DATA 228 Technical Report

<u>NYC Yellow Taxi Analysis</u>

Parag Deshpande
Vaishnavi Samboji
Catherine Raj

Abstract

This project makes use of big data tools to analyze and predict the taxi fares in New York City. Challenges faced by passengers and drivers are also addressed by this project. 13 million records were in the dataset and Pyspark was used to process the data. Hive was used to visualize and MLLib was used to develop the regression model. The preprocessing steps involved removing outliers, feature engineering, normalization and more to ensure that there are accurate predictions. The model that resulted got a Root Mean Square Error of 0.92 with a 7.38% error rate. The visualizations from Hive gave insights on trip and revenue patterns, as well as passenger behaviors. The analysis in the project illustrates the importance of big data analytics when it comes to improving operations and transparency in the industry of transportation.

Introduction

Millions of passengers utilize the New York City Yellow Taxi industry each year. This indicates that the NYC Yellow taxi industry plays a key role in urban transportation. However, many challenges arise due to the unpredictable nature of taxi fares. This causes frustration for both passengers and the drivers when it comes to trip planning. Big data analytics tools were used in this project to discover fare patterns and come up with a reliable predictive model for taxi fares. Pyspark,Hive,MLLib and other tools were utilized to analyze over 13 million records. The availability of the open-source NYC taxi trip records allowed for exploration of this massive and complex dataset. The 13 million records were also transferred from Google Cloud Storage to BigQuery so proper querying can be done. Deeper insights and understanding of fare structures and the factors that influence fare variability came to be. Robust preprocessing techniques were used to make sure that the data was accurate and reliable enough for analysis. Overall, the project emphasizes how important analysis, data engineering and machine learning models are when it comes to addressing challenges faced in the real world and making informed decisions.

Workflow images

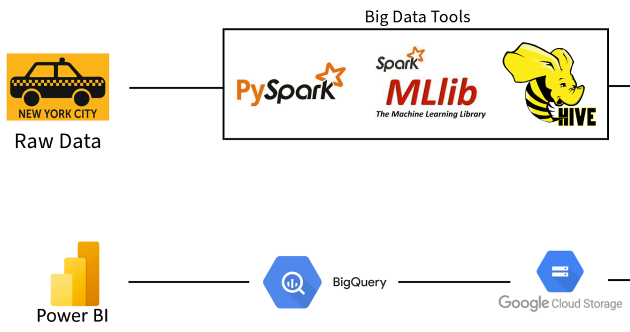
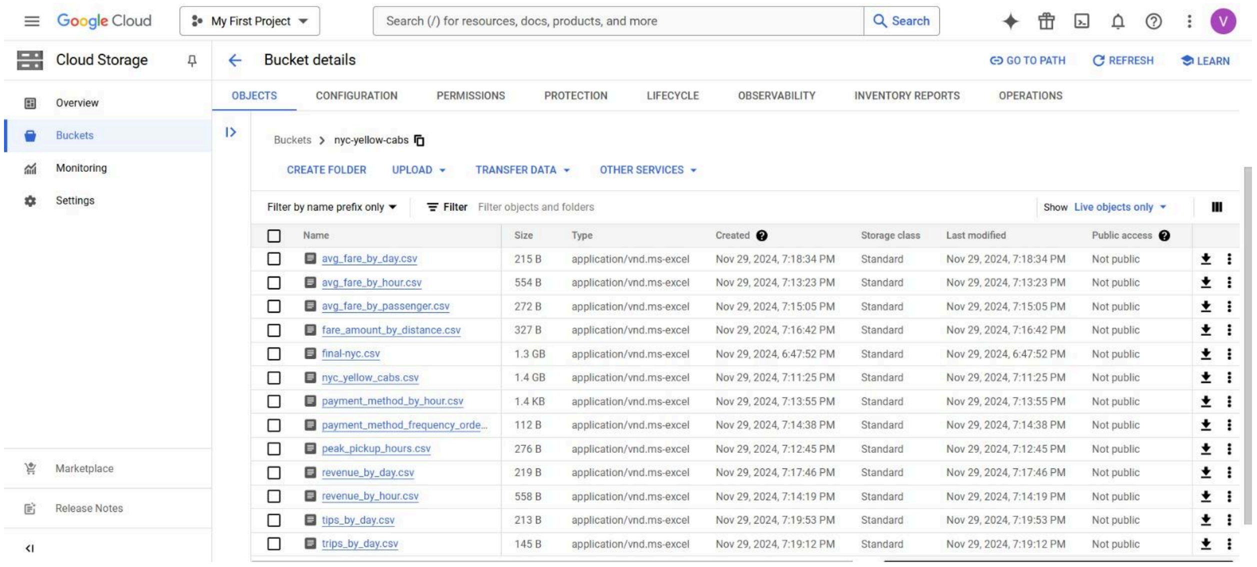


Fig 1: Showcasing the workflow of the project

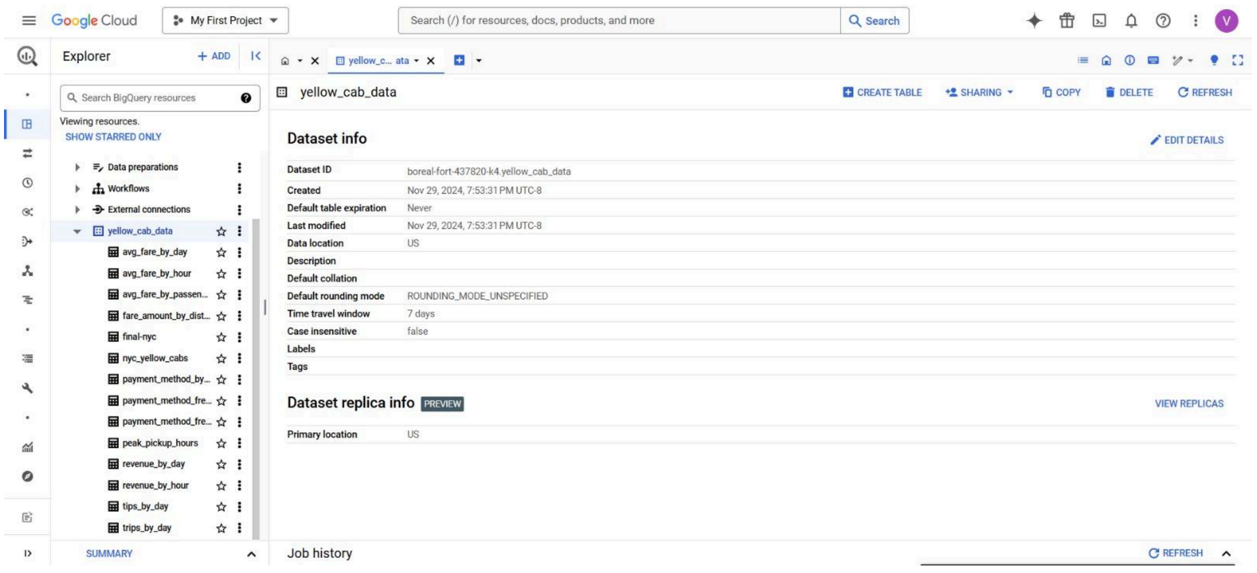
The below images show the utilization of GCP/BigQuery.



Buckets > nyc-yellow-cabs

Name	Size	Type	Created	Storage class	Last modified	Public access
avg_fare_by_day.csv	215 B	application/vnd.ms-excel	Nov 29, 2024, 7:18:34 PM	Standard	Nov 29, 2024, 7:18:34 PM	Not public
avg_fare_by_hour.csv	554 B	application/vnd.ms-excel	Nov 29, 2024, 7:13:23 PM	Standard	Nov 29, 2024, 7:13:23 PM	Not public
avg_fare_by_passenger.csv	272 B	application/vnd.ms-excel	Nov 29, 2024, 7:15:05 PM	Standard	Nov 29, 2024, 7:15:05 PM	Not public
fare_amount_by_distance.csv	327 B	application/vnd.ms-excel	Nov 29, 2024, 7:16:42 PM	Standard	Nov 29, 2024, 7:16:42 PM	Not public
final-nyc.csv	1.3 GB	application/vnd.ms-excel	Nov 29, 2024, 6:47:52 PM	Standard	Nov 29, 2024, 6:47:52 PM	Not public
nyc_yellow_cabs.csv	1.4 GB	application/vnd.ms-excel	Nov 29, 2024, 7:11:25 PM	Standard	Nov 29, 2024, 7:11:25 PM	Not public
payment_method_by_hour.csv	1.4 KB	application/vnd.ms-excel	Nov 29, 2024, 7:13:55 PM	Standard	Nov 29, 2024, 7:13:55 PM	Not public
payment_method_frequency_orde...	112 B	application/vnd.ms-excel	Nov 29, 2024, 7:14:38 PM	Standard	Nov 29, 2024, 7:14:38 PM	Not public
peak_pickup_hours.csv	276 B	application/vnd.ms-excel	Nov 29, 2024, 7:12:45 PM	Standard	Nov 29, 2024, 7:12:45 PM	Not public
revenue_by_day.csv	219 B	application/vnd.ms-excel	Nov 29, 2024, 7:17:46 PM	Standard	Nov 29, 2024, 7:17:46 PM	Not public
revenue_by_hour.csv	558 B	application/vnd.ms-excel	Nov 29, 2024, 7:14:19 PM	Standard	Nov 29, 2024, 7:14:19 PM	Not public
tips_by_day.csv	213 B	application/vnd.ms-excel	Nov 29, 2024, 7:19:53 PM	Standard	Nov 29, 2024, 7:19:53 PM	Not public
trips_by_day.csv	145 B	application/vnd.ms-excel	Nov 29, 2024, 7:19:12 PM	Standard	Nov 29, 2024, 7:19:12 PM	Not public

Fig 2: Screenshot of Google Cloud Storage Bucket



yellow_cab_data

Dataset info

Dataset ID	boreal-fort-437820-k4.yellow_cab_data
Created	Nov 29, 2024, 7:53:31 PM UTC-8
Default table expiration	Never
Last modified	Nov 29, 2024, 7:53:31 PM UTC-8
Data location	US
Description	
Default collation	
Default rounding mode	ROUNDING_MODE_UNSPECIFIED
Time travel window	7 days
Case insensitive	false
Labels	
Tags	

Dataset replica info

Primary location	US

Fig 3: Screenshot of Google BigQuery yellow_cab_data dataset

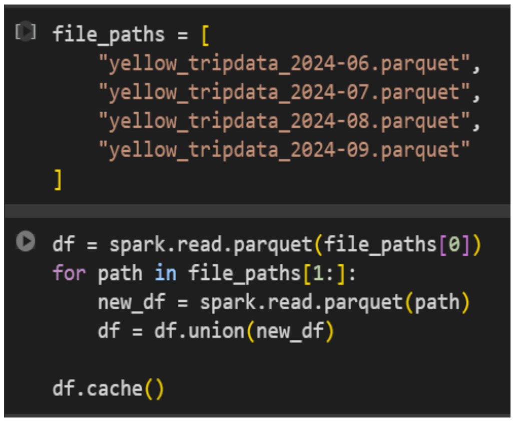
Data Exploration and Processing

There are over 13 million records in this dataset and key features such as trip distance, fare amount, passenger count,payment type, and many others. Data exploration involved loading the data into PySpark and schema inspection. A descriptive analysis was performed as well. To identify patterns, aggregation functions were utilized such as passenger counts per trip, traffic congestion fees and more. Duplicate records were found and removed to make the dataset more simple.

The data preprocessing aspect focused on missing values and outliers. Missing values were imputed using mean values. This allowed for completeness without drastically changing statistical properties. Meanwhile, outliers were detected using the Interquartile Range(IQR) and filtered to make sure that the dataset was clean and ready for analysis. Feature engineering was done to create new metrics such as fare_per_distance, total_per_passenger and more. This allows for more thorough inputs when it comes to model training. Linear correlation was also done with the target feature and all the other features in the dataset. The top 8 features were utilized for the model.Overall, the data exploration and preprocessing steps served as the foundation for building a robust and reliable regression model.

Hive was then used to process the large amounts of data and to visualize the significant insights. These insights are useful for feature selection and refining the model. Analyses such as identifying peak hours for pickups,average fares by time intervals, and more were done using Hive queries. Aggregated datasets such as average fare by hour were used to see the trip patterns and make validation of the trends from the PySpark analyses.

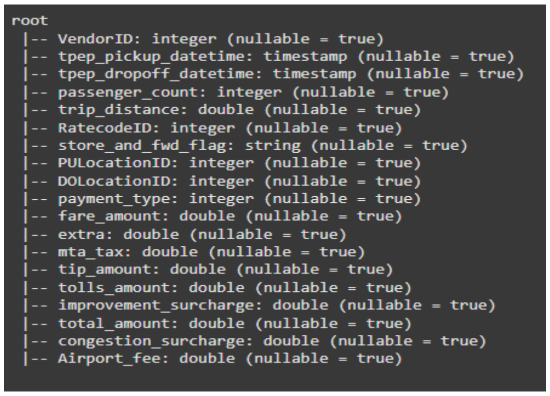
Data Exploration and Processing Images



```
file_paths = [
    "yellow_tripdata_2024-06.parquet",
    "yellow_tripdata_2024-07.parquet",
    "yellow_tripdata_2024-08.parquet",
    "yellow_tripdata_2024-09.parquet"
]
```

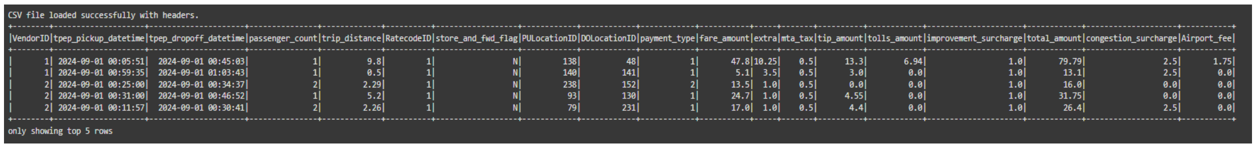
```
df = spark.read.parquet(file_paths[0])
for path in file_paths[1:]:
    new_df = spark.read.parquet(path)
    df = df.union(new_df)

df.cache()
```

Here we are loading the parquet files.



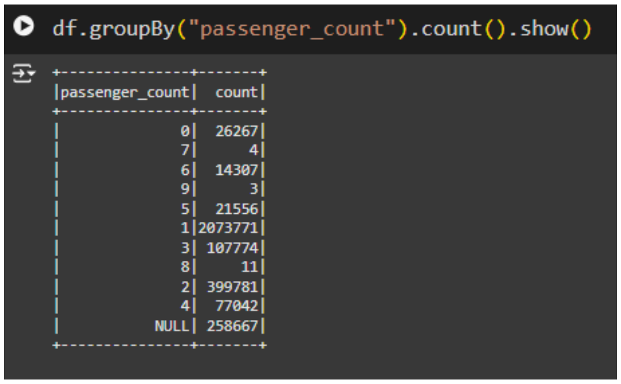
```
root
 |-- VendorID: integer (nullable = true)
 |-- tpep_pickup_datetime: timestamp (nullable = true)
 |-- tpep_dropoff_datetime: timestamp (nullable = true)
 |-- passenger_count: integer (nullable = true)
 |-- trip_distance: double (nullable = true)
 |-- RatecodeID: integer (nullable = true)
 |-- store_and_fwd_flag: string (nullable = true)
 |-- PULocationID: integer (nullable = true)
 |-- DOLocationID: integer (nullable = true)
 |-- payment_type: integer (nullable = true)
 |-- fare_amount: double (nullable = true)
 |-- extra: double (nullable = true)
 |-- mta_tax: double (nullable = true)
 |-- tip_amount: double (nullable = true)
 |-- tolls_amount: double (nullable = true)
 |-- improvement_surcharge: double (nullable = true)
 |-- total_amount: double (nullable = true)
 |-- congestion_surcharge: double (nullable = true)
 |-- Airport_fee: double (nullable = true)
```

All the column names are shown in this image. There are 19 columns in this dataset.



```
CSV file loaded successfully with headers.
+--------+--------------------+---------------------+---------------+-------------+----------+------------------+------------+------------+------------+-----------+-----+-------+----------+------------+---------------------+------------+--------------------+-----------+
|VendorID|tpep_pickup_datetime|tpep_dropoff_datetime|passenger_count|trip_distance|RatecodeID|store_and_fwd_flag|PULocationID|DOLocationID|payment_type|fare_amount|extra|mta_tax|tip_amount|tolls_amount|improvement_surcharge|total_amount|congestion_surcharge|Airport_fee|
+--------+--------------------+---------------------+---------------+-------------+----------+------------------+------------+------------+------------+-----------+-----+-------+----------+------------+---------------------+------------+--------------------+-----------+
|       1| 2024-09-01 00:05:51|  2024-09-01 00:45:03|              1|          9.8|         1|                 N|         138|          48|           1|       47.8|10.25|    0.5|      13.3|        6.94|                  1.0|       79.79|                 2.5|       1.75|
|       1| 2024-09-01 00:59:35|  2024-09-01 01:03:43|              1|          0.5|         1|                 N|         140|         141|           1|        5.1|  3.5|    0.5|       3.0|         0.0|                  1.0|        13.1|                 2.5|        0.0|
|       2| 2024-09-01 00:25:00|  2024-09-01 00:34:37|              2|         2.29|         1|                 N|         238|         152|           2|       13.5|  1.0|    0.5|       0.0|         0.0|                  1.0|        16.0|                 0.0|        0.0|
|       2| 2024-09-01 00:31:00|  2024-09-01 00:46:52|              1|          5.2|         1|                 N|          93|         130|           1|       24.7|  1.0|    0.5|      4.55|         0.0|                  1.0|       31.75|                 0.0|        0.0|
|       2| 2024-09-01 00:11:57|  2024-09-01 00:30:41|              2|         2.26|         1|                 N|          79|         231|           1|       17.0|  1.0|    0.5|       4.4|         0.0|                  1.0|        26.4|                 2.5|        0.0|
+--------+--------------------+---------------------+---------------+-------------+----------+------------------+------------+------------+------------+-----------+-----+-------+----------+------------+---------------------+------------+--------------------+-----------+
only showing top 5 rows
```

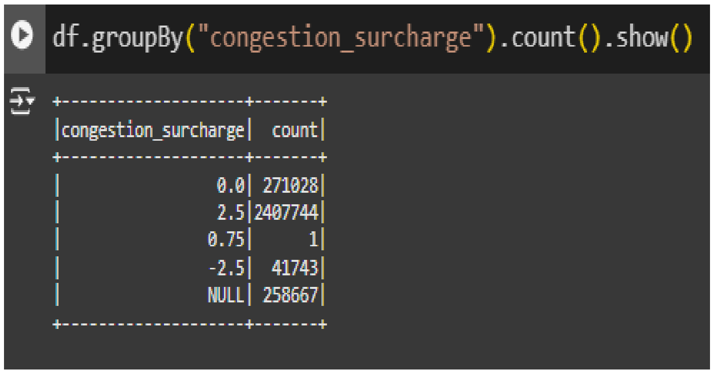
Above are the top 5 rows of the data being printed.



```
df.groupBy("passenger_count").count().show()
```

```
+---------------+-------+
|passenger_count|  count|
+---------------+-------+
|              0|  26267|
|              7|      4|
|              6|  14307|
|              9|      3|
|              5|  21556|
|              1|2073771|
|              3| 107774|
|              8|     11|
|              2| 399781|
|              4|  77042|
|           NULL| 258667|
+---------------+-------+
```

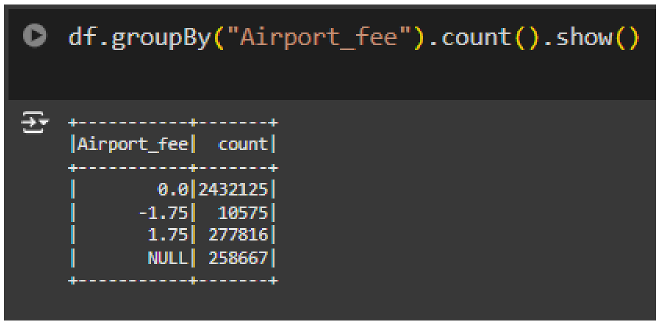
Here are the counts of passengers that travelled in a car in the past 4 months. The groupBy function as well as the count function was used here as aggregation functions .



```
df.groupBy("congestion_surcharge").count().show()
```

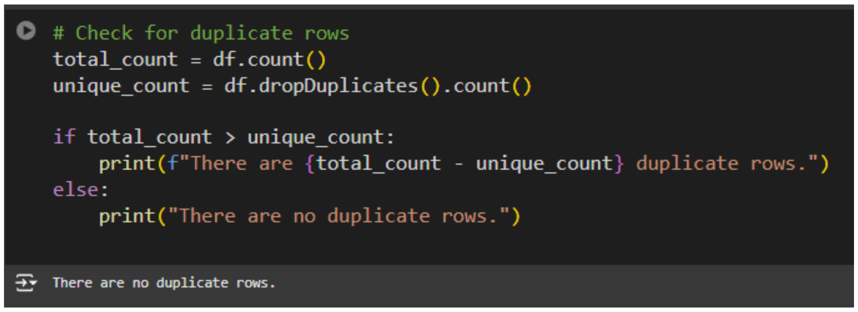
```
+--------------------+-------+
|congestion_surcharge|  count|
+--------------------+-------+
|                 0.0| 271028|
|                 2.5|2407744|
|                0.75|      1|
|                -2.5|  41743|
|                NULL| 258667|
+--------------------+-------+
```

Congestion surcharge occurs when riders are charged extra for traffic congestion.


```
df.groupBy("Airport_fee").count().show()
```

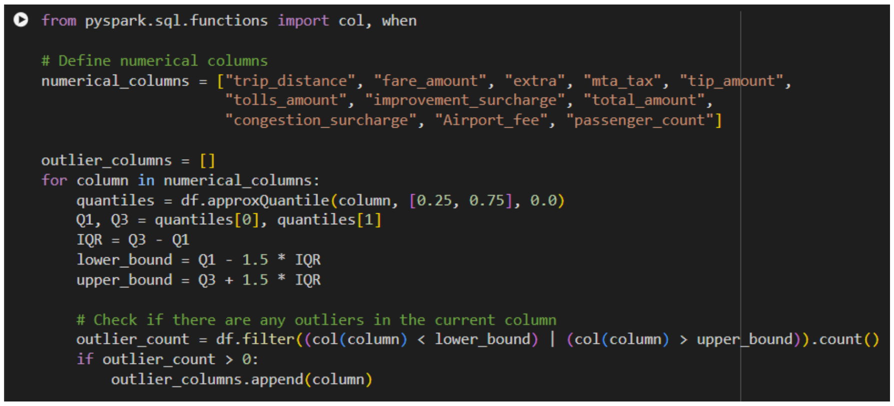
```
+-----------+-------+
|Airport_fee|  count|
+-----------+-------+
|        0.0|2432125|
|      -1.75|  10575|
|       1.75| 277816|
|       NULL| 258667|
+-----------+-------+
```

This image shows the count of passengers that have been travelling to the airport and the extra fee they were charged for doing so.

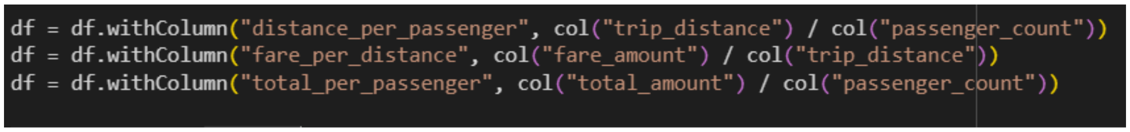

```
# Check for duplicate rows
total_count = df.count()
unique_count = df.dropDuplicates().count()

if total_count > unique_count:
    print(f"There are {total_count - unique_count} duplicate rows.")
else:
    print("There are no duplicate rows.")
```

```
There are no duplicate rows.
```

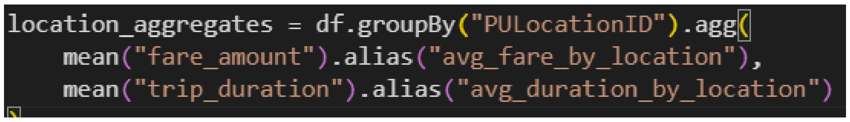
In this image we looked for duplicate rows and saw that there were no duplicate rows.


```
from pyspark.sql.functions import col, when

# Define numerical columns
numerical_columns = ["trip_distance", "fare_amount", "extra", "mta_tax", "tip_amount",
                     "tolls_amount", "improvement_surcharge", "total_amount",
                     "congestion_surcharge", "Airport_fee", "passenger_count"]

outlier_columns = []
for column in numerical_columns:
    quantiles = df.approxQuantile(column, [0.25, 0.75], 0.0)
    Q1, Q3 = quantiles[0], quantiles[1]
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    # Check if there are any outliers in the current column
    outlier_count = df.filter((col(column) < lower_bound) | (col(column) > upper_bound)).count()
    if outlier_count > 0:
        outlier_columns.append(column)
```

In this image, outlier detection was done using the IQR method as mentioned above.
Data points were considered if it ranges between these values: Value < Lower Bound or Value > Upper Bound.We saw that these columns had outliers - *trip_distance, fare_amount, extra, mta_tax, trip_amount, tolls_amount, imporovement_surcharge, total_amount, congestion_surcharge, Airport_fee, passenger_count.*

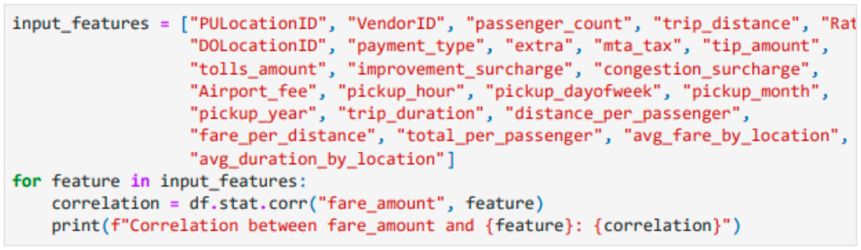

```
df = df.withColumn("distance_per_passenger", col("trip_distance") / col("passenger_count"))
df = df.withColumn("fare_per_distance", col("fare_amount") / col("trip_distance"))
df = df.withColumn("total_per_passenger", col("total_amount") / col("passenger_count"))
```

The existing data was used to derive new features that are required for predicting fares. This was done to help the model make accurate predictions. The new features shown here are distance-per_passenger, fare_per_distance, and total_per_passenger.

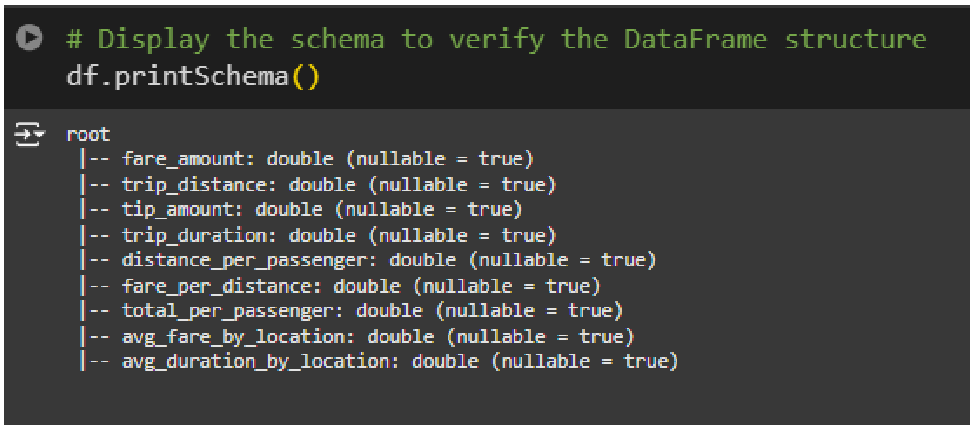


```
location_aggregates = df.groupBy("PULocationID").agg(
    mean("fare_amount").alias("avg_fare_by_location"),
    mean("trip_duration").alias("avg_duration_by_location")
```

This image shows code for the average fare charged based on location using the aggregate function.



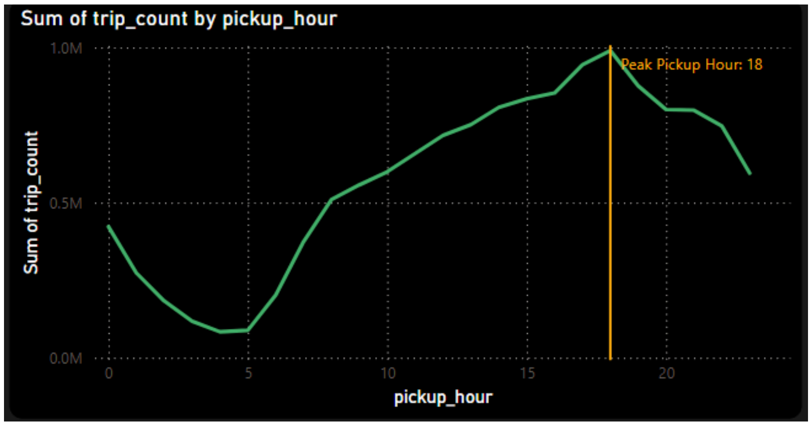
```
input_features = ["PULocationID", "VendorID", "passenger_count", "trip_distance", "Rat
                  "DOLocationID", "payment_type", "extra", "mta_tax", "tip_amount",
                  "tolls_amount", "improvement_surcharge", "congestion_surcharge",
                  "Airport_fee", "pickup_hour", "pickup_dayofweek", "pickup_month",
                  "pickup_year", "trip_duration", "distance_per_passenger",
                  "fare_per_distance", "total_per_passenger", "avg_fare_by_location",
                  "avg_duration_by_location"]
for feature in input_features:
    correlation = df.stat.corr("fare_amount", feature)
    print(f"Correlation between fare_amount and {feature}: {correlation}")
```



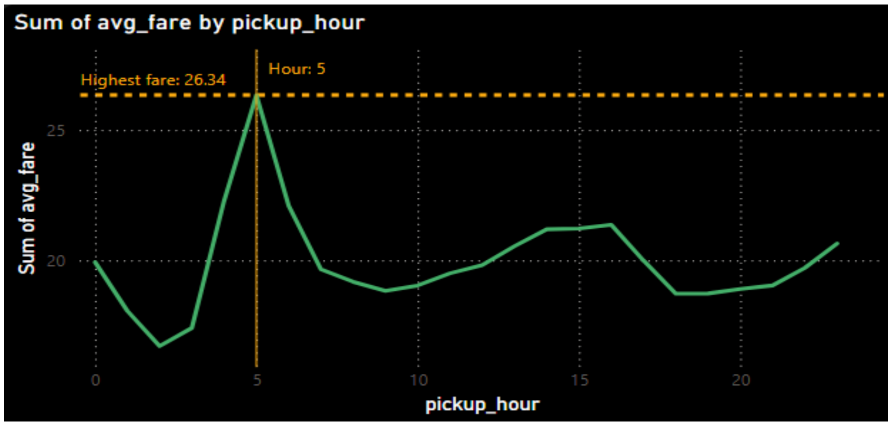
```
# Display the schema to verify the DataFrame structure
df.printSchema()
```

```
root
 |-- fare_amount: double (nullable = true)
 |-- trip_distance: double (nullable = true)
 |-- tip_amount: double (nullable = true)
 |-- trip_duration: double (nullable = true)
 |-- distance_per_passenger: double (nullable = true)
 |-- fare_per_distance: double (nullable = true)
 |-- total_per_passenger: double (nullable = true)
 |-- avg_fare_by_location: double (nullable = true)
 |-- avg_duration_by_location: double (nullable = true)
```

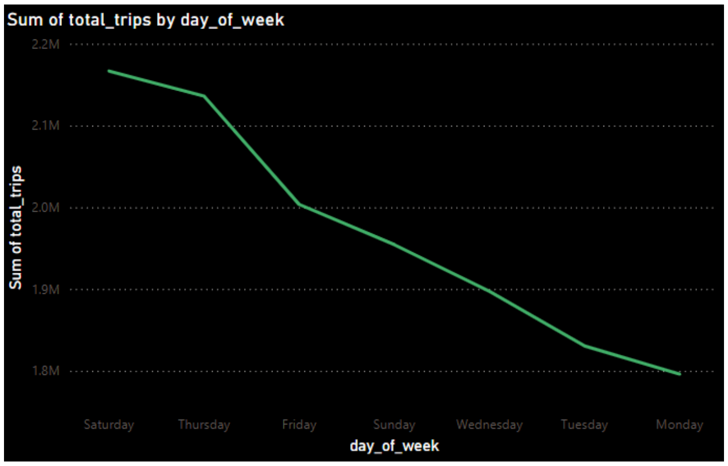
Linear correlation was checked between the target feature which is fare amount and all the other input features. The top 8 features were the ones used for the model training . A key thing to note here is that all the columns are numerical and the new features that were engineered are being utilized.



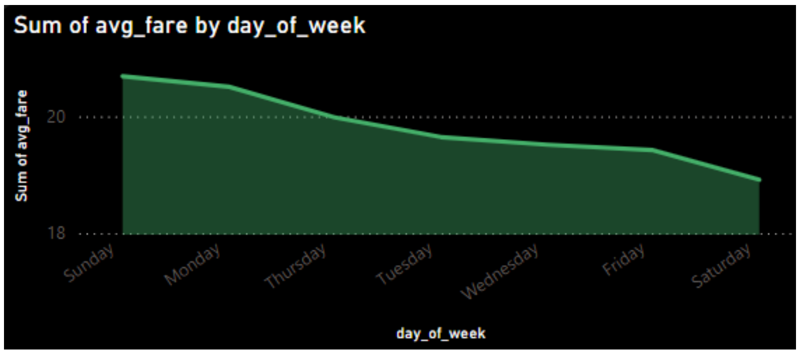
Here is an example of one of the visualizations created using Hive. This graph shows the sum of trip counts by pickup hour. The peak pick up hour seems to be 6pm.



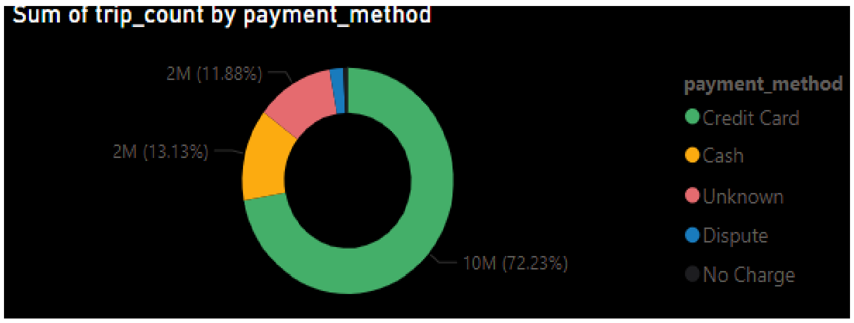
This graph shows the sum of the average far based on the pickup hour. The highest average fare shown here is $26.34 and this seems to occur at 5am.



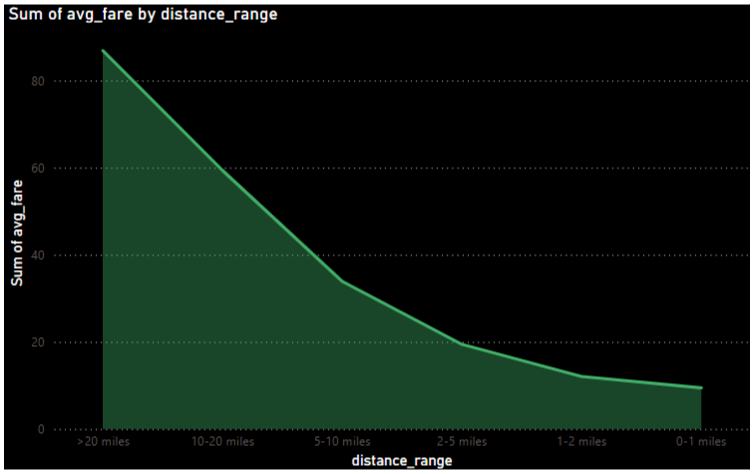
This is the sum of total trips by day of the week. The graph seems to show that the number of trips decline as the week goes on. The highest number of trips seem to be on Saturday, while the lowest number of trips seem to occur on Monday. This could be due to the fact most people have more time to do leisurely activities on the weekends versus the work days.



This graph shows a decline in the fare as the week goes on with prices being high in the beginning of the week(Sunday/Monday) and lower towards the end of the week. This could be due to the lower amount of trips taken on those days which can lead to fare increase.



This graph shows the sum of the trip counts by payment methods. Majority of the trips utilize credit cards as the payment method(72.23%).This could be due to how convenient it is to use credit cards when compared to other payment methods. It could also be due to the fact that carrying cash in a place as busy as New York is quite risky, so credit cards are preferred to reduce the risk of theft.



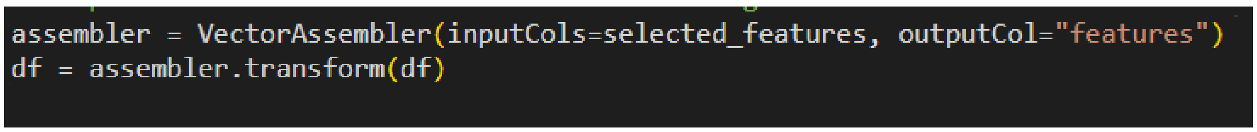
In the graph above, the average fare seems to decrease as the distance for the trip decreases. This makes sense as shorter trips tend to take less time and taxi fare tends to be based on time spent traveling and distance.

Proposed method

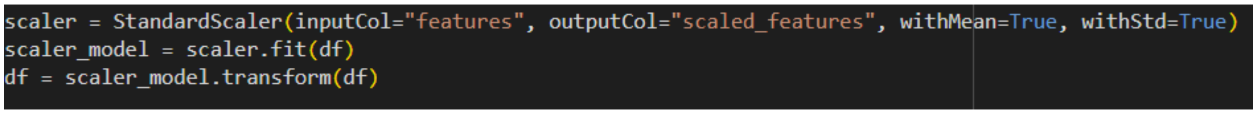
The main proposed method involved the usage of PySpark's MLlib library to predict the taxi fares using a linear regression model. The input features that were chosen were based on their correlation with fare_amount which is our target variable. All of the derived metrics chosen above such as fare_per_distance,distance_per_passenger and total_per_passenger were used as well. The features were then normalized by using a StandardScaler to improve the model performance. A VectorAssembler was utilized to combine the features into one input vector for the model training.

The data was split into training and testing subsets in which the model was trained on 80% of the data and tested on the remaining 20%.Root Mean Square Error(RMSE)and Relative RMSE were evaluation metrics used to assess the performance. The feature engineering and preprocessing done earlier allowed for the model to make more accurate predictions. Both Hive and PySpark allowed for a thorough exploration and analysis.

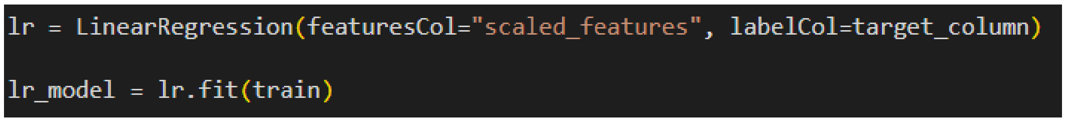
Proposed method images



```
assembler = VectorAssembler(inputCols=selected_features, outputCol="features")
df = assembler.transform(df)
```



```
scaler = StandardScaler(inputCol="features", outputCol="scaled_features", withMean=True, withStd=True)
scaler_model = scaler.fit(df)
df = scaler_model.transform(df)
```



```
lr = LinearRegression(featuresCol="scaled_features", labelCol=target_column)

lr_model = lr.fit(train)
```

In the above images, VectorAssembler takes all the chosen features and puts them into one column. Feature normalization is done using StandardScaler. Linear regression is then done as well.

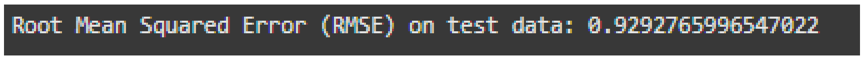
Results

The linear regression model achieved a Root Mean Square Error(RMSE) of 0.93 and a Relative RMSE of 7.38% This means that the model's predictions deviated from the actual values by 7.38% on average. This shows how the model is performing very well. Trip distance was one of the most influential features when it came to predicting. The trip distance feature also showed a high positive correlation of 0.86 with fare_amount(the target variable). The total_per_passenger feature was also significant indicating how passenger count can impact fare predictions.

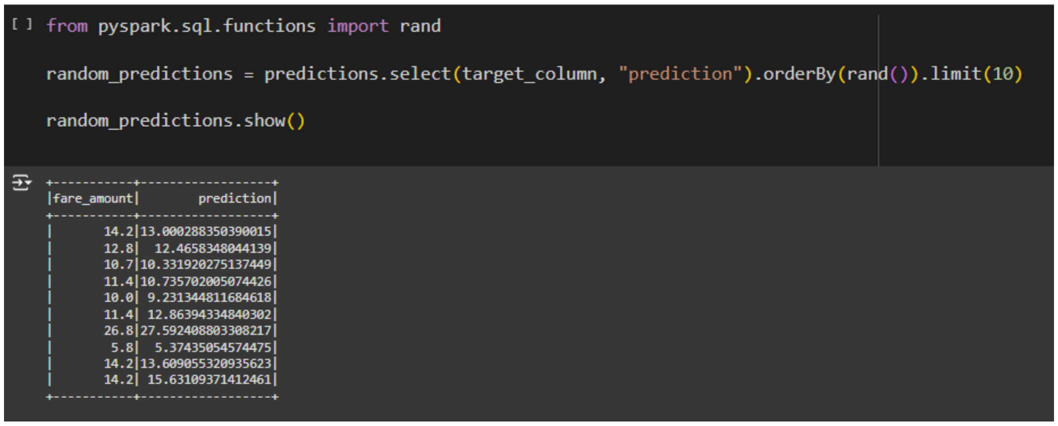
A brief visual analysis was done to show the alignment between actual and predicted fare amounts. This analysis showed that the alignment between them was quite close which indicates the high accuracy of the model. The importance of feature engineering and preprocessing was emphasized with this evaluation. The derived metrics facilitated the model's predictive power as well, by providing more to capture relationships in the data. The aggregated features helped the model with finding and summarizing data patterns. Overall, the results indicate how much of a role feature engineering, data preprocessing and big data tools play when it comes to creating a model that performs well.

Results images

Here is the RMSE found from the model:



```
Root Mean Squared Error (RMSE) on test data: 0.9292765996547022
```



```
from pyspark.sql.functions import rand

random_predictions = predictions.select(target_column, "prediction").orderBy(rand()).limit(10)

random_predictions.show()
```

```
+-----------+------------------+
|fare_amount|        prediction|
+-----------+------------------+
|       14.2|13.000288350390015|
|       12.8|  12.4658348044139|
|       10.7|10.331920275137449|
|       11.4|10.735702005074426|
|       10.0| 9.231344811684618|
|       11.4| 12.86394334840302|
|       26.8|27.592408803308217|
|        5.8|  5.37435054574475|
|       14.2|13.609055320935623|
|       14.2| 15.63109371412461|
+-----------+------------------+
```

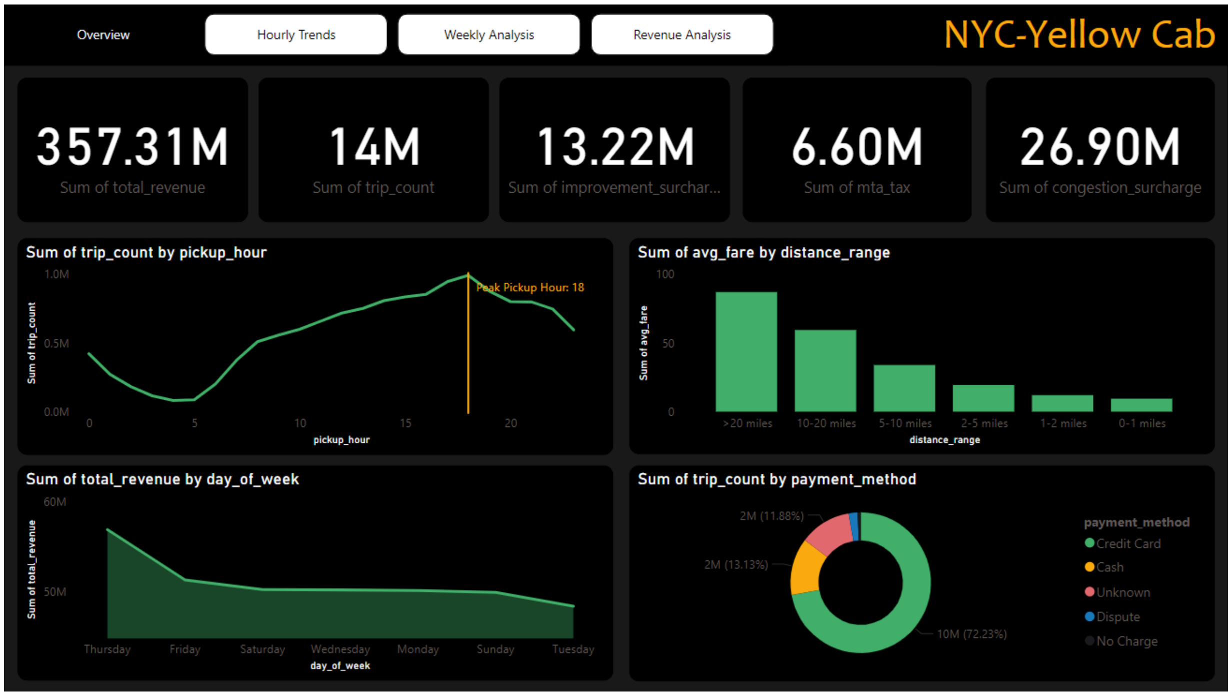
Here is a random set of values which are predicted fare amounts from the model as well as the actual fare amounts. As seen, there is not much of a difference between the values meaning the model is accurately predicting the fare amounts.

Discussion

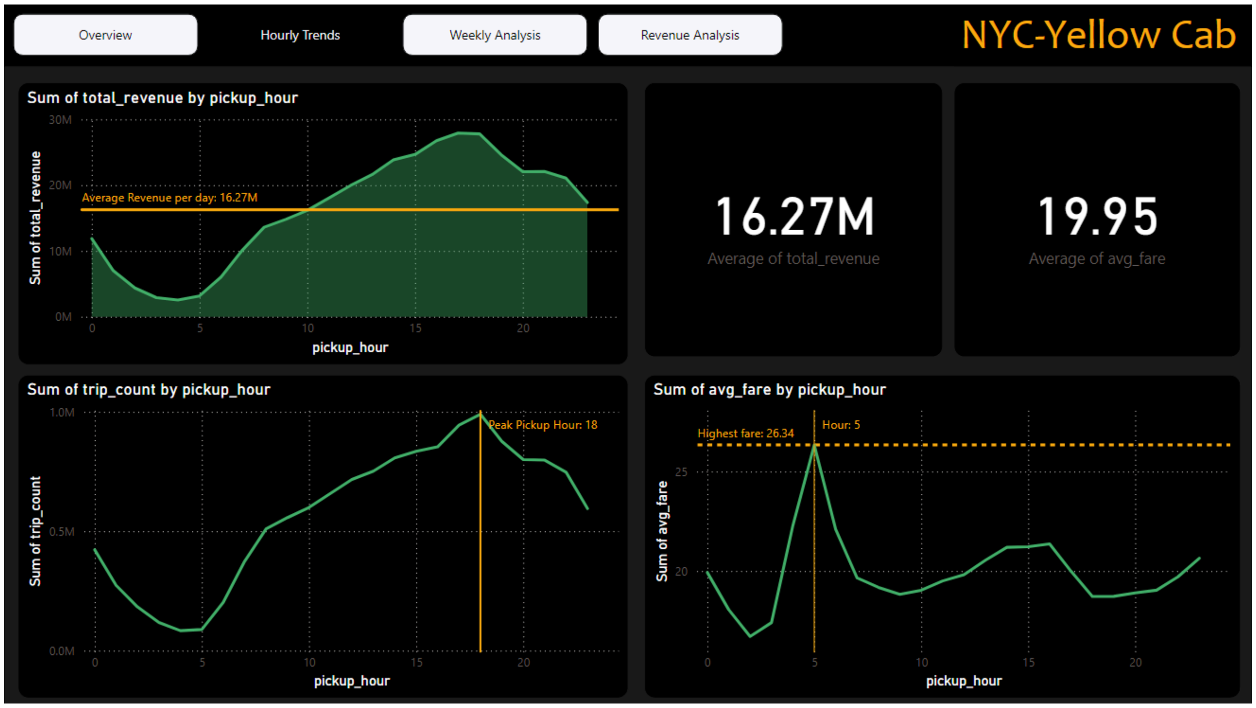
This project made use of many Big Data tools such as PySpark, Hive and MLlib to get a thorough analysis and prediction when it came to NYC Yellow taxi fares. It indicated how data-driven approaches are crucial when it comes to addressing the challenges or frustrations faced by passengers and drivers. Being able to predict fare can help passengers plan trips better and drivers improve their service. Because over 13 million records were being used, it was very crucial to use preprocessing to ensure missing values and outliers were taken care of. Feature engineering was also used to improve the model as it created meaningful inputs. This project emphasizes the role of data preprocessing, machine learning, and feature engineering when it comes to creating a model that performs well in predicting the taxi fares. Some future work can involve using more external variables such as weather or events, and looking at more advanced models. All in all, this project shows how data science, Big Data tools and machine learning are crucial when it comes to addressing transportation challenges and finding ways to facilitate passengers and drivers.

Dashboard

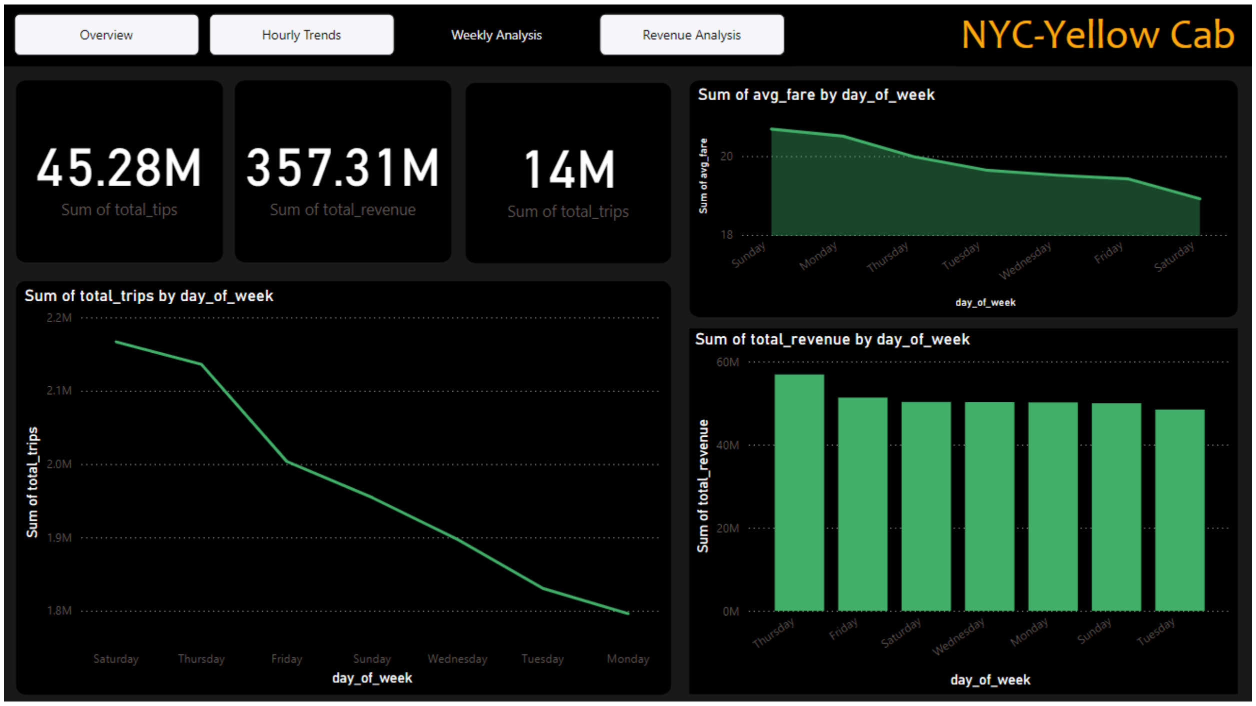
Overview:



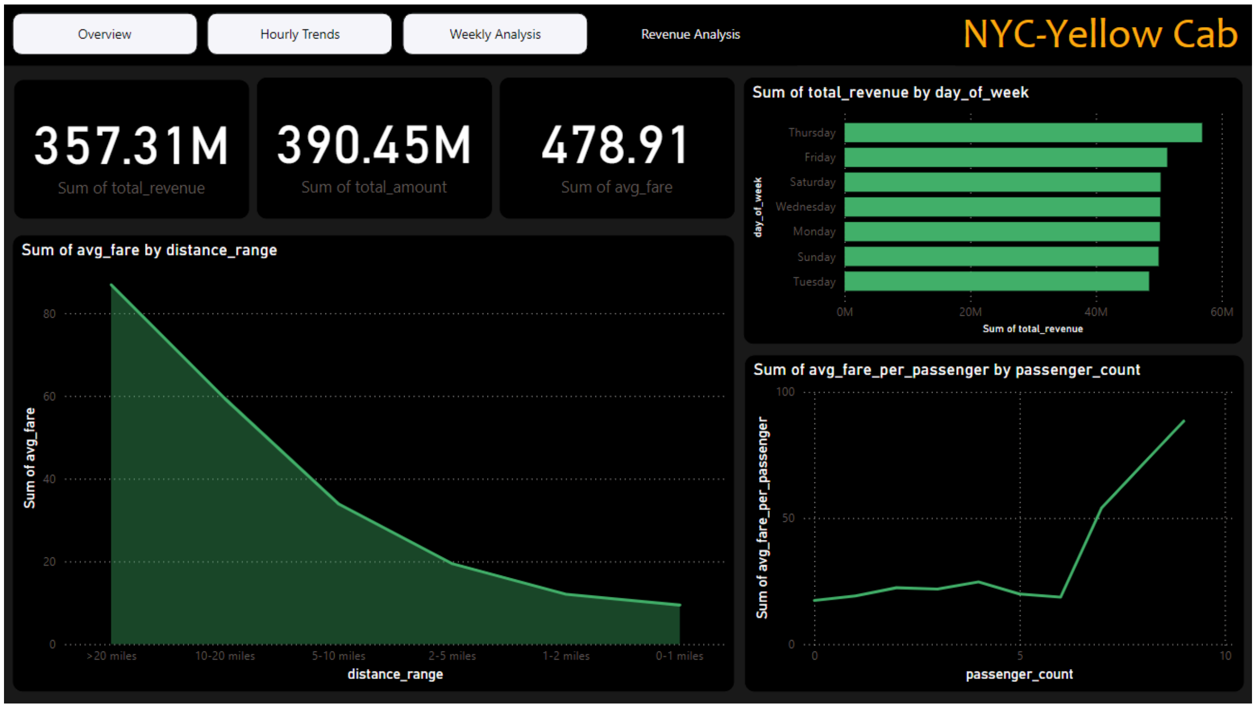
Hourly Analysis:



Weekly Analysis:



Revenue Analysis:


Overview
Hourly Trends
Weekly Analysis
Revenue Analysis
NYC-Yellow Cab
357.31M
Sum of total_revenue
390.45M
Sum of total_amount
478.91
Sum of avg_fare
Sum of total_revenue by day_of_week
day_of_week
Sum of total_revenue
Sum of avg_fare by distance_range
Sum of avg_fare
distance_range
>20 miles
10-20 miles
5-10 miles
2-5 miles
1-2 miles
0-1 miles
Sum of avg_fare_per_passenger by passenger_count
Sum of avg_fare_per_passenger
passenger_count