8th May 2024

CAPSTONE PROJECT - DATA 69099

Title: "EXPLORING TAXI FARE PATTERNS: FACTORS INFLUENCING PRICING DYNAMICS"

Harsha Vardhan Amara - 811302928 - hamara@kent.edu
Naga Sudha Pavani Tirumalasetty - 811294971- ntirumal@kent.edu
Subhasmita Maharana - 811263851 – smaharan@kent.edu

Analysis on Research Question: EXPLORING TAXI FARE PATTERNS: FACTORS INFLUENCING PRICING DYNAMICS- This study explores the various factors such as:

- How does the combination of pickup location and trip distance influence the fare amount for taxi rides in New York City?
- How does the average fare amount for taxi rides differ between weekdays and weekends in New York City?

Understanding the factors that contribute to variations in taxi fare amounts in New York City is important because it helps passengers make informed decisions about transportation costs and allows taxi operators to adjust pricing strategies accordingly. By analyzing these factors, policymakers can also develop regulations thatensure fair pricing practices and promote consumer welfare. Additionally, insights into taxi fare patterns can inform urban planning efforts and contribute to the overall efficiency and sustainability of transportation systemsin the city.

Methods & Analysis Implemented to solve the problem:

Data Visualization: Using libraries Plots, Matplotlib to visualize trends, patterns, and correlations in the dataset. Through dynamic data exploration and deeper insights, stakeholders may make better use of the graphics.

Regression analysis: Quantify the impact of pickup/drop-off locations, trip distance, time of day on taxi fare amounts.

Exploratory data analysis (EDA): To identify correlations and patterns between fare amounts and factors like pickup location, drop-off location, trip distance, time of day.

Time series analysis: Examine fare variations over different times of the day, days of the week.

Geospatial analysis: Analyze fare differences based on pickup and drop-off locations, considering factors like congestion and demand.

Prediction and Machine learning models: To anticipate trip durations and fares, ascertain the significance of various features (e.g., locations, time, and distance) using machine learning techniques like Random Forest, Cross-Validation, and Logistic Regression. This can provide light on the variables that have the most effects on results.

A discussion about how we can determine if we have usefully answered that question, including assessmentsof the accuracy and precision of the model used.

The usefulness of the analysis can be evaluated based on the accuracy and precision of the predictive model in estimating taxi fare amounts. This can be assessed using various evaluation metrics such as mean squared error (MSE), root mean squared error (RMSE), and R-squared value.

Mean Squared Error (MSE): It measures the average squared difference between the predicted values and the actual values.
In the context of analyzing factors influencing taxi fare amounts, MSE can be used to assess the accuracy of the regression model in predicting fare amounts based on factors such as pickup location, drop-off location, trip distance, time of day. Validation techniques such as cross-validation can also help determine the model's abilityto generalize to unseen data.

Root Mean Squared Error (RMSE): RMSE is like Mean Squared Error (MSE), but it represents the square root of the average squared difference between the predicted values and the actual values.

R-squared (R^2): The R-squared value, also known as the coefficient of determination, is a statistical measure used to assess the goodness-of-fit of a regression model. It indicates the proportion of the variance in the dependent variable.
In the context of analyzing factors influencing taxi fare amounts, R-squared can be used to assess the overall effectiveness of the regression model in explaining the variability in fare amounts based on factors such as pickup location, drop-off location, trip distance, time of day.

Mean Absolute Error (MAE): It measures the average absolute difference between the predicted values and the actual values. Unlike Mean Squared Error (MSE) and Root Mean Squared Error (RMSE), MAE does not square the errors, making it less sensitive to outliers.

A discussion about how you might implement the model in the real world to create value for someone.

Implementing our predictive model in real-world scenarios holds immense potential to revolutionize the taxi industry and benefit various stakeholders:
Passengers will experience improved transparency and fairness in fare pricing, thanks to the insights provided by our model. Armed with knowledge about the factors influencing fares, passengers can make informed decisions about their travel routes and timings, leading to greater satisfaction with taxi services.

For taxi companies, our model can offer invaluable insights into fare pricing dynamics, enabling them to tailor pricing strategies dynamically. By adapting fares based on factors like demand fluctuations and time of day, taxi companies can not only increase their revenue but also stay ahead in a competitive market.

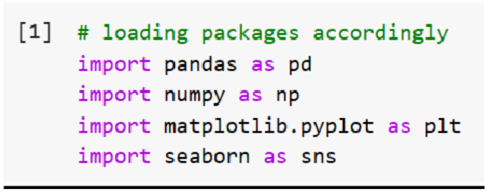
Individual taxi drivers stand to gain significantly from our model's personalized recommendations on pricing and route optimization. With tailored suggestions on optimal routes and peak hours for passenger pickups, drivers can maximize their earnings while minimizing idle time, ultimately enhancing their livelihoods.

Government agencies and policymakers can leverage our model's findings to inform regulatory decisions and transportation policies. By understanding the impact of taxi fare dynamics on urban mobility, policymakers can devise effective strategies such as congestion pricing and infrastructure investments to improve overall transportation systems.

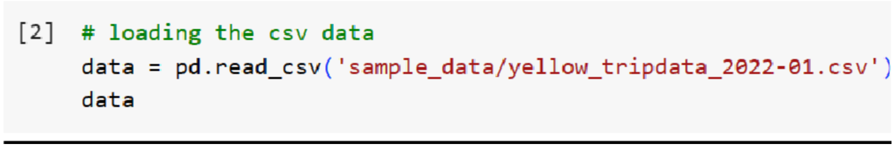
Urban planners and transportation authorities can harness the power of our model to optimize traffic flow and alleviate congestion in cities. By integrating insights about taxi fare patterns with data on infrastructure and public transit, planners can design more efficient transportation networks, ultimately enhancing the quality of life for urban residents.

Initial Data Analysis- read in the data and describe columns, size, etc.

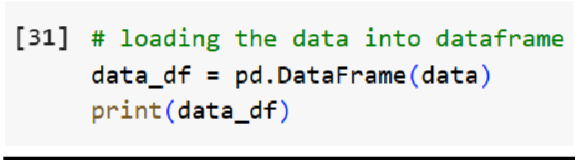
Reading and loading the data: Required packages have been loaded and the dataset is read by using ***pd.read_csv (“)*** command and assigned it to data variable.



```
[1]  # loading packages accordingly
     import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
     import seaborn as sns
```



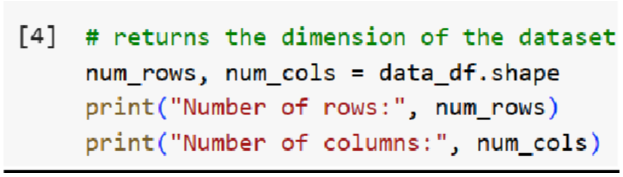
```
[2]  # loading the csv data
     data = pd.read_csv('sample_data/yellow_tripdata_2022-01.csv')
     data
```



```
[31] # loading the data into dataframe
     data_df = pd.DataFrame(data)
     print(data_df)
```

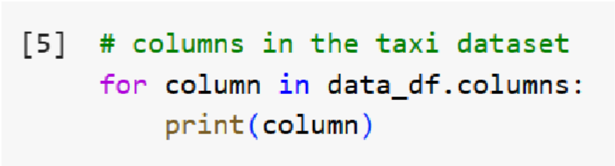
- Here, importing the pandas for data manipulation and analysis, NumPy for scientific computation in Python, matplotlib and seaborn packages for data visualizations.
- Loading the csv data and presenting it into a Dataframe using pd.DataFrame().

Step1: Displays the size of dataset



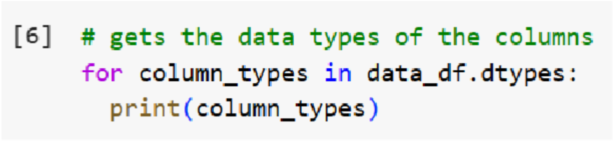
```
[4]  # returns the dimension of the dataset
     num_rows, num_cols = data_df.shape
     print("Number of rows:", num_rows)
     print("Number of columns:", num_cols)
```

Step2: Displays the columns in dataset



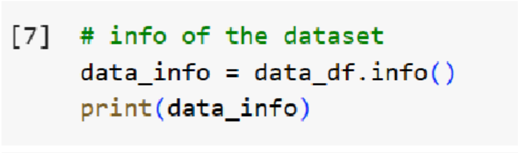
```
[5]  # columns in the taxi dataset
     for column in data_df.columns:
         print(column)
```

Step3: Displays the datatypes of variables



```
[6]  # gets the data types of the columns
     for column_types in data_df.dtypes:
       print(column_types)
```

Step4: Displays the basic info of dataset



```
[7]  # info of the dataset
     data_info = data_df.info()
     print(data_info)
```

The commonly used methods in pandas DataFrame for exploring and understanding the structure and content of a dataset.

- **df.shape-** returns a tuple representing the number of rows and columns.
- **df.columns-** returns the column labels of DataFrame.
- **df.dtypes-** returns the datatypes of each column in DataFrame.
- **df.info-** returns the concise summary of the DataFrame, including information about the index, column names, non-null values, and data types of each column, as well as memory usage.

Description of Columns:

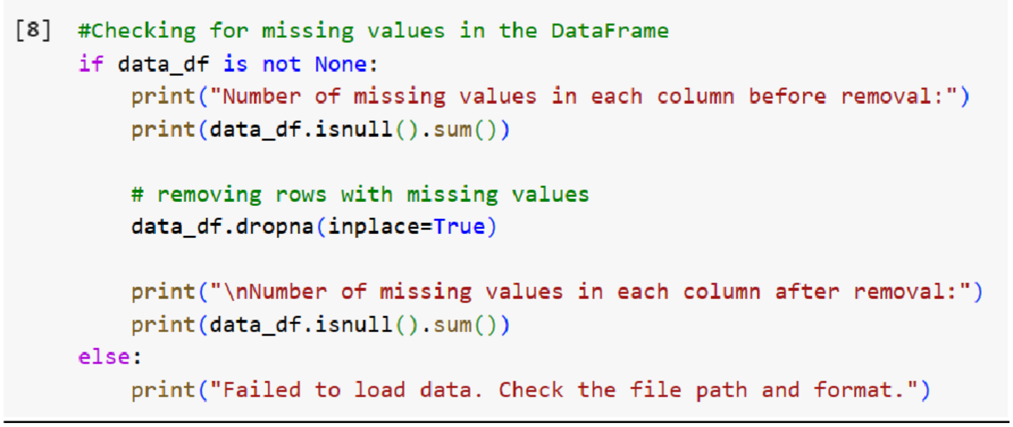
Relevant variables considered in this Analysis:

- **tpep_pickup_datetime**- The date and time when the meter was engaged.
- **tpep_dropoff_datetime**- The date and time when the meter was disengaged.
- **passenger_count**- The number of passengers in the vehicle.
- **trip_distance**- The elapsed trip distance in miles reported by the taximeter.
- **PULocationID**- The locationID of Taxi Zone in which the taximeter was engaged.
- **DOLocationID**- The locationID of Taxi Zone in which the taximeter was disengaged.
- **fare_amount**- The time and distance fare calculated by the meter.
- **payment_type**- A numeric code signifying how the passenger paid for the trip.

Dataset preparation as needed:

Removing missing values:

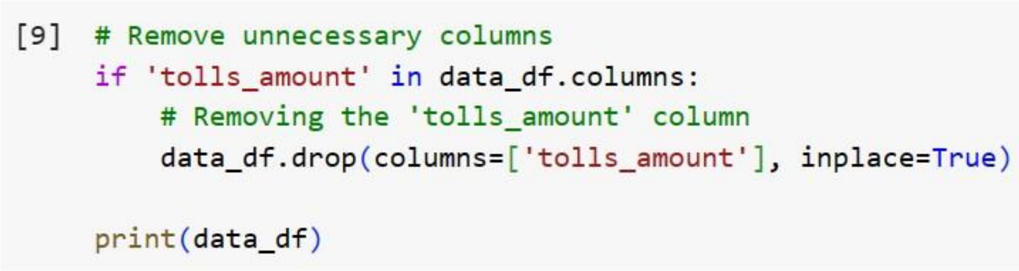
Checking for missing values using isnull(), if the DataFrame exists, it prints the number of missing values in each column before any removal operation. It then removes rows containing missing values from the DataFrame using the dropna() method with inplace=True.



```
[8] #Checking for missing values in the DataFrame
    if data_df is not None:
        print("Number of missing values in each column before removal:")
        print(data_df.isnull().sum())

        # removing rows with missing values
        data_df.dropna(inplace=True)

        print("\nNumber of missing values in each column after removal:")
        print(data_df.isnull().sum())
    else:
        print("Failed to load data. Check the file path and format.")
```

Removing unnecessary variables:

# The below returns the DataFrame without the variable tolls_amount. Here, we have removed the unwanted column using the df.columns and df.drop().

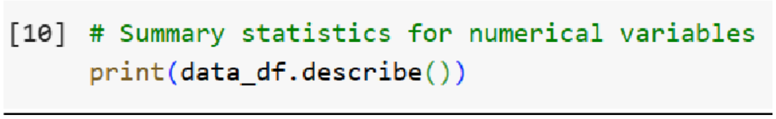


```
[9] # Remove unnecessary columns
    if 'tolls_amount' in data_df.columns:
        # Removing the 'tolls_amount' column
        data_df.drop(columns=['tolls_amount'], inplace=True)

    print(data_df)
```

Initial statistical descriptive statistics:

Generates summary statistics for numerical variables:

# This generates descriptive statistics that summarize the central tendency, dispersion, and shape of a dataset's distribution. And calculates these statistics for numerical columns by default. Calculates some basic statistics such as count (Number of non-null values in the column), mean (Mean or average value of the column), std (standard deviation), min (Minimum value in the column), 25% (25th percentile or first quartile), 50% (50th percentile or median), 75% (75th percentile or third quartile), max (Maximum value in the column).

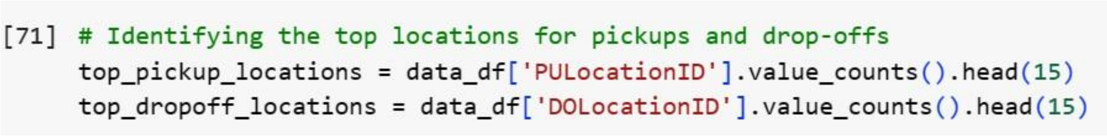


```
[10] # Summary statistics for numerical variables
     print(data_df.describe())
```

Result: For the above descriptive statistics, the result comprehensive overview of the numerical variables in the DataFrame, helping to understand their distribution, central tendency, and variability. They help us identify anomalies, patterns, or issues if any in the data.

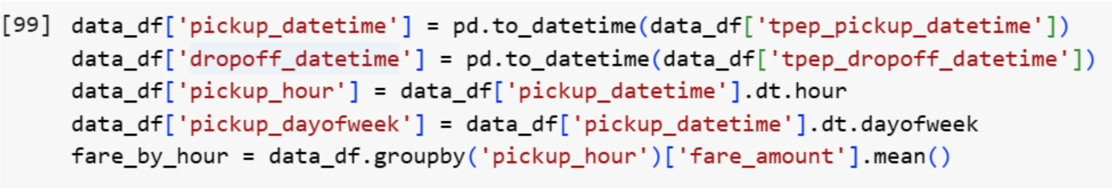
Initial visualization to help understand the data:

# Before visualizing the data, we identify the top_pickup_locations and top_dropoff_locations based on the PULocationID and DOLocationID. Below returns the top 15 pickup and drop-off locations using the counts().



```
[71] # Identifying the top locations for pickups and drop-offs
     top_pickup_locations = data_df['PULocationID'].value_counts().head(15)
     top_dropoff_locations = data_df['DOLocationID'].value_counts().head(15)
```

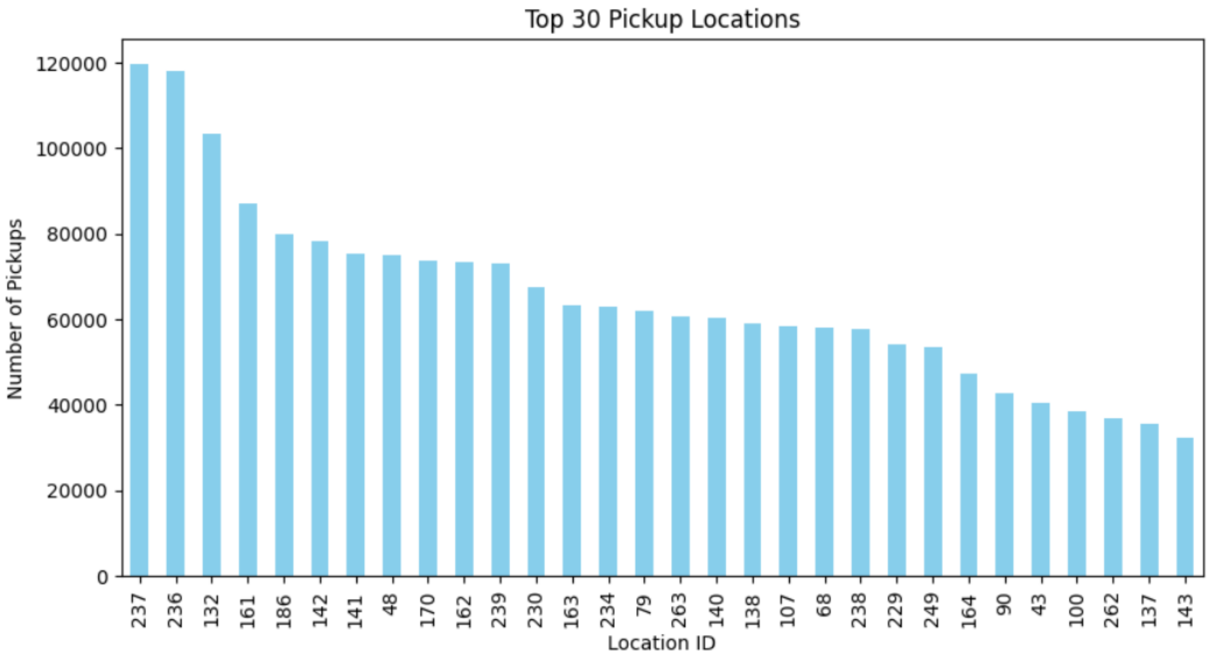
# This code below prepares the DataFrame for further analysis by extracting datetime components and then calculates the average fare amount for each hour of the day. This information could be used to analyze the variation in fare amounts throughout the day.
Converts the datatype of ‘**tpep_pickup_datetime**’, ‘**tpep_dropoff_datetime**’ from object to datetime and further uses to calculate the pickup_hour and pickup_dayofweek accordingly.



```
[99] data_df['pickup_datetime'] = pd.to_datetime(data_df['tpep_pickup_datetime'])
     data_df['dropoff_datetime'] = pd.to_datetime(data_df['tpep_dropoff_datetime'])
     data_df['pickup_hour'] = data_df['pickup_datetime'].dt.hour
     data_df['pickup_dayofweek'] = data_df['pickup_datetime'].dt.dayofweek
     fare_by_hour = data_df.groupby('pickup_hour')['fare_amount'].mean()
```

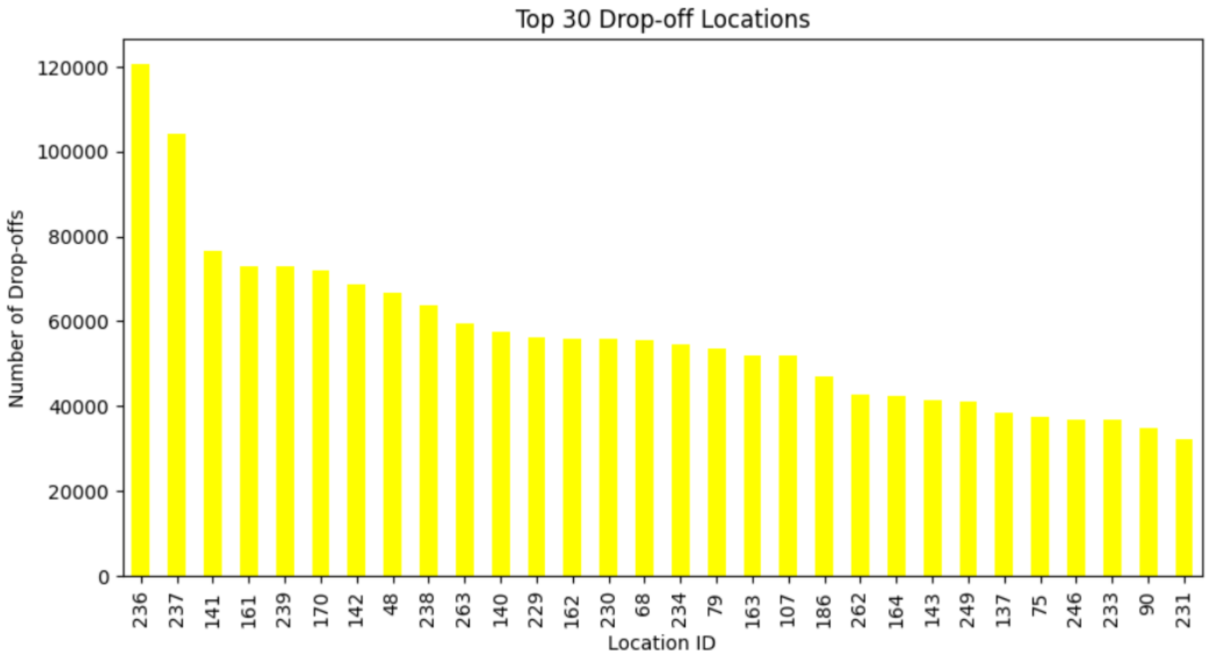
Plot for Number of pickups based on the LocationID (Top 30 Pickup Locations):

# The bar plot visualizes the patterns of the NYC taxi data for the pickup locations. It identifies the top 30 location patterns of number of pickups from a particular location (POLocationID). As we gain insights from the plot, we can say that the LocationID: 237, 236 have highest frequency for number of pickups which means that the locations are the busiest compared to other locations. And LocationID: 143 has the frequency for lowest number of pickups which aren’t that busy. The other locations have an accurate form of pattern for pickup which can be considered as slightly busy. From identifying these patterns, we can generally have a transportation plan, strategies to optimize taxi services and address customer services accordingly.



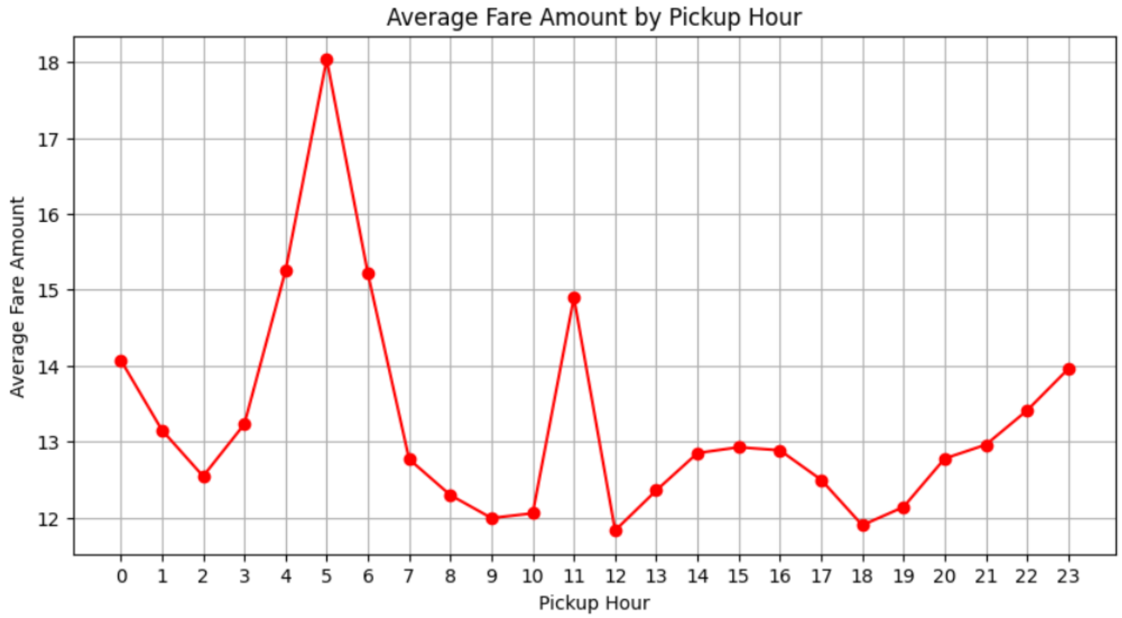
Plot for Number of drop-offs based on the LocationID (Top 30 Pickup Locations):

# The bar plot visualizes the patterns of the NYC taxi data for the drop-off locations. It identifies the top 30 location patterns of number of drop-offs to a particular location (DOLocationID). As we gain insights from the plot, we can say that the LocationID: 236, 237 have highest frequency for number of drop-offs which means that the locations are the busiest compared to other locations. And LocationID: 231 has the frequency for lowest number of drop-offs which aren't that busy. The other locations have an accurate form of pattern for pickup which can be considered as slightly busy. From identifying these patterns, we can generally have a transportation plan, strategies to optimize taxi services and address customer services accordingly.



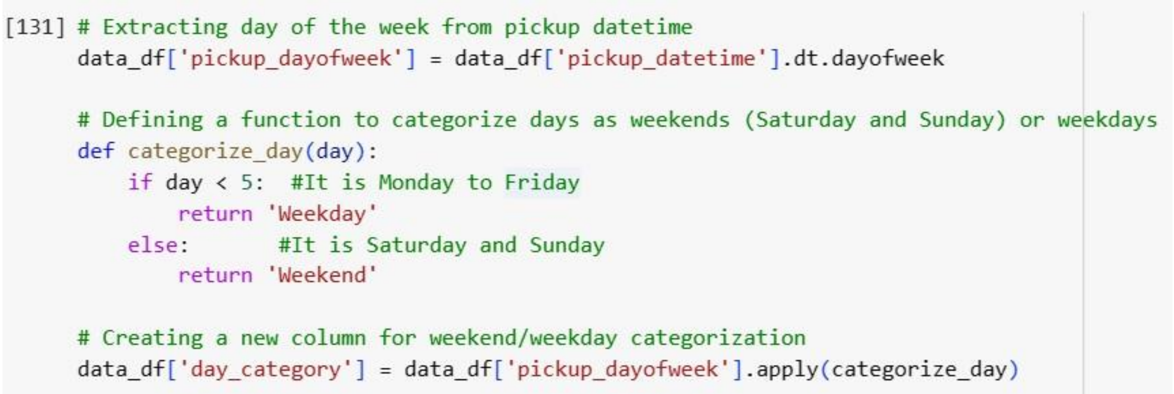
Average Fare amount by Pickup Hour:

# The plot shows how the average fare amount varies throughout the day based on the pickup hour. Each point on the plot represents the average fare amount for trips that started during a specific hour of the day. We can interpret from the above as higher fares during peak hours (5-8 hour) or late-night surges (23 hour). This visualization helps in understanding the relationship between pickup time and fare amounts, which can be useful for various analyses and decision-making processes. We gain insight as such people get rides early in themorning which is peak the average fare amount is high, and again at 11 the fare amount is little moderate. Compared to early morning and afternoon scenes, the average fare amount at late night 11PM, is less.



Categorizing the pickup_datetime variable to calculate the Weekdays and Weekends:

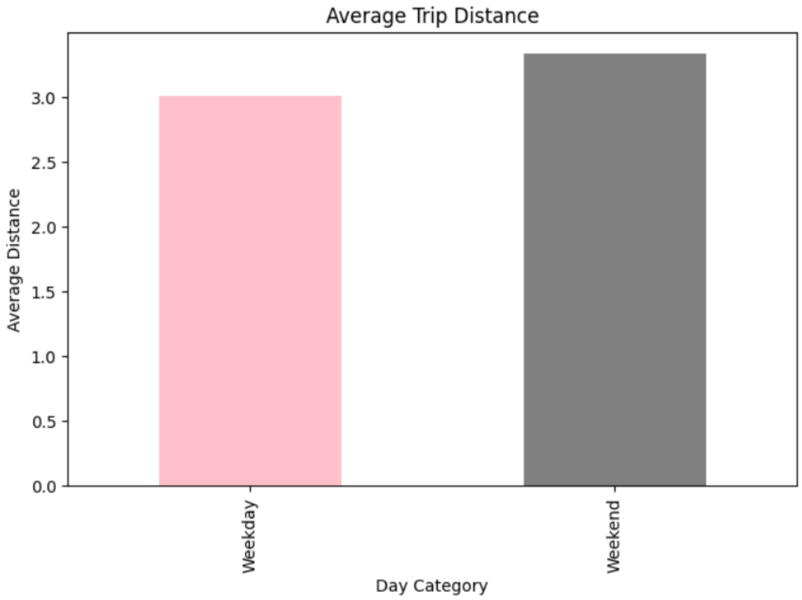
# The below code chunk returns the dayofweek from the pickup_datetime converting into weekdays and weekends. If the day is less than 5 (i.e., 0-4) it considers as Weekdays and greater than 4 considers it as Weekends (Saturday and Sunday).



```
[131] # Extracting day of the week from pickup datetime
      data_df['pickup_dayofweek'] = data_df['pickup_datetime'].dt.dayofweek

      # Defining a function to categorize days as weekends (Saturday and Sunday) or weekdays
      def categorize_day(day):
          if day < 5:  #It is Monday to Friday
              return 'Weekday'
          else:       #It is Saturday and Sunday
              return 'Weekend'

      # Creating a new column for weekend/weekday categorization
      data_df['day_category'] = data_df['pickup_dayofweek'].apply(categorize_day)
```

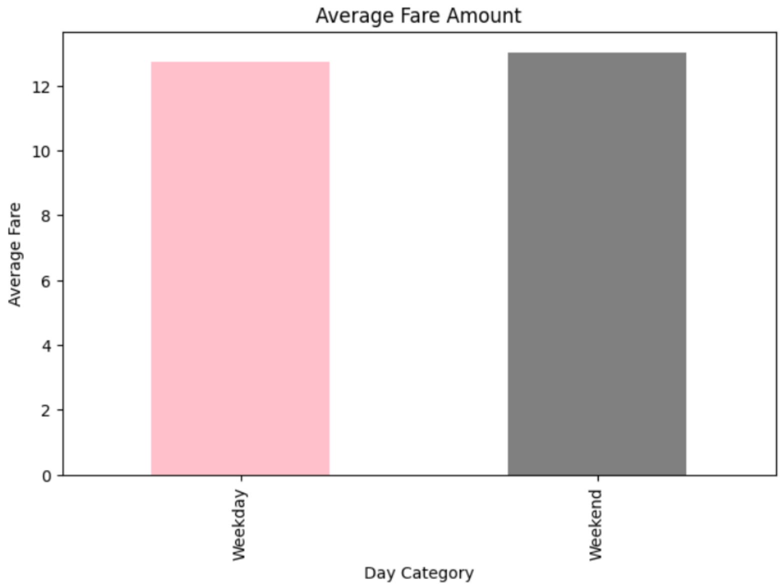
Average Trip distance variation between Weekdays and Weekends:

# The plot visualizes the average trip distance for different categories of days. Each bar represents the average trip distance for a specific category of days, such as weekdays and weekends. We can interpret that trips tend to be longer on weekends compared to weekdays. By the insights we can say that more taxis get booked on the weekends and the trip distance is more compared to on the weekdays.



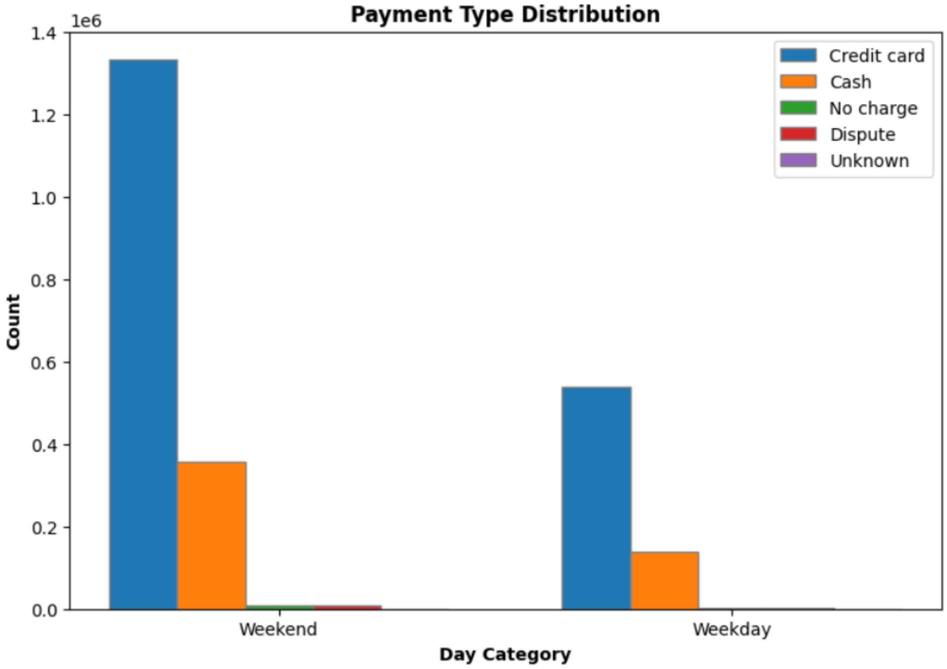
Average Fare amount variation on Weekdays and Weekends:

# The plot visualizes the average fare amount for different categories of days. Each bar represents the average fare amount for a specific category of days, such as weekdays and weekends. We can interpret that fares tend to be higher on weekends compared to weekdays. From the below categorization we can visualize that based on the weekdays and weekend the average fare amount is slightly higher on the weekend compared with weekdays.



Payment type distribution variation on Weekdays and Weekends:

# The bar plot visualizes the distribution of payment types for different categories of days. Each bar represents the total count of trips for a specific day category, and within each bar, the segments represent the count of trips for each payment type. We can interpret that credit card payments and cash payments dominate on weekends, while both the payment types are less on weekdays. By this we can say that the payment types such as Credit card, Cash, no cash are generally identified as a pattern type for the payment stakeholders enabling the taxi passengers to optimize the transactions which is easy and accessible everywhere. While dispute and unknown are rarely used by the customers.



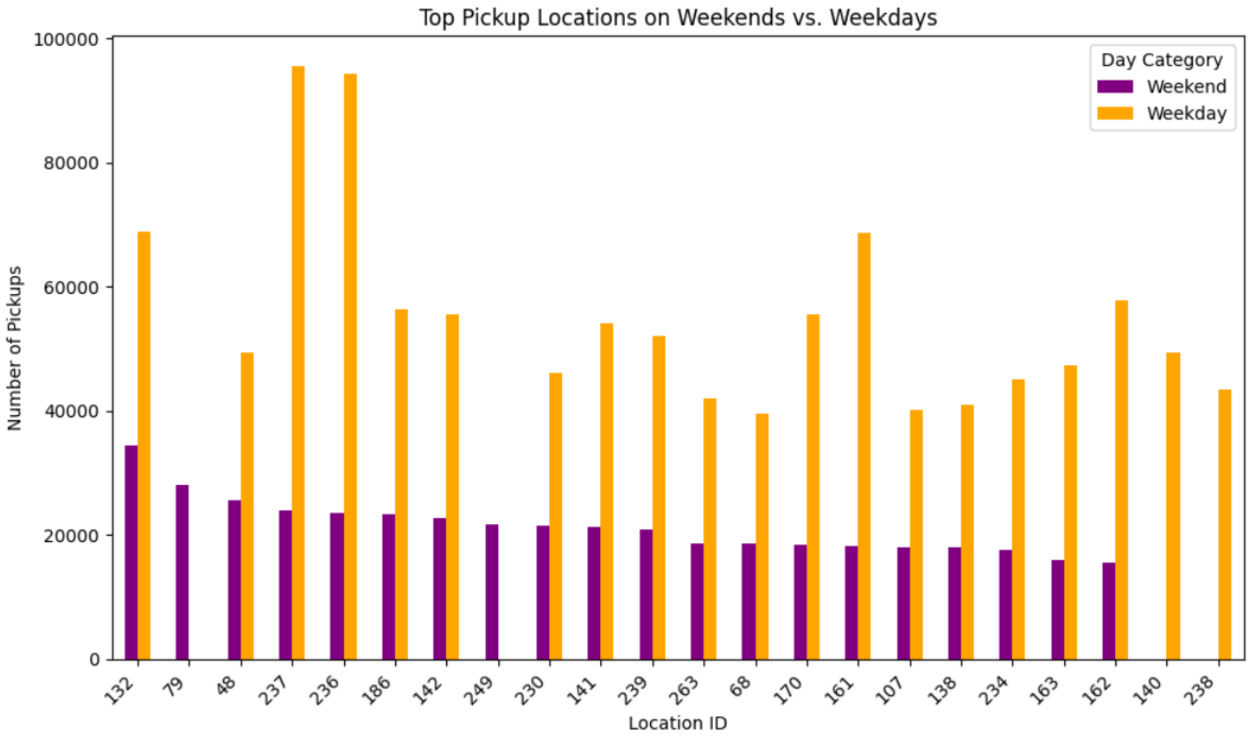
Pickup Location variations on Weekdays and Weekends:

# The bar plot visualizes the taxi data pickup locations on Weekdays and Weekends. Based on the pickup locationID (POLocationID), the number of pickups varies accordingly.

Weekdays: From the result plot we can analyze that the taxi pickups are more on weekdays for the Location ID: 237 and 236 compared to no pickups for the LocationID: 249 on Weekday. LocationID: 132 and 161 are slightly busy for the pickups on weekdays.

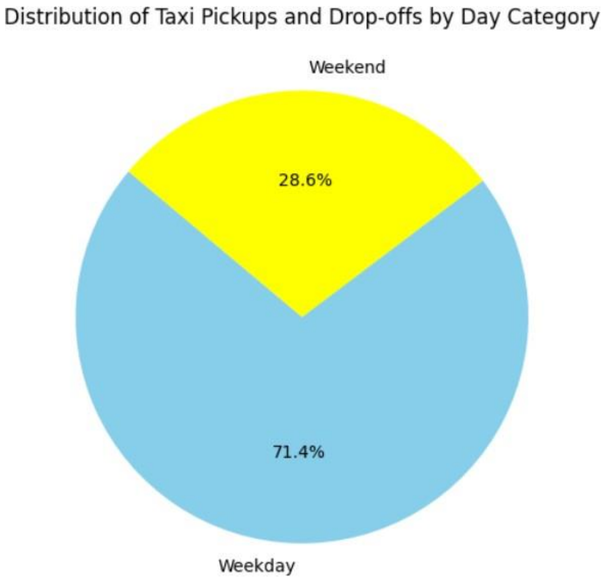
Weekends: From the result plot we can say that for the Weekends the taxi pickups are more for the LocationID: 132 and 79 compared with no pickups for the LocationID: 140 and 238.

Differentiating both Weekdays and Weekends we conclude that the taxis are busier on the Weekdays and pickups are more on weekdays compared to Weekends. This analysis can help taxi companies to produce more taxis on the weekdays and provide good customer service.



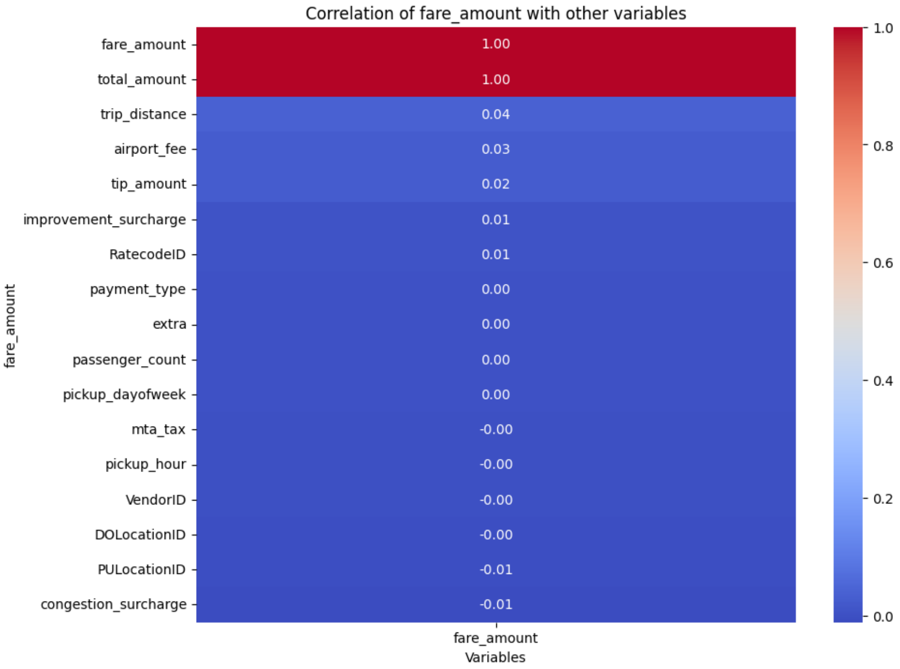
Distribution of Taxi pickups and drop-offs by Day Category:

# The below pie chart visualizes the distribution of taxi pickups and drop-off by day wise category. The chart displays the proportions of the Weekdays and Weekends to the total, providing a clear snap of the most pickups and drop-offs. We analyze that the pickups and drop-offs on Weekdays dominate the pickups and drop-offs on Weekends.



Correlation Map between the features:

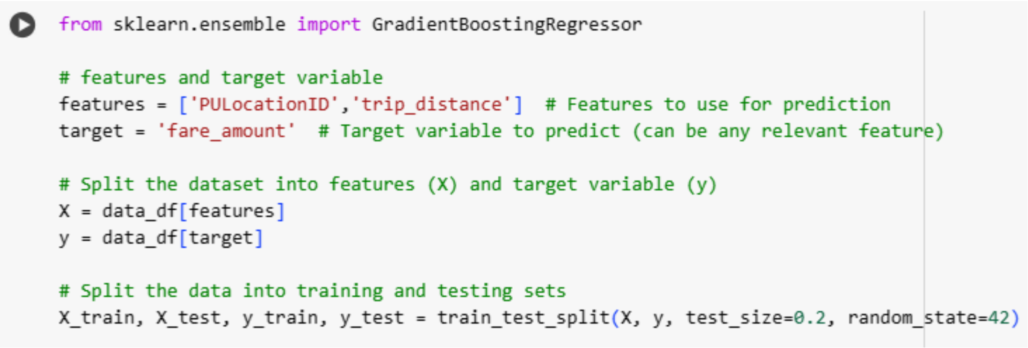
The correlation analysis reveals insights into factors influencing taxi fare amounts. Notably, fare amount shows minimal correlation with trip distance (0.039), airport fee (0.027), tip amount (0.024), and improvement surcharge (0.007). This suggests that while these factors may have some influence, their impact on fare amount is relatively weak. However, it's essential to consider them collectively when determining fare structures and pricing strategies. Understanding these correlations aids in optimizing fare calculations and providing transparent pricing mechanisms in urban transportation systems.



Prepare datasets for modeling (training sets, validation tests, test sets, etc.)

Datasets preparation for Gradient Boosting Regression (Implementation-1):

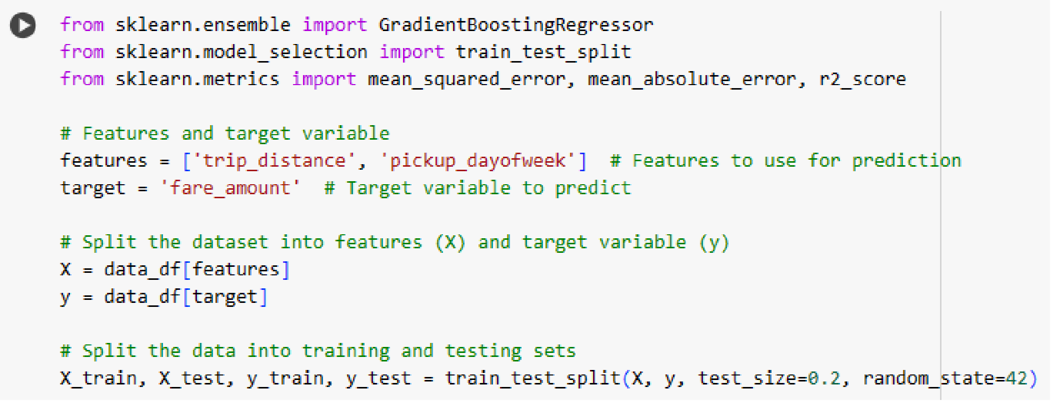
# Datasets are prepared by considering some relevant features according to the research question. Feature variables: **PULocationID, trip_distance**; Target variables: **fare_amount**. Then the train and test datasets are split into as train=80% and test=20%.



```
from sklearn.ensemble import GradientBoostingRegressor

# features and target variable
features = ['PULocationID','trip_distance']  # Features to use for prediction
target = 'fare_amount'  # Target variable to predict (can be any relevant feature)

# Split the dataset into features (X) and target variable (y)
X = data_df[features]
y = data_df[target]

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Datasets preparation for Gradient Boosting Regression (Implementation-2):

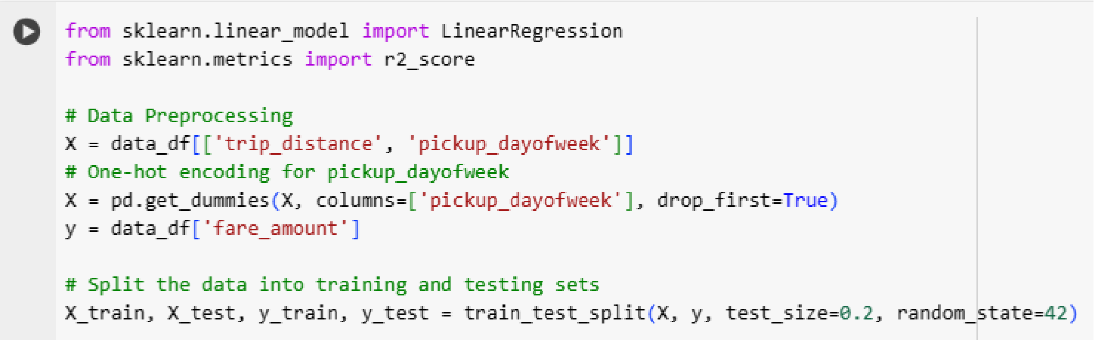
# Datasets are prepared by considering some relevant features according to the research question. Feature variables: **trip_distance, pickup_dayofweek**; Target variables: **fare_amount**. Then the train and test datasets aresplit into as train=80% and test=20%.



```
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

# Features and target variable
features = ['trip_distance', 'pickup_dayofweek']  # Features to use for prediction
target = 'fare_amount'  # Target variable to predict

# Split the dataset into features (X) and target variable (y)
X = data_df[features]
y = data_df[target]

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Datasets preparation for Linear Regression (Implementation-2):

# Datasets are prepared by considering some relevant features according to the research question. Feature variables: **trip_distance, pickup_dayofweek**; Target variables: **fare_amount**. Then the train and test datasets are split into as train=80% and test=20%.



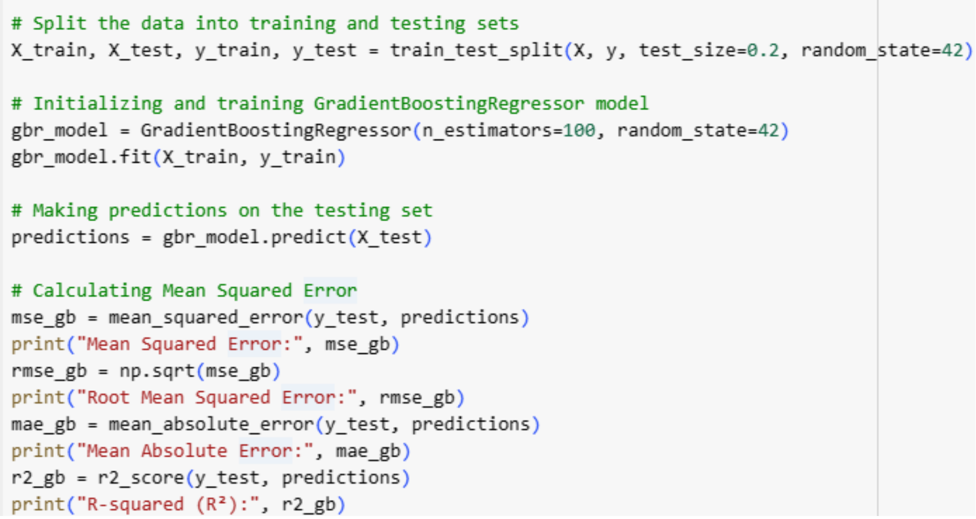
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score

# Data Preprocessing
X = data_df[['trip_distance', 'pickup_dayofweek']]
# One-hot encoding for pickup_dayofweek
X = pd.get_dummies(X, columns=['pickup_dayofweek'], drop_first=True)
y = data_df['fare_amount']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Perform initial modeling:

# We have performed Gradient Boosting Modeling which improves prediction accuracy by combining multiple weak predictors into a strong model, especially effective for complex, non-linear relationships within data likethe Yellow Taxi Trip Records. Effective modeling with complex multiple variables without high risk.

Gradient Boosting Modeling (Implementation-1):

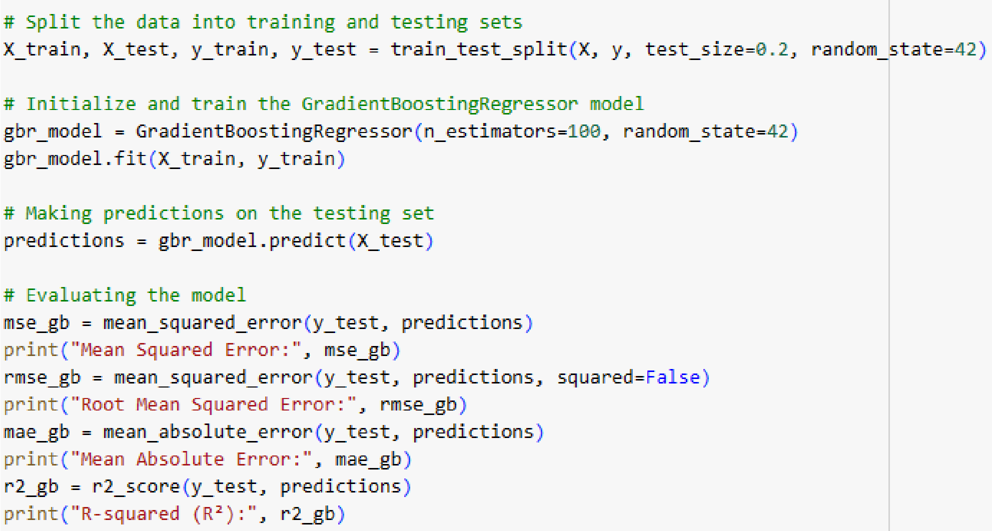
# For the below modeling, we have trained the model using the GradientBoostingRegressor function for feature variables **PULocationID, trip_distance** and fittedthe model using 80% of trained data. And predicted the model using 20% of test data. And calculated the Mean Squared Error (MSE), RMSE (Root Mean Squared Error), MAE (Mean Absolute Error), R-Squared accordingly on the predicted value.



```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initializing and training GradientBoostingRegressor model
gbr_model = GradientBoostingRegressor(n_estimators=100, random_state=42)
gbr_model.fit(X_train, y_train)

# Making predictions on the testing set
predictions = gbr_model.predict(X_test)

# Calculating Mean Squared Error
mse_gb = mean_squared_error(y_test, predictions)
print("Mean Squared Error:", mse_gb)
rmse_gb = np.sqrt(mse_gb)
print("Root Mean Squared Error:", rmse_gb)
mae_gb = mean_absolute_error(y_test, predictions)
print("Mean Absolute Error:", mae_gb)
r2_gb = r2_score(y_test, predictions)
print("R-squared (R²):", r2_gb)
```

Gradient Boosting Modeling (Implementation-2):

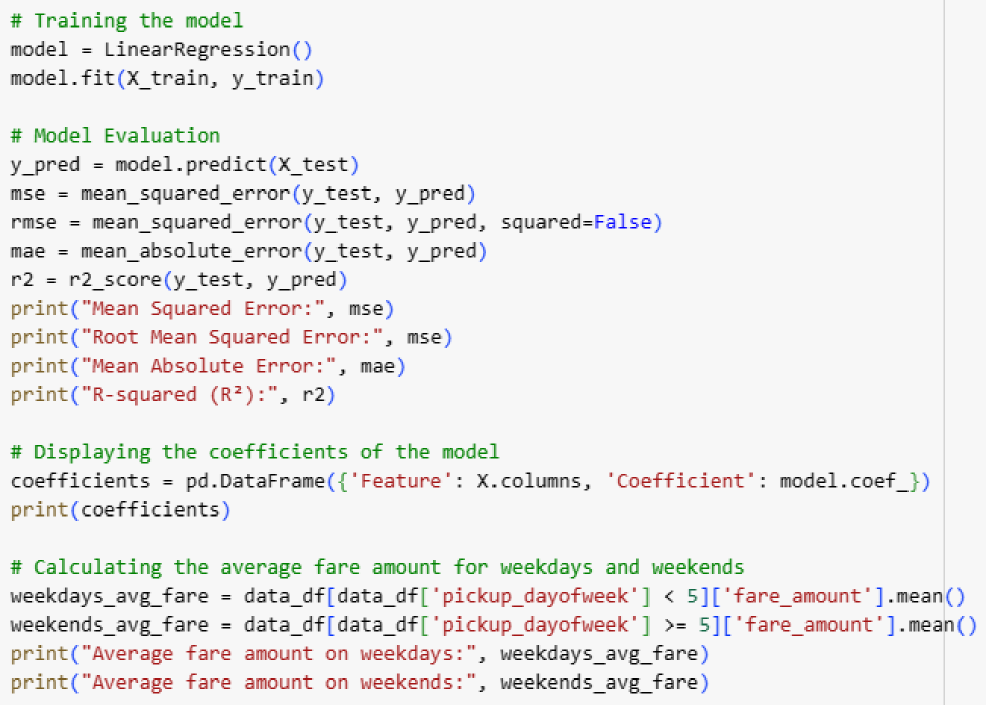
# For the below modeling, we have trained the model using the GradientBoostingRegressor function for feature variables **trip_distance, pickup_dayofweek** and fitted the model using 80% of trained data. And predicted the model using 20% of test data. And calculated the Mean Squared Error (MSE), RMSE (Root Mean Squared Error), MAE (Mean Absolute Error), R-Squared accordingly on the predicted value.



```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize and train the GradientBoostingRegressor model
gbr_model = GradientBoostingRegressor(n_estimators=100, random_state=42)
gbr_model.fit(X_train, y_train)

# Making predictions on the testing set
predictions = gbr_model.predict(X_test)

# Evaluating the model
mse_gb = mean_squared_error(y_test, predictions)
print("Mean Squared Error:", mse_gb)
rmse_gb = mean_squared_error(y_test, predictions, squared=False)
print("Root Mean Squared Error:", rmse_gb)
mae_gb = mean_absolute_error(y_test, predictions)
print("Mean Absolute Error:", mae_gb)
r2_gb = r2_score(y_test, predictions)
print("R-squared (R²):", r2_gb)
```

Linear Regression Modeling (Implementation-2):

# For the Linear Regression modeling below, we have trained the model using LinearRegression function for feature variables **trip_distance, pickup_dayofweek** and fitted the model using 80% of trained data. And predicted the model using 20% of test data. And calculated the Mean Squared Error (MSE), RMSE (Root Mean Squared Error), MAE (Mean Absolute Error), R-Squared accordingly on the predicted value.



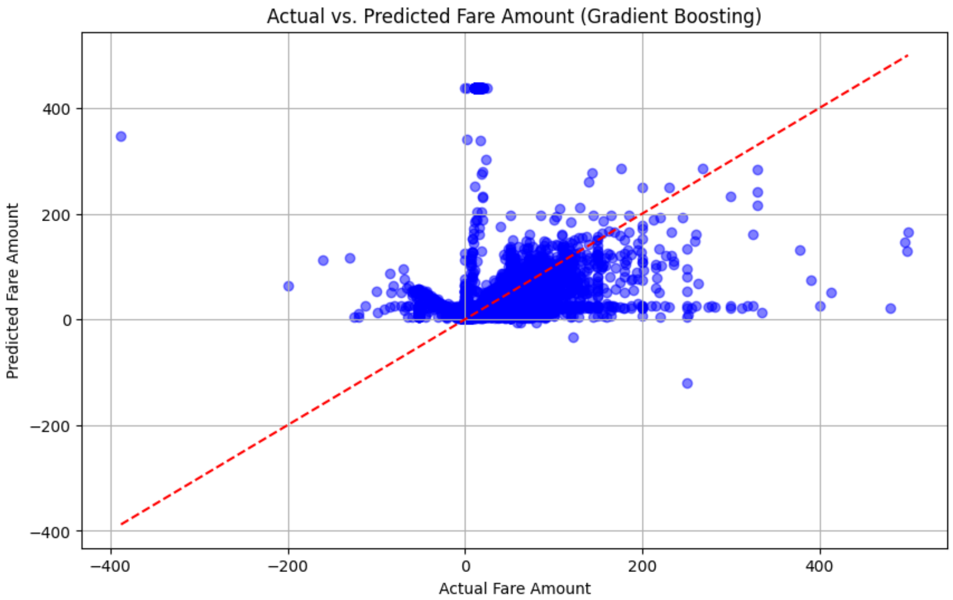
```
# Training the model
model = LinearRegression()
model.fit(X_train, y_train)

# Model Evaluation
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
rmse = mean_squared_error(y_test, y_pred, squared=False)
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print("Mean Squared Error:", mse)
print("Root Mean Squared Error:", mse)
print("Mean Absolute Error:", mae)
print("R-squared (R²):", r2)

# Displaying the coefficients of the model
coefficients = pd.DataFrame({'Feature': X.columns, 'Coefficient': model.coef_})
print(coefficients)

# Calculating the average fare amount for weekdays and weekends
weekdays_avg_fare = data_df[data_df['pickup_dayofweek'] < 5]['fare_amount'].mean()
weekends_avg_fare = data_df[data_df['pickup_dayofweek'] >= 5]['fare_amount'].mean()
print("Average fare amount on weekdays:", weekdays_avg_fare)
print("Average fare amount on weekends:", weekends_avg_fare)
```

Display visualizations describe the success of the modeling effort:

Visualization for Predicted Trip distance (Implementation-1):

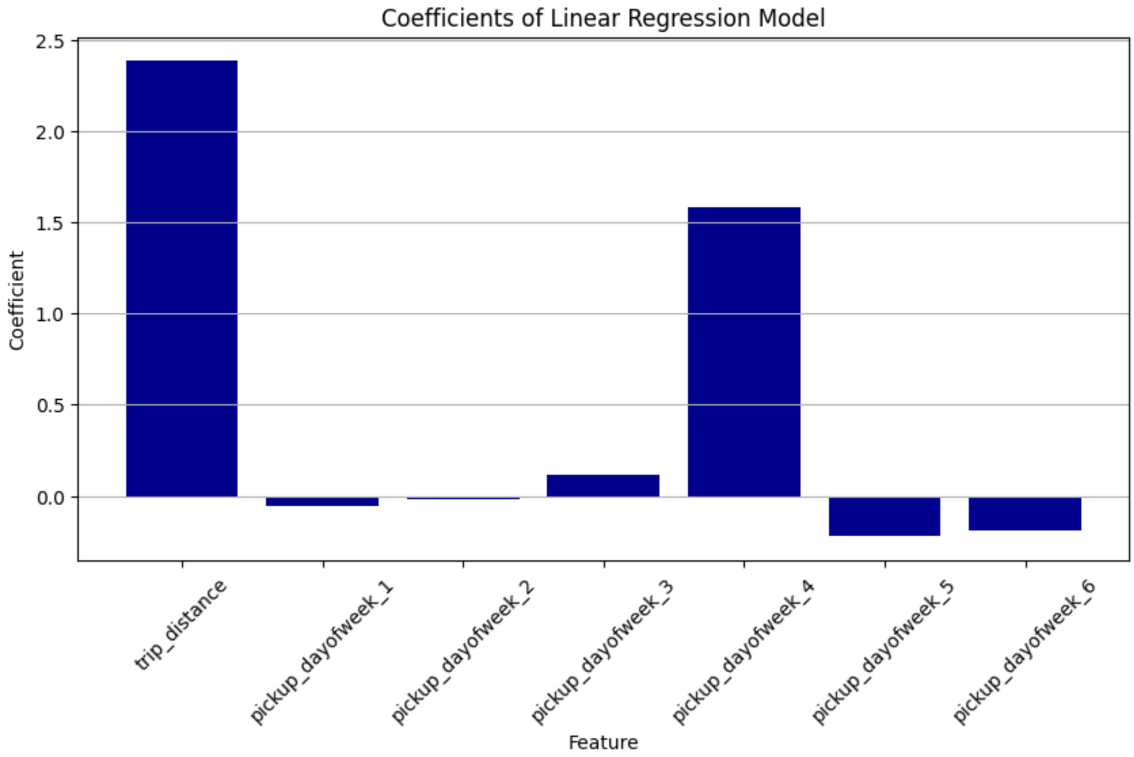
# The below residual plot visualizes the actual and predicted fare amount for the modeling technique, prediction on the test data. Residuals near zero: points near y=0 indicate that the difference between the actual and predicted values (residuals) is minimal. When residuals are close to zero, it means that the model'spredictions are very accurate.

Consistently seeing points clustered near the dashed line across the range of predicted fare amount indicatesthat the model is performing consistently well across different prediction scenarios.



Visualization plot Coefficient of Linear Regression Model (Implementation-2):

# The below bar plot visualizes the coefficients of linear regression, each coefficient quantifies the expected change in the target variable resulting from a one-unit increase in a corresponding feature, holding all other features constant. For example, a coefficient of 0.5 for trip distance suggests that each additional unit of distance increases the target variable's predicted value by 0.5. The signs of these coefficients are crucial for understanding the direction of influence; positive coefficients indicate an increase, while negative coefficientssuggest a decrease in the target variable with respect to the feature.



Demonstrate use of the model to predict results and metrics in the test set:

Root Mean Squared Error (RMSE): RMSE measures the average magnitude of the errors between predictedand actual values. It represents the standard deviation of the residuals. In this context:
If RMSE is close to 0, it indicates that the model's predictions are very close to the actual values on average.If RMSE is relatively low, it implies that the model is making accurate predictions with small errors.
If RMSE is high, it suggests that the model's predictions have large errors compared to the actual values.

Mean Absolute Error (MAE): MAE measures the average absolute difference between predicted and actual values. It provides a more intuitive understanding of the error magnitude compared to RMSE. In this context: If MAE is close to 0, it indicates that the model's predictions are very close to the actual values on average.
If MAE is relatively low, it implies that the model is making accurate predictions with small errors.
If MAE is high, it suggests that the model's predictions have large errors compared to the actual values.

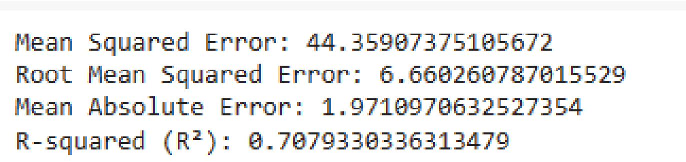
R-squared (R^2): R-squared represents the proportion of the variance in the dependent variable (target) that is predictable from the independent variables (features). It ranges from 0 to 1, where 1 indicates a perfect fit and 0 indicates no linear relationship between the variables. In this context:

If R² is close to 1, it suggests that the model explains a large portion of the variance in the target variable, indicating a good fit.
If R² is relatively low, it implies that the model does not explain much of the variance in the target variable, indicating a poor fit.
If R² is negative, it suggests that the model performs worse than a horizontal line (the mean of the target variable).

Model quality metrics:

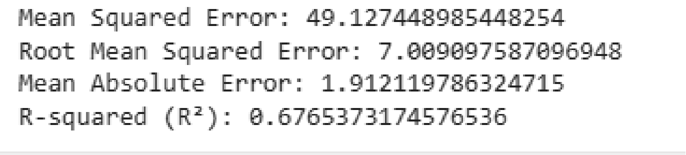
Below are the results for model metrics predictions for Gradient Boosting, and Linear Regression.

Results for Gradient Boosting Regression Modeling (Implementation-1):



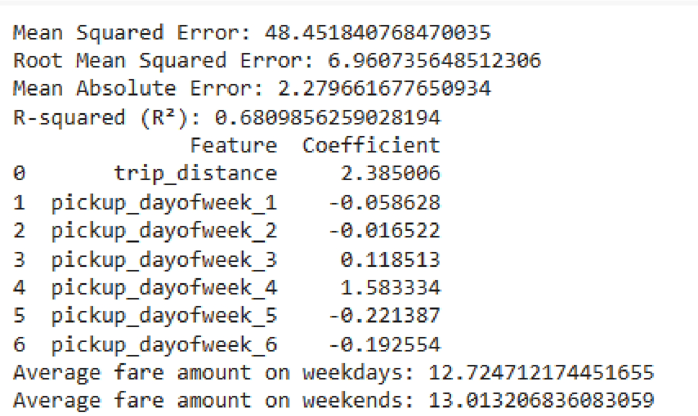
```
Mean Squared Error: 44.35907375105672
Root Mean Squared Error: 6.660260787015529
Mean Absolute Error: 1.9710970632527354
R-squared (R²): 0.7079330336313479
```

Results for Gradient Boosting Regression Modeling (Implementation-2):



```
Mean Squared Error: 49.127448985448254
Root Mean Squared Error: 7.009097587096948
Mean Absolute Error: 1.912119786324715
R-squared (R²): 0.6765373174576536
```

Results for Linear Regression Modeling (Implementation-2):



```
Mean Squared Error: 48.451840768470035
Root Mean Squared Error: 6.960735648512306
Mean Absolute Error: 2.279661677650934
R-squared (R²): 0.6809856259028194
              Feature  Coefficient
0       trip_distance     2.385006
1  pickup_dayofweek_1    -0.058628
2  pickup_dayofweek_2    -0.016522
3  pickup_dayofweek_3     0.118513
4  pickup_dayofweek_4     1.583334
5  pickup_dayofweek_5    -0.221387
6  pickup_dayofweek_6    -0.192554
Average fare amount on weekdays: 12.724712174451655
Average fare amount on weekends: 13.013206836083059
```

Results of final modeling efforts:

Comparing the metrics for Gradient Boosting and Linear Regression Modeling (Implementation-2):

# The plot compares the performance of two machine learning models, Gradient Boosting and Linear Regression, on 4 different metrics: mean squared error (MSE), root mean squared error (RMSE), mean absolute error (MAE), and R-squared (R2).

Mean Squared Error (MSE):

Overall, Linear Regression Mean Squared Error (MSE) is smaller than Gradient Boosting.

This implies that, on average, Linear Regression forecasts are more accurate than Gradient model which has lower MSE value.

Root Mean Squared Error (RMSE):

Linear Regression has a lower RMSE than Gradient Boosting in all cases, which is similar to MSE. This indicates that forecasts generated by Linear Regression Modeling approach are moreaccurate.

Mean Absolute Error (MAE):

The MAE results are conclusive. For the first instance, MAE for Gradient Boosting is 1.91.

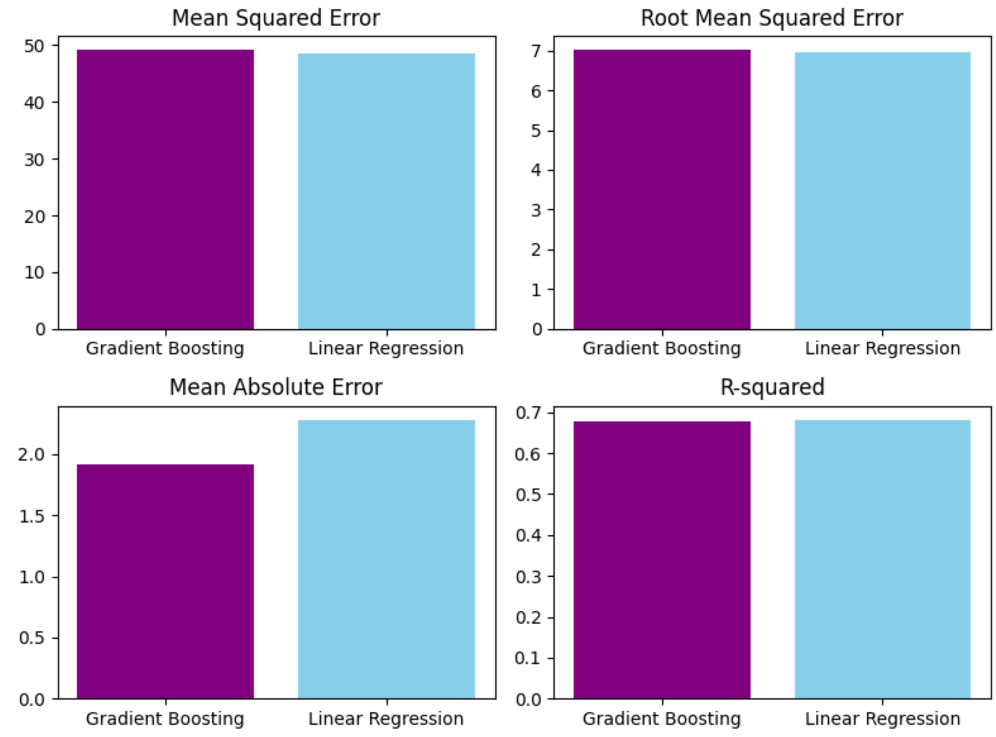
MAE closer to 0, it indicates that the model's predictions are very close to the actual values on average.

Indicating that Gradient Boosting modeling is accurate compared to Linera Regression.

R-Squared (R2):

R-squared is a metric that measures how well the variation in your actual data is explained by the predictionsof your model.

A higher R-squared value generally indicates a better fit. Below indicates that both Linear Regression and Gradient Boosting modeling have approx. 68% which is accurate.



<u>Discuss overall results of the experiment:</u>

Overall Results for Gradient Boosting (Implementation-1):

# The Gradient Boosting Regressor and Linear Regressor models are essential for assessing how effectively each model forecasts the amounts of taxi fares according to the information and characteristics.

Gradient Boosting Regressor Model (Implementation-1):

Mean Squared Error (MSE) of 44.36: This metric tells us that on average, the square of the errors (the difference between the predicted fare amounts and the actual fare amounts) is 44.36. This square helps to highlight larger errors more so than smaller ones because squaring larger differences results in much higher values.

Root Mean Squared Error (RMSE) of 6.66: By taking the square root of the MSE, we get the RMSE. An RMSE of 6.66 suggests that typically, the model's predictions are off by about $6.66 from the actual fare. This metric is in the same units as the fare itself, making it a more interpretable measure of prediction error.

Mean Absolute Error (MAE) of 1.97: This indicates that on average, the absolute differences between the predicted fares and actual fares are $1.97. Unlike MSE or RMSE, MAE gives a direct indication of average error magnitude without emphasizing larger errors disproportionately.

R-squared (R^2) of 0.71: This value means that about 71% of the variability in the actual fare amounts can be explained by the model's inputs. A higher R^2 value typically indicates a better fit of the model to the data.

Overall Results for Gradient and Linear Regression Modeling (Implementation-2):

Mean Squared Error (MSE) of 49.13: This is slightly higher than the MSE for the Gradient Boosting model, suggesting that the Linear Regressor model, on average, squares errors that are larger, indicating possibly poorer performance in some scenarios.

Root Mean Squared Error (RMSE) of 7.01: Like the RMSE of the Gradient Boosting model, but slightly higher, indicating that typical prediction errors are around $7.01, which is slightly worse compared to the previous model.

Mean Absolute Error (MAE) of 1.91: Interestingly, this model has a slightly lower MAE than the Gradient Boosting model, indicating that, on average, the magnitude of its prediction errors is slightly smaller.

R-squared (R^2) of 0.68: This shows that 68% of the fare variability is explained by the model, which is slightly less effective than the Gradient Boosting model.

Suggestions for Future work or improvements:

Future work on the "Exploring Taxi Fare Patterns: Factors Influencing Pricing Dynamics" project could concentrate on improving or adding more pertinent data, such as historical data from local events, to give a deeper understanding of the dynamics affecting taxi fares. Taxi operators may be able to increasing revenue and enhance service efficiency, by creating dynamic pricing models utilizing time-series forecasting and geospatial demand forecasting.

Apart from technological enhancements, the project might gain a great deal from creating tools that are easyto use and carrying out thorough evaluations of the economic and legal implications. Transparency and user engagement might be improved by developing interactive dashboards for planners and taxi operators, as wellas by incorporating fare calculation tools into consumer-facing applications. Market practices would be fair and competitive if the economic effects of dynamic pricing were thoroughly examined, and regulatoryframeworks evaluated.

Conclusions of the experiment:

The analysis of taxi fare structures in New York City highlights the significant roles of pickup location and trip distance on fare amounts. Due to the longer journey distance, taxis picked up from certain locations—especially those further out from downtown or in less crowded zones—typically have higher prices. Furthermore, the fee increases with the length of the trip in order to account for the combined expenses of travel time and distance. This illustrates how important geographical variables are in setting taxi charges, and how careful planning is thus required for urban transportation.

The fare research reveals an interesting finding that average rates are higher on weekends than on weekdays. This finding contradicts the widely held belief that weekday costs would be higher owing to commuter volume. This implies that weekend recreation, nightlife, and fewer public transportation options might increase demand for taxis and raise prices. These findings, which highlight the possible advantages of dynamic price pricing and better transportation alternatives during periods of peak demand, are crucial for city planners and taxi services in order to maximize service efficiency and pricing fairness.

Appendix A:

The analysis of New York City taxi fare trends and the identification of important variables influencing fare dynamics was effectively accomplished by our team. The utilization of modern data analytics and machine learning approaches allowed us to deliver insights that help cab companies and consumers alike enhance theiroperational and travel plans. With improved demand forecasting and dynamic pricing, the findings let serviceproviders make more informed decisions, boosting client travel planning and increasing operational efficiency.

Evaluation metrics such as MSE, RMSE, MAE, and R2 were calculated. Various visualizations were plotted according to the features available.
In general, the initiative made a substantial contribution to improving New York City's urban transportation system's responsiveness and efficiency.

Appendix B: Team Member Contributions

Harsha Vardhan Amara

Data Management and Collection: Harsha was in charge of compiling and overseeing the dataset that was utilized for the project. This required gathering data from open sources, verifying its accuracy, and suitably cleaning and organizing it before it was analyzed.

Model Development: Using Gradient Boosting methods, Harsha took the lead in creating predictive models.To guarantee ideal performance, he put several models into practice, adjusted hyperparameters, and examined model outputs.

Documentation: He also made a substantial contribution to the final report's documentation of the procedures and outcomes, which helped to guarantee the team's conclusions were presented with clarity and thoroughness.

Subhasmita Maharana

Exploratory Data Analysis (EDA): Under the direction of Subhasmita, the data was analyzed to find patterns, trends, and anomalies using statistical tools and visualization approaches. Her work served as a foundationfor further research and the development of models.

Model Evaluation: To guarantee the models' correctness and dependability, she was in charge ofassessing the performance of the created models using a variety of metrics, including MSE, RMSE, and R-squared.

Visualization: Subhasmita produced perceptive visuals for the project presentation that aided in convincingstakeholders of the findings.

Naga Sudha Pavani Tirumalasetty

Geospatial analysis: Pavani concentrated on geospatial analysis, evaluating the effects of pickup and drop-offsites on taxi prices. Her job required her to map data points and evaluate spatial data, both of which were essential for comprehending how geography affected fare variances.

Implementing Additional Machine Learning Models: Pavani played a key role in putting Random Forest and Linear Regression into practice and assessing their efficacy in comparison to baseline models.

Preparing the Report and Presentation: She was essential in putting together the project's final report and creating the presentation's slide deck, making sure that all analytical procedures and conclusions were communicated succinctly.

Google Colab Link:

https://colab.research.google.com/drive/1TLEP-Ee8s_H1M5xDNeYDp8R0o7XpZREn#scrollTo=L0gJ5z-I36ka