se ha devuelto. Revisa tu código.")
39 else:
40     # Acceder a las métricas de entrenamiento
41     losses_lstm = history.history['loss']
42     plt.figure(figsize=(12, 4))
43     plt.xticks(np.arange(0, 21, 1))
44     plt.plot(range(len(losses_lstm)), losses_lstm, label='Training Loss')
45     plt.legend()
46     plt.show()

```

Figura 18 – Código python Modelo LSTM

Y da como resultado:

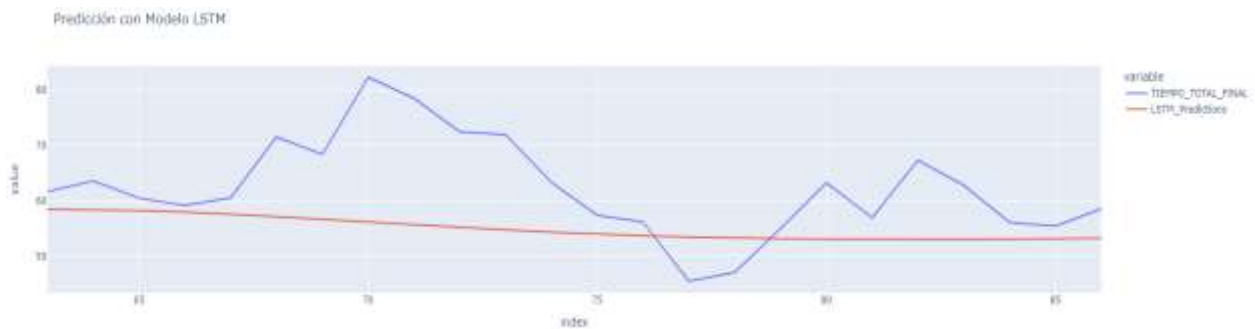


Figura 19 – Resultado Modelo LSTM

Y al ejecutar el código de Métricas:

```
Evaluation metric results:-
MSE is : 119.32418373978271
MAE is : 8.501764912988989
RMSE is : 10.923560945945361
MAPE is : 12.78407627021983
R2 is : -0.6019176144232468
```

Figura 20 – Resultado Métricas Modelo LSTM

Análisis del resultado: no se acerca en nada a la tendencia del dato histórico, y además el R2 está más lejos del 0.



## 9. RESULTADOS – DISCUSIONES

Se generan diferentes graficas de análisis para evaluar cuál de los modelos de predicción se acerca más a la realidad de los datos históricos.

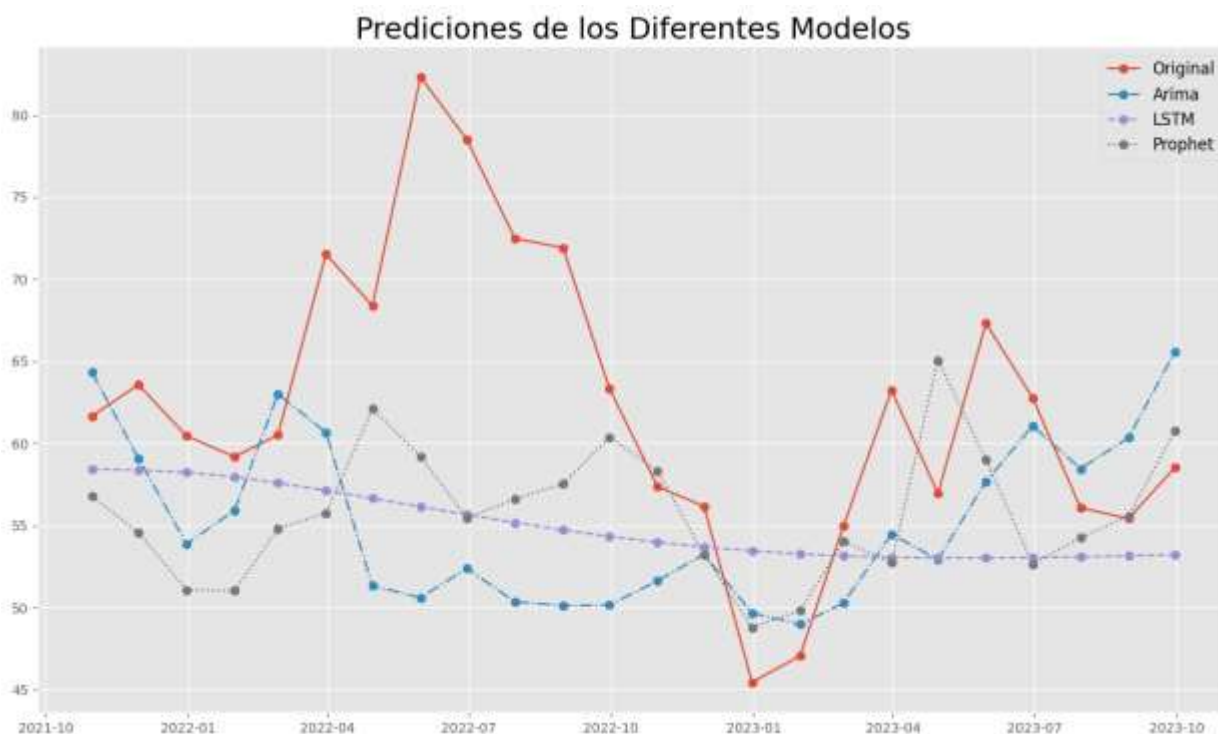


Figura 21 – Comparación Grafica de los diferentes modelos de series de tiempo

Al analizar los datos recolectados en la función de evaluación de Métricas se evidencia que el modelo Prophet está más cercano al cero (0) y es más cercano a los datos históricos

	TimeSeries	MSE	MAE	RMSE	MAPE	R2
0	Prophet	104.123758	7.949468	10.204105	11.858376	-0.397853
1	ARIMA	153.197501	9.182392	12.377298	13.574875	-1.056664
2	LSTM	119.324184	8.501765	10.923561	12.784076	-0.601918

Figura 22 – Comparación Evaluación Métricas de los diferentes modelos de series de tiempo

Al analizar los datos detallados de cada mes de los tiempos promedio recolectados y la predicción de cada uno de los modelos se puede evaluar que la mayoría de los datos del modelo Prophet se acerca más a los datos de tiempos de atención histórica.

	Month	TIEMPO_TOTAL_FINAL	Prophet_Predictions	ARIMA_Predictions	LSTM_Predictions
63	2021-10	61.669154	56.775329	64.328074	58.424081
64	2021-11	63.571233	54.563291	59.087427	58.379876
65	2021-12	60.453111	51.083772	53.886896	58.234386
66	2022-01	59.193422	51.026953	55.913274	57.964057
67	2022-02	60.508857	54.813553	63.004093	57.589442
68	2022-03	71.524030	55.753660	60.660901	57.134253
69	2022-04	68.387080	62.119462	51.320390	56.648305
70	2022-05	82.326158	59.187532	50.611559	56.150630
71	2022-06	78.494542	55.459448	52.378270	55.654593
72	2022-07	72.472787	56.615090	50.350319	55.177159
73	2022-08	71.920960	57.530229	50.118544	54.732115
74	2022-09	63.336275	60.375453	50.154529	54.333089
75	2022-10	57.375565	58.336352	51.628670	53.984484
76	2022-11	56.166005	53.234445	53.239092	53.690466
77	2022-12	45.448452	48.799970	49.667102	53.452645
78	2023-01	47.066502	49.816740	48.986359	53.270216
79	2023-02	54.964683	54.021699	50.270431	53.141346
80	2023-03	63.231415	52.741140	54.467245	53.062962
81	2023-04	56.948957	65.073603	52.925684	53.026247
82	2023-05	67.322675	59.017028	57.682213	53.024839
83	2023-06	62.789188	52.637363	61.064407	53.047489
84	2023-07	56.079714	54.253355	58.459155	53.089191
85	2023-08	55.417946	55.585098	60.345484	53.149787
86	2023-09	58.541141	60.771543	65.600669	53.221650

Figura 23 – Comparación datos de tiempos de atención y la predicción de cada uno de los modelos

## 10. IMPLEMENTACIÓN DEL MODELO SELECCIONADO

### 6.1. Requerimiento 1: Herramienta descriptiva (Tablero de Mando de Históricos para tiempos de Espera)

Por medio de varias herramientas se realizó el primer borrador de un tablero de visualización de datos Históricos. (realizada en Power BI), el cual esta solución contempla un tablero de control compuesto de dos páginas con información de: Históricos de Atenciones, Predicción de datos.

Tablero 1: Históricos de Atenciones



Figura 24 – Graficas de Power BI – Histórico de atenciones

Tablero 2: Predicción de datos

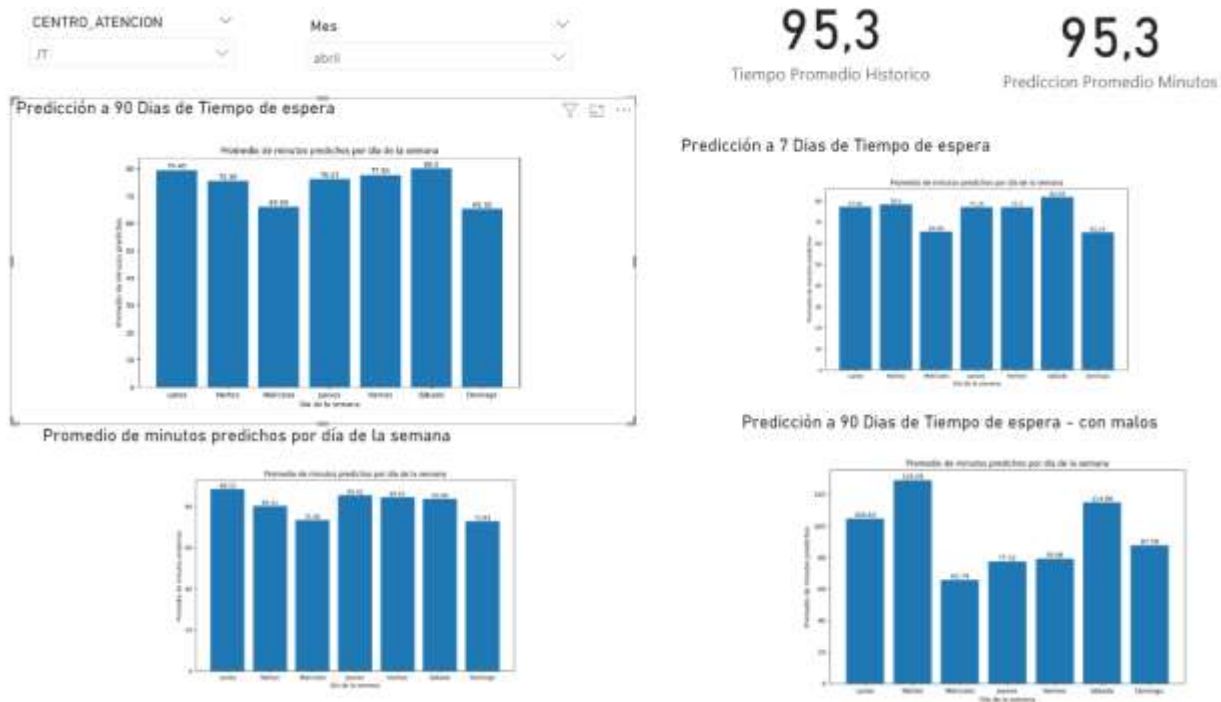


Figura 25 – Graficas de Power BI – Predicción de datos

## 6.2. Requerimiento 2: Herramienta de predicción de tiempos de atención (Aplicativo web)

De acuerdo al capítulo anterior (capítulo 9 –RESULTADOS) se da como conclusión el usar el **Modelo Prophet** para nuestro proyecto de predicción de tiempos de atención de los usuarios de acuerdo a Horas, días, semanas, turnos, meses

Y así por medio de varias herramientas tales como GitHub (repositorio), Visual Studio Code (para realizar programas de Python), Streamlit (herramienta para publicar proyecto como

página web), se desarrolla diferentes tableros, aplicaciones, gráficas y como solución final la publicación web de la visualización de datos Históricos y predicción de datos, por medio de filtros

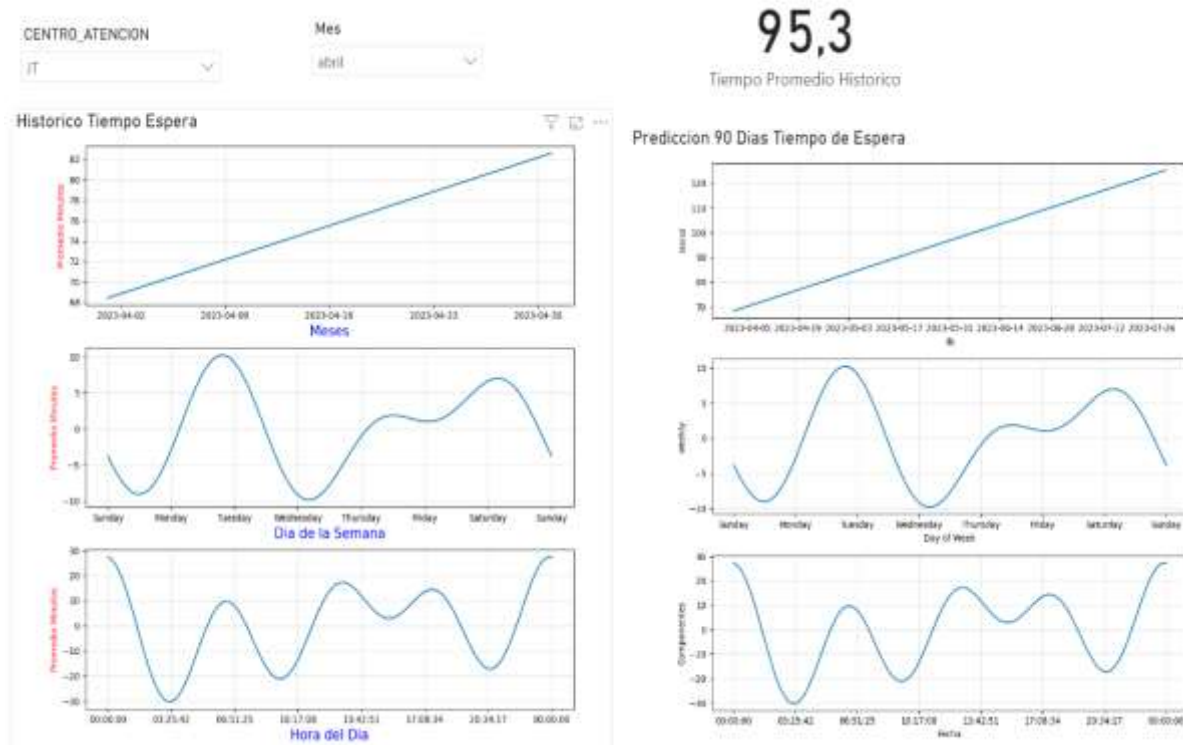


Figura 26 – Graficas de Power BI – Predicción de datos en línea

Grafica de la Aplicación Web publicada en la pagina de Streamlit : <https://tiemposesp-au5vhalmehgtxmvdvbu9fy.streamlit.app/>

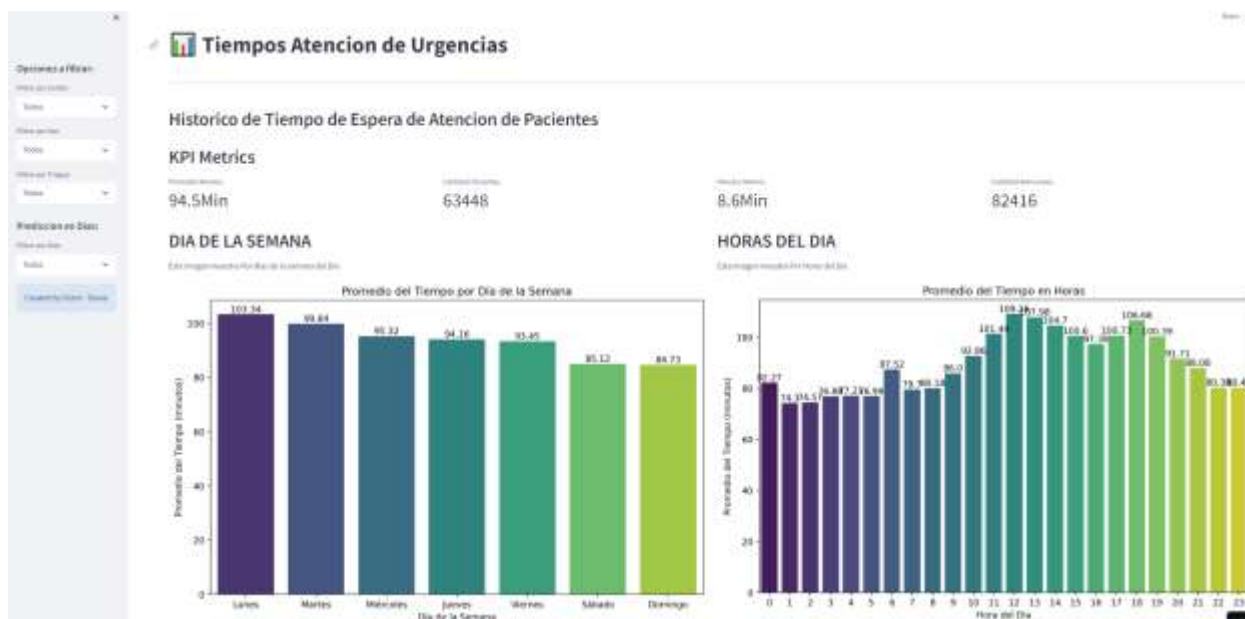


Figura 27 – Graficas de aplicación Web – Tiempos de atención de Urgencias - Histórico



Figura 28 – Graficas de aplicación Web – Tiempos de atención de Urgencias - Predicción