IBA

Institute of Business Administration
Big Data Analytics Project

Big Data Analytics using Hadoop, Hive and Superset – Machine Learning using Apache Spark (pyspark)

Dataset by Kaggle: https://www.kaggle.com/datasets/threnjen/2019-airline-delays-and-cancellations?select=train.csv

Submitted by

Tabina Navaid (ERP: 27265)
Marium Mohammad Nasir (ERP: 17876)

1. Dataset

The dataset was taken from Kaggle and is a classification dataset that consists of Airline delays with weather and airport detail. The dataset has the following attributes.

Data size	1.68 Gigabytes
Data Format	CSV
Data Description	2019 Airline Delays with Weather and Airport Detail
Dataset shape	30 columns, 6489062 rows

The details of the features of the dataset are as follows.

Column Name	Description
MONTH	Month of the flight
DEP_TIME_BLK	Departure time Block
DAY_OF_WEEK	Day of the week
DEP_DEL15	Binary indicator of departure delay over 15 minutes (1 is yes)
DISTANCE_GROUP	Distance group to be flown by departing aircraft
DEP_BLOCK	Departure block
SEGMENT_NUMBER	The segment that this tail number is on for the day
CONCURRENT_FLIGHTS	Concurrent flights leaving from the airport in the same departure block
NUMBER_OF_SEATS	Number of seats on the aircraft
CARRIER_NAME	Carrier name
AIRPORT_FLIGHTS_MONTH	Average airport flights per month
AIRLINE_FLIGHTS_MONTH	Average airline flights per month
AIRLINE_AIRPORT_FLIGHTS_MONTH	Average flights per month for both airline and airport
AVG_MONTHLY_PASS_AIRPORT	Average passengers for the departing airport for the month
AVG_MONTHLY_PASS_AIRLINE	Average passengers for the airline for the month
FLT_ATTENDANTS_PER_PASS	Flight attendants per passenger for the airline
GROUND_SERV_PER_PASS	Ground service employees (service desk) per passenger for the airline
PLANE_AGE	Age of departing aircraft
DEPARTING_AIRPORT	Departing airport
LATITUDE	Latitude of departing airport
LONGITUDE	Longitude of departing airport
PREVIOUS_AIRPORT	Previous airport that the aircraft departed from
PRCP	Inches of precipitation for the day
SNOW	Inches of snowfall for the day
SNWD	Inches of snow on the ground for the day
TMAX	Max temperature for the day
AWND	Max wind speed for the day
CARRIER_HISTORICAL	Carrier history
DEP_BLOCK_HIST	Departure Block History
DAY_HISTORICAL	Day History
DEP_AIRPORT_HIST	Departure airport History

2. Problem Statement

The project objective is to apply scalable data analytics tools and techniques to gain useful insights to analyze and identify the factors that cause airline delays for the year. The project source is Kaggle with complete data for the year 2019, USA. The project activities include ingesting data into Hadoop and using Hive to run queries and extract meaningful information.

Another objective of this project is to develop dashboards using Apache Superset to analyze the data and identify the most important attributes that contribute to the factors that cause delays like the weather conditions. We also aim to find relationships between these factors.

We also aim to train machine learning algorithms on this data set using PySpark and find an algorithm that works best for this classification problem.

3. Big Data Pipeline

We used the following tools for the big data pipeline to complete our data analytics.

4. Containers

- Starting the Hadoop cluster (Hadoop, hive and spark)

  Downloaded the cluster from this GitHub repository: https://github.com/jopereira/docker-fullstack
  Ran the container using the docker-compose up command, and the running containers can be seen below.

```
PS D:\MSDS\Big Data Analytics\Project\docker-fullstack-main\docker-fullstack-main> docker ps
CONTAINER ID   IMAGE                                            COMMAND                  CREATED         STATUS
          PORTS                                             NAMES
6cf9856e514a   bde2020/spark-worker:2.4.5-hadoop2.7             "/bin/bash /worker.sh"   5 minutes ago   Up 4 minutes
          0.0.0.0:8081->8081/tcp                            spark-worker-1
a44e25699d52   bde2020/hive:2.3.2-postgresql-metastore          "entrypoint.sh /bin/…"   5 minutes ago   Up 4 minutes
          0.0.0.0:10000->10000/tcp, 10002/tcp               hive-server
849495d7ba18   bde2020/hive:2.3.2-postgresql-metastore          "entrypoint.sh /opt/…"   5 minutes ago   Up 4 minutes
          10000/tcp, 0.0.0.0:9083->9083/tcp, 10002/tcp      hive-metastore
21a0925ba472   bde2020/hive-metastore-postgresql:2.3.0          "/docker-entrypoint.…"   5 minutes ago   Up 4 minutes
          5432/tcp                                          docker-fullstack-main-hive-metastore-postgresql-1
77efe8bdd27f   bde2020/spark-master:2.4.5-hadoop2.7             "/bin/bash /master.sh"   5 minutes ago   Up 4 minutes
          0.0.0.0:7077->7077/tcp, 6066/tcp, 0.0.0.0:8080->8080/tcp   spark-master
b9b82f1a9769   bde2020/hadoop-datanode:2.0.0-hadoop2.7.4-java8  "/entrypoint.sh /run…"   5 minutes ago   Up 4 minutes (
healthy)  0.0.0.0:50075->50075/tcp                          datanode
bfd73e1920fb   bde2020/hadoop-namenode:2.0.0-hadoop2.7.4-java8  "/entrypoint.sh /run…"   5 minutes ago   Up 4 minutes (
healthy)  0.0.0.0:50070->50070/tcp                          namenode
e31df5b5b85d   bde2020/hbase-master:1.0.0-hbase1.2.6            "/entrypoint.sh /run…"   5 minutes ago   Up 4 minutes
          16000/tcp, 0.0.0.0:16010->16010/tcp               hbase-master
2634eff2e1cc   zookeeper:3.4.10                                 "/docker-entrypoint.…"   5 minutes ago   Up 4 minutes
          2888/tcp, 0.0.0.0:2181->2181/tcp, 3888/tcp        zoo
446de8344084   bde2020/hbase-regionserver:1.0.0-hbase1.2.6      "/entrypoint.sh /run…"   5 minutes ago   Up 4 minutes
          16020/tcp, 0.0.0.0:16030->16030/tcp               hbase-regionserver
PS D:\MSDS\Big Data Analytics\Project\docker-fullstack-main\docker-fullstack-main>
```

Pulled the Jupyter Notebook image as well and and ran the container for Machine learning for PySpark:

```
PS D:\MSDS\Big Data Analytics\Project\docker-fullstack-main\docker-fullstack-main> docker pull jupyter/pyspark-notebook
Using default tag: latest
latest: Pulling from jupyter/pyspark-notebook
dbf6a9befcde: Downloading [>                                                  ]  308.1kB/29.53MB
febda94ed4af: Pulling fs layer
c883f1cfa2af: Download complete
4f4fb700ef54: Waiting
1dac1028dfdc: Waiting
f07f69497478: Waiting
d220a5b039d0: Waiting
32fa0bb4deab: Waiting
e5440bf53163: Waiting
dab799a54eb1: Waiting
6362e77514fc: Waiting
65fe0dfd1cdc: Waiting
```

5. Data preprocessing

The data was available in train and test and we had to concatenate the data using python so that the file can ingested into Hadoop. We also had to transform one column so that the data can be read by hive.

- Reading the data and concatenating it.

```python
import numpy as np
import pandas as pd
```
[1] ✓ 5.7s

```python
df_train = pd.read_csv("train.csv")
df_test = pd.read_csv("test.csv")
```
[2] ✓ 55.0s

```python
df = pd.concat([df_train, df_test])
```
[3] ✓ 2.3s

```python
print(df_train.shape)
print(df.shape)
```
[4] ✓ 0.1s

```
(4542343     30)
(6489062     30)
```

- This is the concatenated data. As we can see the column DEP_TIME_BLK shows a range of the time which can't be read in hive. so, we transformed this column by taking mean of the range.

[10] ✓ 6.8s Python

	MONTH	DAY_OF_WEEK	DEP_DEL15	DEP_TIME_BLK	DISTANCE_GROUP	SEGMENT_NUMBER	CONCURRENT_FLIGHTS	NUMBER_OF_SEATS	CARRIER_NAME	AIRPORT_FLIGHTS_MONTH	...	PRCP	SNOW	S
0	7	7	0	1500-1559	3	3	26	160	American Airlines Inc.	19534	...	0.00	0.0	
1	4	1	0	1300-1359	4	4	63	50	SkyWest Airlines Inc.	18788	...	0.00	0.0	
2	11	4	0	0001-0559	2	1	3	76	American Eagle Airlines Inc.	1148	...	0.00	0.0	
3	3	2	0	1500-1559	7	5	14	143	Southwest Airlines Co.	7612	...	0.00	0.0	
4	7	3	0	0800-0859	1	2	85	50	American Eagle Airlines Inc.	29376	...	0.01	0.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1946714	5	1	0	0800-0859	3	1	48	160	Delta Air Lines Inc.	20794	...	0.03	0.0	
1946715	4	4	0	0800-0859	3	2	28	76	Endeavor Air Inc.	12669	...	0.00	0.0	

- We also encountered that there were these characters (‘,’’) in the CARRIER_NAME column. So, we transformed it as well. We also dropped the original DEP_TIME_BLK after transforming it.

```python
# Function to calculate the mean from a range string
def calculate_mean_from_range(range_string):
    start_value, end_value = map(int, range_string.split('-'))
    mean = (start_value + end_value) / 2
    return mean
```
[5] ✓ 0.1s

```python
df['MEAN_DEP_TIME_BLK'] = df['DEP_TIME_BLK'].apply(calculate_mean_from_range)
```
[6] ✓ 10.9s

```python
df1 = df.drop(columns=['DEP_TIME_BLK'])
```
[7] ✓ 2.0s

```python
df1['CARRIER_NAME'] = df1['CARRIER_NAME'].str.replace(',', '')
```
[8] ✓ 4.4s

```python
df1.to_csv('airline_data.csv', index=False)
```
[9] ✓ 3m 50.7s

- The final data can be seen as follows after the initial preprocessing, which was then loaded to HDFS.

```python
df1
```
[11] ✓ 4.2s Python

ARRIER_NAME	AIRPORT_FLIGHTS_MONTH	AIRLINE_FLIGHTS_MONTH	...	PRCP	SNOW	SNWD	TMAX	AWND	CARRIER_HISTORICAL	DEP_AIRPORT_HIST	DAY_HISTORICAL	DEP_BLOCK_HIST	MEAN_DEP_TIME_BLK
American Airlines Inc.	19534	79247	...	0.00	0.0	0.0	95.0	4.25	0.237709	0.273236	0.222538	0.255479	1529.5
SkyWest Airlines Inc.	18788	67082	...	0.00	0.0	0.0	71.0	11.41	0.154651	0.121849	0.237972	0.197503	1329.5
American Eagle Airlines Inc.	1148	25517	...	0.00	0.0	0.0	54.0	0.45	0.117559	0.187867	0.139886	0.060327	280.0
Southwest Airlines Co.	7612	114119	...	0.00	0.0	0.0	64.0	8.05	0.204389	0.141446	0.132868	0.202037	1529.5
American Eagle Airlines Inc.	29376	28267	...	0.01	0.0	0.0	94.0	10.51	0.203263	0.193761	0.203027	0.113050	829.5
...	...	...	...	...	...	...	...	...	...	...	...	...	...
Delta Air Lines Inc.	20794	85579	...	0.03	0.0	0.0	70.0	9.40	0.141341	0.187883	0.193668	0.111033	829.5
ndeavor Air Inc.	12669	20645	...	0.00	0.0	0.0	63.0	10.96	0.188378	0.149965	0.171317	0.106597	829.5
Atlantic Southeast Airlines	15165	10970	...	0.20	0.0	0.0	84.0	8.50	0.243554	0.187883	0.199784	0.210959	1229.5
JetBlue Airways	8560	24966	...	0.00	0.0	0.0	85.0	14.09	0.267584	0.187883	0.177124	0.075131	629.5
American Airlines Inc.	9612	79228	...	0.82	0.0	0.0	38.0	9.40	0.178345	0.187883	0.199784	0.081939	629.5

6. Data Storage in Hdfs

HDFS is used to store data. Data is then accessed by Hive and Spark containers.

Airline data csv file is copied into the hadoop container in namenode.

docker cp airline_data.csv namenode:/tmp/

```
ocker-fullstack-main\docker-fullstack-main>
ocker-fullstack-main\docker-fullstack-main> docker cp airline_data.csv namenode:/tmp/
```

Enter the hadoop bash environment
docker exec -it namenode /bin/bash

```
ocker-fullstack-main\docker-fullstack-main>
ocker-fullstack-main\docker-fullstack-main> docker cp airline_data.csv namenode:/tmp/
ocker-fullstack-main\docker-fullstack-main> docker exec -it namenode /bin/bash
```

Create an input directory in hdfs to store the file
hdfs dfs -mkdir -p /user/root/input

```
root@5604cb1e9a22:/# hdfs dfs -mkdir -p /user/root/input
```

Copy the csv file from hadoop into hdfs directory
hdfs dfs -copyFromLocal /tmp/airline_data.csv /user/root/input

```
root@b962d25b071f:/# hdfs dfs -copyFromLocal /tmp/airline_data.csv /user/root/input
```

Check if file is copied successfully in hdfs
hdfs dfs -ls /user/root/input

```
root@e38d45101308:/# hdfs dfs -ls /user/root/input
Found 1 items
-rwxr-xr-x   3 root supergroup 1801252083 2023-06-04 14:14 /user/root/input/airline_data.csv
```

7. Data Analysis in Hive

Hive is used to perform exploratory analysis using queries and get a deeper insight into data. Database and table are created in hive where data is loaded from HDFS. From the tables, queries are performed to get a better insight into the data.

Enter the hive bash environment
docker-compose exec hive-server bash
/opt/hive/bin/beeline -u jdbc:hive2://localhost:10000

```
PS D:\MSDS\Big Data Analytics\Project\docker-fullstack-main\docker-fullstack-main>  docker-compose exec hive-server bash
root@925b892e4378:/opt#  /opt/hive/bin/beeline -u jdbc:hive2://localhost:10000
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/opt/hive/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/opt/hadoop-2.7.4/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Connecting to jdbc:hive2://localhost:10000
Connected to: Apache Hive (version 2.3.2)
Driver: Hive JDBC (version 2.3.2)
Transaction isolation: TRANSACTION_REPEATABLE_READ
Beeline version 2.3.2 by Apache Hive
0: jdbc:hive2://localhost:10000> show dbs
```

Create Database
create database if not exists airline.

Create Table

```
CREATE EXTERNAL TABLE AIRLINE_DELAY (
    MONTH INT,
    DAY_OF_WEEK INT,
    DEP_DEL15 INT,
    DISTANCE_GROUP INT,
    SEGMENT_NUMBER INT,
    CONCURRENT_FLIGHTS INT,
    NUMBER_OF_SEATS INT,
    CARRIER_NAME STRING,
    AIRPORT_FLIGHTS_MONTH INT,
    AIRLINE_FLIGHTS_MONTH INT,
    AIRLINE_AIRPORT_FLIGHTS_MONTH INT,
    AVG_MONTHLY_PASS_AIRPORT INT,
    AVG_MONTHLY_PASS_AIRLINE INT,
    FLT_ATTENDANTS_PER_PASS DOUBLE,
    GROUND_SERV_PER_PASS DOUBLE,
    PLANE_AGE INT,
    DEPARTING_AIRPORT STRING,
    LATITUDE DOUBLE,
    LONGITUDE DOUBLE,
    PREVIOUS_AIRPORT STRING,
    PRCP DOUBLE,
    SNOW INT,
    SNWD INT,
    TMAX INT,
    AWND DOUBLE,
    CARRIER_HISTORICAL DOUBLE,
    DEP_AIRPORT_HIST DOUBLE,
    DAY_HISTORICAL DOUBLE,
    DEP_BLOCK_HIST DOUBLE,
    MEAN_DEP_TIME_BLK DOUBLE)
PARTITIONED BY (DEP_DEL15 INT)
 ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE
LOCATION '/user/root/input';
```

Load the data from HDFS into Hive Table

```
LOAD DATA INPATH '/user/root/input/airline_data.csv' INTO TABLE AIRLINE_DELAY
PARTITION (DEP_DEL15=0);
```

```
0: jdbc:hive2://localhost:10000> CREATE EXTERNAL TABLE AIRLINE_DELAY(MONTH INT,
. . . . . . . . . . . . . . . .> DAY_OF_WEEK INT,
. . . . . . . . . . . . . . . .> MEAN_DEP_TIME_BLK DOUBLE,
. . . . . . . . . . . . . . . .> DISTANCE_GROUP INT,
. . . . . . . . . . . . . . . .> SEGMENT_NUMBER INT,
. . . . . . . . . . . . . . . .> CONCURRENT_FLIGHTS INT,
. . . . . . . . . . . . . . . .> NUMBER_OF_SEATS INT,
. . . . . . . . . . . . . . . .> CARRIER_NAME STRING,
. . . . . . . . . . . . . . . .> AIRPORT_FLIGHTS_MONTH INT,
. . . . . . . . . . . . . . . .> AIRLINE_FLIGHTS_MONTH INT,
. . . . . . . . . . . . . . . .> AIRLINE_AIRPORT_FLIGHTS_MONTH INT,
. . . . . . . . . . . . . . . .> AVG_MONTHLY_PASS_AIRPORT INT,
. . . . . . . . . . . . . . . .> AVG_MONTHLY_PASS_AIRLINE INT,
. . . . . . . . . . . . . . . .> FLT_ATTENDANTS_PER_PASS DOUBLE,
. . . . . . . . . . . . . . . .> GROUND_SERV_PER_PASS DOUBLE,
. . . . . . . . . . . . . . . .> PLANE_AGE INT,
. . . . . . . . . . . . . . . .> DEPARTING_AIRPORT STRING,
. . . . . . . . . . . . . . . .> LATITUDE DOUBLE,
. . . . . . . . . . . . . . . .> LONGITUDE DOUBLE,
. . . . . . . . . . . . . . . .> PREVIOUS_AIRPORT STRING,
. . . . . . . . . . . . . . . .> PRCP DOUBLE,
. . . . . . . . . . . . . . . .> SNOW INT,
. . . . . . . . . . . . . . . .> SNWD INT,
. . . . . . . . . . . . . . . .> TMAX INT,
. . . . . . . . . . . . . . . .> AWND DOUBLE,
. . . . . . . . . . . . . . . .> CARRIER_HISTORICAL DOUBLE,
. . . . . . . . . . . . . . . .> DEP_AIRPORT_HIST DOUBLE,
. . . . . . . . . . . . . . . .> DAY_HISTORICAL DOUBLE,
. . . . . . . . . . . . . . . .> DEP_BLOCK_HIST DOUBLE)
. . . . . . . . . . . . . . . .> PARTITIONED BY (DEP_DEL15 INT)
. . . . . . . . . . . . . . . .> ROW FORMAT DELIMITED
. . . . . . . . . . . . . . . .> FIELDS TERMINATED BY ','
. . . . . . . . . . . . . . . .> STORED AS TEXTFILE
. . . . . . . . . . . . . . . .> LOCATION '/user/root/input';
No rows affected (0.276 seconds)
0: jdbc:hive2://localhost:10000> LOAD DATA INPATH '/user/root/input/airline_data.csv' INTO TABLE AIRLINE_DELAY
. . . . . . . . . . . . . . . .> PARTITION (DEP_DEL15=0);
No rows affected (1.594 seconds)
```

We ran the following queries separately for this created table:

```
0: jdbc:hive2://localhost:10000> SELECT DEPARTING_AIRPORT, COUNT(SEGMENT_NUMBER) AS flight_count
. . . . . . . . . . . . . . . .> FROM airline_delay
. . . . . . . . . . . . . . . .> GROUP BY DEPARTING_AIRPORT
. . . . . . . . . . . . . . . .> LIMIT 20;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (
ve 1.X releases.
+-----------------------------------+---------------+
|         departing_airport         | flight_count  |
+-----------------------------------+---------------+
| 6                                 | 29114         |
| Adams Field                       | 11893         |
| 12                                | 44219         |
| 5                                 | 33567         |
| Albuquerque International Sunport | 23086         |
| Atlanta Municipal                 | 386718        |
| 11                                | 50273         |
| Anchorage International           | 18828         |
| 13                                | 16370         |
| 2                                 | 19895         |
| 10                                | 3218          |
| Birmingham Airport                | 18697         |
| 15                                | 29704         |
| 4                                 | 28101         |
| Austin - Bergstrom International  | 65253         |
| Boise Air Terminal                | 18934         |
| 14                                | 39120         |
| 3                                 | 6573          |
| Albany International              | 5461          |
| Bradley International             | 27409         |
+-----------------------------------+---------------+
20 rows selected (26.332 seconds)
0: jdbc:hive2://localhost:10000>
```

```
0: jdbc:hive2://localhost:10000> SELECT CARRIER_NAME, COUNT(*) AS flight_count
. . . . . . . . . . . . . . . . .> FROM airline_delay
. . . . . . . . . . . . . . . . .> GROUP BY CARRIER_NAME
. . . . . . . . . . . . . . . . .> ORDER BY flight_count DESC
. . . . . . . . . . . . . . . . .> LIMIT 20;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different
ve 1.X releases.
+-------------------------------+---------------+
|         carrier_name          | flight_count  |
+-------------------------------+---------------+
| Southwest Airlines Co.        | 1296329       |
| Delta Air Lines Inc.          | 938346        |
| American Airlines Inc.        | 903640        |
| United Air Lines Inc.         | 601044        |
| SkyWest Airlines Inc.         | 584204        |
| "Midwest Airline              | 300154        |
| JetBlue Airways               | 269596        |
| Alaska Airlines Inc.          | 239337        |
| American Eagle Airlines Inc.  | 228792        |
| Comair Inc.                   | 219324        |
| Endeavor Air Inc.             | 203827        |
| Spirit Air Lines              | 189419        |
| Mesa Airlines Inc.            | 177600        |
| Frontier Airlines Inc.        | 120872        |
| Atlantic Southeast Airlines   | 99044         |
| Hawaiian Airlines Inc.        | 74898         |
| Allegiant Air                 | 42636         |
| CARRIER_NAME                  | 1             |
+-------------------------------+---------------+
18 rows selected (13.894 seconds)
0: jdbc:hive2://localhost:10000>
```

```
0: jdbc:hive2://localhost:10000> SELECT MONTH, AVG(PRCP) AS avg_precipitation, MAX(TMAX) AS max_tempera
. . . . . . . . . . . . . . . . .> FROM airline_delay
. . . . . . . . . . . . . . . . .> GROUP BY MONTH
. . . . . . . . . . . . . . . . .> LIMIT 20;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider u
ve 1.X releases.
+--------+----------------------+------------------+
| month  |  avg_precipitation   | max_temperature  |
+--------+----------------------+------------------+
| 6      | 0.1336220638373257   | 112              |
| 1      | 0.09946374062923609  | 85               |
| 9      | 0.07701217807788006  | 112              |
| 5      | 0.1297695105410213   | 102              |
| NULL   | NULL                 | NULL             |
| 8      | 0.10399327391537645  | 115              |
| 2      | 0.11073152717941934  | 88               |
| 10     | 0.11848030036311749  | 101              |
| 7      | 0.09408176479419313  | 115              |
| 4      | 0.11055895629004013  | 104              |
| 12     | 0.11568679582826517  | 87               |
| 3      | 0.07149453649553734  | 91               |
| 11     | 0.0653569293481005   | 93               |
+--------+----------------------+------------------+
13 rows selected (18.514 seconds)
0: jdbc:hive2://localhost:10000>
```

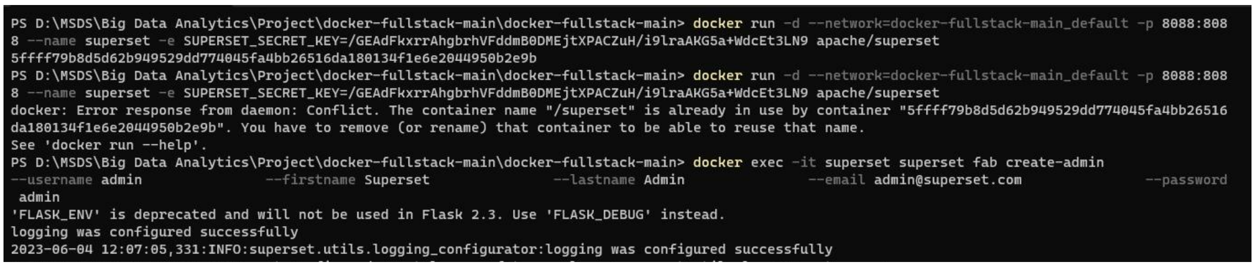
8. Data Visualization using Superset

Apache Superset is a data exploration and visualization platform. It provides SQL query tab where we can perform query on the dataset and show the results in various charts. It provides a way to make dashboards which contain various charts that help us to get deeper insights into data.

- Generate a secret key
  openssl rand -base64 32

- Create a docker container
  docker run -d --network=docker_fullstack-main_default -p 8088:8088 --name superset –e SUPERSET_SECRET_KEY=/GEAdFkxrrAhgbrhVFddmB0DMEjtXPACZuH/i9lraAKG5a+WdcEt3LN9 apache/superset

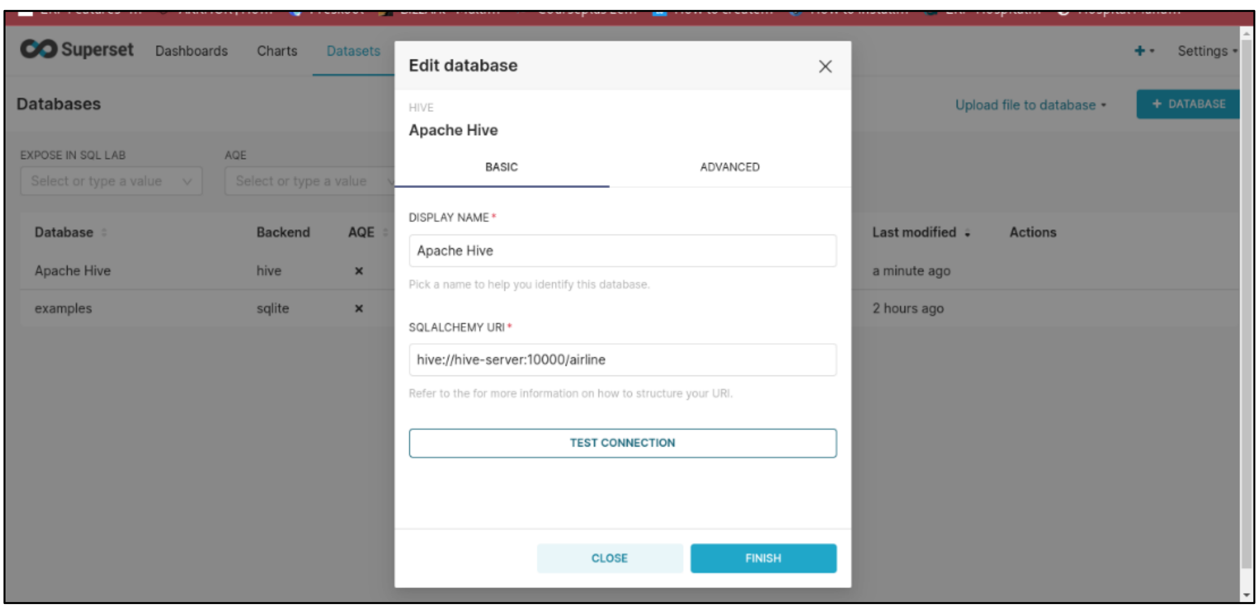
- Configure docker container and add credentials to superset
  docker exec -it superset superset fab create-admin --username admin --firstname Superset --lastname Admin --email admin@superset.com --password admin



Upgrade database and initialize superset in container

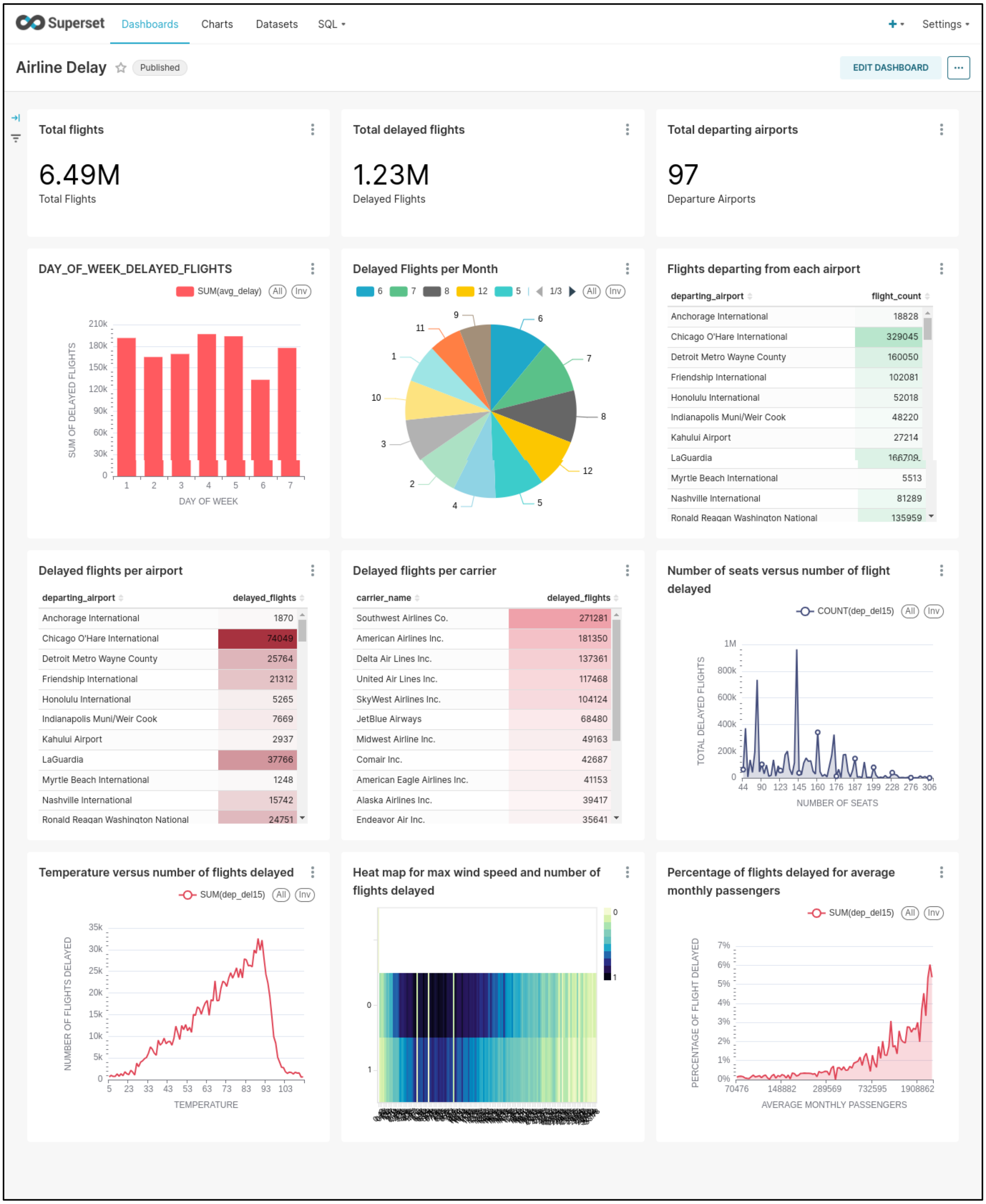
docker exec -it superset superset db upgrade
docker exec -it superset superset init

- **Connect Hive with Superset**
  Go to database page and add a new database select source as Apache hive and give the connection details like container name, port and database name.



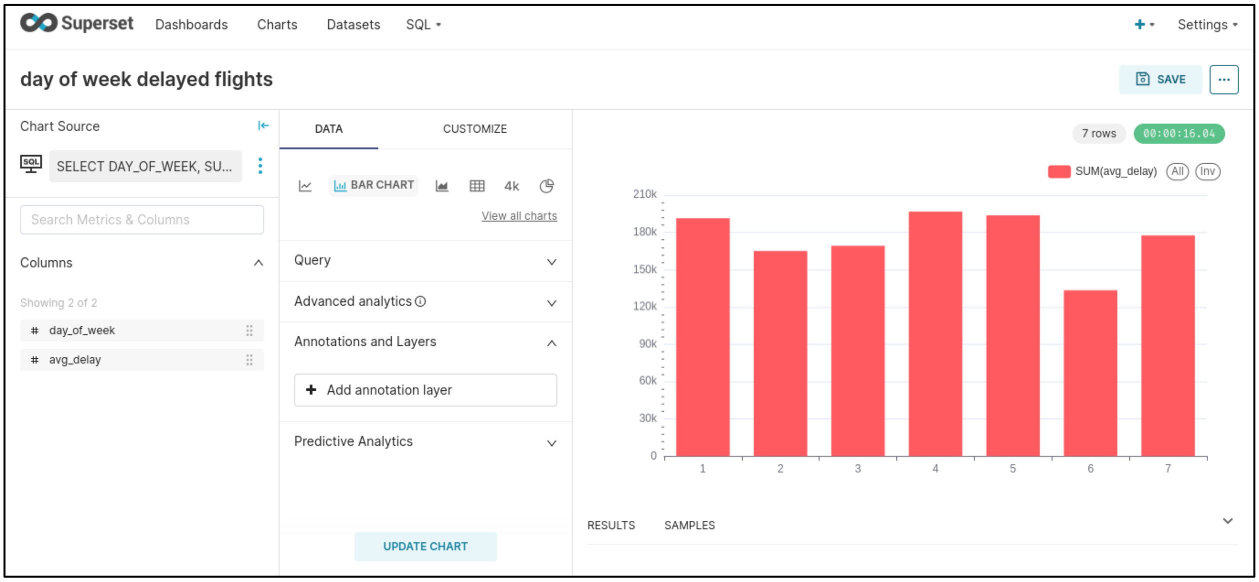
Hive-SQL Query is done SQL Lab in Superset.

Created a dashboard that help to get insights into data.

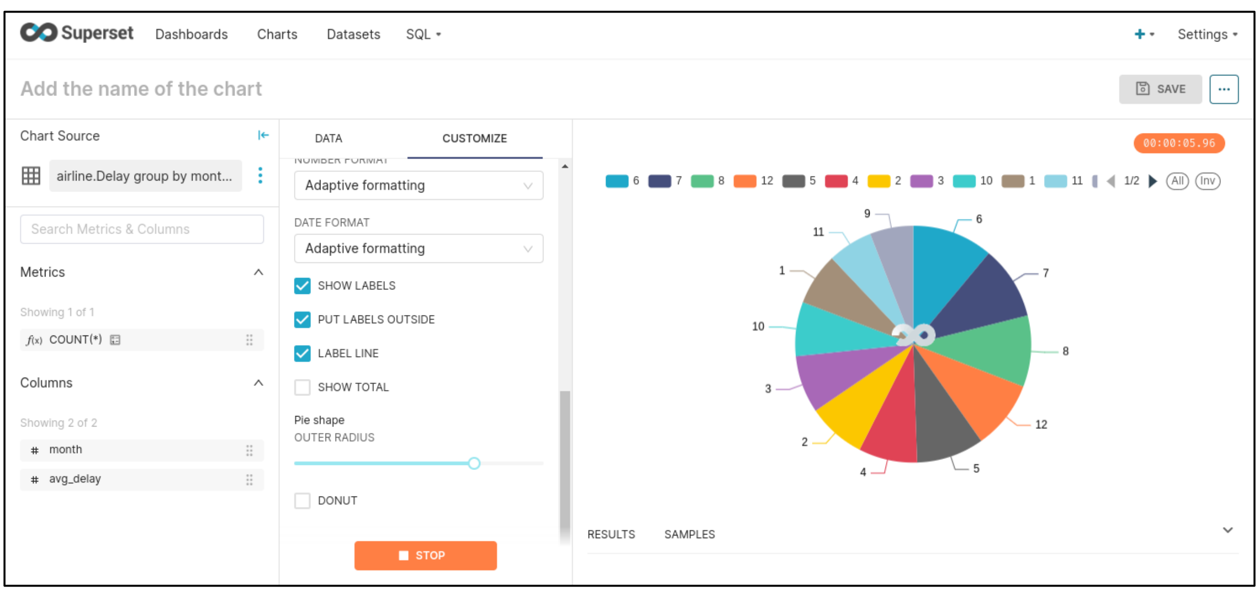


Queries performed and their equivalent charts.

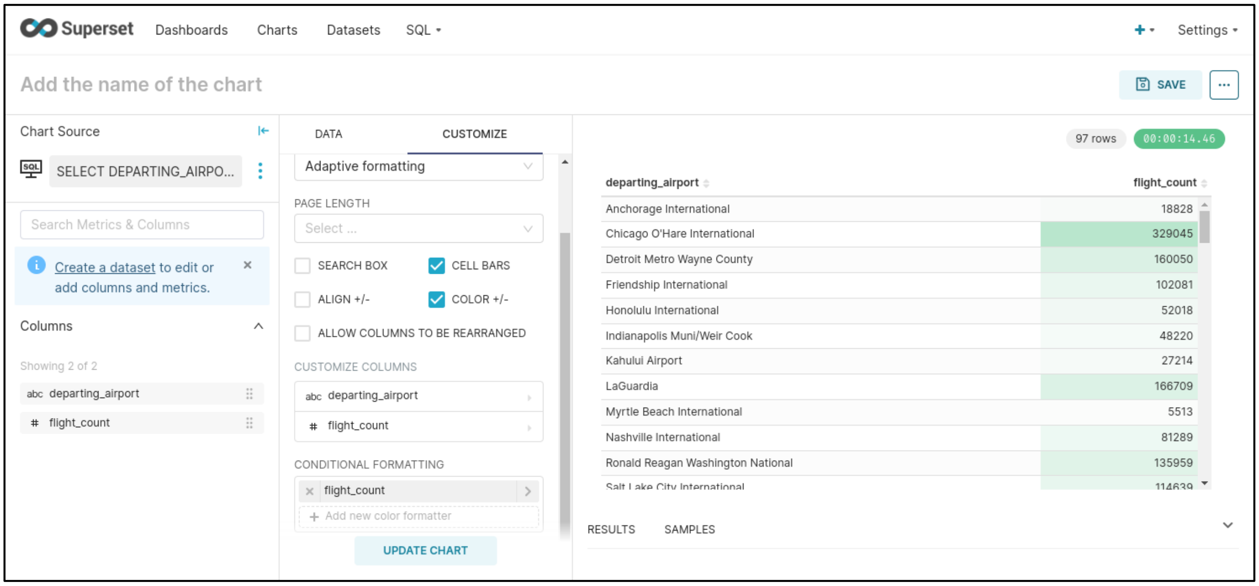
- How many flights are delayed on each day of the week. This gives insight that on which day of the week flights are normally delayed
  SELECT DAY_OF_WEEK, SUM(DEP_DEL15) AS avg_delay
  FROM airline_delay
  GROUP BY DAY_OF_WEEK;



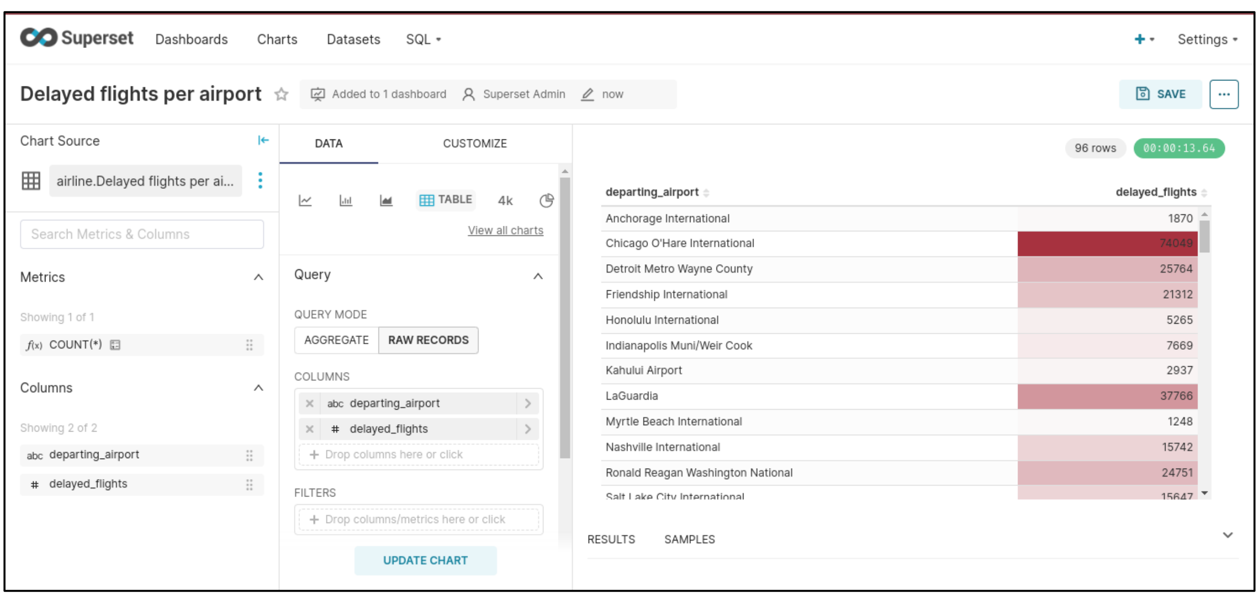
- How many flights are delayed each month – this gives insights into which month, flights are usually delayed.
  SELECT month, SUM(DEP_DEL15) AS avg_delay
  FROM airline_delay
  GROUP BY month;



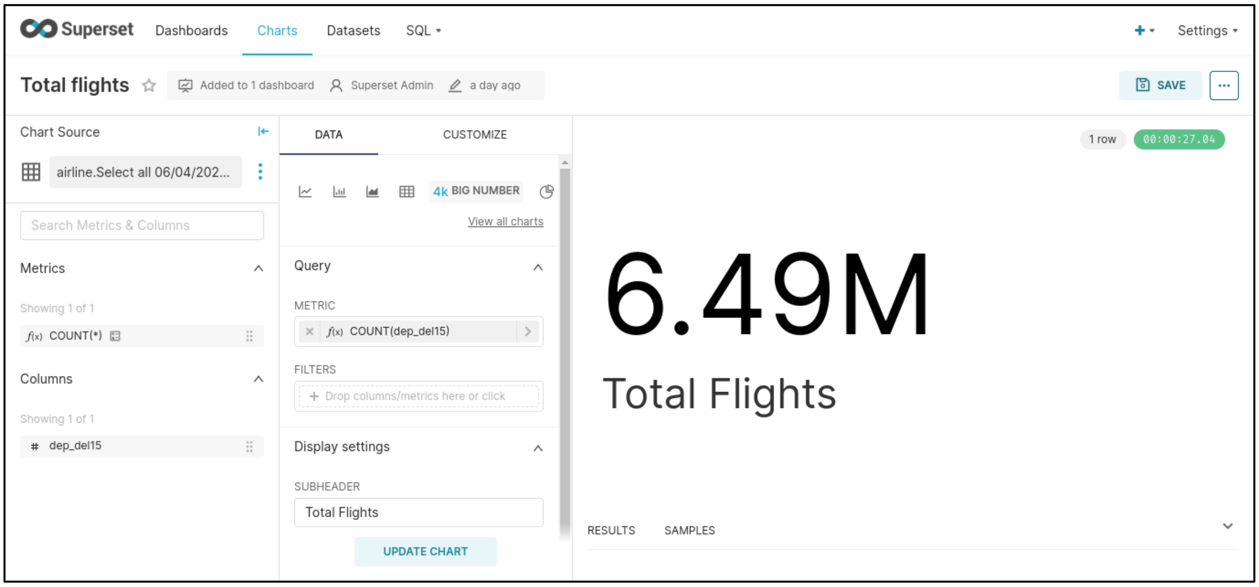
- This chart give insights into how many flights per airport had.
  SELECT departing_airport, COUNT(segment_number) AS flight_count
  FROM airline_delay
  GROUP BY departing_airport;



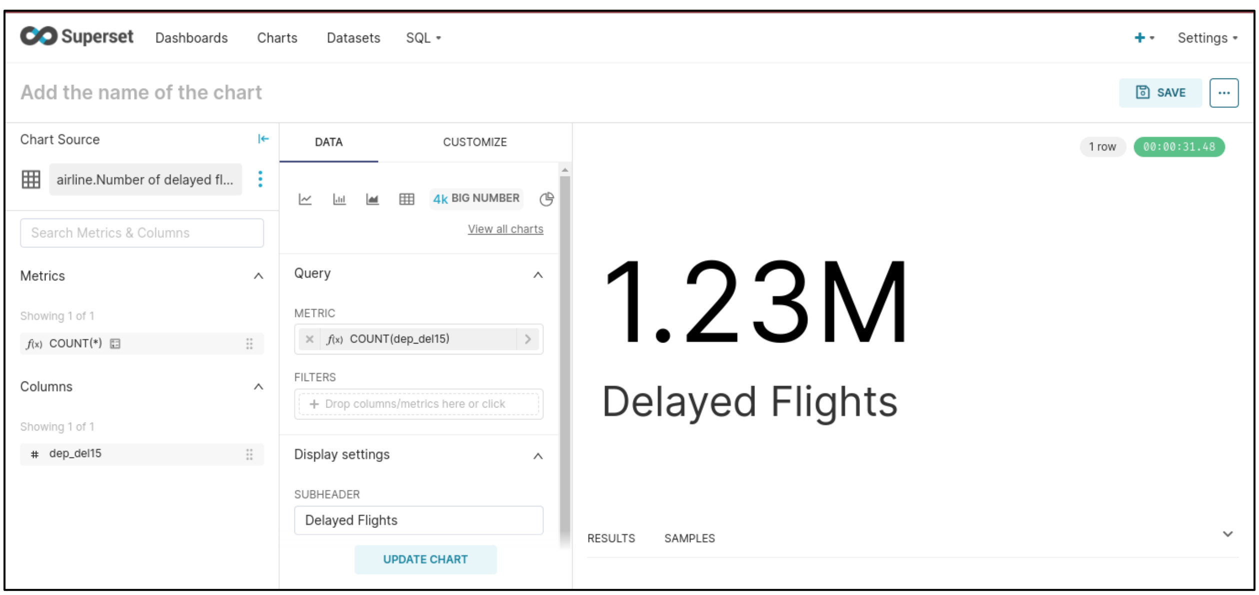
- This chart gives insights into how many delayed flights per airport had.
  SELECT departing_airport, COUNT(dep_del15) AS delayed_flights
  FROM airline_delay
  GROUP BY departing_airport;



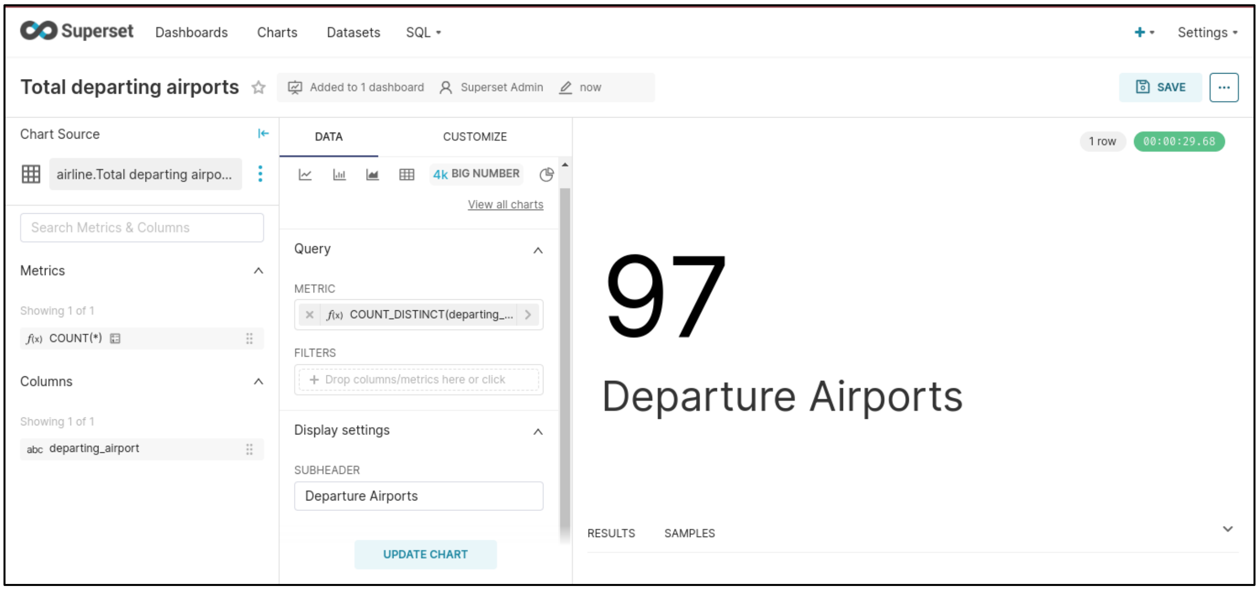
- This chart shows a KPI for the total number of flights.
  SELECT COUNT(*)
  FROM airline_delay;



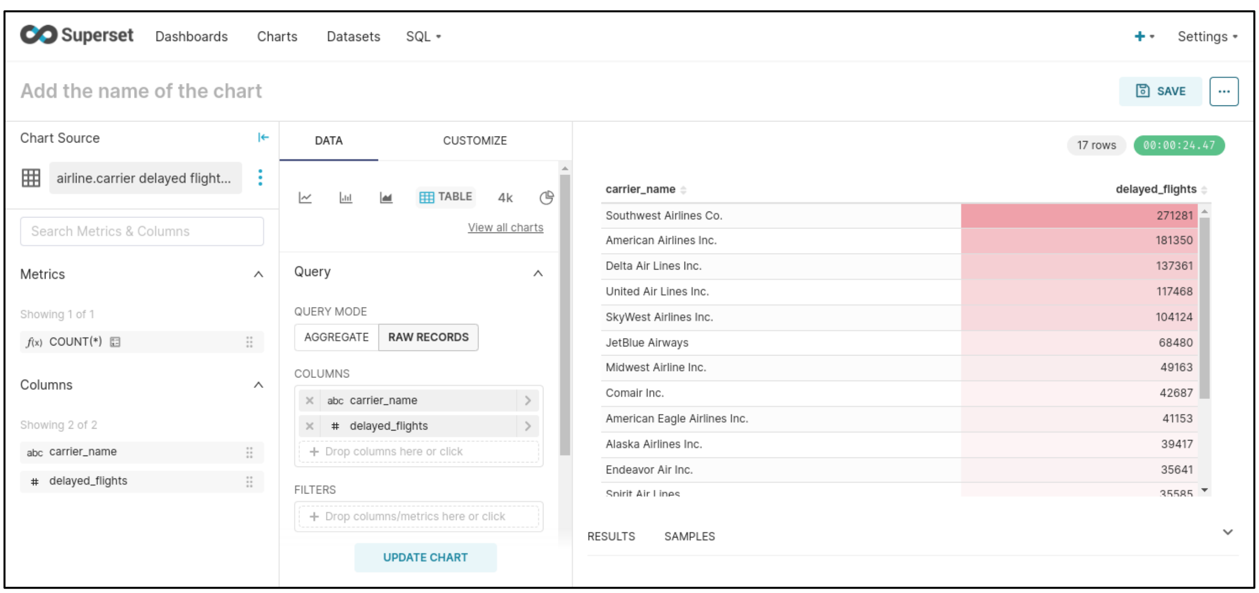
- This chart shows a KPI for the number of delayed flights.
  SELECT COUNT(*)
  FROM airline_delay
  WHERE dep_del15 = 1;



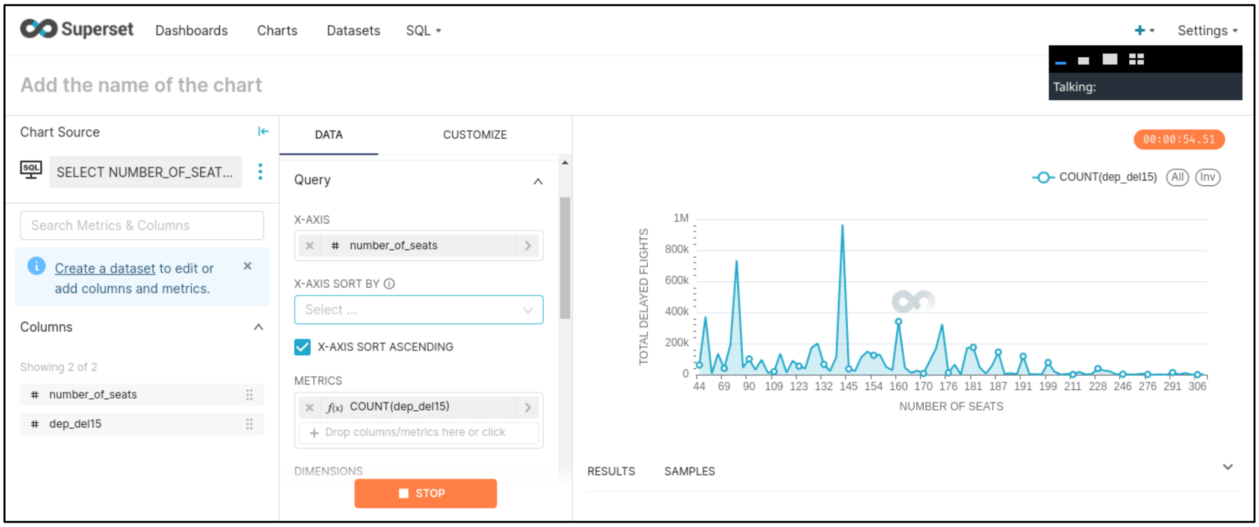
- This chart shows a KPI for the number of departing airports.
  SELECT DISTICNT COUNT(*)
  FROM airline_delay
  GROUP BY departing_airports;



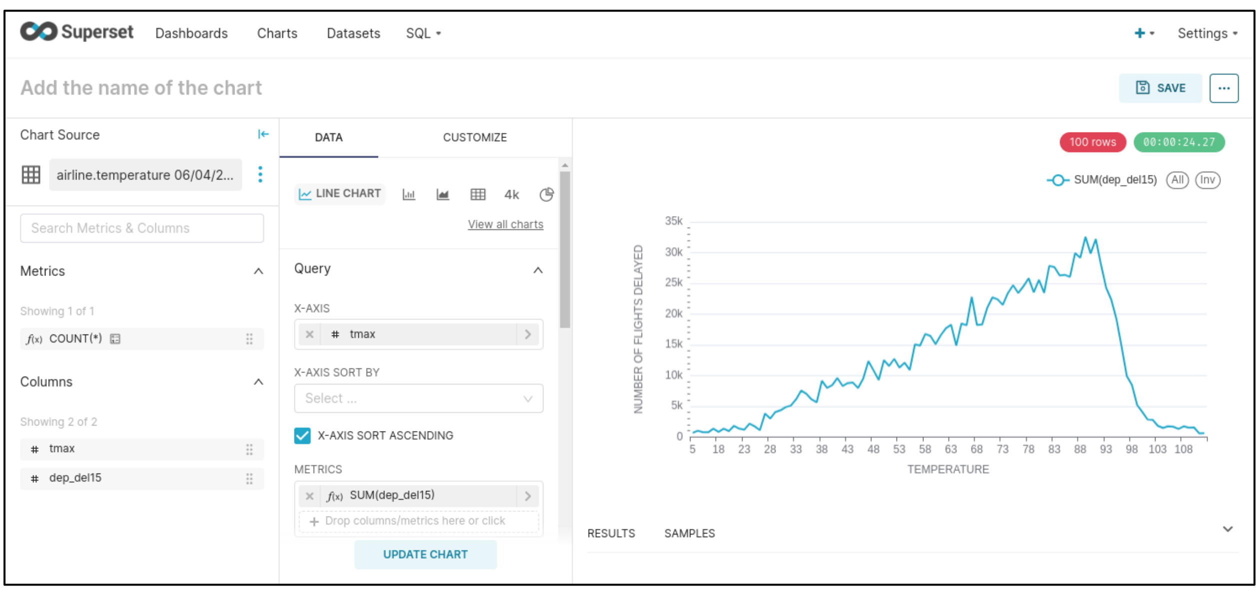
- This chart gives insights into how many delayed flights per carrier had.
  SELECT carrier_name, COUNT(*) AS delayed_flights
  FROM airline_delay
  WHERE dep_del15 = 1
  GROUP BY carrier_name
  ORDER BY delayed_flights DESC;



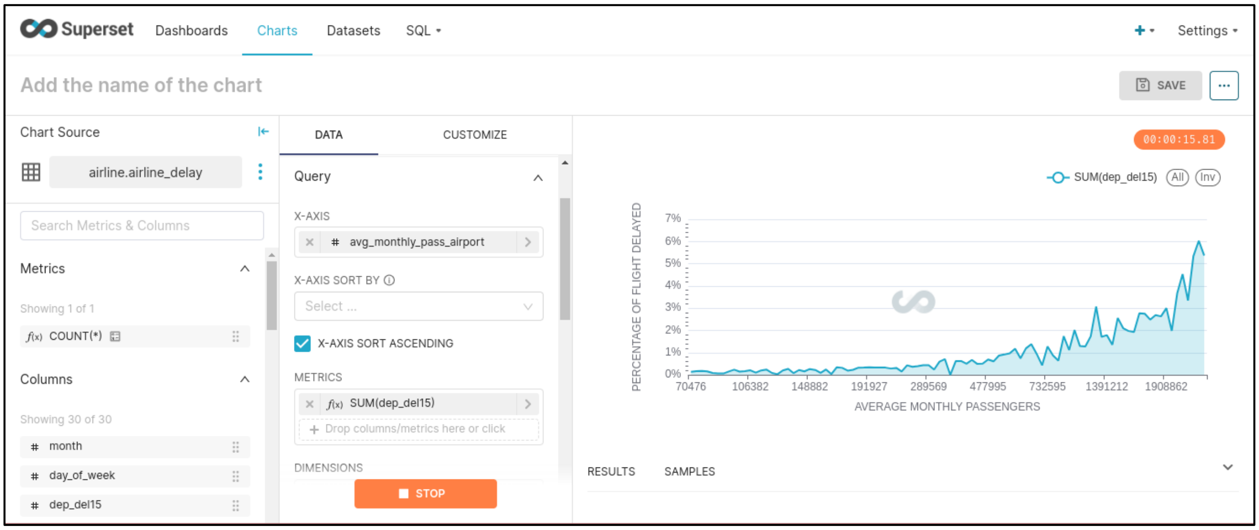
- This chart number of seats versus the total number of delayed flights.
  SELECT number_of_seats, dep_del15
  FROM airline_delay;



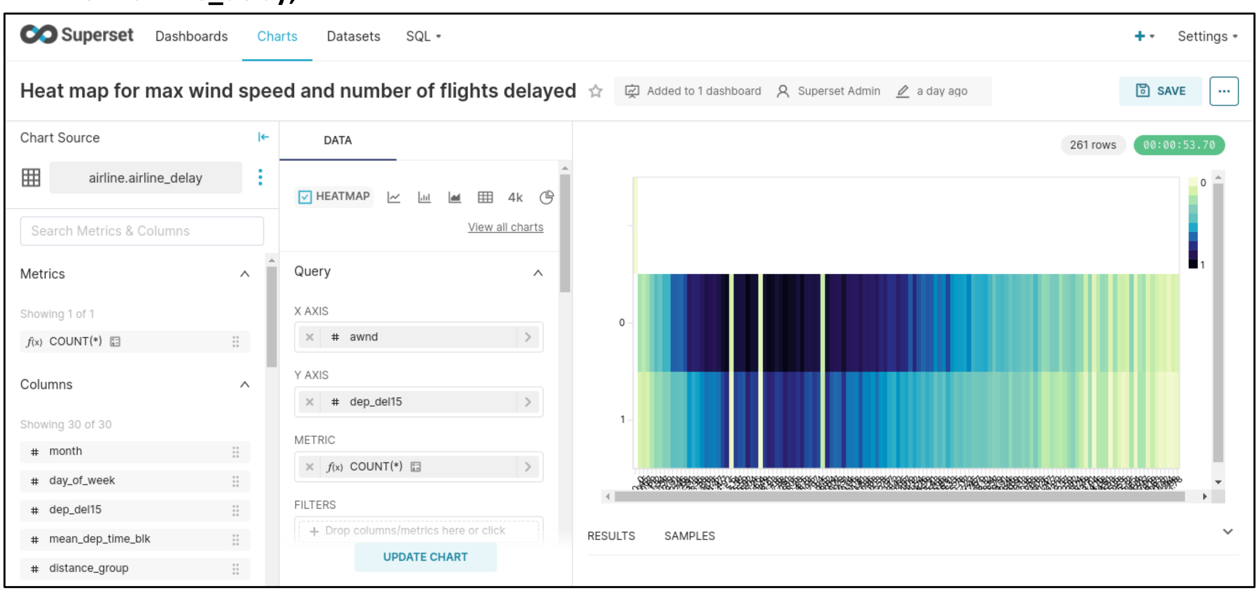
- This chart shows the effect of temperature on number of delayed flights
  SELECT tmax, dep_del15
  FROM airline_delay;



- This chart shows the effect of average monthly passengers on number of delayed flights
  SELECT avg_monthly_pass_airport, dep_del15
  FROM airline_delay;



- This chart shows the effect of wind on number of delayed flights
  SELECT awnd, dep_del15
  FROM airline_delay;

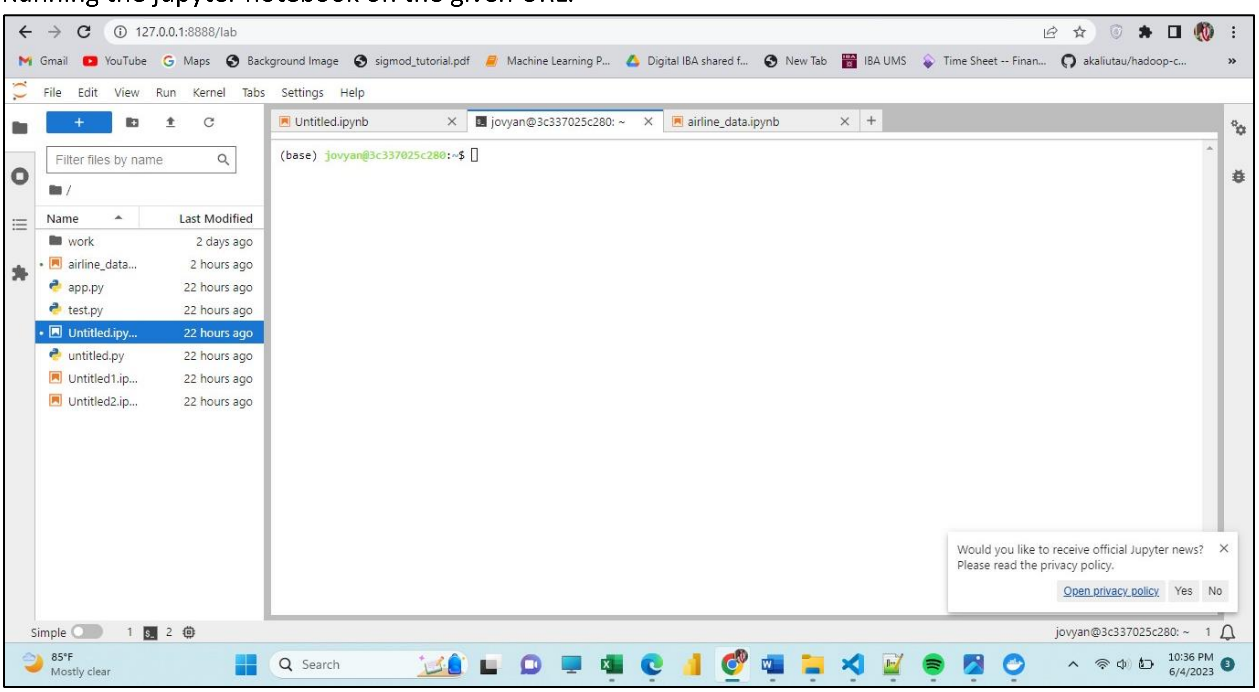


9. Machine Learning Model using PySpark

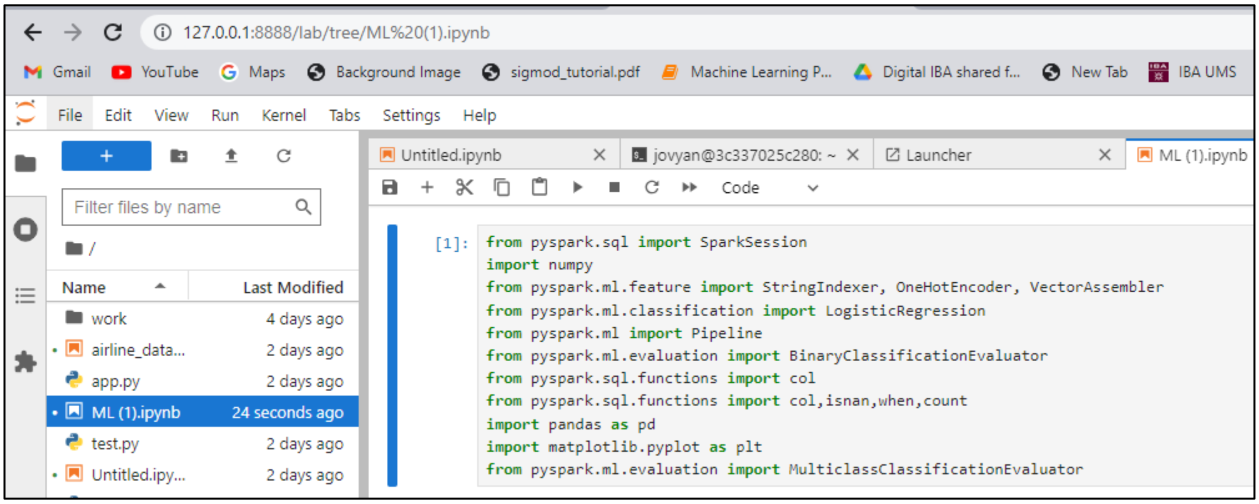
- Bash into spark-master.
  docker exec -it spark-master /bin/bash
- Bash into the Jupyter Notebook
  vi ~/.jupyter/jupyter_notebook_config.py
  jupyter notebook –allow-root

```
PS D:\MSDS\Big Data Analytics\Project\docker-fullstack-main\docker-fullstack-main> docker exec -it spark-master /bin/bash
bash-5.0# nano ~/.jupyter/jupyter_notebook_config.py
bash: nano: command not found
bash-5.0# vi ~/.jupyter/jupyter_notebook_config.py
bash-5.0# jupyter notebook --allow-root
[I 10:49:44.116 NotebookApp] Writing notebook server cookie secret to /root/.local/share/jupyter/runtime/notebook_cookie_secret

  _   _          _      _
 | | | |_ __  __| |__ _| |_ ___
 | |_| | '_ \/ _` / _` |  _/ -_)
  \___/| .__/\__,_\__,_|\__\___|
       |_|

Read the migration plan to Notebook 7 to learn about the new features and the actions to take if you are using extensions.

https://jupyter-notebook.readthedocs.io/en/latest/migrate_to_notebook7.html

Please note that updating to Notebook 7 might break some of your extensions.

[I 10:49:44.456 NotebookApp] Serving notebooks from local directory: /
[I 10:49:44.456 NotebookApp] Jupyter Notebook 6.5.4 is running at:
[I 10:49:44.456 NotebookApp] http://fd4187955c21:8888/?token=107e99bbdf27fde1a7c75f75b7a63e28d15fdf85fabc5359
[I 10:49:44.456 NotebookApp]  or http://127.0.0.1:8888/?token=107e99bbdf27fde1a7c75f75b7a63e28d15fdf85fabc5359
[I 10:49:44.456 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[W 10:49:44.465 NotebookApp] No web browser found: could not locate runnable browser.
[C 10:49:44.465 NotebookApp]

    To access the notebook, open this file in a browser:
        file:///root/.local/share/jupyter/runtime/nbserver-22414-open.html
    Or copy and paste one of these URLs:
        http://fd4187955c21:8888/?token=107e99bbdf27fde1a7c75f75b7a63e28d15fdf85fabc5359
     or http://127.0.0.1:8888/?token=107e99bbdf27fde1a7c75f75b7a63e28d15fdf85fabc5359
```

Running the jupyter notebook on the given URL.



Importing all required libraries.



```
[1]: from pyspark.sql import SparkSession
     import numpy
     from pyspark.ml.feature import StringIndexer, OneHotEncoder, VectorAssembler
     from pyspark.ml.classification import LogisticRegression
     from pyspark.ml import Pipeline
     from pyspark.ml.evaluation import BinaryClassificationEvaluator
     from pyspark.sql.functions import col
     from pyspark.sql.functions import col,isnan,when,count
     import pandas as pd
     import matplotlib.pyplot as plt
     from pyspark.ml.evaluation import MulticlassClassificationEvaluator
```

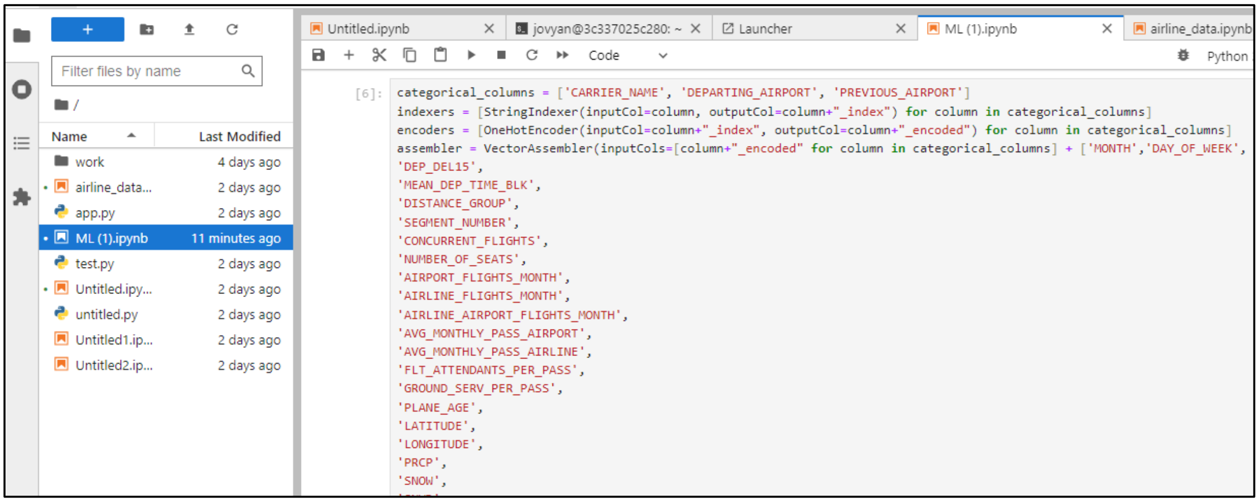
Creating a Spark Session

```
[2]: spark = SparkSession.builder \
         .appName("PySpark Classification Example") \
         .config("spark.driver.memory", "15g") \
         .getOrCreate()
```

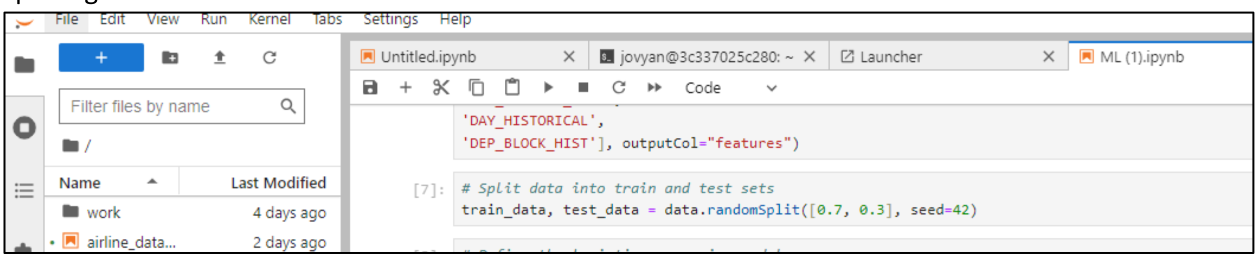
Loading the data from Hdfs and finding missing values. (This data did not have any missing values).

```
[5]: data = spark.read.format("csv") \
         .option("header", "true") \
         .option("inferSchema", "true") \
         .load("hdfs://namenode:8020/user/root/input/airline_data.csv")
     columns = data.columns
     for column in columns:
         print(column)
     missing_counts = data.select([count(when(col(c).isNull() | isnan(c), c)).alias(c) for c in data.columns])

MONTH
DAY_OF_WEEK
DEP_DEL15
DISTANCE_GROUP
SEGMENT_NUMBER
CONCURRENT_FLIGHTS
NUMBER_OF_SEATS
CARRIER_NAME
AIRPORT_FLIGHTS_MONTH
AIRLINE_FLIGHTS_MONTH
AIRLINE_AIRPORT_FLIGHTS_MONTH
AVG_MONTHLY_PASS_AIRPORT
AVG_MONTHLY_PASS_AIRLINE
FLT_ATTENDANTS_PER_PASS
GROUND_SERV_PER_PASS
PLANE_AGE
DEPARTING_AIRPORT
LATITUDE
```

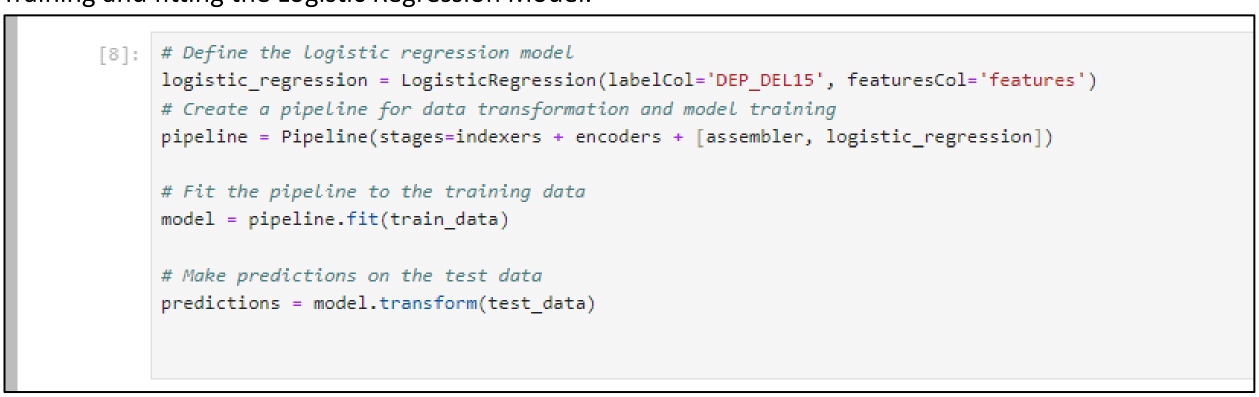
Creating categorical and numerical columns for one-hot encoding.



Splitting into train and test.



Training and fitting the Logistic Regression Model.

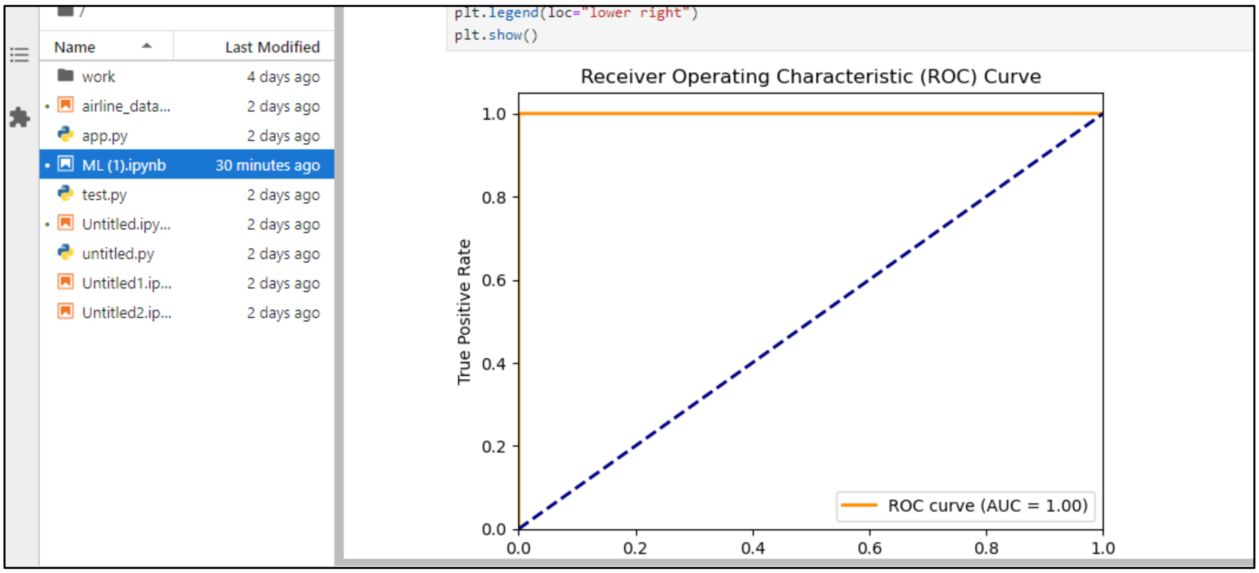


The accuracy of the model is 0.99

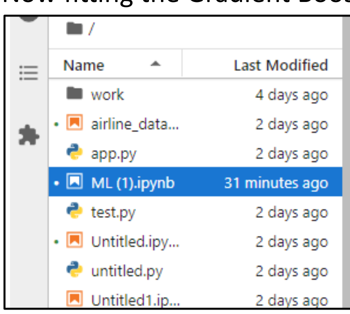
```
# Evaluate the model
evaluator = BinaryClassificationEvaluator(labelCol='DEP_DEL15')
accuracy = evaluator.evaluate(predictions)


print("Accuracy:", accuracy)
```

```
Accuracy: 0.9999996496645805
```

The ROC is 1 for this model.

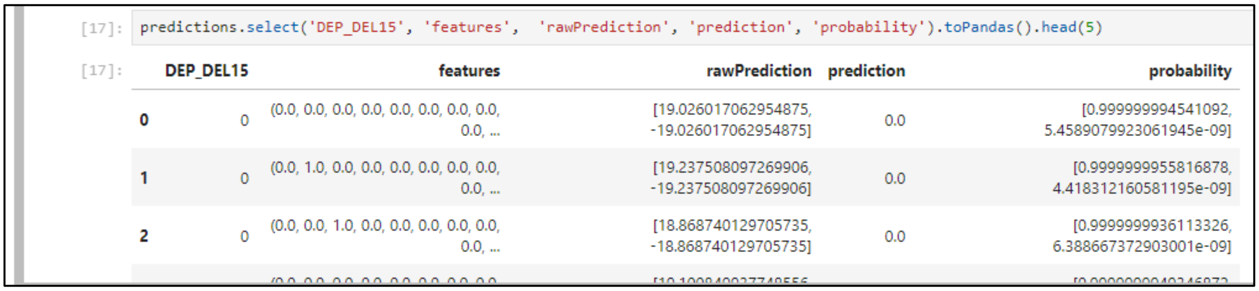


Now fitting the Gradient Boosting Model.




```
[14]: from pyspark.ml.classification import GBTClassifier
# Create a Gradient Boosting Classifier
gbt = GBTClassifier(labelCol="DEP_DEL15", featuresCol="features")

# Create a pipeline for data transformation and model training
pipeline = Pipeline(stages=indexers + encoders + [assembler, logistic_regression])

# Fit the pipeline to the training data
model = pipeline.fit(train_data)

# Make predictions on the test data
predictions = model.transform(test_data)
```

The comparison of the actual prediction and the prediction made by the model.

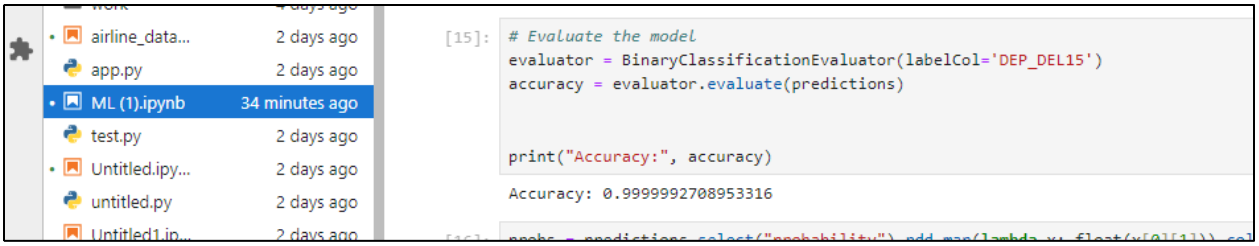


```
[17]: predictions.select('DEP_DEL15', 'features',  'rawPrediction', 'prediction', 'probability').toPandas().head(5)
```

[17]:

	DEP_DEL15	features	rawPrediction	prediction	probability
0	0	(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	[19.026017062954875, -19.026017062954875]	0.0	[0.999999994541092, 5.4589079923061945e-09]
1	0	(0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	[19.237508097269906, -19.237508097269906]	0.0	[0.9999999955816878, 4.418312160581195e-09]
2	0	(0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	[18.868740129705735, -18.868740129705735]	0.0	[0.9999999936113326, 6.388667372903001e-09]

The Accuracy of this model is also 0.99.




```
[15]: # Evaluate the model
evaluator = BinaryClassificationEvaluator(labelCol='DEP_DEL15')
accuracy = evaluator.evaluate(predictions)


print("Accuracy:", accuracy)
```

```
Accuracy: 0.9999992708953316
```

The ROC for this model is 1.