assignment2-pyspark-in-databricks

November 24, 2024

0.0.1 PySpark Assignment in Databricks: Analyzing and Modeling Flight Delays Data

0.0.2 Step 1: Data Loading and Initial Exploration

[0]:
```
# Import Spark Session
from pyspark.sql import SparkSession
```

[0]:
```
# Importing Necessary libraries
from pyspark.sql.functions import col, when, concat, lit, to_date, dayofweek,␣
 ↪count,month,year
from pyspark.sql.functions import regexp_replace
from pyspark.sql import functions as F
from pyspark.ml.feature import Imputer
# Plot using Matplotlib
import matplotlib.pyplot as plt
```

[0]:
```
# Initialize Spark session
spark = SparkSession.builder.appName("FlightDelays&Cancellation").getOrCreate()
```

1. Load Data

[0]:
```
# Load the dataset from the uploaded location
file_path = "dbfs:/FileStore/2016.csv"
df = spark.read.csv(file_path, header=True, inferSchema=True)
```

2. Initial Exploration

[0]:
```
# Display schema
df.printSchema()
```

```
root
 |-- FLIGHT_DATE : string (nullable = true)
 |-- AirlineIdentifier: string (nullable = true)
 |-- FlightNumber: integer (nullable = true)
 |-- Origin: string (nullable = true)
 |-- Dest: string (nullable = true)
 |-- PlannedDepTime: integer (nullable = true)
 |-- ActualDepTime: integer (nullable = true)
```

```
|-- DepDelay: integer (nullable = true)
|-- TaxiOut: integer (nullable = true)
|-- WheelsOff: integer (nullable = true)
|-- WheelsOn: integer (nullable = true)
|-- TaxiIn: integer (nullable = true)
|-- PlannedArrivalTime: integer (nullable = true)
|-- ArrivalTime: integer (nullable = true)
|-- ArrivalDelay: integer (nullable = true)
|-- AirTime: integer (nullable = true)
|-- Distance: integer (nullable = true)
|-- CANCELLED: integer (nullable = true)
|-- CANCELLATION_REASON: string (nullable = true)
|-- AIR_SYSTEM_DELAY: integer (nullable = true)
|-- SECURITY_DELAY: integer (nullable = true)
|-- AIRLINE_DELAY: integer (nullable = true)
|-- LATE_AIRCRAFT_DELAY: integer (nullable = true)
```

[0]:
```
# Display the first 10 rows of the DataFrame
display(df.limit(10))
```

Count the number of rows and columns.

[0]:
```
# Count the number of rows and columns.
num_rows = df.count()
num_columns = len(df.columns)
print(f"Number of rows: {num_rows}, Number of columns: {num_columns}")
```

```
Number of rows: 872860, Number of columns: 23
```

Questions:

How many rows and columns does the dataset contain? - Answer : There are 872860 rows and 23 columns.

What are the data types of each column?

Answer: The data types of each column are displayed using printSchema(). Here are the data types of each column in text form: **FLIGHT_DATE**: string, **AirlineIdentifier**: string, **FlightNumber**: integer, **Origin**: string, **Dest**: string, **PlannedDepTime**: integer, **ActualDepTime**: integer, **DepDelay**: integer, **TaxiOut**: integer, **WheelsOff**: integer, **WheelsOn**: integer, **TaxiIn**: integer, **PlannedArrivalTime**: integer, **ArrivalTime**: integer, **ArrivalDelay**: integer, **AirTime**: integer, **Distance**: integer, **CANCELLED** : integer, **CANCELLATION_REASON** : string, **AIR_SYSTEM_DELAY** : integer, **SECURITY_DELAY**: integer, **AIRLINE_DELAY** : integer, **LATE_AIRCRAFT_DELAY** : integer

0.0.3 Step 2: Data Cleaning and Transformation

1. Handling Missing Values

[0]:
```
# Count the number of missing values in each column
missing_counts = df.select([F.count(F.when(F.col(c).isNull(), c)).alias(c) for␣
 ↪c in df.columns])
display(missing_counts)
```

[0]:
```
# Impute missing values for numerical columns with mean or median
from pyspark.ml.feature import Imputer

numerical_columns = ['ActualDepTime', 'DepDelay', 'TaxiOut', 'WheelsOff',
                     'WheelsOn', 'TaxiIn', 'ArrivalTime','ArrivalDelay',␣
 ↪'AirTime','AIR_SYSTEM_DELAY', 'SECURITY_DELAY', 'AIRLINE_DELAY',␣
 ↪'LATE_AIRCRAFT_DELAY']

imputer = Imputer(inputCols=numerical_columns, outputCols=[f"{col}_imputed" for␣
 ↪col in numerical_columns]).setStrategy("median")
df = imputer.fit(df).transform(df)
```

[0]:
```
# Impute categorical columns with "Unknown" for missing values
categorical_columns = ['CANCELLATION_REASON']

for column in categorical_columns:
    df = df.na.fill({column: 'Unknown'})
```

2. Feature Engineering

[0]:
```
# Remove anything after the actual date, e.g., "(2)"
df = df.withColumn("FLIGHT_DATE ", regexp_replace(col("FLIGHT_DATE "), r"(\s*\(.
 ↪*\))", ""))
```

[0]:
```
# Convert FLIGHT_DATE  to a proper date type
df = df.withColumn("FLIGHT_DATE ", to_date(col("FLIGHT_DATE "), "yyyy-MM-dd"))
```

[0]:
```
# Feature Engineering
# Extract day of week, month, and year from FLIGHT_DATE
df = df.withColumn("DAY_OF_WEEK", dayofweek(col("FLIGHT_DATE "))) \
                        .withColumn("MONTH", month(col("FLIGHT_DATE "))) \
                        .withColumn("YEAR", year(col("FLIGHT_DATE ")))
```

[0]:
```
# Create a binary column indicating if a flight was delayed

df = df.withColumn("WasDelayed", when(col("ArrivalDelay") > 0, 1).otherwise(0))
df = df.withColumn("WasDepartureDelayed", when(col("DepDelay") > 0, 1).
 ↪otherwise(0))
df = df.withColumn("WasArrivalLate", when(col("ArrivalDelay") > 0, 1).
 ↪otherwise(0))
df = df.withColumn("AirSystemDelayOccurred", when(col("AIR_SYSTEM_DELAY") > 0,␣
 ↪1).otherwise(0))
```

```
df = df.withColumn("SecurityDelayOccurred", when(col("SECURITY_DELAY") > 0, 1).
 ↪otherwise(0))
df = df.withColumn("AirlineDelayOccurred", when(col("AIRLINE_DELAY") > 0, 1).
 ↪otherwise(0))
df = df.withColumn("LateAircraftDelayOccurred", when(col("LATE_AIRCRAFT_DELAY")␣
 ↪> 0, 1).otherwise(0))
```

[0]:
```
# Show updated schema and a sample of the transformed data
df.printSchema()
display(df.limit(10))
```

```
root
 |-- FLIGHT_DATE : date (nullable = true)
 |-- AirlineIdentifier: string (nullable = true)
 |-- FlightNumber: integer (nullable = true)
 |-- Origin: string (nullable = true)
 |-- Dest: string (nullable = true)
 |-- PlannedDepTime: integer (nullable = true)
 |-- ActualDepTime: integer (nullable = true)
 |-- DepDelay: integer (nullable = true)
 |-- TaxiOut: integer (nullable = true)
 |-- WheelsOff: integer (nullable = true)
 |-- WheelsOn: integer (nullable = true)
 |-- TaxiIn: integer (nullable = true)
 |-- PlannedArrivalTime: integer (nullable = true)
 |-- ArrivalTime: integer (nullable = true)
 |-- ArrivalDelay: integer (nullable = true)
 |-- AirTime: integer (nullable = true)
 |-- Distance: integer (nullable = true)
 |-- CANCELLED: integer (nullable = true)
 |-- CANCELLATION_REASON: string (nullable = false)
 |-- AIR_SYSTEM_DELAY: integer (nullable = true)
 |-- SECURITY_DELAY: integer (nullable = true)
 |-- AIRLINE_DELAY: integer (nullable = true)
 |-- LATE_AIRCRAFT_DELAY: integer (nullable = true)
 |-- ActualDepTime_imputed: integer (nullable = true)
 |-- DepDelay_imputed: integer (nullable = true)
 |-- TaxiOut_imputed: integer (nullable = true)
 |-- WheelsOff_imputed: integer (nullable = true)
 |-- WheelsOn_imputed: integer (nullable = true)
 |-- TaxiIn_imputed: integer (nullable = true)
 |-- ArrivalTime_imputed: integer (nullable = true)
 |-- ArrivalDelay_imputed: integer (nullable = true)
 |-- AirTime_imputed: integer (nullable = true)
 |-- AIR_SYSTEM_DELAY_imputed: integer (nullable = true)
 |-- SECURITY_DELAY_imputed: integer (nullable = true)
 |-- AIRLINE_DELAY_imputed: integer (nullable = true)
```

```
|-- LATE_AIRCRAFT_DELAY_imputed: integer (nullable = true)
|-- DAY_OF_WEEK: integer (nullable = true)
|-- MONTH: integer (nullable = true)
|-- YEAR: integer (nullable = true)
|-- WasDelayed: integer (nullable = false)
|-- WasDepartureDelayed: integer (nullable = false)
|-- WasArrivalLate: integer (nullable = false)
|-- AirSystemDelayOccurred: integer (nullable = false)
|-- SecurityDelayOccurred: integer (nullable = false)
|-- AirlineDelayOccurred: integer (nullable = false)
|-- LateAircraftDelayOccurred: integer (nullable = false)
```

Which columns had missing values, and how did you handle them?
The following columns contained missing values: ActualDepTime, DepDelay, TaxiOut, WheelsOn, Taxiln, ArrivalTime, ArrivalDelay, AirTime, AIR_SYSTEM_DELAY, SECURITY_DELAY, and LATE_AIRCRAFT_DELAY.

To handle the missing values in numerical columns, we applied the median imputation technique, which replaces missing entries with the median of the respective column. This approach was chosen because it is less affected by outliers, making it more reliable for datasets with extreme values.

For categorical columns, missing values were replaced with "Unknown" to preserve their interpretability while minimizing the data loss.

What additional features were created, and what was their purpose?

New features such as DAY_OF_WEEK, MONTH, YEAR, and WasDelayed were created to enhance the analysis of flight delays. DAY_OF_WEEK, MONTH, and YEAR were derived from the FLIGHT_DATE column to identify temporal patterns, like weekly or seasonal trends, that might affect delays. WasDelayed, a binary feature generated from ArrivalDelay, indicates whether a flight was delayed, with a value of 1 for delays greater than zero and 0 otherwise. These features provide valuable insights into the influence of time on delays and make predictive modeling more effective by introducing straightforward and interpretable categories.

0.0.4 Step 3: Exploratory Data Analysis (EDA) using Spark SQL

1. Register DataFrame as a Temporary Table:

[0]:
```
# Register the DataFrame as a temporary table
df.createOrReplaceTempView("flights")
```

2. SQL Queries

Calculate the average delay time for each airline.

[0]:
```
# Average Delay time for Each Airline
average_delay_airline = spark.sql("""
    SELECT AirlineIdentifier, AVG(ArrivalDelay) AS AverageArrivalDelay
    FROM flights
    GROUP BY AirlineIdentifier
```

```
    ORDER BY AverageArrivalDelay DESC
""")

# Show the results
average_delay_airline.show()
```

```
+-----------------+-------------------+
|AirlineIdentifier|AverageArrivalDelay|
+-----------------+-------------------+
|               NK| 16.120191609812746|
|               B6|  13.26877415472912|
|               OO|  5.413006143074179|
|               VX| 4.3464681844716875|
|               EV| 2.8303697216616532|
|               AA| 1.4225160099555356|
|               DL| -1.210882902499669|
|               HA|-1.4672557601783798|
|               WN| -1.569883802464189|
|               UA| -3.216790744727763|
|               F9| -3.669669226225635|
|               AS| -4.854136110296275|
+-----------------+-------------------+
```

Identify the top 5 airports with the most delayed departures.

[0]:
```
# Top 5 airports with the most delayed departures.
top_delayed_airports = spark.sql("""
    SELECT Origin, COUNT(*) AS TotalDelays
    FROM flights
    WHERE DepDelay > 0
    GROUP BY Origin
    ORDER BY TotalDelays DESC
    LIMIT 5
""")

# Show the results
top_delayed_airports.show()
```

```
+------+-----------+
|Origin|TotalDelays|
+------+-----------+
|   ATL|      20495|
|   ORD|      14780|
|   LAX|      11815|
|   DEN|      10868|
|   DFW|       9550|
+------+-----------+
```

Determine the most common reason for flight cancellations.

[0]:
```
# Determine the most common reason for flight cancellations using the␣
 ↪CANCELLATION_REASON column
most_common_cancellation_reason = spark.sql("""
    SELECT CANCELLATION_REASON, COUNT(*) AS Count
    FROM flights
    WHERE CANCELLED = 1
    GROUP BY CANCELLATION_REASON
    ORDER BY Count DESC
    LIMIT 1
""")
most_common_cancellation_reason.show()
```

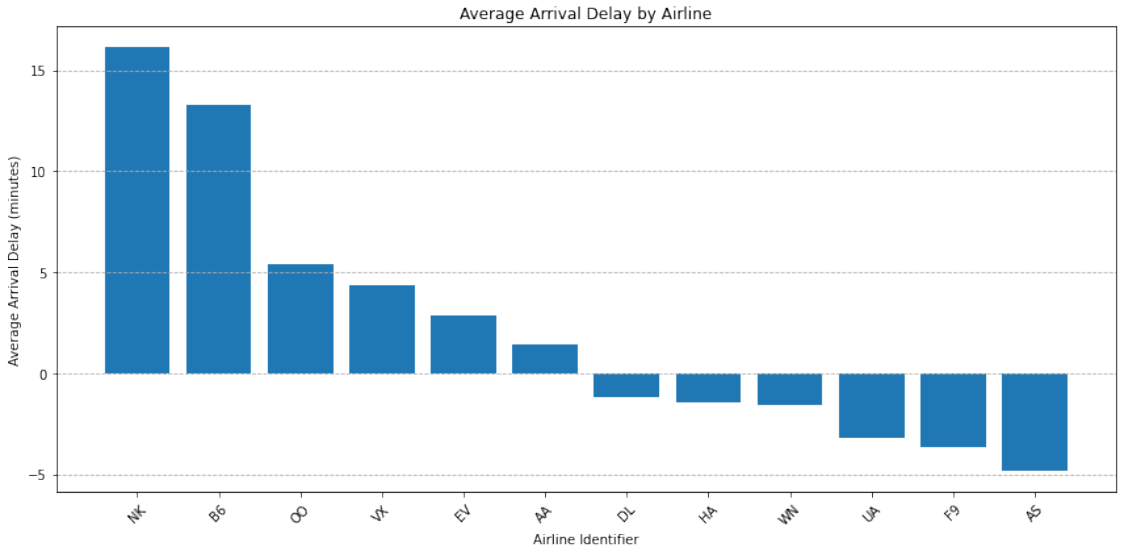
```
+-------------------+-----+
|CANCELLATION_REASON|Count|
+-------------------+-----+
|                  B| 7344|
+-------------------+-----+
```

3. Visualization:

Plot the average delay by airline.

[0]:
```
# Convert to Pandas DataFrame for visualization
average_delay_airline_pandas = average_delay_airline.toPandas()

plt.figure(figsize=(12, 6))
plt.bar(average_delay_airline_pandas['AirlineIdentifier'],␣
 ↪average_delay_airline_pandas['AverageArrivalDelay'])
plt.xlabel('Airline Identifier')
plt.ylabel('Average Arrival Delay (minutes)')
plt.title('Average Arrival Delay by Airline')
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--')
plt.tight_layout()
plt.show()
```



Visualize delay patterns over days of the week or months.

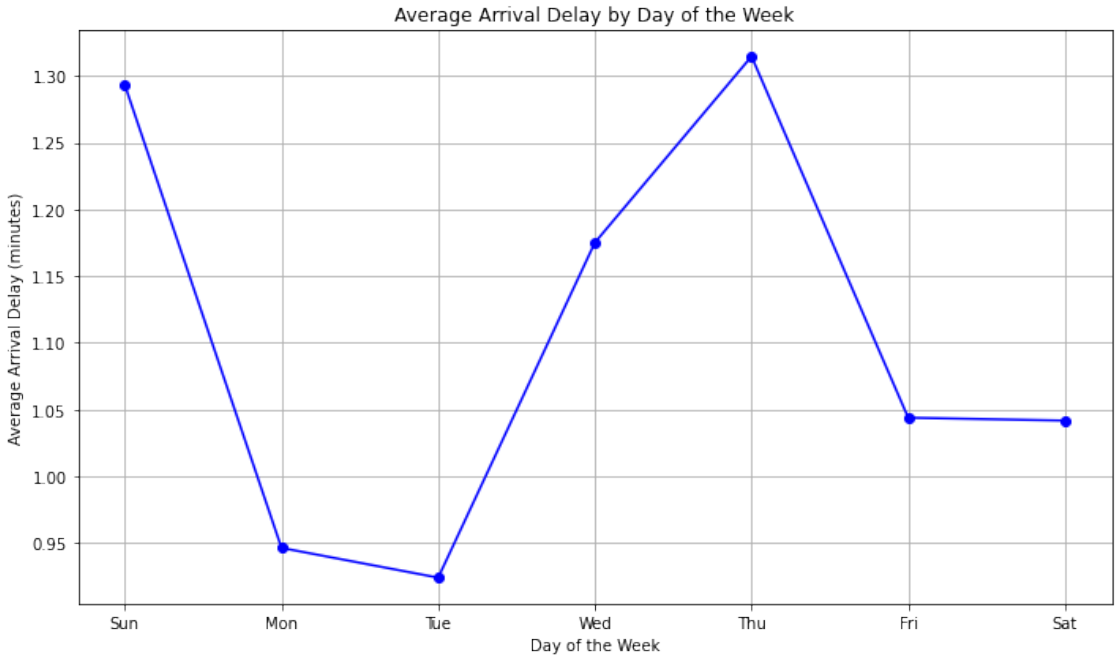
[0]:
```
# Calculate average delay by day of the week
average_delay_by_day = spark.sql("""
    SELECT DAY_OF_WEEK, AVG(ArrivalDelay) AS AverageArrivalDelay
    FROM flights
    GROUP BY DAY_OF_WEEK
    ORDER BY DAY_OF_WEEK
""")

# Show the result to confirm
display(average_delay_by_day)


# Convert to Pandas DataFrame for visualization
average_delay_by_day_pandas = average_delay_by_day.toPandas()

# Plot using Matplotlib
plt.figure(figsize=(10, 6))
plt.plot(average_delay_by_day_pandas['DAY_OF_WEEK'],␣
 ↪average_delay_by_day_pandas['AverageArrivalDelay'], marker='o',␣
 ↪linestyle='-', color='b')
plt.xlabel('Day of the Week')
plt.ylabel('Average Arrival Delay (minutes)')
plt.title('Average Arrival Delay by Day of the Week')
plt.xticks(ticks=[1, 2, 3, 4, 5, 6, 7], labels=['Sun', 'Mon', 'Tue', 'Wed',␣
 ↪'Thu', 'Fri', 'Sat'])
plt.grid(True)
plt.tight_layout()
```

```
plt.show()
```



Which airlines had the highest average delays?

The Average ArrivalDelay by Time plot reveals that NK (Spirit Airlines) has the highest average delay, exceeding 15 minutes, followed by B6 (JetBlue Airways) with an average delay of around 13 minutes. These two airlines stand out with significantly higher delays compared to others. While carriers like OO, VX, and EV also experience notable delays, they are not as substantial as those of NK and B6. In contrast, airlines such as AS (Alaska Airlines), F9 (Frontier Airlines), and UA (United Airlines) have negative average delays, indicating they are typically early or on schedule.

What patterns did you observe in delays by day of the week?

The line plot of average arrival delay by day of the week shows distinct patterns. Sundays experience the highest delays, averaging around 1.3 minutes, likely due to weekend congestion or increased travel activity as people return home. Another notable peak occurs on Thursdays, which may reflect heavy traffic as the workweek comes to a close. In contrast, Mondays and Tuesdays have much lower average delays, possibly due to lighter air traffic or fewer disruptions, resulting in better punctuality. Fridays and Saturdays show moderate and consistent delays, indicating stable operations without significant fluctuations on these days.

0.0.5 Step 4: Conclusion

Key Observations from the Analysis The analysis revealed that Spirit Airlines (NK) and JetBlue Airways (B6) had the highest average delays, exceeding 15 and 13 minutes, respectively, while Alaska Airlines (AS) and Frontier Airlines (F9) often performed better with negative average delays. Among airports, ATL, ORD, LAX, DEN, and DFW reported the highest number of delayed departures, suggesting congestion or operational challenges at these hubs. The most common cause

of flight cancellations was security delays (B), accounting for 7,344 cancellations. Additionally, delay patterns showed Sundays and Thursdays had the highest average delays, with Sundays averaging 1.3 minutes, while Mondays and Tuesdays experienced the least delays, reflecting smoother operations. These findings highlight key areas for improving on-time performance and resource management.