Informe Proyecto Final
Ciencia de Datos

Vicente Perelli Tassara	201756594-2
Nandy Troncoso Giacaman	201973570-5
Alan Zapata Silva	201956567-2

5 de Julio del 2023

Índice

Introducción

El objetivo de este proyecto es predecir la cantidad de delitos de distintos tipos que ocurrirían en distintos barrios. Para ello, se trabajo con datos reales del departamento de policía de la ciudad de Nueva York que van desde el 2006 al 2020 con el fin de desarrollar un modelo, el cual se entreno con estos datos.

El modelo busco ser capaz de predecir la cantidad de delitos de cada tipo que ocurrirán en cada barrio semanalmente. Para los datos en los cuales se basó para las predicciones, el modelo fue entrenado con datos del 2008 hasta el 2019 y para probar la precisión del modelo generado, se predijo la cantidad de delitos ha ocurrir de distintos tipos por cada barrio en el 2020 y fue testeado/comparado con los datos del 2020.

Descripción de Dataset

El Dataset del cual se obtuvo la información tiene 19 columnas, de las cuales las relevantes/utilizadas para el proyecto son:

- ✪ **ARREST_DATE**: Fecha en la cuál se efectuó el crimen. Viene con el formato MM/DD/AAAA
- ✪ **LAW_CAT_CD**: Crimen cometido, los cuales pueden ser Delitos (Felonies), Faltas (Misdemeanors) o Violaciones (Violations)
- ✪ **ARREST_BORO**: Barrio donde fue cometido el crimen, los cuales pueden ser Staten Island, Brooklyn, Queens, Manhattan o Bronx

Análisis exploratorio de datos.

Data Slicing:

Dado que se pide utilizar toda la información histórica del dataset hasta el año 2019 para entrenamiento, y se busco predecir el año 2020, se crearon 2 nuevos archivos .csv:

- **NYPD_ARRESTS_DATA_2019.csv:** Data histórica hasta el 2019
- **NYPD_ARRESTS_DATA_2020.csv:** Test data, para comparar con las predicciones.

Para los datos de entrenamiento no se tomó en consideración todo el Dataset, se seleccionaron fechas del 2008 hasta el 2019, ya que habían fechas sin datos en el 2007 que no entregaban información al modelo.

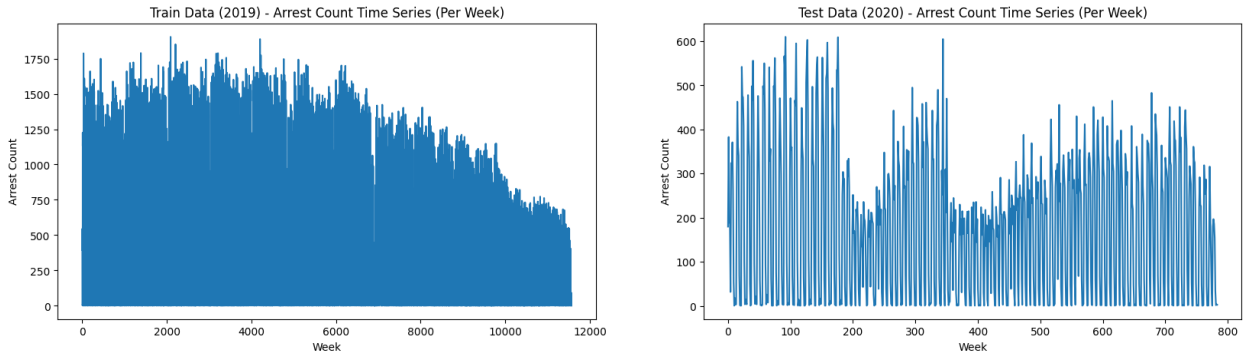


Fig_1: Dataframes de entrenamiento y prueba.

Luego de esto, se revisó si habían datos null y duplicados, encontrando algunos registros sin datos para la columna law_cat_cd, por lo que se eliminaron dichas filas.

Para efecto de las predicciones, se seleccionaron 3 columnas: **ARREST_DATE**, **LAW_CAT_CD** y **ARREST_BORO**, donde a estas dos últimas se reemplazaron las siglas por el significado real de sus registros, con el fin de hacer más legible el dataset.

Se creó un vector llamado ARREST_COUNT, con el fin de contabilizar la cantidad de arrestos semanales por cada Barrio/Crimen.

	ARREST_DATE	LAW_CAT_CD	ARREST_BORO	ARREST_COUNT
0	2008-01-06	Felony	Bronx	394
1	2008-01-06	Felony	Brooklyn	545
2	2008-01-06	Felony	Manhattan	477
3	2008-01-06	Felony	Queens	313
4	2008-01-06	Felony	Staten Island	53

Fig_2: train_arrest_count.head()

Una vez hecho esto, se busco crear un modelo por cada Barrio/Crimen para predecirlos todos.

Para este propósito, se pivoteó train_arrest_count para obtener un dataframe que tuviera una columna por Barrio/Crimen, utilizando como índice ARREST_DATE.

Este fue el dataset iterado, columna por columna, para entrenar los modelos y obtener predicciones.

LAW_CAT_CD	Felony					I				Misdemeanor					Violation					I
ARREST_BORO	Bronx	Brooklyn	Manhattan	Queens	Staten Island	Bronx	Brooklyn	Manhattan	Queens	Bronx	Brooklyn	Manhattan	Queens	Staten Island	Bronx	Brooklyn	Manhattan	Queens	Staten Island	Staten Island
ARREST_DATE																				
2008-01-06	394.0	545.0	477.0	313.0	53.0	4.0	6.0	3.0	33.0	1226.0	1132.0	1160.0	770.0	148.0	20.0	99.0	129.0	90.0	2.0	0.0
2008-01-13	549.0	733.0	622.0	393.0	86.0	3.0	9.0	14.0	28.0	1787.0	1620.0	1606.0	1059.0	217.0	39.0	165.0	170.0	131.0	4.0	1.0
2008-01-20	509.0	706.0	518.0	488.0	75.0	5.0	14.0	5.0	32.0	1609.0	1483.0	1552.0	971.0	171.0	41.0	145.0	168.0	115.0	2.0	1.0
2008-01-27	451.0	571.0	542.0	344.0	71.0	5.0	7.0	10.0	31.0	1375.0	1330.0	1417.0	880.0	191.0	31.0	120.0	159.0	112.0	1.0	2.0
2008-02-03	528.0	593.0	563.0	336.0	87.0	6.0	8.0	11.0	27.0	1370.0	1292.0	1329.0	902.0	195.0	22.0	123.0	178.0	158.0	4.0	0.0

Fig_3: train_arrest_pivott.head()

El procedimiento anterior también se efectuó con la información del 2020, a modo de lograr un dataframe que tuviera información a comparar con lo obtenido de los modelos, amen de poder imprimir gráficos comparativos entre predicciones y data actual del 2020.

Analizando Time Series

Para cada una de las combinaciones Barrio/Crimen, se aplicaron test ADF para determinar si los datos correspondían a estacionarios o no.

ADFuller:

Usando ADFuller, se estudiaron los datos de cada serie temporal correspondiente a cada barrio y tipo de delito para determinar si son estacionarias o no. Si el p-value es menor a 0.05, entonces era factible rechazar la hipótesis nula y decir que la serie es estacionaria con un 95% de confianza.De lo contrario, si el p-value es mayor a 0.05, entonces no se podia rechazar la hipótesis nula y decir que la serie no es estacionaria.

Se ejecuto el test para cada columna de la pivot table, y se obtuvieron los siguientes resultados:

```
stationary = [('Bronx', 'I'), ('Brooklyn', 'Felony'), ('Brooklyn', 'I'),
('Queens', 'Felony'), ('Staten Island', 'Felony'), ('Staten Island', 'I')]
```

Kwiatkowski-Phillips-Schmidt-Shin Test:

Se efectuó el mismo análisis, para corroborar estacionalidad a través del KPSS test.

KPSS es una prueba estadística utilizada para corroborar si una serie temporal es o no estacionaria. Su hipótesis nula es que la serie es estacionaria, mientras que su hipótesis alternativa es que no es estacionaria. La prueba KPSS compara esta estadística de prueba con un valor crítico de distribución para determinar si se debe rechazar la hipótesis nula. El valor crítico depende del nivel de significación y de la longitud de la serie temporal. El teorema que respalda la prueba KPSS. El teorema se llama Teorema de Ljung-Box y establece que, si una serie temporal es estacionaria, entonces los términos de autocorrelación de la serie deberían caer rápidamente a cero. En otras palabras, la autocorrelación de una serie temporal estacionaria debería ser insignificante para los retrasos más allá de un cierto punto. Por lo tanto, si la autocorrelación de la serie es significativa incluso para los retrasos más altos, entonces la serie puede ser no estacionaria.

Este test determina si la serie es estacionaria si encuentra un p-value > 0.05, rechazando la hipótesis nula.

Los resultados fueron:

```
stationary = []
```

Con estos resultados contradictorios entre ambos tests, no se pudo establecer estacionalidad en las series, por lo que se procedió a evaluar las tendencias y estacionalidad utilizando el método Rolling Mean.

Rolling Mean method:

Para analizar la tendencia y estacionalidad de las series de tiempo, analizaremos primero el promedio sobre el tiempo usando el método de rolling mean. El método de rolling mean calcula el promedio de los últimos n periodos. Esto es muy útil para suavizar las series de tiempo y resaltar las tendencias subyacentes o los patrones de la serie de tiempo.

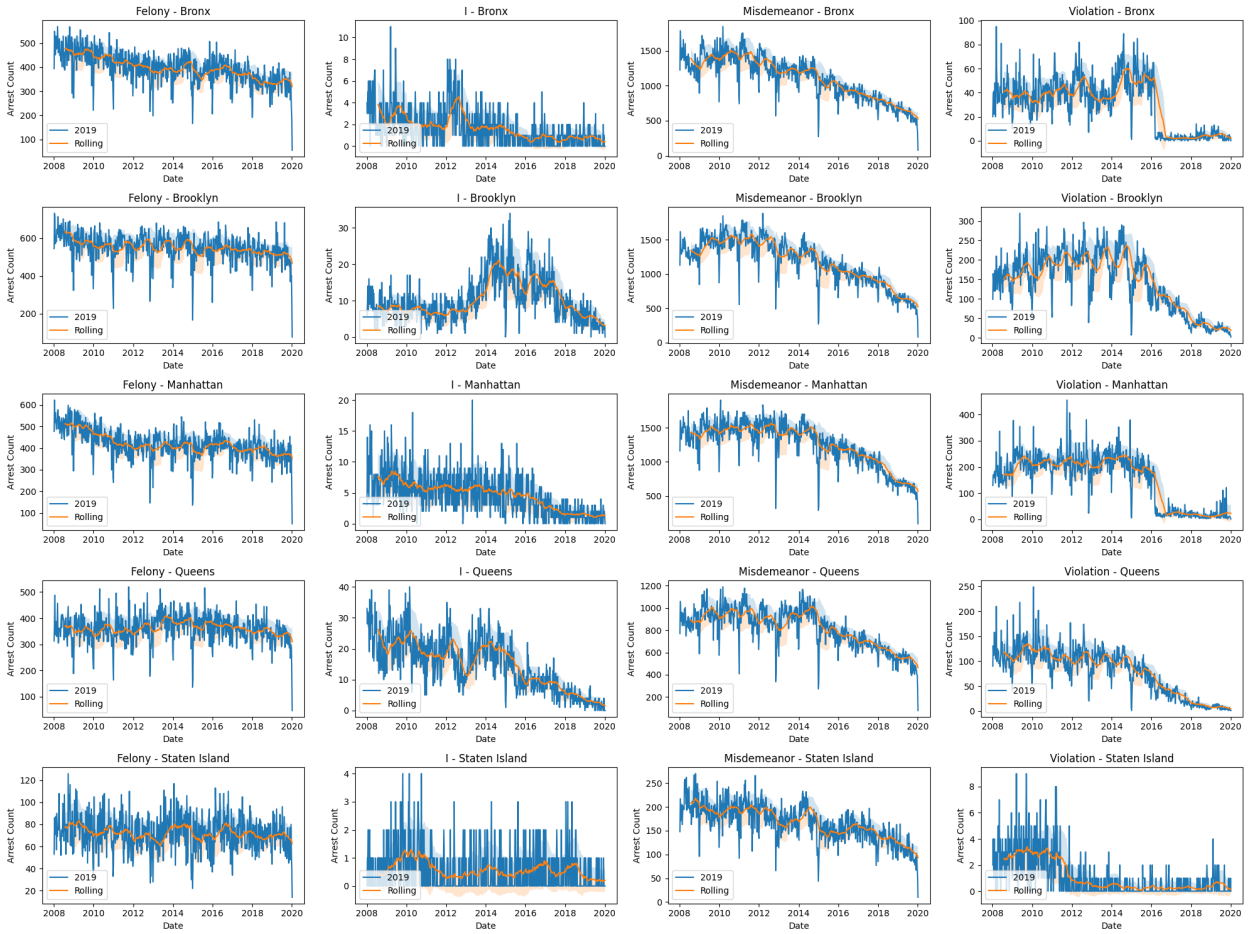


Fig_4: Train Dataset con Trends

De estos trends, podemos notar visualmente que si existe una estacionalidad en los datos, por lo que se determinó utilizar modelos que admitan seasonality.

Se debió haber extraído la componente estacional de nuestro train data, pero amen del tiempo no logramos ejecutar correctamente el código por lo que se omitió este paso.

Modelo

Habiendo determinado que los datos tienen un patrón a través del tiempo, se tomaron en consideración los siguientes modelos:

- ✪ **SARIMAX**
- ✪ **PROPHET**

SARIMAX Hyperparameter Tuning:

En base a lo anterior, se decidió buscar los mejores hiperparametros para “order” y “seasonal_order” necesarios para obtener mejores predicciones en el modelo.

Para esto, se utilizo auto_arima, una función del package pmdarima que permite ejecutar iterativamente Arima haciendo un GridSearch automatizado con los parámetros que se le entreguen.

```
auto_train_array = train_arrest_pivot.values.flatten()

model = pm.auto_arima(
    auto_train_array,
    seasonal=True,
    m=4,
    suppress_warnings=True,
    stepwise=True,
    error_action='ignore',
    trace=True,
    start_p=1,
    start_q=1,
    test='adf',
    max_p=3,
    max_q=3,)

print(model.summary())

best_order = model.order
best_seasonal_order = model.seasonal_order
```

Fig_5: auto_arima parameters.

Se escogieron los valores max_p = 3 y max_q = 3 experimentalmente, ya que al usar valores mayores las predicciones producían valores negativos para algunas combinaciones de Barrio/Crimen, lo cual no tendría sentido (*¿sería como liberar arrestos durante esos días?*).

A continuación, se ejecuta SARIMAX con los mejores parámetros encontrados, los cuales son los siguientes:

- **order** = (1,0,2)
- **seasonal_order** = (2, 0, 2, 4)

SARIMAX:

Para ejecutar SARIMAX con los datos de nuestro dataset y compararlos con los arrestos reales por Barrio/Crimen en el 2020, se itera en ambos dataset reducidos, train y test:

```
# Iterate over each borough and law category in train_arrest_pivot
for i, borough in enumerate(train_arrest_pivot.columns.get_level_values('ARREST_BORO').unique()):
    for j, law_cat_cd in enumerate(train_arrest_pivot.columns.get_level_values('LAW_CAT_CD').unique()):
        # Prepare the data for the specific borough and law category
        data = train_arrest_pivot.xs((law_cat_cd, borough), level=('LAW_CAT_CD', 'ARREST_BORO'), axis=1)
        test = test_arrest_pivot.xs((law_cat_cd, borough), level=('LAW_CAT_CD', 'ARREST_BORO'), axis=1)

        # Fit the SARIMA model with the best order and seasonal order to the data
        model = sm.tsa.SARIMAX(data, order=best_order, seasonal_order=best_seasonal_order)
        model_fit = model.fit()

        # Make predictions using index positions
        predictions = model_fit.predict(start="2020-01-06", end="2021-01-05")

        prediction_series = pd.Series(predictions, index=predictions.index)

        #axs[i,j].plot(data, label='2019')
        axs[i,j].plot(test, label='2020')
        axs[i,j].plot(prediction_series, label='Prediction')
        axs[i,j].legend()
        # axs[i,j].xticks(test.index.month)
        axs[i,j].set_xlabel('Week')
        axs[i,j].set_ylabel('Arrest Count')
        axs[i,j].set_title(f'{law_cat_cd} - {borough}')

plt.tight_layout()
plt.show()
```

Fig_6: SARIMAX model training.

De la predicción se obtuvieron los siguientes resultados:

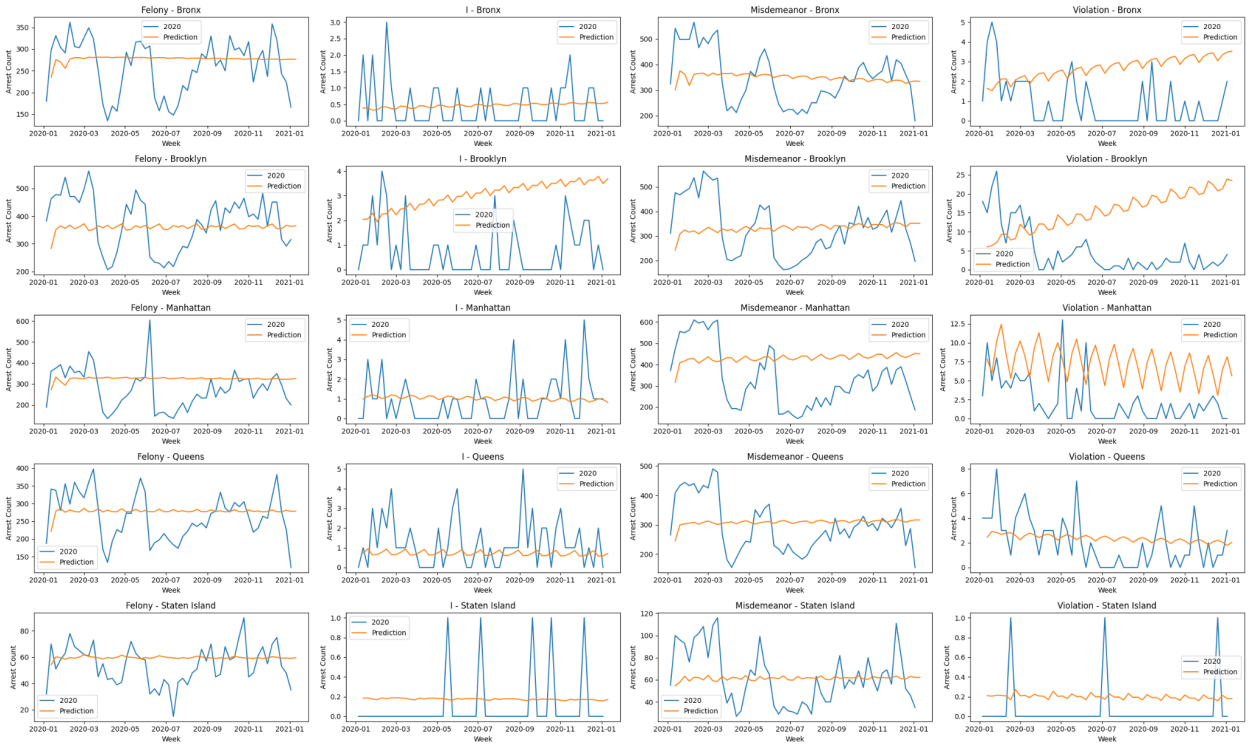


Fig_7: SARIMAX results.

PROPHET

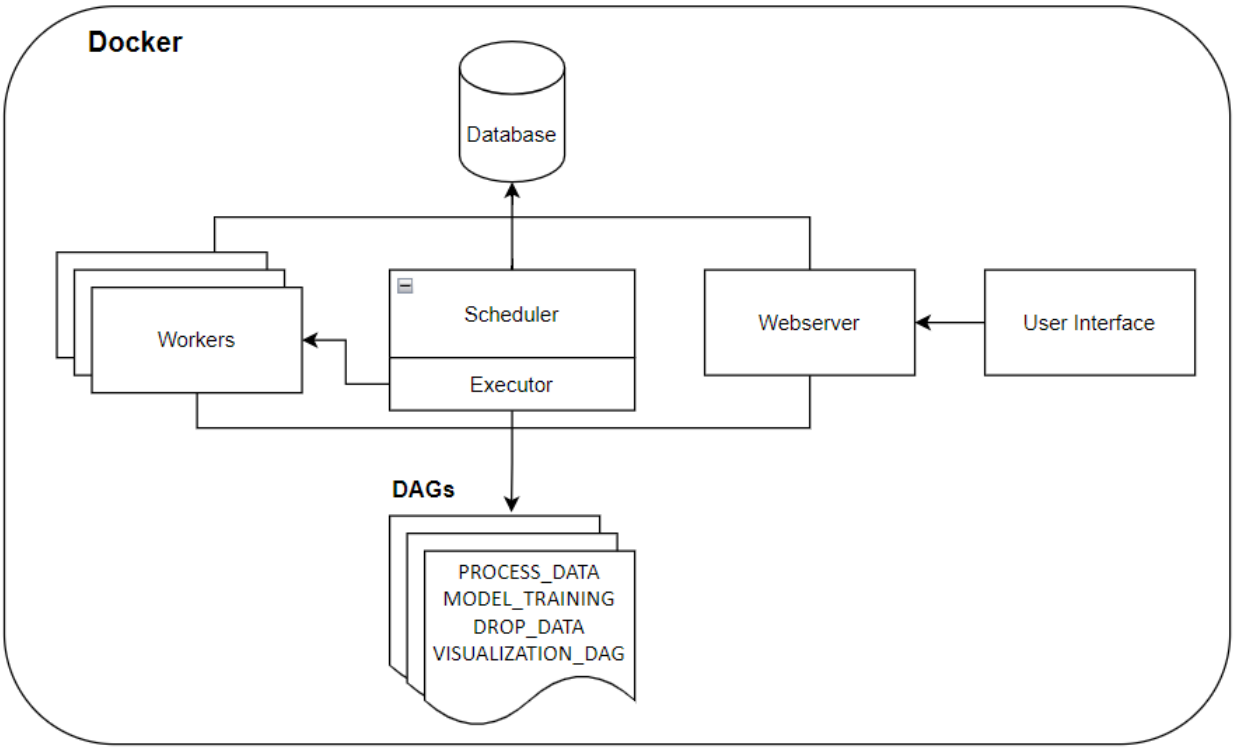
Se intentó efectuar el mismo procedimiento con PROPHET, pero por tiempo no se alcanzó a tunear hiperparámetros.

Por lo anterior, se tomó la decisión de mencionar que también PROPHET podría ser de utilidad para realizar predicciones sobre este dataset, pero no fue ocupado ni en el notebook exploratorio ni en Apache Airflow.

Apache Airflow

Infraestructura

La estructura sobre la que ocurre el flujo con el cual se trabajó se representa en la siguiente imagen



DAGs

Se crearon cuatro DAGs para realizar las tareas necesarias en Apache Airflow, estos fueron:

- **PROCESS_DATA:**

  Extrae la información del csv original ubicada en *"/tmp/Data"*, se limpia acorde a la información necesaria y con base en esta información ya tratada se crean las tablas **train** y **test** en postgres con la información limpia.

- **MODEL_TRAINING:**

  Crea la tabla de resultados de predicciones, entrena el modelo con los datos que extrae de la base de datos y guarda los resultados de las predicciones obtenidas de este.

- **DROP_DATA:**

En caso de que las predicciones salgan mal (tanto en errores como calidad de predicciones), se elimina la tabla “Predicciones” y se vuelve a generar la tabla vacía.

Este DAG fue utilizado solo en periodos de prueba, no debería utilizarse normalmente, ya que en la entrega final no existen errores que necesiten utilizarla.

- **VISUALIZATION_DAG:**

  Lee la tabla **predictions** y **test,** y crea gráficos comparativos entre las predicciones y los datos actuales para 2020, a modo de facilitar la visualización de los resultados, tal como en el jupyter notebook.

  Estos resultados son almacenados en el volumen compartido: Data.

Conclusiones

Los resultados predichos por el modelo SARIMAX están bastante lejos de lo que realmente registran los datos del año 2020. Esto puede deberse en gran parte a que el año 2020 fue un año atípico, ya que por decreto gubernamental, en New York se llevó a cabo una cuarentena por el COVID-19 durante algunos meses.

Se puede decir que a pesar de que no es el mejor, de todas formas en algunos casos la predicción se condice con los datos actuales, obteniendo intersecciones entre las rectas de predicción y test data.