Informe: Prueba técnica Data Science

Responsable	Jacob Tinoco
Fecha de creación	@10 de febrero de 2025 8:19
Clase	Data analitics, Inteligencia Artificial, Programación
Tipo	Reporte/Informe
Adjuntos	Prueba DS Arkon_v2 (1).pdf
Revisado	☑
Correo electrónico	jacobtinoco.ao@gmail.com
Estado	En curso
Número de teléfono	722 647 9735

Bicicletas Compartidas en Los Ángeles, EU.

Introducción

Este proyecto se centra en el análisis de datos de un sistema compartido de bicicletas en Los Ángeles, utilizando un conjunto de datos que abarca desde 2016 al 2022. El objetivo principal es explorar la demanda de servicio y prever el crecimiento de los planes de uso de bicicletas. Se emplearon modelos analíticos como la **Regresión Lineal** e identificación de tendencias por medio de **Prophet** para realizar predicciones y análisis de tendencias.

Estructura del Proyecto:

```
Solución_prueba_tecnica_100225_V1.5.0_JT/
Data_science_data_analitics_example/
│
├── data/
│   ├── train_set.csv
│   ├── test_set.csv
│   └── sample_submission.csv
│
├── data_prosseced
│   ├── test_set_Preproceced_090225_V1.0.0_JT
│   ├── train_set_Preproceced_070225_V1.1.0_JT.csv
│   └── sample_submission.csv
│
├── grafic/
│   ├── conteo_viajes_por_año.png
│   ├── distribucion_duracion_viajes.png
│   ├── distribucion_duracion_(año).png
│   ├── ...
│   ├── Modelo_analitcico_Prophet_predicción_3años.png
│   ├── Modelo_analitcico_Prophet_Tendencias.png
│   ├── Modelo_analitcico_Prophet_Tendencias_actuales.png
│   └── Modelo_analitico_(LR).png
│
├── notebook/
│   └── notebook.ipynb
│
├── src/
│   ├── estaciones_saturadas.py
│   ├── grafic_lines.py
│   ├── horarios_saturados.py
│   ├── modelo_LR.py
│   ├── modelo_prophet.py
│   ├── modelo_prophet.py
│   └── visualización_general.py
│
```

```
├── README.md
└── requirements.txt
```

Descripción de la Estructura

- **data/**: Contiene los datos originales utilizados en el análisis para el preprocesamiento.
- data_prosseced**/**: Contiene los datos procesados.
- **grafic/**: Almacena las gráficas generadas a partir del análisis.
- **notebook/:** Incluye el Jupyter Notebooks con los codigos de prueba para el análisis exploratorio y los modelos analíticos.
- **notebook/grafic/**: Almacena las gráficas generadas a partir del análisis.
- **src/**: Contiene los scripts para visualizaciones del análisis exploratorio y modelos analíticos.
- **README.md**: Documento que proporciona una visión general del proyecto y sus objetivos.
- **requirements.txt**: Lista de dependencias necesarias para ejecutar el proyecto.

0. Comprensión del problema

Para determinar la trayectoria del análisis primero se comprendido que se busca saber la saturación del servició, y las tendencias para determinar los planes a futuro, previamente en el documento pdf de la prueba se especulaba una tendencia al alta en el uso del servicio, lo cual origino preguntas guías a responder:

- ¿Cuál es la tendencia de uso en el servicio a lo largo del tiempo?
- ¿Cuál es la saturación del servicio por estación, horarios, y días a lo largo del tiempo?
- ¿Qué datos pueden ayudar a responder las preguntas anteriores dentro del data set?

- Con los datos anteriores y conociendo el tipo de servicio, ¿Cuáles son los modelos predictivos adecuados para el análisis?
- ¿Cómo será la predicción de uso para el servicio a 3 años?

1. Análisis Exploratorio de datos

Se realizó un análisis exploratorio de datos (EDA) para entender mejor la distribución y características del conjunto de datos:

2. Visualización exploratoria de datos en el csv:

- **Archivos de respaldo**: se creó una carpeta de respaldo para los datos originales nombrados como `train_set_original` y `test_set_original`
- **Verificación de datos**: limpieza de datos duplicados.
- **Identificación de datos relevantes**: `start_time`, `end_time`, `start_sation`, `end_satation`, `passholder_type`
- **Identificación de formatos**: visualizar como era el formato en el `train_set_original`

Durante esta etapa no se encontraron datos duplicados, pero si se pudo identificar que el formato de fechas en `start_time` y en `end_time` eran inconsistentes.

3. Preprocesamiento de datos

- Formateo de datos: Se formatearon los datos a la norma ISO 8601 y se creó un archivo nuevo llamado: `test_set_Preproceced_090225_V1.0.0_JT`
- Filtrado: se filtraron los datos más relevantes en notas personales para su uso en el procesamiento y análisis.

4. Procesamiento de datos

Para conocer la saturación del servicio Se generaron gráficos para visualizar:

- **Conteo de viajes a lo largo del tiempo**: Se utilizó un gráfico de líneas para mostrar la tendencia de viajes desde 2016 hasta 2021, pueden ver el código en Anexo 1.
- **Distribución de la duración de los viajes**: Se generó un histograma para visualizar la duración de los viajes, limitando a un rango de 0 a 60 minutos,

pueden ver el código en Anexo 2.

- **Top 10 rutas más utilizadas**: Se identificaron las rutas más frecuentadas y se analizó la duración media de los viajes, pueden ver el código en Anexo 3.
- **Top 4 horarios de mayor saturación para las rutas más utilizadas**: Se identificaron los horarios más frecuentados y se analizó la duración media de los viajes, pueden ver el código en Anexo 4.

5. Análisis de datos

En la fase previa se logró identificar una tendencia creciente en el uso del sistema desde 2016 a 2018, y justo después un pronunciamiento decreciente desde el 2019 al 2021, contrario a la especulación que teníamos al del proyecto, para analizar estas tendencias y comportamientos se compararon algunos modelos, ver Anexo 6, posteriormente se implementaron dos modelos analíticos regresión lineal, ver en Anexo 5, y uso de Prophet de Meta en Pyhton, ver en Anexo 6.

5.1 Regresión Lineal

Se eligió la regresión lineal debido a su simplicidad y eficacia para predecir tendencias basadas en datos históricos. Este modelo permite comprender la relación entre variables, como el tiempo y el número de viajes, pero está limitado en su comprensión del entorno.

Justificación:

- **Interpretabilidad**: Los coeficientes del modelo son fáciles de interpretar, lo que ayuda a entender cómo cada variable impacta en la predicción.
- **Eficiencia**: Es más optimo y usa menos recursos de harware en comparación con modelos más complejos.
- **Adecuación**: La regresión lineal se adapta bien a los datos de viajes.

Pueden visualizar el código en el Anexo 5.0.

5.2 Prophet

Prophet fue utilizado para realizar predicciones de series temporales, aprovechando su capacidad para manejar estacionalidades (Año, mes, dias) y

tendencias.

Justificación:

- **Flexibilidad**: Prophet permite modelar datos con tendencias no lineales y cambios estacionales.
- **Interactividad**: Permite ajustes en los componentes del modelo, lo que facilita la personalización.
- **Predicciones a largo plazo**: Es ideal para pronósticos a largo plazo, como se requiere en este proyecto.

Pueden visualizar el código en el Anexo 5.1.

6. Resultados y Visualizaciones

Se generaron visualizaciones que muestran las tendencias y proyecciones a futuro, incluyendo:

- **Gráficos de tendencias**: Para visualizar la evolución de los viajes.
- **Proyección a 3 años**: Utilizando gráficos de barras para mostrar las predicciones anuales.

En el Gráfico puede verse que al iniciar el registro de viajes en 2016 se incrementó su demanda hasta el 2018, **fue durante y después de 2019 entrando la pandemia a los EU,** que la demanda baja significativamente hasta los últimos registros en **2021,** en dicho año fue donde la pandemia empezó a tener una mejor regularización de control sanitario, demostrando en 2021 no tener un déficit igual de pronunciado como de 2019 a 2020.

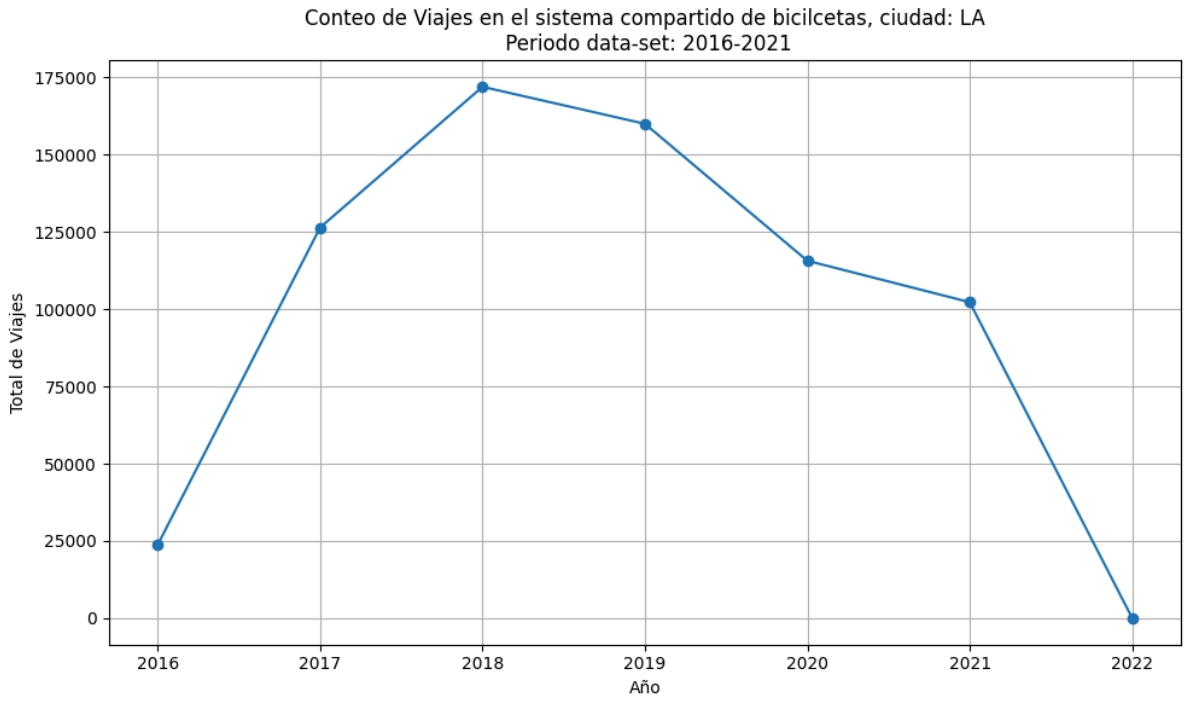


Gráfico 1: Conteo de viajes al año a lo largo del tiempo dentro del data set.

Al usar la regresión lineal del Gráfico 2, se pude inferir, en caso de que la pandemia siguiera con sus restricciones de cuarentena que el déficit en demanda para el servicio pudo incrementarse, pero este modelo no es confiable debido a sus limitaciones en la comprensión del entorno, resultado en un MSE elevado.

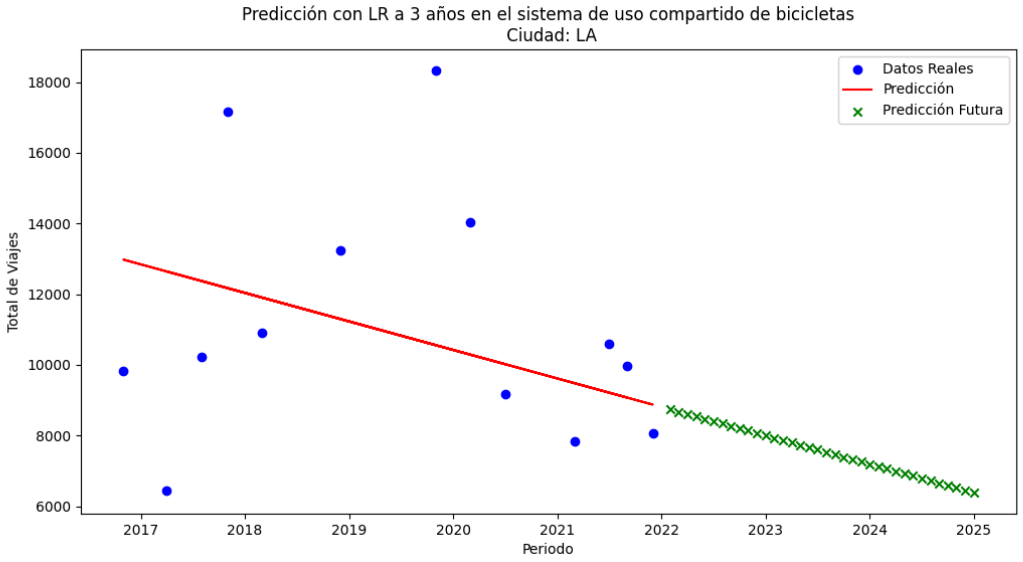


Gráfico 2: Uso de regresión lineal para la predicción a 3 años.

Por otro lado, dentro de la versatilidad en Prophet para su personalización, se identificaron las tendencias del servicio en uso: anual, mensual, semanal. Además de una predicción a futuro más confiable. En la Gráfica 3 se visualizan las tendencias del servicio, destacando que los fines de semana son de mayor demanda, con un alza en el uso del servicio desde el día jueves y un decremento hacia el día lunes. Además, puede identificarse que la saturación del servicio se encuentra durante el otoño. Por último, se identificó que los horarios más concurridos para las estaciones con de mayor demanda son en primer lugar alrededor de la media noche (22 hr a 03 hrs) y en segundo lugar por la tarde (14 hrs a 16 hrs), ver Anexo 7.

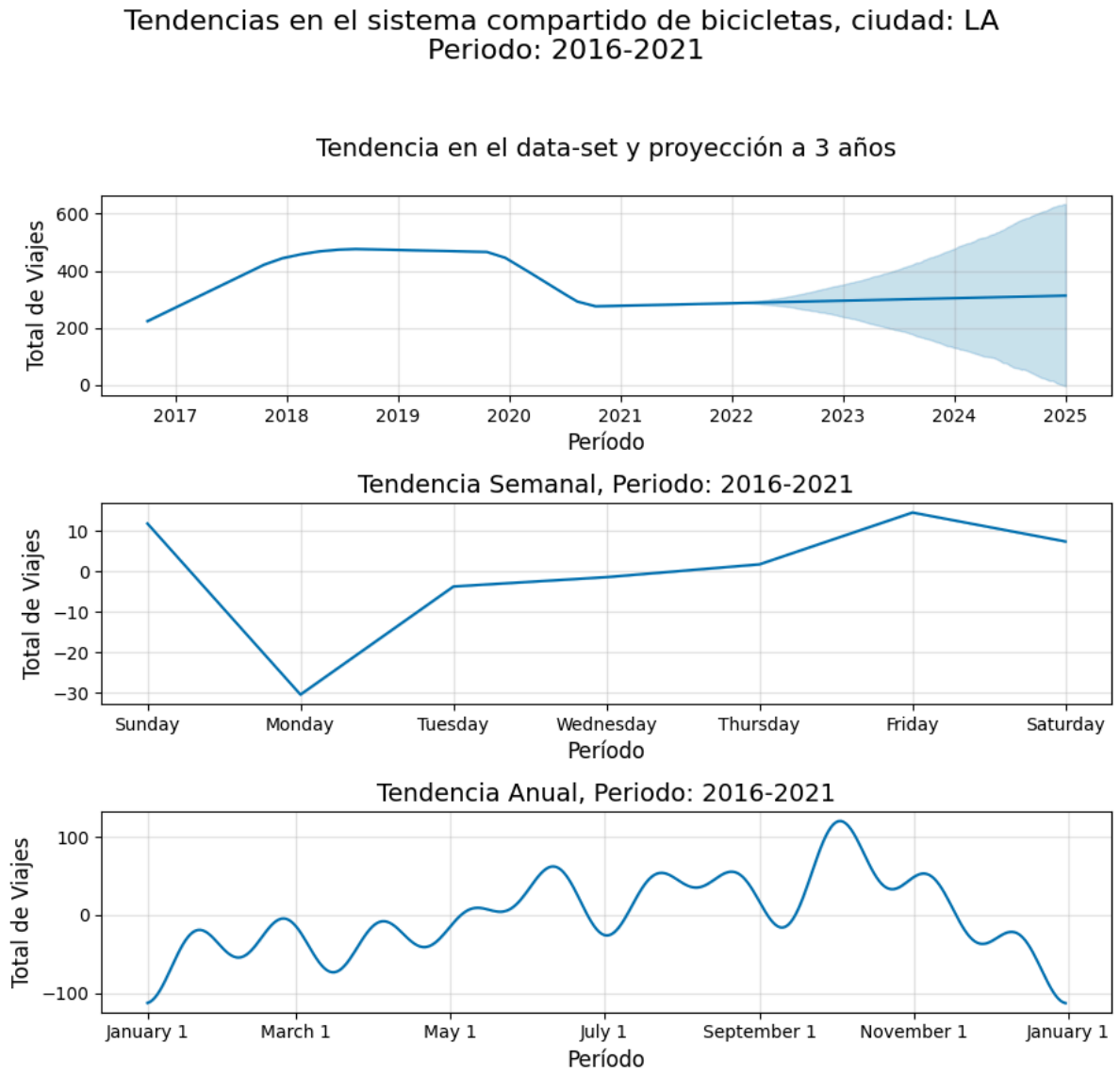


Gráfico 3. Tendencias en el uso del servicio por: Año, mes, semana, y una predicción a 3 años en le uso del servicio

Por último, en el Gráfico 4, se muestran la tendencia en el uso del servicio del data set con una predicción a 3 años y en el Gráfico 5 se muestra el conteo de viajes del registro y una predicción a 3 años en la demanda.

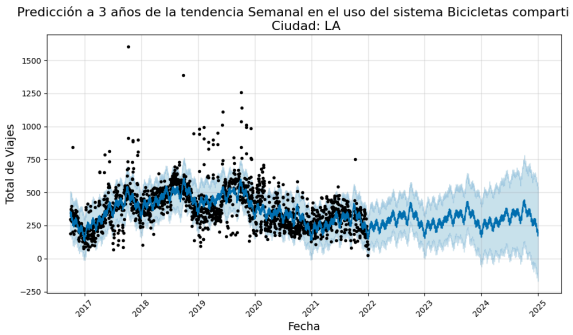


Gráfico 4. Tendencia anual a lo largo del tiempo con predicción a 3 años

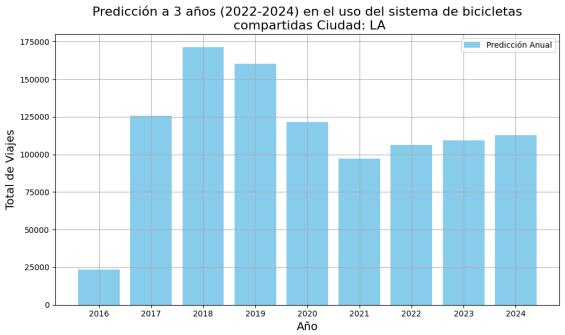


Gráfico 5. Predicción de los siguientes 3 años, respecto a la información a lo largo del tiempo

7. Conclusiones

- **Prophet** predice un incremento en la demanda del servicio, lo cual tiene sentido al concluir los años de pandemia.
- Se identificaron patrones significativos en el uso del servicio: baja demanda iniciando la pandemia en 2019, y una regularización en 2021.
- Se recomienda considerar variables adicionales, como condiciones climáticas y eventos locales, para mejorar la precisión del modelo.
- La época con mayor demanda es otoño.
- Los días de mayor demanda son en fin de semana, iniciando el jueves y terminando el domingo
- Los horarios de mayor saturación son cercanos a la media noche, seguido de los horarios entre las 14 hrs y 16 hrs.

8. Recomendaciones

- Continuar monitoreando los datos y ajustar los modelos según sea necesario.
- Explorar modelos más complejos si se dispone de más datos y recursos.
- Modularizar los Scripts para ejecutar en conjunto, y para crear un modelo escalable.

Anexos:

Anexo 1: Script: visualización_general.py

Códigos para conteo de Viajes a lo largo del tiempo:

```
import pandas as pd
import matplotlib.pyplot as plt
import os

df = pd.read_csv(r"data_prosseced\test_set_Preproceced_090225_V1.0
df['start_date'] = pd.to_datetime(df['start_date'])
df['year'] = df['start_date'].dt.year
total_trips_per_year = df.groupby('year').size().reset_index(name='total_

plt.figure(figsize=(10, 6))
plt.plot(total_trips_per_year['year'], total_trips_per_year['total_trips'], ma
plt.title('Conteo de Viajes en el sistema compartido de bicilcetas, ciudad
plt.xlabel('Año')
plt.ylabel('Total de Viajes')
plt.xticks(total_trips_per_year['year'])
plt.grid()
plt.tight_layout()

if not os.path.exists('grafic'):
    os.makedirs('grafic')

plt.savefig('grafic/conteo_viajes_por_año.png')
plt.show()
```

Anexo 2: Script: Top 10 rutas más utilizadas periodo 2016 - 2021

Histograma de viajes.

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import os

df = pd.read_csv(r"data_prosseced\test_set_Preproceced_090225_V1.0
```

```
df = df.dropna()
df_filtered = df[(df['duration'] >= 0) & (df['duration'] <= 60)]

if not os.path.exists('grafic'):
    os.makedirs('grafic')

plt.figure(figsize=(10, 6))
sns.histplot(df_filtered['duration'], bins=30, kde=True)
plt.title('Distribución de la Duración de los Viajes (0 a 60 minutos)')
plt.xlabel('Duración (minutos)')
plt.ylabel('Frecuencia')
plt.xlim(0, 60)
plt.tight_layout()
plt.savefig('grafic/distribucion_duracion_viajes.png')
plt.close()
```

▼ Anexo 3: Script: estaciones_saturadas.py

Top 10 rutas más utilizadas periodo 2016 - 2021.

```
import pandas as pd
import numpy as np

df = pd.read_csv(r"data_prosseced\test_set_Preproceced_090225_V1.0

station_pairs = df.groupby(['start_station', 'end_station']).size().reset_in
station_pairs = station_pairs.sort_values('count', ascending=False)

print('Rutas más utilizadas:')
print(station_pairs.head(10))

total_trips = len(df)
print(f'Número total de viajes: {total_trips}')

mean_duration = df['duration'].mean()
print(f'Duración media de los viajes: {mean_duration:.2f} minutos')

station_usage = df.groupby('start_station').size().reset_index(name='co
station_usage = station_usage.sort_values('count', ascending=False)
```

```
print('Frecuencia de uso por estación:')
print(station_usage)

top_stations_by_trips = station_usage.sort_values('count', ascending=Fa
print('Estaciones más concurridas por número de viajes:')
print(top_stations_by_trips.head(10))

station_durations = df.groupby('start_station')['duration'].mean().reset_i
top_stations_by_duration = station_durations.sort_values('mean_duratio
print('Estaciones más concurridas por duración media de los viajes:')
print(top_stations_by_duration.head(10))
```

▼ Anexo 5: Script: horarios_saturados

Este código identifica las 10 estaciones más saturadas con sus 4 horarios mas concurridos para cada una de las estaciones antes mencionada

```
import pandas as pd

df = pd.read_csv(r"data_prosseced\test_set_Preproceced_090225_V1.0

station_pairs = df.groupby(['start_station', 'end_station']).size().reset_in
top_routes = station_pairs.sort_values('count', ascending=False).head(1


print('Top 10 rutas más utilizadas:')
print(top_routes)

for idx, route in top_routes.iterrows():
    start_station = route['start_station']
    end_station = route['end_station']
    filtered_data = df[(df['start_station'] == start_station) & (df['end_statio
    filtered_data['hour'] = pd.to_datetime(filtered_data['start_time']).dt.ho
    hourly_counts = filtered_data.groupby(['start_station', 'end_station', 'h
    top_hours = hourly_counts.sort_values('count', ascending=False).hea

    print(f'\nTop 4 horarios más saturados para la ruta iniciando en: {star
    print(top_hours)
```

▼ Anexo 6: Script: modelo_LR.py

Modelo de Regresión Lineal

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt
import numpy as np

df = pd.read_csv(r"data_prosseced\test_set_Preproceced_090225_V1.0

df['start_datetime'] = pd.to_datetime(df['start_date'] + ' ' + df['start_hou
df['year'] = df['start_datetime'].dt.year
df['month'] = df['start_datetime'].dt.month
df['day'] = df['start_datetime'].dt.day
df['day_of_week'] = df['start_datetime'].dt.dayofweek
df['hour'] = df['start_datetime'].dt.hour

total_trips_per_month = df.groupby(['year', 'month']).size().reset_index(
total_trips_per_month['time'] = total_trips_per_month['year'] + total_trip

X = total_trips_per_month[['time']]
y = total_trips_per_month['total_trips']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, rando

model = LinearRegression()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

mse = mean_squared_error(y_test, y_pred)
print(f'Error Cuadrático Medio: {mse}')

future_years_months = []
for year in range(total_trips_per_month['year'].max(), total_trips_per_mo
    for month in range(1, 13):
        future_years_months.append(year + month / 12)
```

```
future_predictions = model.predict(np.array(future_years_months).resha

for year_month, prediction in zip(future_years_months, future_predictior
    year = int(year_month)
    month = int((year_month - year) * 12) + 1
    print(f'Predicción de viajes para {year}-{month:02d}: {prediction}')

plt.figure(figsize=(12, 6))
plt.scatter(X_test, y_test, color='blue', label='Datos Reales')
plt.plot(X_test, y_pred, color='red', label='Predicción')
plt.scatter(future_years_months, future_predictions, color='green', label
plt.title('Predicción con LR a 3 años en el sistema de uso compartido de
plt.xlabel('Periodo')
plt.ylabel('Total de Viajes')
plt.legend()
plt.savefig('grafic/Modelo_analitico_(LR).png')
plt.show()
```

▼ Anexo 7. Script modelo_prophet

Modelo analítico usando Prophet de Meta

```
import pandas as pd
import matplotlib.pyplot as plt
import os
from prophet import Prophet

df = pd.read_csv(r"data_prosseced\test_set_Preproceced_090225_V1.0

df['start_date'] = pd.to_datetime(df['start_date'])

df['year'] = df['start_date'].dt.year
df['month'] = df['start_date'].dt.month
df['day_of_week'] = df['start_date'].dt.day_name()

total_trips_per_year = df.groupby('year').size().reset_index(name='total_
```

```
plt.figure(figsize=(10, 6))
plt.plot(total_trips_per_year['year'], total_trips_per_year['total_trips'], ma
plt.title('Conteo de Viajes en el sistema compartido de bicilcetas, ciudad
plt.xlabel('Año')
plt.ylabel('Total de Viajes')
plt.xticks(total_trips_per_year['year'])
plt.grid()
plt.tight_layout()
plt.show()

total_trips_per_day = df.groupby('start_date').size().reset_index(name='

total_trips_per_day.columns = ['ds', 'y']

model = Prophet(weekly_seasonality=True, yearly_seasonality=True)
model.fit(total_trips_per_day)

future = model.make_future_dataframe(periods=1095)
forecast = model.predict(future)
fig = model.plot_components(forecast)

for ax in fig.axes:
    ax.set_xlabel('Período', fontsize=12)
    ax.set_ylabel('Total de Viajes', fontsize=12)
    ax.grid(True)

fig.axes[0].set_title('Tendencia en el data-set y proyección a 3 años\n',
fig.axes[1].set_title('Tendencia Semanal, Periodo: 2016-2021', fontsize=1
fig.axes[2].set_title('Tendencia Anual, Periodo: 2016-2021', fontsize=14)

plt.suptitle('Tendencias en el sistema compartido de bicicletas, ciudad:
plt.tight_layout(rect=[0, 0.03, 1, 0.95])
plt.savefig('grafic/Modelo_analitcico_Prophet_Tendencias.png')
plt.show()

fig2 = model.plot(forecast)
plt.title('Predicción a 3 años de la tendencia Semanal en el uso del siste
plt.xlabel('Fecha', fontsize=14)
```

```
plt.ylabel('Total de Viajes', fontsize=14)
plt.grid(True)
plt.xticks(rotation=45)
plt.tight_layout()
plt.savefig('grafic/Modelo_analitcico_Prophet_Tendencias_actuales.png
plt.show()

annual_forecast = forecast.resample('Y', on='ds').sum()
plt.figure(figsize=(10, 6))
plt.bar(annual_forecast.index.year, annual_forecast['yhat'], color='skyb
plt.title('Predicción a 3 años (2022-2024) en el uso del sistema de bicic
plt.xlabel('Año', fontsize=14)
plt.ylabel('Total de Viajes', fontsize=14)
plt.xticks(annual_forecast.index.year)
plt.grid()
plt.tight_layout()
plt.legend()
plt.savefig('grafic/Modelo_analitcico_Prophet_predicción_3años.png')
plt.show()
```

▼ Anexo 8: Comparación de modelos

Modelos Analíticos

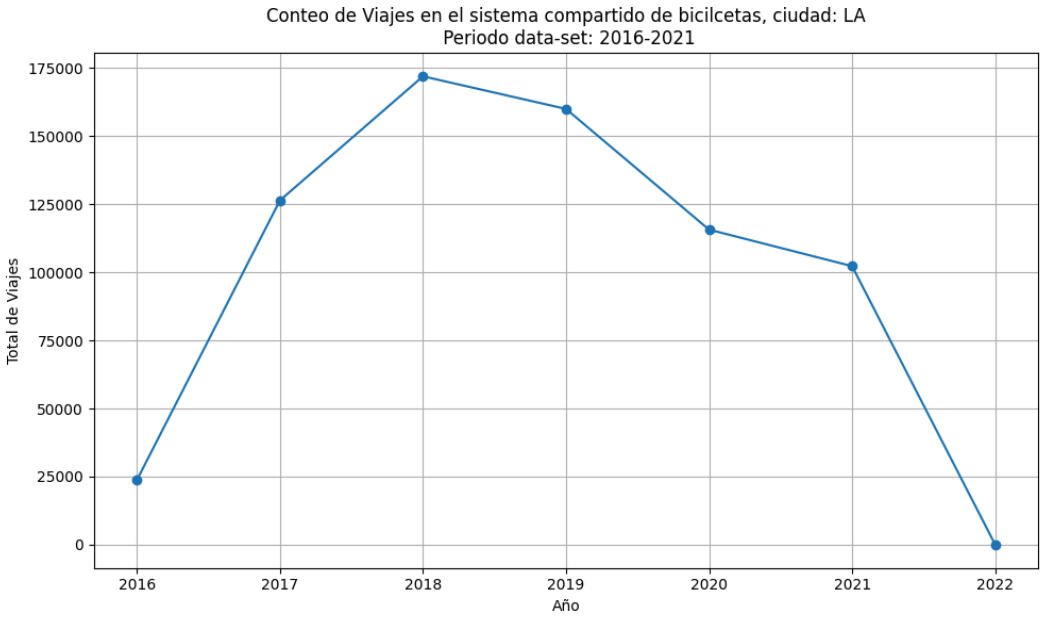
Modelo	Sobre el Módelo
Prophet	- Maneja temporadas y tendencias.
	- Fácil de interpretar y ajustar.
Regresión Lineal	- Fácil de interpretar.
	- Usa menos recursos de hadware
Árboles de Decisión	- Fácil de visualizar.
	- No requiere normalización de datos.
	- Maneja tanto datos categóricos como numéricos.
Random Forest	- Mejora la precisión al promediar múltiples árboles.
	- Reduce el riesgo de sobreajuste.
Redes Neuronales	- Capacidad para aprender patrones complejos.
	- Puede manejar múltiples entradas y salidas.
	- Buen rendimiento en tareas no lineales.

Consideraciones Finales

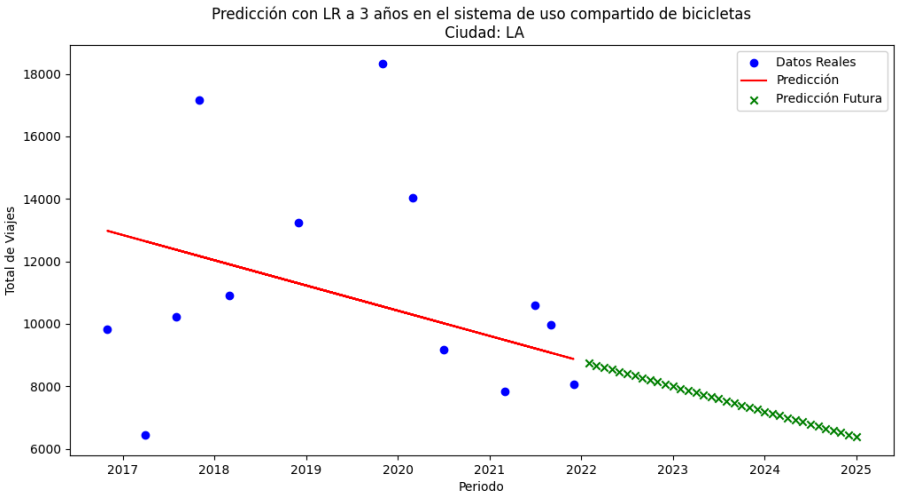
- **Prophet** y **Regresión Lineal:** recomendados para este proyecto debido a su capacidad para manejar series temporales y su facilidad de interpretación.
- Modelos más complejos como **Redes Neuronales** o **Random Forest** pueden ser útiles, pero requieren más datos y recursos.

▼ Gráficos:

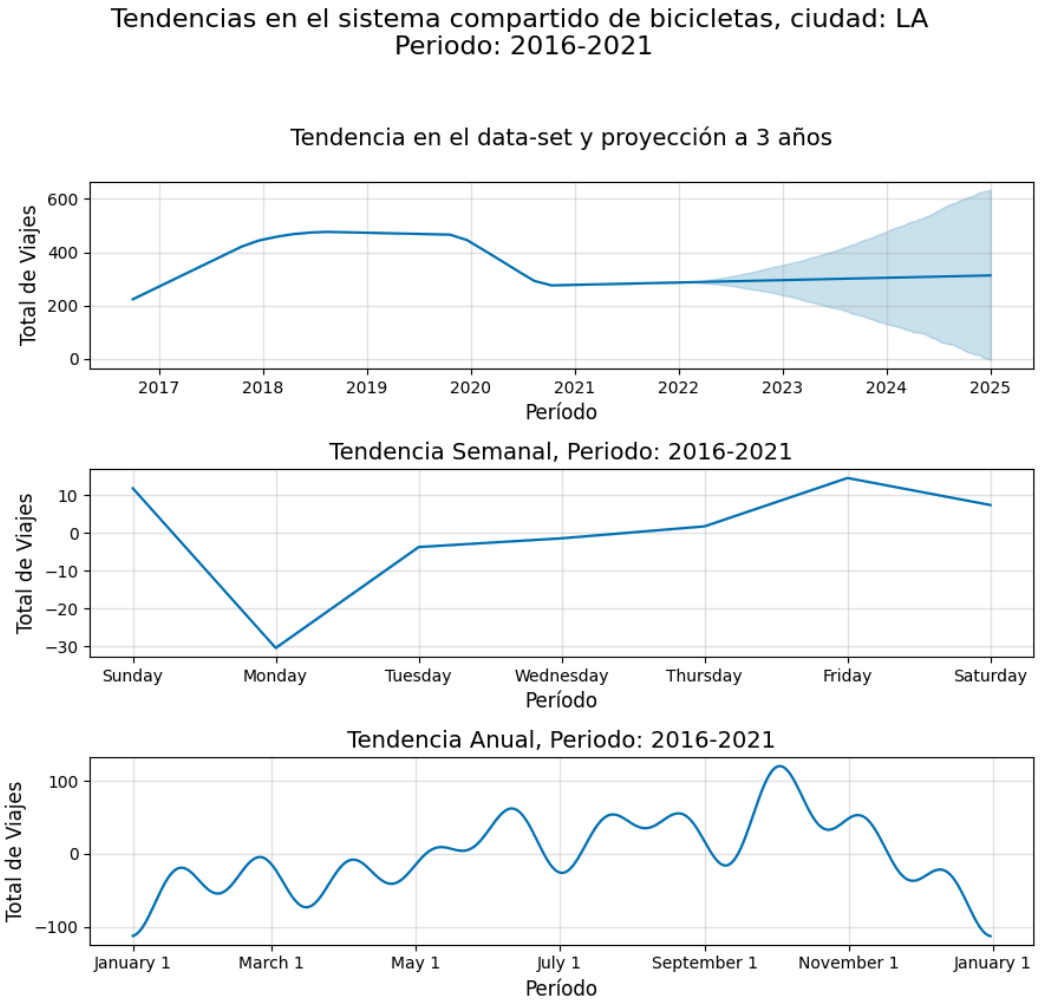
▼ Gráfico 1: Conteo de viajes al año a lo largo del tiempo dentro del data set.



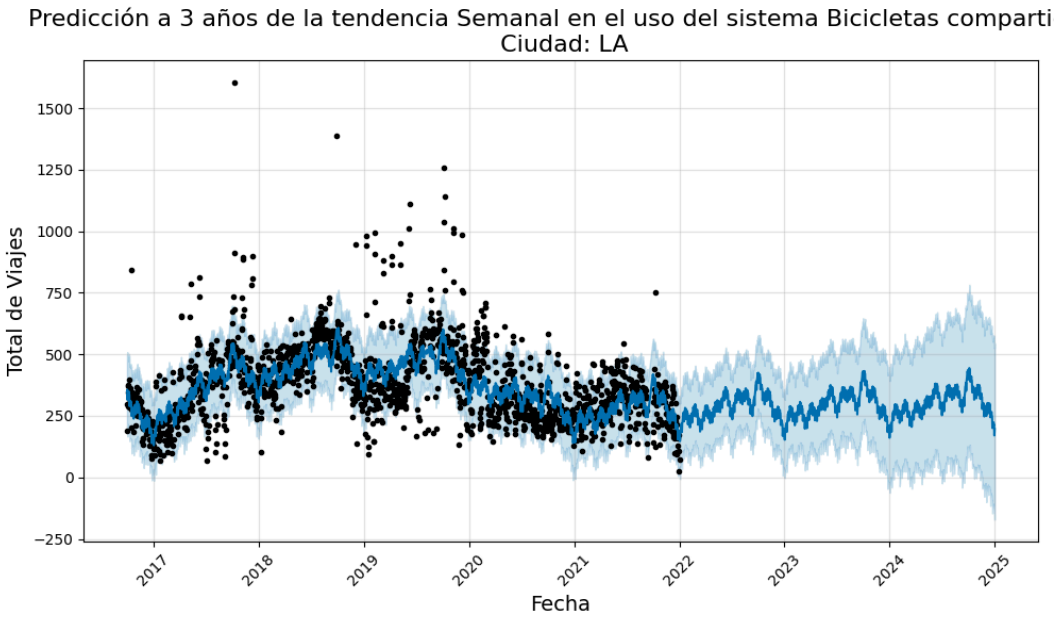
▼ Gráfico 2: Uso de regresión lineal para la predicción a 3 años.



▼ Gráfico 3. Tendencias en el uso del servicio por: Año, mes, semana, y una predicción a 3 años en el uso del servicio



▼ Gráfico 4. Tendencia anual a lo largo del tiempo con predicción a 3 años



▼ Gráfico 5. Predicción de los siguientes 3 años, respecto a la información a lo largo del tiempo