HERRAMIENTAS DE SOFTWARE PARA BIG DATA

FACULTAD DE INGENIERÍA

Algoritmo de predicción de probabilidad en arribo en tiempo de vuelos

Autor: Ing. Yader Luis Coca Ribas

Índice

Problema: Se necesita saber teniendo los datos históricos de varias aerolíneas dentro de Estados Unidos cual es la probabilidad de que dada una aerolínea una fecha de salida, un punto de salida y de llegada que llegue temprano.

Objetivo General**:** Desarrollar algoritmo Random Forest para los vuelos domésticos de Estados Unidos.

Objetivos Específicos**:**

1. Definir ingeniería de atributos, seleccionar el conjunto de datos, correlación de los atributos y establecer los datos para hacer una recomendación.
2. Seleccionar el algoritmo y el conjunto de datos de entrenamiento y de prueba.
3. Matriz de confusión para establecer el rendimiento del algoritmo seleccionado
4. Automatizar el algoritmo.

Alcance:

Recopilar información de aeropuertos, vuelos y aerolíneas sobre los vuelos en Estados Unidos en el año 2015. Introducir estos datos en un sistema de archivos distribuidos (HDFS). Leer estos archivos haciendo uso de Hive. Utilizar Spark para levantar la información de las tablas en

formato Parquet. Implementar ingeniería de atributos y utilizar el algoritmo de Random Forest para encontrar la predicción sobre si llegara temprano.

Arquitectura de la Solución:

Para realizar el desarrollo de este trabajo se usó la arquitectura lambda, ya que se ajusta perfectamente en el cumplimiento de las características que precisamos para resolver el problema. Dado a que no contamos con una fuente de datos en tiempo real, decidimos hacer la ingestión de los datos usando un lote batch, que en sentido práctico sería un stream acotado. Estos lotes están representados por un dataset que se encuentra público en kaggle[i]. La capa de Fuente (Source) estaría encargada de Real-Time Ingestión (o near realtime). Dado a las características de los datos y el proyecto no hice uso de la capa de ingestión. La capa de almacenamiento se confecciono con el uso de Apache Hadoop que nos permite tener archivos distribuidos en datanodes y hacer uno efectivo de map reduce. Permitiendo sobre el utilizar varias aplicaciones del llamado ecosistema de Hadoop. La destilación de los datos (Destillation tier) se realizo utilizando Jupyter y Sklearn que serian el ambiente de desarrollo que permite el uso de R y Python y el manejo de los algoritmos de ingeniería de atributos y featuring respectivamente. También se utilizó Apache Spark con su librería pyspark para traer el archivo parquet que tenemos en Hive; Hive trabaja en la capa de utilización y funciona como nuestro enlace a Hadoop y este cuenta con tablas columnares que contienen las referencias a los ficheros distribuidos de Hadoop. A ser referencias nos permite rápidas consultas y un mejor manejo de la información en un formato muy parecido a SQL. Se utilizo para las consultas y el manejo de la utilización Hue que seria nuestra ultima capa de componentes antes del usuario final.

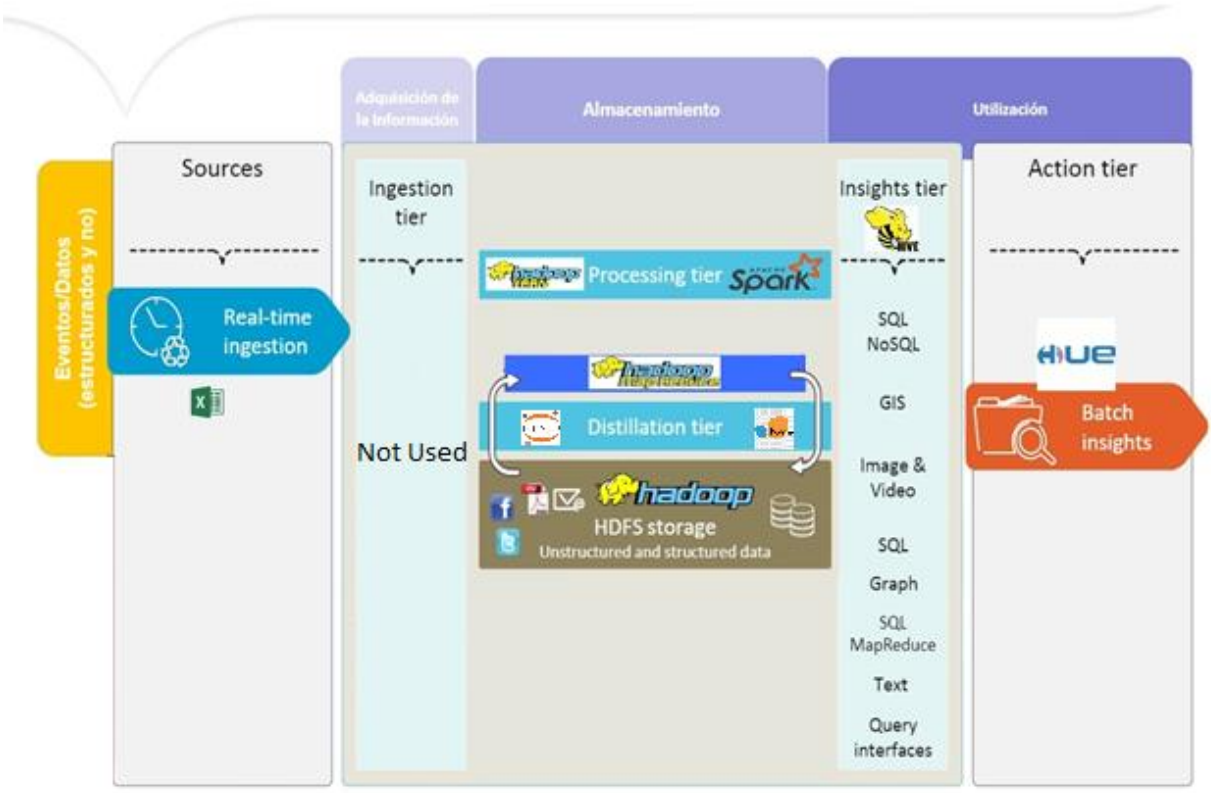

Imagen 1: Arquitectura Propuesta

Collection Tier:

Este proyecto de Big Data se llevó a cabo sobre un sistema de archivos distribuido, usando el sistema de almacenamiento distribuido del ecosistema de Hadoop (HDFS). A partir del dataset provisto por kaggle, sobre los vuelos y sus estadísticas en Estados Unidos ordenados por fecha los cuales se encontraban en archivos .csv y se insertaron de forma manual sobre HDFS usando su interfaz de usuario.

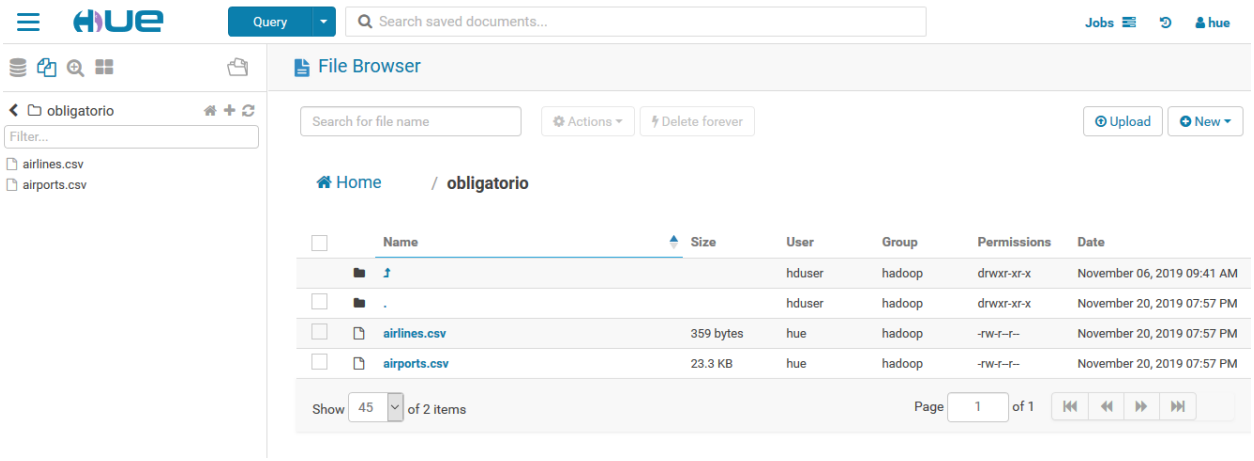

Imagen 2: Repositorio de archivos en HDFS vistos desde Hue.

Con esto nos aseguramos de que Hadoop se encargue de hacer el map reduce y hacer uso de la distribución, y Yarn se encarga de la gestión de los recursos del clúster.

Se uso el patrón cliente/servidor el cual es stateless, no necesito saber el estado de un cliente que esta realizando un requerimiento. También es posible sumar instancias del servicio sin alterar instancias de ejecución lo cual lo hace escalabre horizontalmente.

Message Queuing Tier:

Los datos fueron escritos en formato columnar usando el formato Apache Parquet para hacer más fácil el proceso de map reduce. Para esta acción use Hive ya que los datos los había importado de HDFS a Hive para su procesamiento y consulta. En Hive con una simple consulta hice una unión de las tablas y convertí esa unión en una tabla parquet.

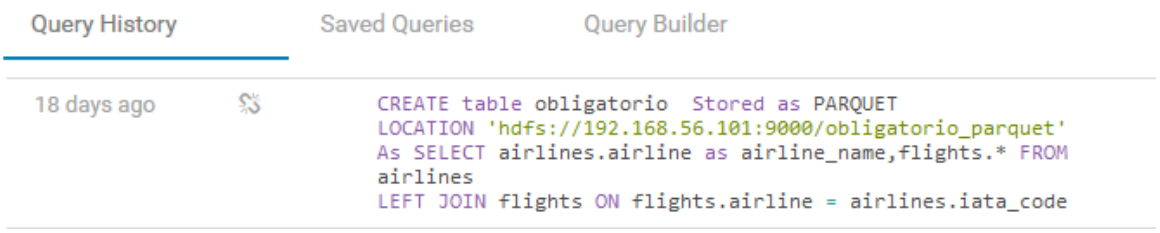

Imagen 3: Creación de table en Hive como Parquet

Analysis tier:

En esta capa se levantaron los datos de hive usando Spark, en especial la librería para Python pyspark.

```
In [1]: import pandas as pd

        from pyspark import SparkContext, SparkConf
        from pyspark.sql import SparkSession, HiveContext
        from pyspark.sql import SQLContext
```

Imagen 3: Librería PySpark

Los datos fueron levantados desde la previamente creada tabla en parquet

```
In [2]: #Spark Session
        spark = SparkSession.builder \
            .master("local[*]")  \
            .appName('Obligatorio') \
            .config('spark.executor.memory', '5gb') \
            .config("spark.cores.max", "6") \
            .getOrCreate()

        sc = spark.sparkContext
        sqlContext = SQLContext(sc)

        spark_df = sqlContext.read.parquet('hdfs://192.168.56.101:9000/obligatorio_parquet')
```

Imagen 4 : Spark obteniendo los datos desde Hive

El desarrollo del feature engineering se realizó en Python, así como la implementación de el algoritmo de clasificación que en este caso fue elegido Random Forest. Para el manejo de los data sets elegí usar pandas ya que me resultaba más sencillo la manipulación y transformación de los datos. Se eligió dado el volumen de datos solo realizar el estudio predictivo para la aerolínea American Airlines y en el mes de septiembre.

```
In [5]: pandas_df.head()
```

Out[5]:

	airline_name	year	month	day	day_of_week	airline	flight_number	tail_number	origin_airport	destination_airport	...	arrival_time	arrival_delay	diverted
0	American Airlines Inc.	2015	9	13	7	AA	2326	N012AA	IAH	MIA	...	2236.0	-23.0	None
1	American Airlines Inc.	2015	9	28	1	AA	2224	N528AA	MSP	ORD	...	1627.0	168.0	None
2	American Airlines Inc.	2015	9	13	7	AA	2315	N3JNAA	MCO	MIA	...	2040.0	4.0	None
3	American Airlines Inc.	2015	9	13	7	AA	218	N864AA	SFO	LAX	...	2143.0	42.0	None
4	American Airlines Inc.	2015	9	28	1	AA	97	N3KGAA	DCA	DFW	...	1430.0	-5.0	None

5 rows × 32 columns

Imagen 5: Primeros 5 elementos del dataset

Long-term storage tier:

Con la intención de realizar con precisión y facilidad las operaciones estadísticas y matemáticas sobre el obtenido y filtrado dataset se eligió utilizar la librería *sklearn*[ii] . Con la cual se realizaron desde las particiones del dataset en train y test hasta la implementación del algoritmo de clasificación y la medición de las métricas estadísticas.

```
In [17]: from sklearn.model_selection import train_test_split
         train_x, test_x, train_y, test_y = train_test_split(pandas_df.drop('late', axis=1), pandas_df['late'], test_size=0.2, random_
         state=42)

In [18]: train_x.shape
Out[18]: (58703, 180)

In [19]: test_x.shape
Out[19]: (14676, 180)

In [20]: from sklearn.ensemble import RandomForestClassifier

         model = RandomForestClassifier(random_state=13)
         model.fit(train_x, train_y)
```

Imagen 6 Uso de la librería Scikit

Se realizaron varios análisis estadísticos a cerca de la predicción como el área bajo la curva

```
In [21]: # mean accuracy
         predicted = model.predict(test_x)
         model.score(test_x, test_y)

Out[21]: 0.7074816026165167

In [37]: prediction = pd.DataFrame(predicted, columns=['predictions']).to_csv('predict

In [22]: #prediction of probability ,chance of being on time
         from sklearn.metrics import roc_auc_score
         probabilities = model.predict_proba(test_x)
         roc_auc_score(test_y, probabilities[:, 1])

Out[22]: 0.6574425577020072

In [32]: print(probabilities)

         [[0.9 0.1]
          [1.  0. ]
          [0.9 0.1]
          ...
          [0.8 0.2]
          [1.  0. ]
          [0.9 0.1]]
```

Image 7 : Predicción y el área bajo la curva

ROC

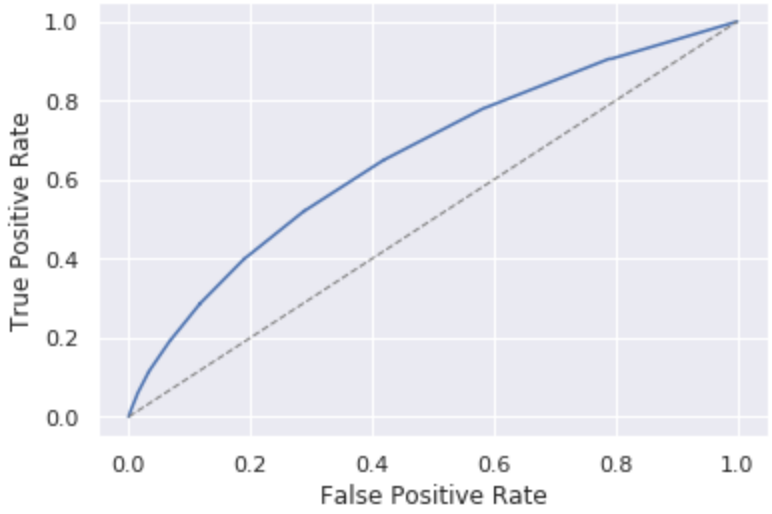

Imagen 8 : ROC

Así como se estableció la efectividad del modelo a través de la matriz de confusión, el cálculo del recall y la precisión.

```
In [23]: #confusion matrix
         from sklearn.metrics import confusion_matrix
         confusion_matrix(test_y, predicted)

Out[23]: array([[9146, 1222],
                [3071, 1237]])

In [24]: #presicion
         from sklearn.metrics import precision_score

         train_predictions = model.predict(train_x)
         precision_score(train_y, train_predictions)

Out[24]: 0.9885370118515641

In [25]: #recall
         from sklearn.metrics import recall_score

         recall_score(train_y, train_predictions)

Out[25]: 0.9130278741476253
```

Ilustración 9 : Librería scikit learn para las métricas

Data Access:

Para la realización de los reportes multidimensionales con análisis de profundidad sobre la arquitectura se utilizó Apache Hue que fue la interfaz web por de facto con la que se manejó el acceso a Hive y a Hadoop ya que permite el gobierno de estas aplicaciones desde una perspectiva superior sin tener que acceder a ellas directamente.

Para mostrar gráficamente la probabilidad del modelo en fechas acotadas usé numpty y se mostraron las probabilidades de arribo en tiempo para las fechas de septiembre desde el 17/9 al 23/9

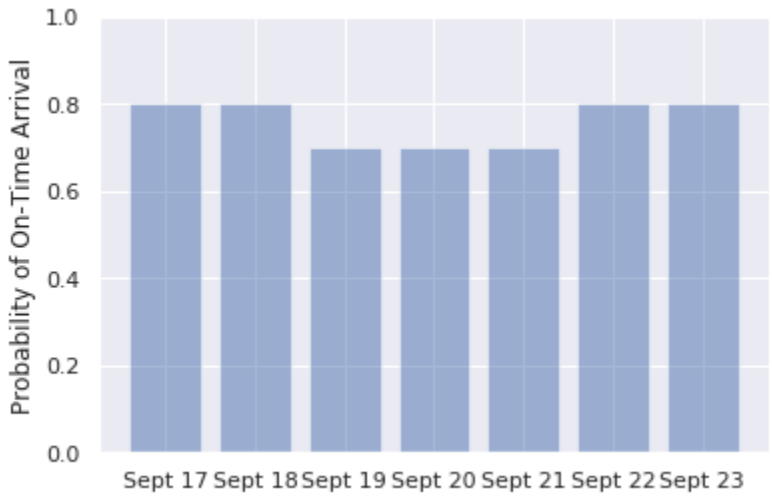

Imagen 10: Gráfico de la probabilidad de arribo en tiempo

Resultado de la predicción, escrito en una tabla …implementar con la exportación de las predicciones en hue o en power bi

Implementación:

Para la implementación se usaron varias herramientas y recursos conformados por una máquina virtual con CentOS 7 instalada con Hue, Hadoop, Hive y a su vez contiene otra en Docker para la utilización de la notebook Jupyter.

Siendo la siguiente figura parte del filesystem de Hadoop

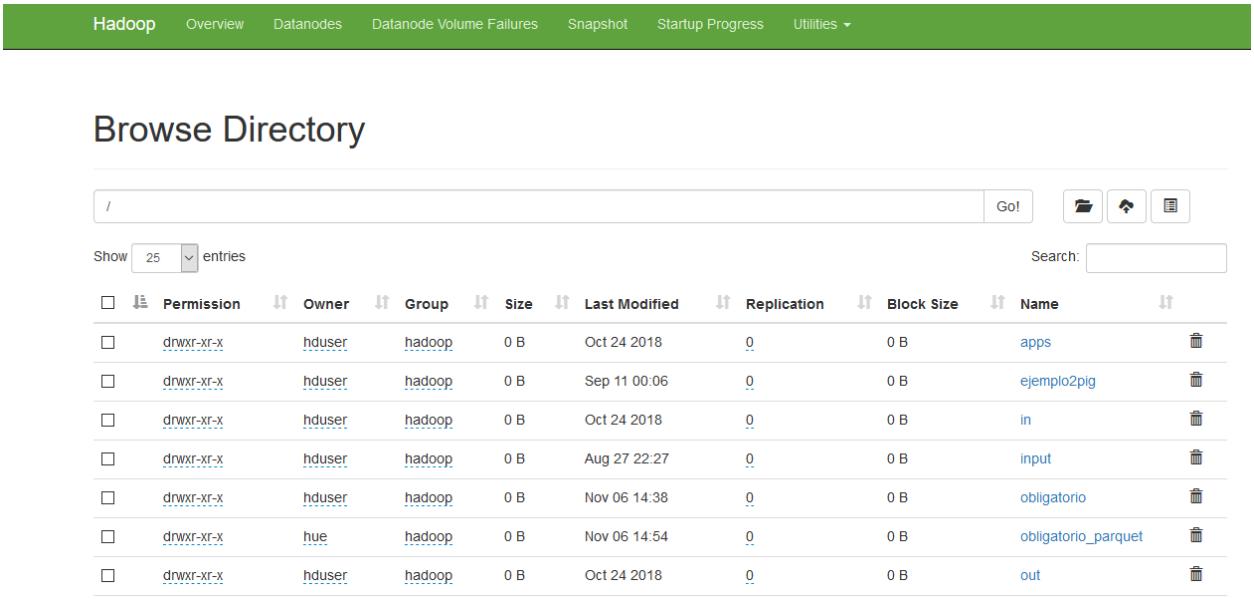

Imagen 11: Hadoop File System

Hive y las tablas utilizadas

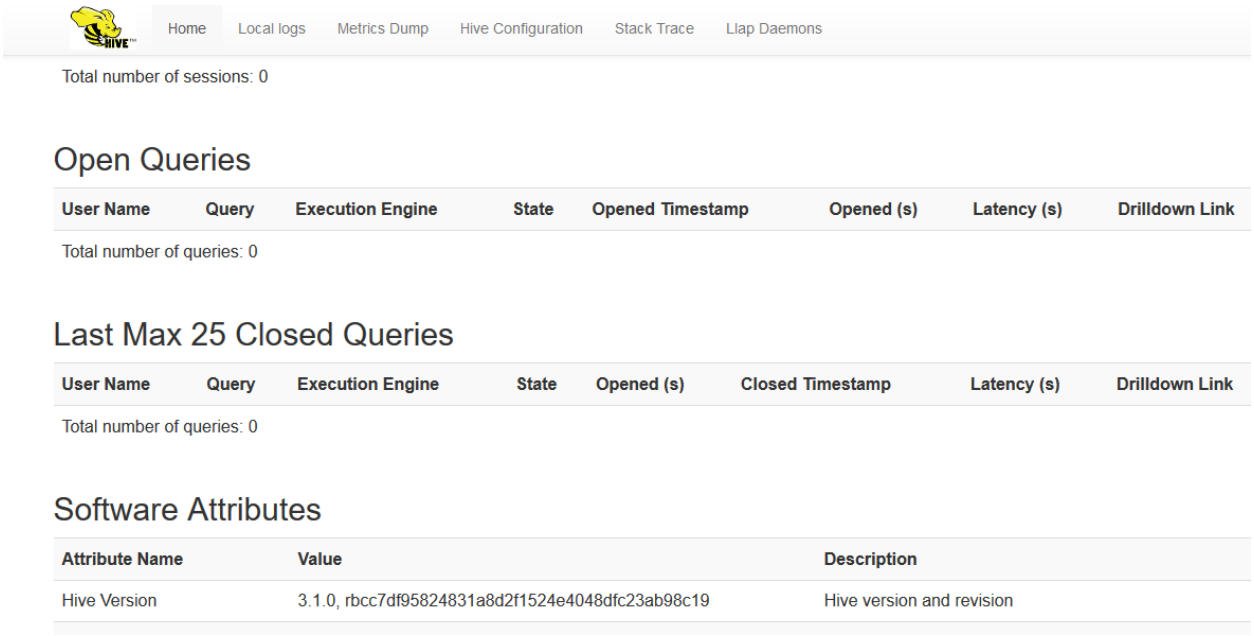

Imagen 12: Interfaz de Hive

El acceso y manejo de las tablas de hive las realice desde Hue.

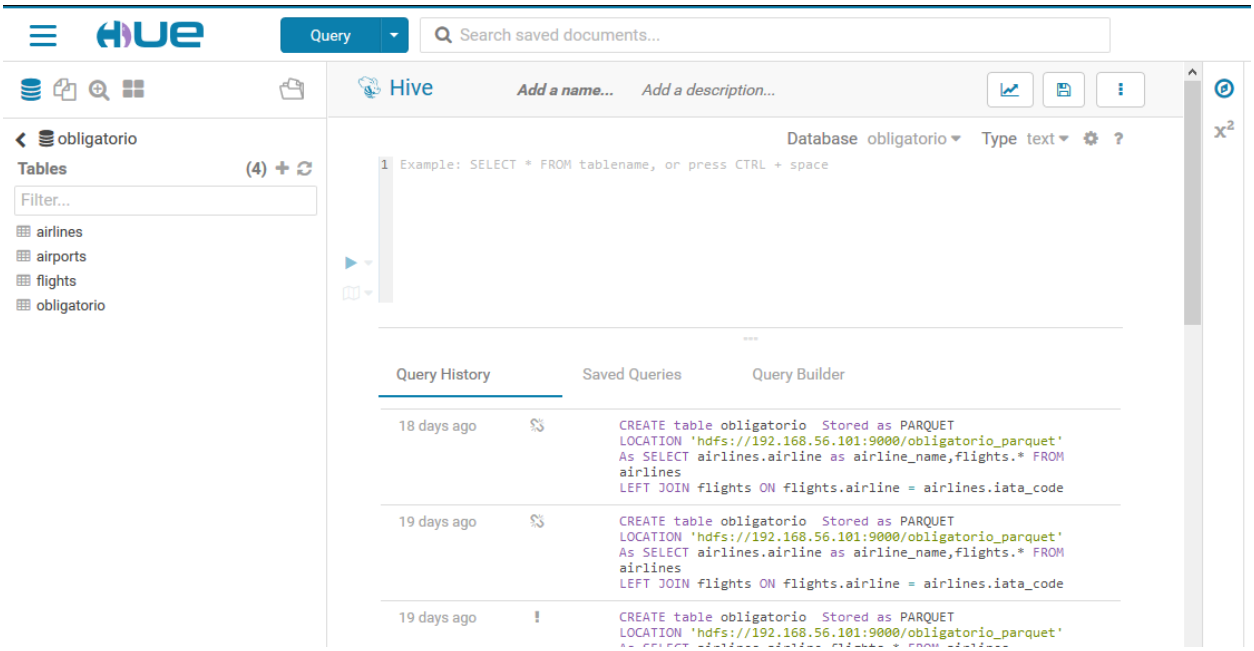

Imagen 13: Espacio de trabajo de Hive con las tablas desde Hue.

Conclusiones

Uno de los aspectos mas importantes de la preparación del dataset fue la selección de los features que son relevantes a nuestros resultados. Filtré las columnas que no afectaran los resultados para así evitar sesgo y multicolinealidad. Otra tarea impórtate para la ingeniería de atributos lo constituyó la eliminación de valores vacíos o faltantes y donde estaban.

```
In [10]: pandas_df.isnull().values.any()
Out[10]: True
```

Imagen 14 : Pandas mostrando la existencia de ficheros nulos

```
In [11]: pandas_df.isnull().sum()

Out[11]: airline_name              0
         year                      0
         month                     0
         day                       0
         day_of_week               0
         airline                   0
         flight_number             0
         tail_number               0
         origin_airport            0
         destination_airport       0
         scheduled_departure       0
         departure_time          321
         departure_delay         321
         taxi_out                350
         wheels_off              350
         scheduled_time            0
         elapsed_time            491
         air_time                491
         distance                  0
         wheels_on               365
         taxi_in                 365
         scheduled_arrival         0
         arrival_time            365
         arrival_delay           491
         diverted              73379
         cancelled             73379
         cancellation_reason       0
         air_system_delay          0
         security_delay            0
         airline_delay             0
         late_aircraft_delay       0
         weather_delay             0
         late                      0
```

Imagen 15: Listado de las columnas con datos nulos

A partir de ahí me quede solo con las columnas (*features* que me interesaban)

```
In [12]: pandas_df = pandas_df[["month", "day", "day_of_week", "origin_airport", "destination_airport", "scheduled_departure", "late"]]
         pandas_df.isnull().sum()

Out[12]: month                0
         day                  0
         day_of_week          0
         origin_airport       0
         destination_airport  0
         scheduled_departure  0
         late                 0
         dtype: int64
```

Imagen 16 : Columnas con las que vamos a trabajar

Al no contar con una columna que me permitirá efectivamente clasificar el resultado de si llegaba tarde el vuelo o no, agregué una columna llamada *late* que contiene el valor de 1 o 0 dependiendo de si la columna *arrival delay* es mayor que cero.

```
In [6]: def late(row):
            if row['arrival_delay'] > 0:
                val = 1
            else:
                val = 0
            return val

In [7]: pandas_df ['late'] = pandas_df.apply(late, axis=1)
```

Imagen 17 : Creación de nueva columna con los valores a predecir(Y)

. Sobre esta columna evalué la predicción. Si llega tarde se evalúa la columna en 1 y si no se evalúa en cero. La idea es evaluar la probabilidad de un vuelo dado llega temprano a su destino, dígase tiene valor 0.

Luego me era importante agrupar las fechas a la hora mas cercana para que no estuvieran fraccionadas y así facilitar el trabajo de predicción.

	month	day	day_of_week	origin_airport	destination_airport	scheduled_departure	late
0	9	13	7	IAH	MIA	1930	0
1	9	28	1	MSP	ORD	1211	1
2	9	13	7	MCO	MIA	1930	1
3	9	13	7	SFO	LAX	1930	1
4	9	28	1	DCA	DFW	1212	0

In [14]:
```
import math
def binned(df):
    for index, row in df.iterrows():
        df.loc[index, 'scheduled_departure'] = math.floor(row['scheduled_departure'] / 100)
```

In [15]:
```
binned(pandas_df)
pandas_df.head()
```

Out[15]:

	month	day	day_of_week	origin_airport	destination_airport	scheduled_departure	late
0	9	13	7	IAH	MIA	19	0
1	9	28	1	MSP	ORD	12	1
2	9	13	7	MCO	MIA	19	1
3	9	13	7	SFO	LAX	19	1
4	9	28	1	DCA	DFW	12	0

Imagen 18 : Muestra de "binding" a la fecha

Aquí los datos quedaron listos para utilizar un modelo sobre ellos. Para el modelo elegí hacer una clasificación usando la librería **scikit-learn** que incluye una amplia variedad de clases para implementar modelos de machine learning comunes. También se seleccionaron las particiones de entrenamiento y prueba usando la mencionada librería que nos facilita con una función simple este trabajo.

In [17]:
```
from sklearn.model_selection import train_test_split
train_x, test_x, train_y, test_y = train_test_split(pandas_df.drop('late', axis=1), pandas_df['late'], test_size=0.2, random_state=42)
```

Imagen 19 : Partición en test y train

Luego de haber corrido el algoritmo predictivo se estableció basados en la librería **scikit-learn** la matriz de confusión

```
In [23]: #confusion matrix
         from sklearn.metrics import confusion_matrix
         confusion_matrix(test_y, predicted)

Out[23]: array([[9146, 1222],
                [3071, 1237]])
```

Imagen 20 : Matriz de confución

Esta como se puede observar contiene una alta cantidad de falsos positivos y negativos, pero pienso que puede darse a que no se tuneo precisamente el algoritmo. Además de esto se utilizaron otras medidas para detectar la efectividad del modelo como lo fue el recall

```
In [25]: #recall
         from sklearn.metrics import recall_score

         recall_score(train_y, train_predictions)

Out[25]: 0.9130278741476253
```

Imagen 21 : Recall

Mientras que el recall fue positivo, la precisión fue mejor

```
In [24]: #presicion
         from sklearn.metrics import precision_score

         train_predictions = model.predict(train_x)
         precision_score(train_y, train_predictions)

Out[24]: 0.9885370118515641
```

Imagen 22 : Presición

Este algoritmo debe correr como una tarea programada ya sea usando un *cron job* o una tarea programada dependiendo del sistema operativo del servidor todos los días y así reentrenar el modelo y devolver las predicciones en demanda.

Visualización

Para la visualización se uso Power BI , se adjunta un fichero para su uso así como un pdf con el reporte ejemplo de la visualización de las predicciones con respecto los datos.

Bibliografía

Bonillo, P. B. (s.f.). Propuesta de una Arquitectura de Gestión de Grandes Volúmenes de Datos para la Analítica en Tiempo Real bajo Software Libre. Recuperado 30 noviembre, 2019, de https://www.linkedin.com/pulse/propuesta-de-una-arquitectura-gesti%C3%B3n-grandes-datos-la-bonillo-ramos

Chakrabarty, N. C. (2019, 15 marzo). A Data Mining Approach to Flight Arrival Delay Prediction for American Airlines. Recuperado 1 diciembre, 2019, de https://deepai.org/publication/a-data-mining-approach-to-flight-arrival-delay-prediction-for-american-airlines

Domínguez, J. D. (2019, 1 noviembre). De Lambda a Kappa: evolución de las arquitecturas Big Data. Recuperado 30 noviembre, 2019, de https://www.paradigmadigital.com/techbiz/de-lambda-a-kappa-evolucion-de-las-arquitecturas-big-data/

Zorrilla, M. Z., García, D. G., & Saiz, S. (2017, enero). Arquitecturas y tecnologías para el bigdata [Documento]. Recuperado 30 noviembre, 2019, de https://ocw.unican.es/pluginfile.php/2396/course/section/2473/tema%203.2%20Arquitecturas%20y%20tecnologi%CC%81as%20para%20el%20big%20data.pdf

[i] https://www.kaggle.com/usdot/flight-delays
[ii] https://scikit-learn.org/stable/