Team: **Course: Data Science**

Saran Kumar Sallagundla **Date: 05/08/2024**

Ramya Alluri

Gowthami Pakanati

Tip Estimation in Taxi Trips: Leveraging Fare and Payment Data for Predictive Insights

Statement of Research Question:

How do fare amounts and payment methods affect the tipping behavior of taxi passengers?

Why the answer to this question would be valuable?

The amount that taxi drivers make is significantly influenced by the amount that customers understand the fare and how much to tip them, which also increases customer happiness. Knowing what motivates tipping tendencies allows taxi drivers to target particular clientele—those most inclined to leave larger gratuities based on fare amounts or preferred methods of payment. Financially speaking, it makes sense for drivers to focus on this clientele since it allows them to make more money than they would from other clientele categories while also fostering a feeling of community. Additionally, regulatory authorities can utilize this knowledge to ensure fair compensation practices and promote transparency in tipping procedures across the industry. Overall, the analysis of tipping behavior provides valuable data-driven insights that drive informed decision-making and contribute to the improvement of taxi services and customer satisfaction.

What are the Possible ways to answer the question?

Starting to clean up the data might potentially address outliers and missing numbers. We may better understand distributions and variable connections with the use of exploratory data analysis, or EDA. Perhaps defining new characteristics or changing existing ones, such as temporal and spatiotemporal, could serve as an example. Among other things, feature engineering entails creating new qualities and altering existing ones according to place or period. Once suitable regression models have been selected, the dataset is split into two groups, one for training and the other for testing, based on pre-determined criteria such as mean squared error (MSE). The model is then trained, and evaluated. Once the model meets the desired accuracy, it can be deployed into production. Continuous monitoring ensures its performance remains reliable over time. This approach leverages available data effectively to build a predictive model that accurately estimates the tip amounts, providing valuable insights for taxi companies, passengers, and policymakers in the transportation sector.

Previous attempts to solve the problem and what we have learned from them

Here, the main goal of our earlier research on the subject of tipping behavior in taxis was to ascertain how fare prices, payment options, and tip sizes relate to one another. These studies have shown a positive correlation between the amount of fare charged and tips, indicating that passengers are more likely to give larger sums for tips with higher prices. In addition, the payment method has been shown to influence tip behavior, and credit card users are more likely to tip as opposed to cash payers.

The main findings of previous research have been that passenger tipping depends on how payments are made and how much fare is paid. These studies have helped lay the foundation for further exploration of the problem and guided our research approach.

Discussion about the methods our team used to solve the problem.

To solve the issue, the team combined techniques from data analysis, machine learning, and visualization:

Data Gathering and Preparation: The data cleaning process involves handling missing values, addressing outliers, and removing negative fare amounts. We have performed data transformations and conversions to ensure our data was in a suitable format for analysis.

Exploratory Data Analysis (EDA): We have conducted a comprehensive EDA to find relationships and patterns in the data. Potential relationships were found by statistical analysis of the response variable and predictors.

Modeling Approaches: The team built a linear regression model and a Random Forest model to predict tip amounts based on fare amounts and payment methods. Metrics like Mean Squared Error (MSE), Mean Absolute Error (MSE), and R-squared values were used to evaluate the model performance.

Cross-Validation: In order to evaluate the model's performance and to ensure that it generalizes well across different subsets of data, we used a 10-fold cross-validation.

Discussion of the results our team obtained from the work.

The team found that:

Correlation:
There is a moderate positive correlation between fare amounts and tip amounts, suggesting that higher fare amounts lead to higher tip amounts.

Payment Type:
The type of payment method influences tipping behavior, with credit card payments associated with higher tip amounts.

Model Performance:
Both the linear regression and Random Forest models showed reasonable performance in predicting tip amounts. Slightly better results were obtained using the Random Forest model, which had a higher R-squared score and lower MAE and MSE.

How we can determine if we have usefully answered that question?

Determining the usefulness of answering this question regarding predicting taxi tip amounts involves assessing both the accuracy and precision of the model used.

<u>Accuracy Assessment:</u>

Mean Absolute Error (MAE):
Calculate the average absolute difference between the predicted tip amounts and the actual tip amounts across all taxi trips. A lower MAE indicates higher accuracy.

Mean Squared Error (MSE):
Square the differences between predicted and actual tip amounts, then calculate the average. This penalizes larger errors more heavily than MAE.

Residual Analysis:
Examining the residuals of the model for patterns and ensuring they are randomly distributed indicates that the model captures the underlying relationships in the data effectively.

Precision Evaluation:

Determination Coefficient (R-squared):
Counting the percentage of the target variable's variation (tip amounts) that can be predicted based on the characteristics that were entered. Precision improves with a greater R-squared value.

Confidence Intervals:
Determining the confidence intervals around the predicted tip amounts tells us about the precision. A narrower confidence interval indicates higher precision in predictions.

Prediction Intervals:
Assess the range within which future tip amounts are expected to fall with a certain level of confidence. A narrower prediction interval suggests greater precision in predicting individual tip amounts.

Cross Validation:
To make sure that the model's performance extends well to new data, apply the k-fold cross-validation procedure. This aids in the validation of the model's precision and accuracy across various data subsets.

Comparative Analysis:
By evaluating several machine learning models and algorithms' performances against one another using measures like R-squared, MAE, MSE, and RMSE.Select the model with the lowest error and highest R-squared value for optimal accuracy and precision.
Hence by employing these evaluation methods, we can determine whether we have effectively answered the question of predicting taxi fare amounts A well-performing model with high accuracy and precision may provide trustworthy predictions, enabling all stakeholders to make decisions that will result in the execution of their respective outcomes based on the information found in the data.

How you might deploy the model in the real world to create value for someone?

Implementing a model to predict tip amounts based on fare amount and payment type can create value for various stakeholders in the taxi industry:

Taxi Drivers:
Taxi drivers can use the model to estimate the expected tip amount for a given fare amount and payment type. This information can help them optimize their earnings by providing better service to passengers likely to give higher tips or by strategically accepting or declining fares based on potential tip amounts.

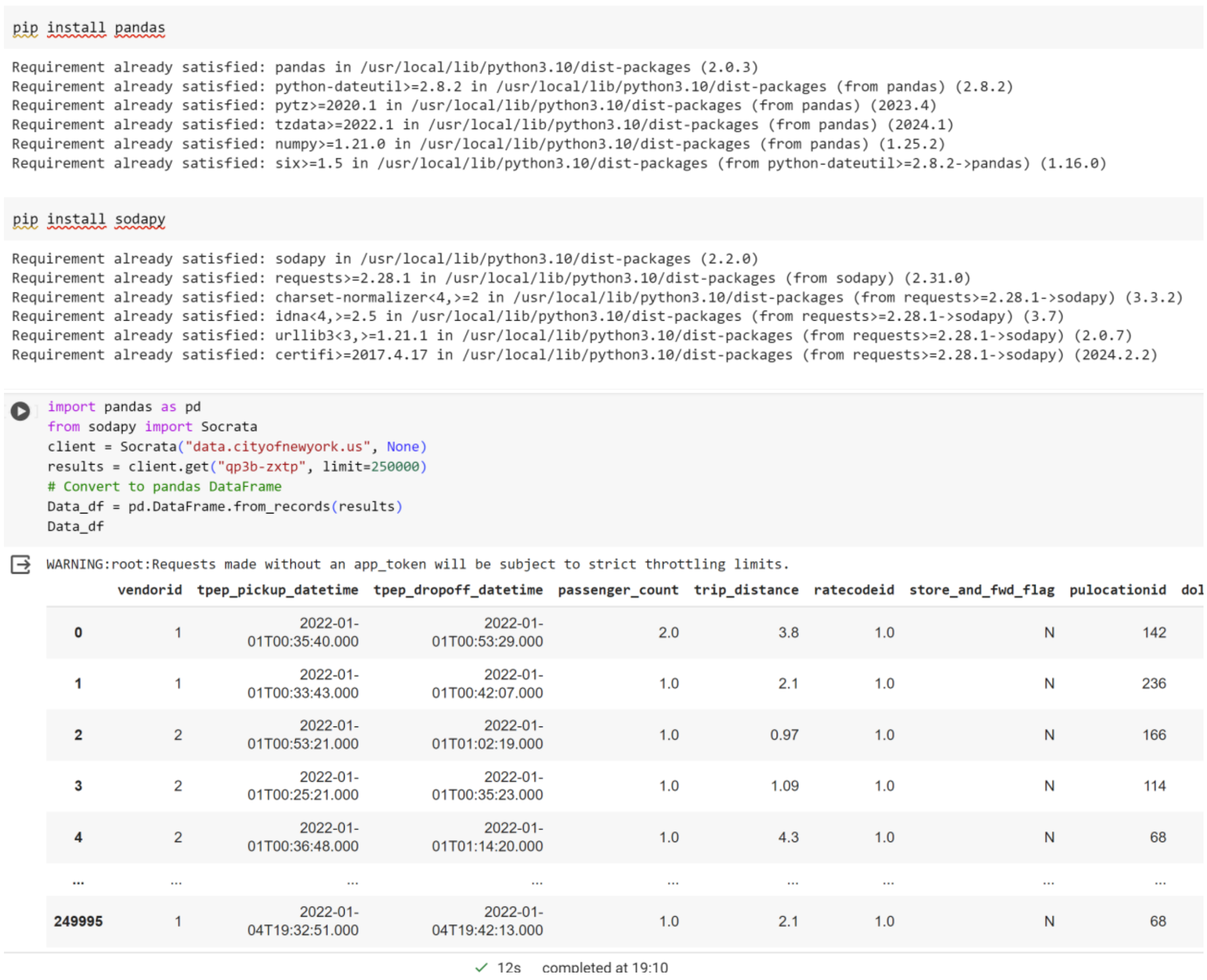
Taxi Companies:
Taxi companies can leverage the model to guide their drivers on how to maximize tip earnings. They can also use the insights from the model to design incentive programs or training sessions aimed at improving customer service and increasing tip amounts.

Passengers:
Cab drivers, whose goal is to maximise their tips, will provide passengers with superior service. The process's regularity and transparency, which might lead to an increase in customers' overall satisfaction with taxi services, are especially appreciated by credit card users.

Regulatory Authorities:
Policy choices aiming at advancing equitable pay practices and enhancing the general passenger experience may be informed by the model's findings, which can be utilised by regulatory bodies that supervise the taxi business.

Initial Data Analysis:

Yellow Taxi data set for the year 2022 consists of 39.7 million rows and 19 columns. But here we have considered 250000 rows and 19 columns for our analysis.
We read the taxi data as a **JSON** file from the **API** using the **SODAPY** package.



```
pip install pandas
```

```
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (2.0.3)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2023.4)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2024.1)
Requirement already satisfied: numpy>=1.21.0 in /usr/local/lib/python3.10/dist-packages (from pandas) (1.25.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.2->pandas) (1.16.0)
```

```
pip install sodapy
```

```
Requirement already satisfied: sodapy in /usr/local/lib/python3.10/dist-packages (2.2.0)
Requirement already satisfied: requests>=2.28.1 in /usr/local/lib/python3.10/dist-packages (from sodapy) (2.31.0)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests>=2.28.1->sodapy) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests>=2.28.1->sodapy) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests>=2.28.1->sodapy) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests>=2.28.1->sodapy) (2024.2.2)
```

```
import pandas as pd
from sodapy import Socrata
client = Socrata("data.cityofnewyork.us", None)
results = client.get("qp3b-zxtp", limit=250000)
# Convert to pandas DataFrame
Data_df = pd.DataFrame.from_records(results)
Data_df
```

```
WARNING:root:Requests made without an app_token will be subject to strict throttling limits.
```

	vendorid	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance	ratecodeid	store_and_fwd_flag	pulocationid	dol
0	1	2022-01-01T00:35:40.000	2022-01-01T00:53:29.000	2.0	3.8	1.0	N	142	
1	1	2022-01-01T00:33:43.000	2022-01-01T00:42:07.000	1.0	2.1	1.0	N	236	
2	2	2022-01-01T00:53:21.000	2022-01-01T01:02:19.000	1.0	0.97	1.0	N	166	
3	2	2022-01-01T00:25:21.000	2022-01-01T00:35:23.000	1.0	1.09	1.0	N	114	
4	2	2022-01-01T00:36:48.000	2022-01-01T01:14:20.000	1.0	4.3	1.0	N	68	
...	...	...	...	...	...	...	...	...	
249995	1	2022-01-04T19:32:51.000	2022-01-04T19:42:13.000	1.0	2.1	1.0	N	68	

✓ 12s completed at 19:10

Here we have described each column in our data set.

Vendor ID: A code indicating the TPEP provider that provided the record.

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.

RatecodeID: The final rate code is in effect at the end of the trip.

store_and_fwd_flag: This flag indicates whether the trip record was held in vehicle memory before sending to the vendor, aka “store and forward,” because the vehicle did not have a connection to the server.

PULocationID: TLC Taxi Zone in which the taximeter was engaged.

DOLocationID: TLC Taxi Zone in which the taximeter was disengaged.

payment_type: A numeric code signifying how the passenger paid for the trip.

fare_amount: The time-and-distance fare calculated by the meter.

extra: Miscellaneous extras and surcharges.

mta_tax: Tax that is automatically triggered based on the metered rate in use.

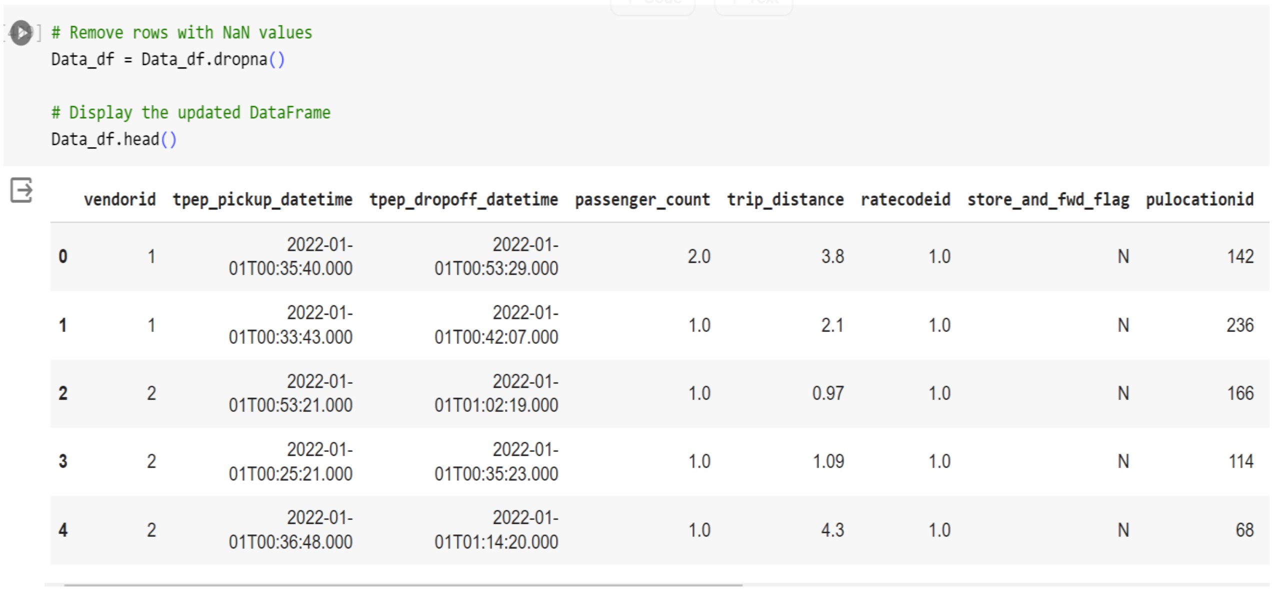
tip_amount: This field is automatically populated for credit card tips. Cash tips are not included.

tolls_amount: Total amount of all tolls paid in the trip.
improvement_surcharge: Improvement surcharge assessed trips at the flag drop. The improvement surcharge began being levied in 2015.
total_amount: The total amount charged to passengers. Does not include cash tips.
congestion_surcharge: Total amount collected in trip for NYS congestion surcharge.
airport_fee: For pick up only at LaGuardia and John F. Kennedy Airports.

<u>Data Preparation:</u>

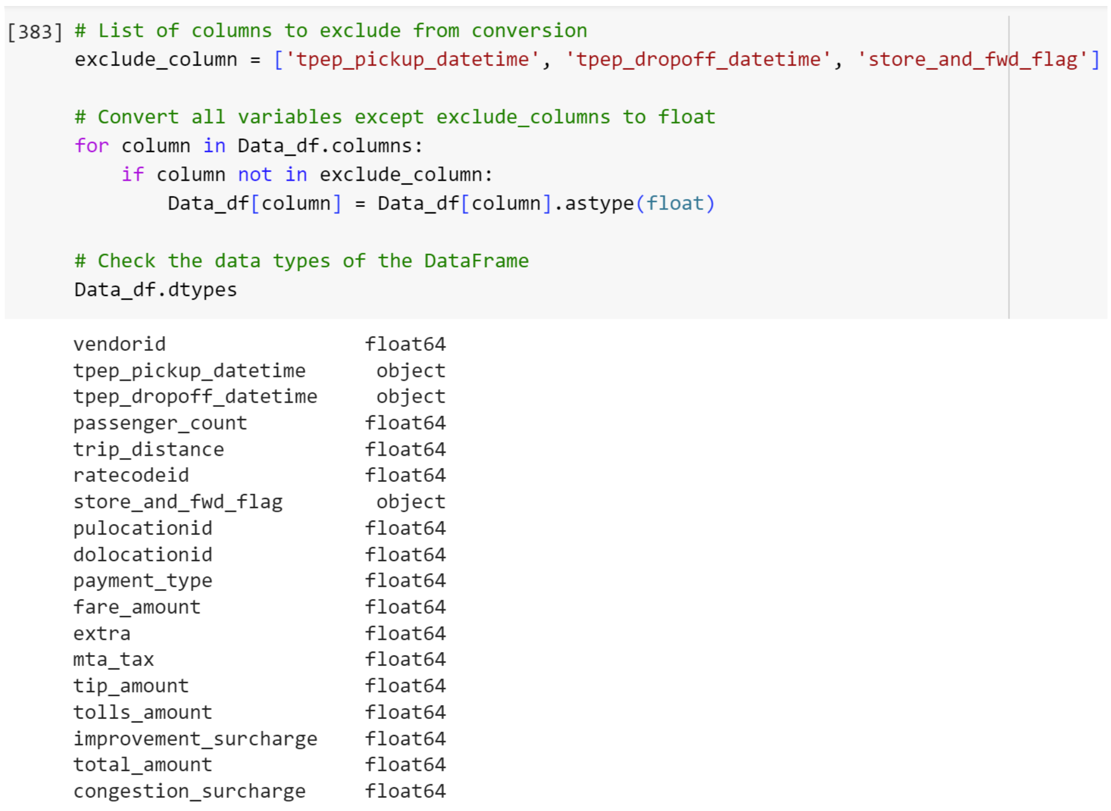
We have removed unknown values from our data set and checked the data types of all the variables in our taxi data.

Later converted the data types of desired variables to proceed further into our analysis.



```
# Remove rows with NaN values
Data_df = Data_df.dropna()

# Display the updated DataFrame
Data_df.head()
```

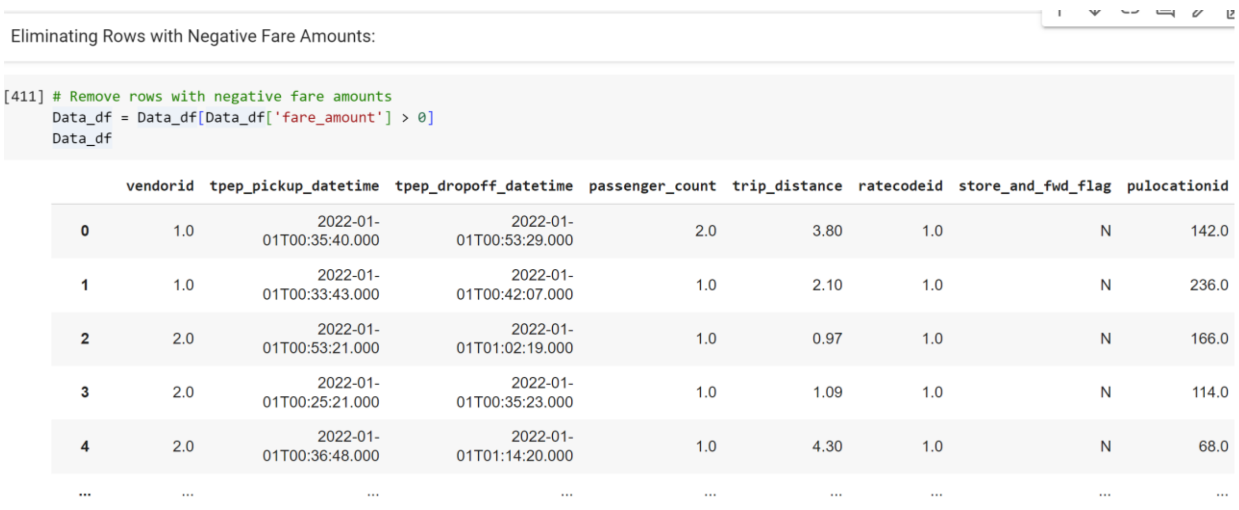
	vendorid	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance	ratecodeid	store_and_fwd_flag	pulocationid
0	1	2022-01-01T00:35:40.000	2022-01-01T00:53:29.000	2.0	3.8	1.0	N	142
1	1	2022-01-01T00:33:43.000	2022-01-01T00:42:07.000	1.0	2.1	1.0	N	236
2	2	2022-01-01T00:53:21.000	2022-01-01T01:02:19.000	1.0	0.97	1.0	N	166
3	2	2022-01-01T00:25:21.000	2022-01-01T00:35:23.000	1.0	1.09	1.0	N	114
4	2	2022-01-01T00:36:48.000	2022-01-01T01:14:20.000	1.0	4.3	1.0	N	68



```
[383] # List of columns to exclude from conversion
      exclude_column = ['tpep_pickup_datetime', 'tpep_dropoff_datetime', 'store_and_fwd_flag']

      # Convert all variables except exclude_columns to float
      for column in Data_df.columns:
          if column not in exclude_column:
              Data_df[column] = Data_df[column].astype(float)

      # Check the data types of the DataFrame
      Data_df.dtypes
```

```
vendorid                 float64
tpep_pickup_datetime      object
tpep_dropoff_datetime     object
passenger_count          float64
trip_distance            float64
ratecodeid               float64
store_and_fwd_flag        object
pulocationid             float64
dolocationid             float64
payment_type             float64
fare_amount              float64
extra                    float64
mta_tax                  float64
tip_amount               float64
tolls_amount             float64
improvement_surcharge    float64
total_amount             float64
congestion_surcharge     float64
```

Upon checking we found that the data set contains negative fare amounts which might be an error. So we have filtered them out and displayed the data set.



Eliminating Rows with Negative Fare Amounts:

```
[411] # Remove rows with negative fare amounts
      Data_df = Data_df[Data_df['fare_amount'] > 0]
      Data_df
```

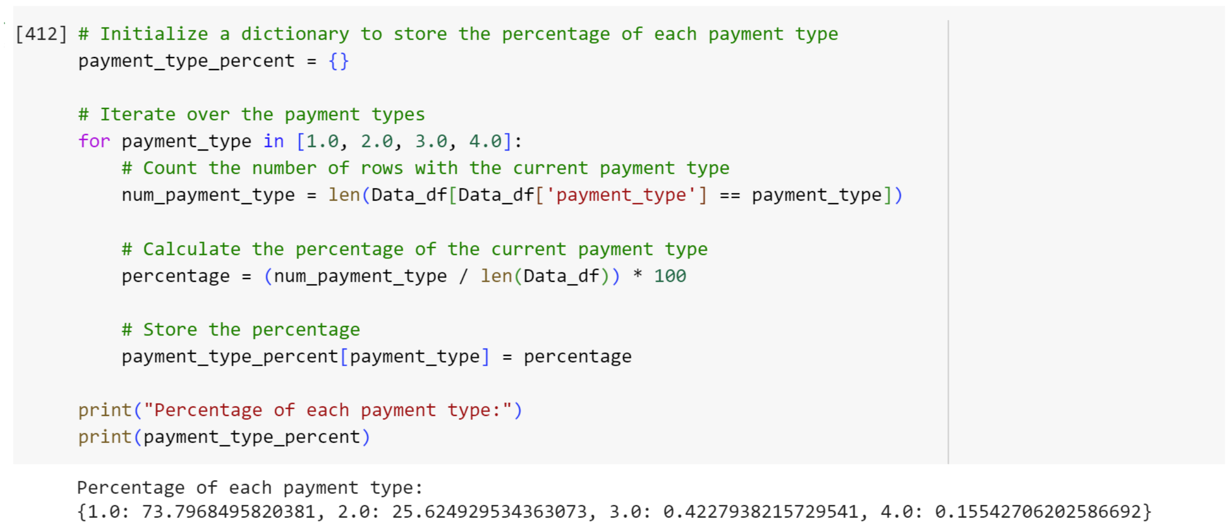
	vendorid	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance	ratecodeid	store_and_fwd_flag	pulocationid
0	1.0	2022-01-01T00:35:40.000	2022-01-01T00:53:29.000	2.0	3.80	1.0	N	142.0
1	1.0	2022-01-01T00:33:43.000	2022-01-01T00:42:07.000	1.0	2.10	1.0	N	236.0
2	2.0	2022-01-01T00:53:21.000	2022-01-01T01:02:19.000	1.0	0.97	1.0	N	166.0
3	2.0	2022-01-01T00:25:21.000	2022-01-01T00:35:23.000	1.0	1.09	1.0	N	114.0
4	2.0	2022-01-01T00:36:48.000	2022-01-01T01:14:20.000	1.0	4.30	1.0	N	68.0
...	...	...	...	...	...	...	...	...

<u>Statistical Analysis for “payment_type” predictor:</u>

We have evaluated the proportion of passengers present within each type of payment category and also evaluated the proportion of individuals within each payment type category who provide a tip.

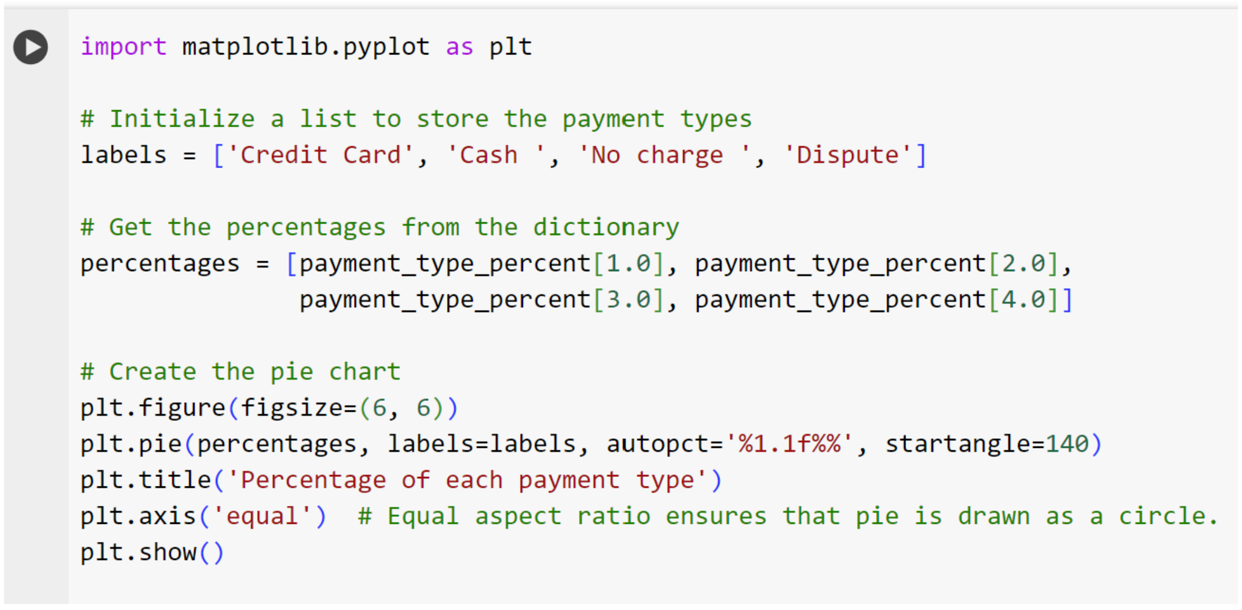
Here payment_type values suggest:
1= Credit card
2= Cash
3= No charge
4= Dispute

Evaluating the proportion of individuals within each payment type category:

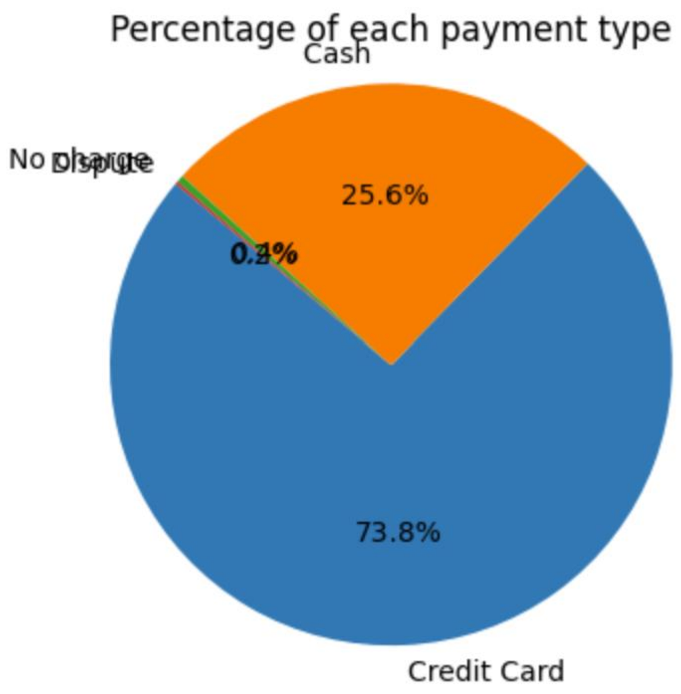


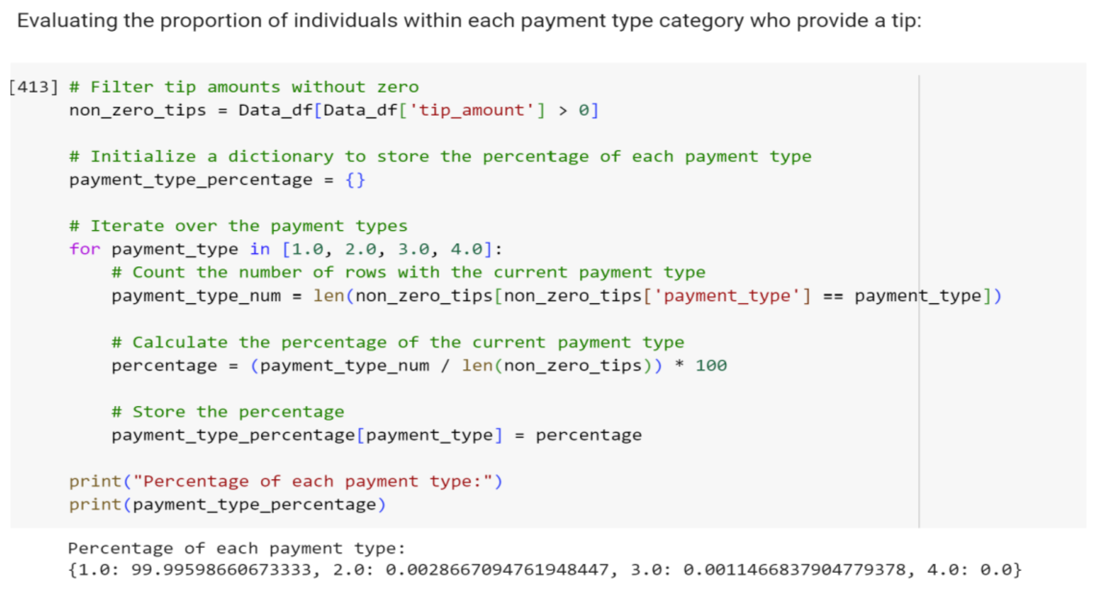
```
[412] # Initialize a dictionary to store the percentage of each payment type
      payment_type_percent = {}

      # Iterate over the payment types
      for payment_type in [1.0, 2.0, 3.0, 4.0]:
          # Count the number of rows with the current payment type
          num_payment_type = len(Data_df[Data_df['payment_type'] == payment_type])

          # Calculate the percentage of the current payment type
          percentage = (num_payment_type / len(Data_df)) * 100

          # Store the percentage
          payment_type_percent[payment_type] = percentage

      print("Percentage of each payment type:")
      print(payment_type_percent)
```

```
Percentage of each payment type:
{1.0: 73.7968495820381, 2.0: 25.624929534363073, 3.0: 0.4227938215729541, 4.0: 0.15542706202586692}
```



```
import matplotlib.pyplot as plt

# Initialize a list to store the payment types
labels = ['Credit Card', 'Cash ', 'No charge ', 'Dispute']

# Get the percentages from the dictionary
percentages = [payment_type_percent[1.0], payment_type_percent[2.0],
               payment_type_percent[3.0], payment_type_percent[4.0]]

# Create the pie chart
plt.figure(figsize=(6, 6))
plt.pie(percentages, labels=labels, autopct='%1.1f%%', startangle=140)
plt.title('Percentage of each payment type')
plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.
plt.show()
```




Evaluating the proportion of individuals within each payment type category who provide a tip:

```
[413] # Filter tip amounts without zero
      non_zero_tips = Data_df[Data_df['tip_amount'] > 0]

      # Initialize a dictionary to store the percentage of each payment type
      payment_type_percentage = {}

      # Iterate over the payment types
      for payment_type in [1.0, 2.0, 3.0, 4.0]:
          # Count the number of rows with the current payment type
          payment_type_num = len(non_zero_tips[non_zero_tips['payment_type'] == payment_type])

          # Calculate the percentage of the current payment type
          percentage = (payment_type_num / len(non_zero_tips)) * 100

          # Store the percentage
          payment_type_percentage[payment_type] = percentage

      print("Percentage of each payment type:")
      print(payment_type_percentage)
```

```
Percentage of each payment type:
{1.0: 99.99598660673333, 2.0: 0.0028667094761948447, 3.0: 0.0011466837904779378, 4.0: 0.0}
```

Finally, we have calculated the average tip amount provided by passengers from each payment_type category.

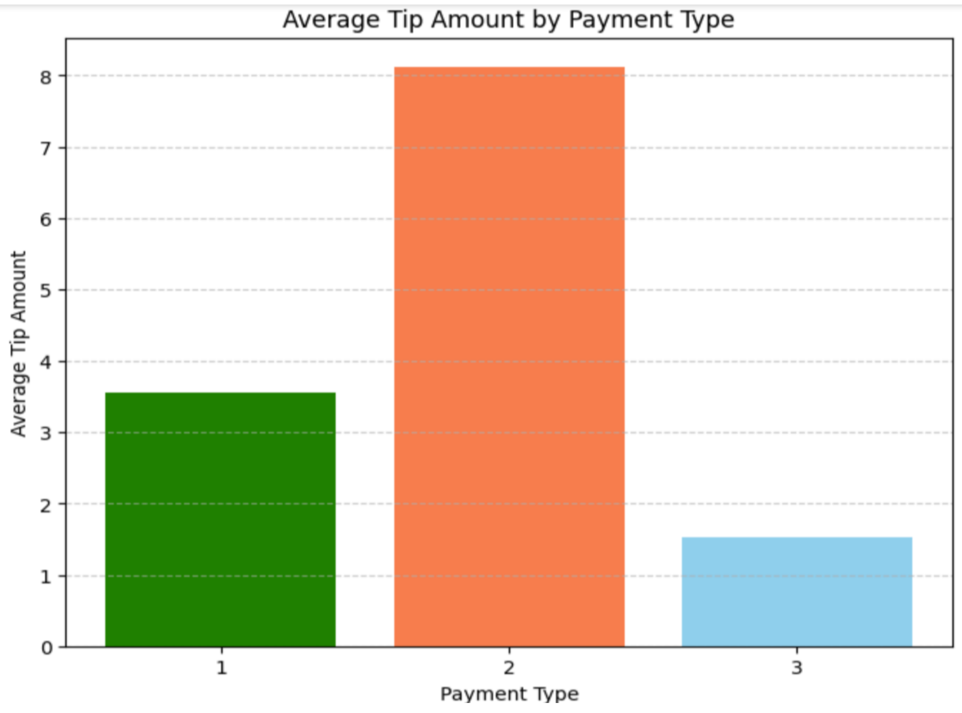
payment_type	Average tip_amount
1.0	3.566482$
2.0	8.124000$
3.0	1.520000$

Calculating the mean tip amount for each payment_type:

```python
# Group the data by 'payment_type'
average_tip_by_payment_type = non_zero_tips.groupby('payment_type')['tip_amount'].mean()

print("Average tip provided by each payment_type passenger:")
print(average_tip_by_payment_type)
```

```
Average tip provided by each payment_type passenger:
payment_type
1.0    3.566482
2.0    8.124000
3.0    1.520000
Name: tip_amount, dtype: float64
```

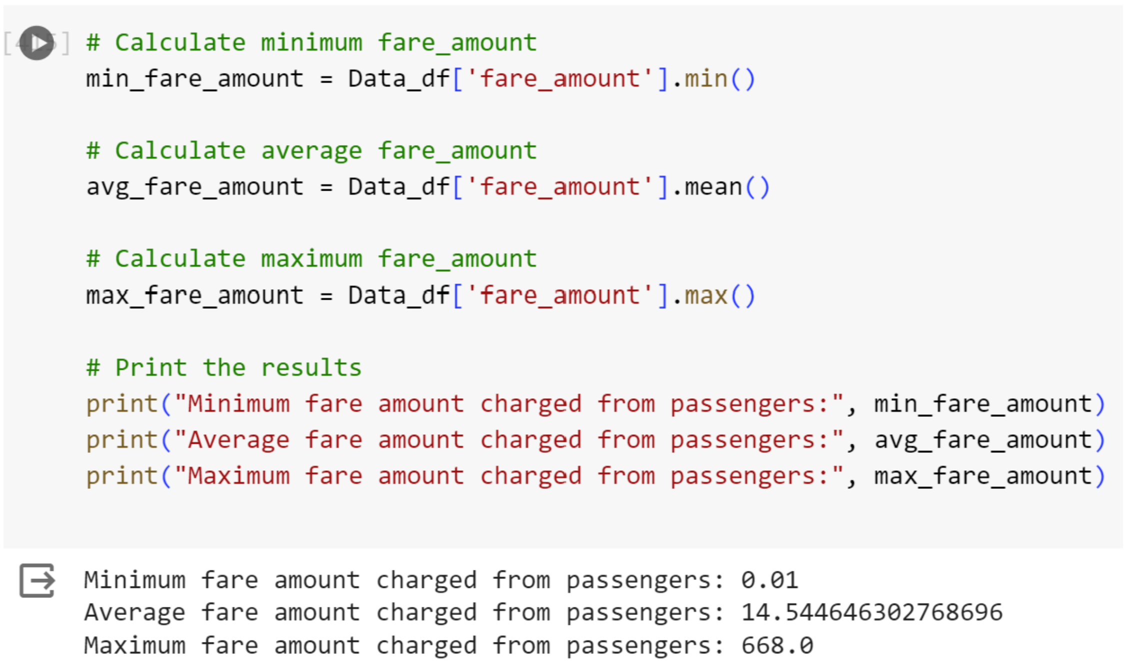
```python
import matplotlib.pyplot as plt

# Define colors for each payment type
colors = ['green', 'lightsalmon', 'skyblue', 'coral']

# Get the payment types and corresponding average tips
payment_types = average_tip_by_payment_type.index
average_tips = average_tip_by_payment_type.values

# Create the histogram with custom colors
plt.figure(figsize=(8, 6))
plt.bar(payment_types, average_tips, color=colors)
plt.xlabel('Payment Type')
plt.ylabel('Average Tip Amount')
plt.title('Average Tip Amount by Payment Type')
plt.xticks(payment_types)  # Ensure all payment types are shown on x-axis
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```



Statistical Analysis for “fare_amount” predictor:

Calculated minimum, average, and maximum fare amounts present in our data. Also, calculated average tip amounts in each range after dividing fare amounts into different ranges.

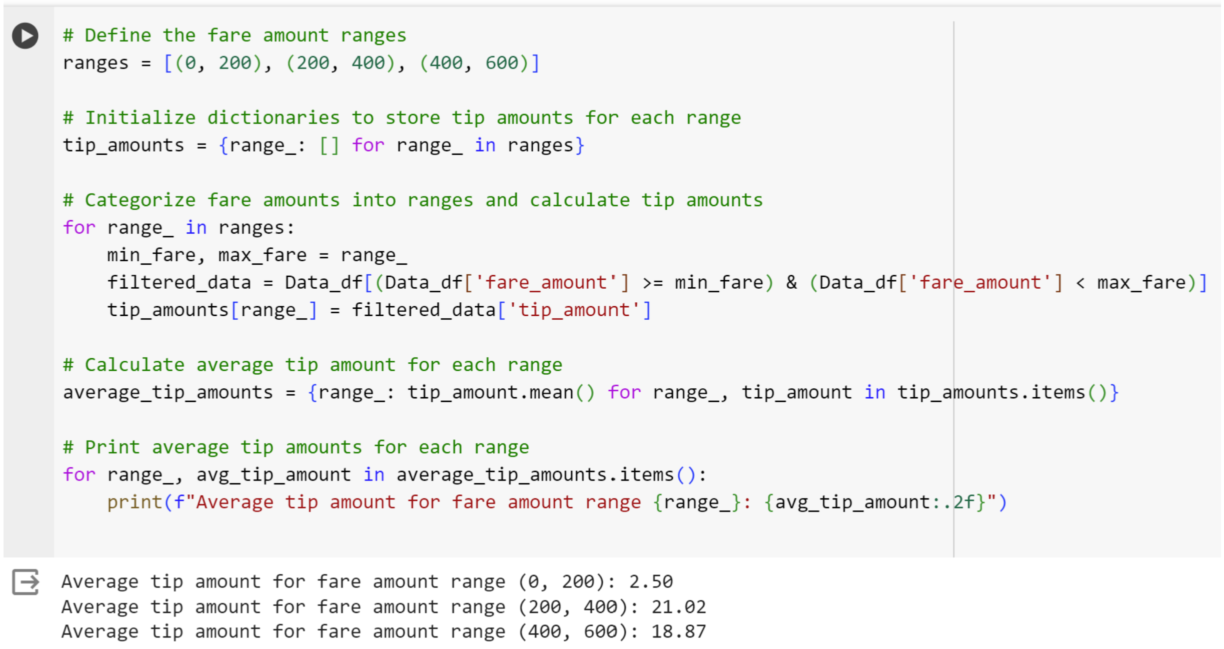
Calculating Minimum, Average and Maximun fare amounts in the data:



```
# Calculate minimum fare_amount
min_fare_amount = Data_df['fare_amount'].min()

# Calculate average fare_amount
avg_fare_amount = Data_df['fare_amount'].mean()

# Calculate maximum fare_amount
max_fare_amount = Data_df['fare_amount'].max()

# Print the results
print("Minimum fare amount charged from passengers:", min_fare_amount)
print("Average fare amount charged from passengers:", avg_fare_amount)
print("Maximum fare amount charged from passengers:", max_fare_amount)
```

```
Minimum fare amount charged from passengers: 0.01
Average fare amount charged from passengers: 14.544646302768696
Maximum fare amount charged from passengers: 668.0
```

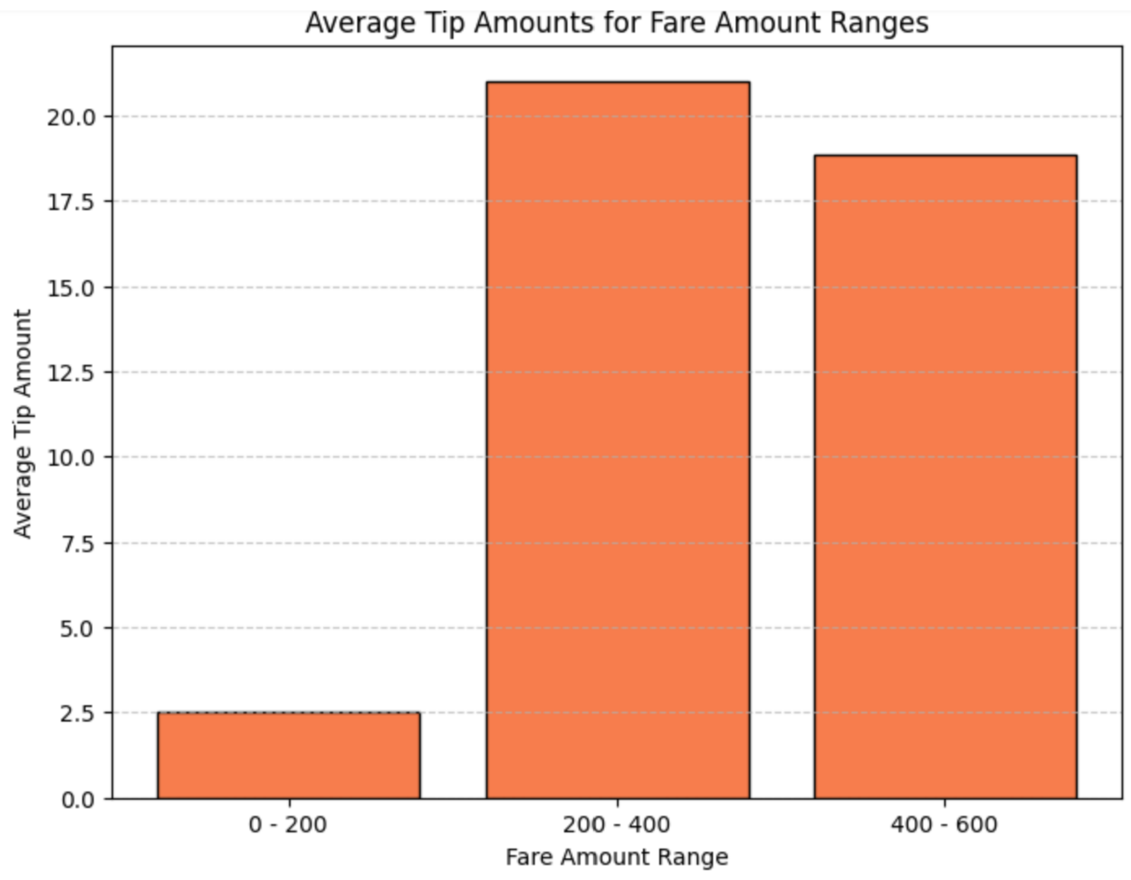
Calculating the average tip amounts in different ranges of fare amonut:



```
# Define the fare amount ranges
ranges = [(0, 200), (200, 400), (400, 600)]

# Initialize dictionaries to store tip amounts for each range
tip_amounts = {range_: [] for range_ in ranges}

# Categorize fare amounts into ranges and calculate tip amounts
for range_ in ranges:
    min_fare, max_fare = range_
    filtered_data = Data_df[(Data_df['fare_amount'] >= min_fare) & (Data_df['fare_amount'] < max_fare)]
    tip_amounts[range_] = filtered_data['tip_amount']

# Calculate average tip amount for each range
average_tip_amounts = {range_: tip_amount.mean() for range_, tip_amount in tip_amounts.items()}

# Print average tip amounts for each range
for range_, avg_tip_amount in average_tip_amounts.items():
    print(f"Average tip amount for fare amount range {range_}: {avg_tip_amount:.2f}")
```

```
Average tip amount for fare amount range (0, 200): 2.50
Average tip amount for fare amount range (200, 400): 21.02
Average tip amount for fare amount range (400, 600): 18.87
```

```
[15] import matplotlib.pyplot as plt

     # Extract the ranges and average tip amounts
     ranges = [f"{range_[0]} - {range_[1]}" for range_ in average_tip_amounts.keys()]
     average_tips = list(average_tip_amounts.values())

     # Create the bar plot
     plt.figure(figsize=(8, 6))
     plt.bar(ranges, average_tips, color='coral', edgecolor='black')

     # Add labels and title
     plt.xlabel('Fare Amount Range')
     plt.ylabel('Average Tip Amount')
     plt.title('Average Tip Amounts for Fare Amount Ranges')

     # Add grid
     plt.grid(axis='y', linestyle='--', alpha=0.7)

     # Show the plot
     plt.show()
```

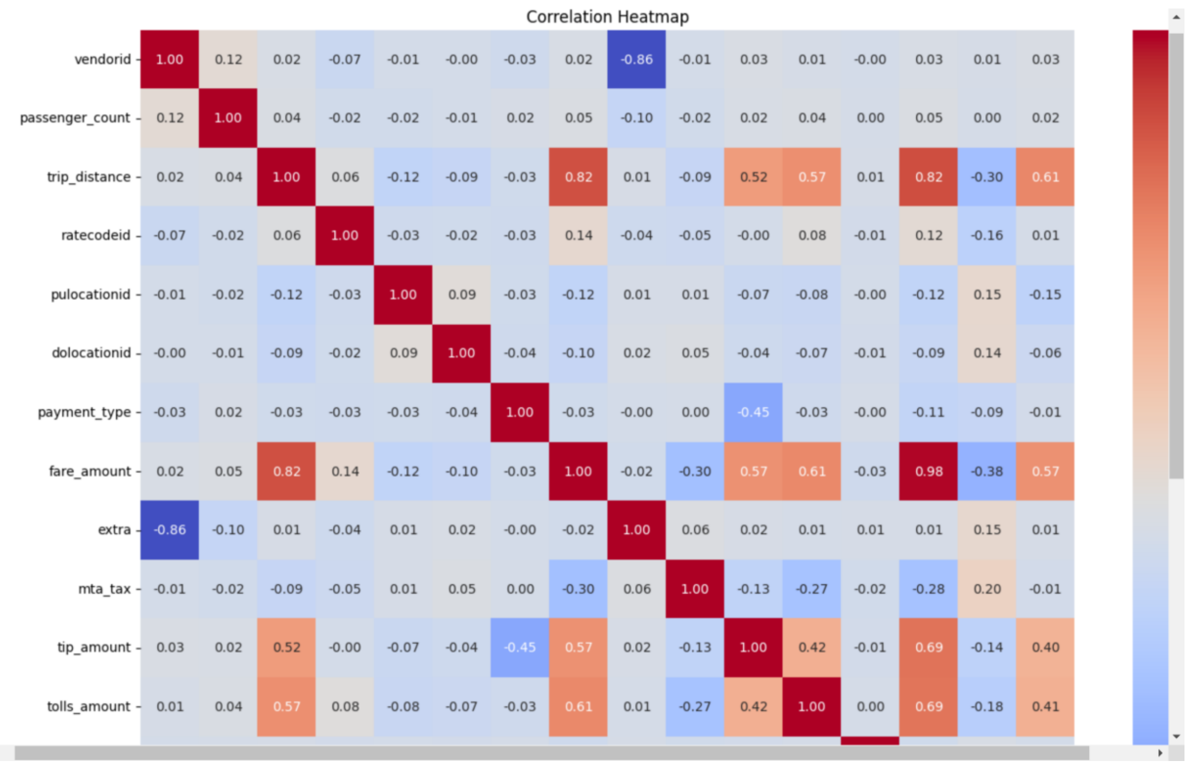


As we can see here people who are paying higher fare amounts tend to give high amounts of tip.

<u>Checking Correlation between variables:</u>

Here we analyzed which predictor is more useful for determining the tip amount based on correlation coefficient value using a correlation heat map.

```
[416] import pandas as pd
      import seaborn as sns
      import matplotlib.pyplot as plt

      # Convert all variables except exclude_columns to float
      numeric_data= Data_df.drop(columns=exclude_column).astype(float)

      # Calculate correlation matrix
      correlation_matrix1 = numeric_data.corr()

      # Plot heatmap
      plt.figure(figsize=(16, 13))
      sns.heatmap(correlation_matrix1, annot=True, cmap='coolwarm', fmt=".2f")
      plt.title('Correlation Heatmap')
      plt.show()
```



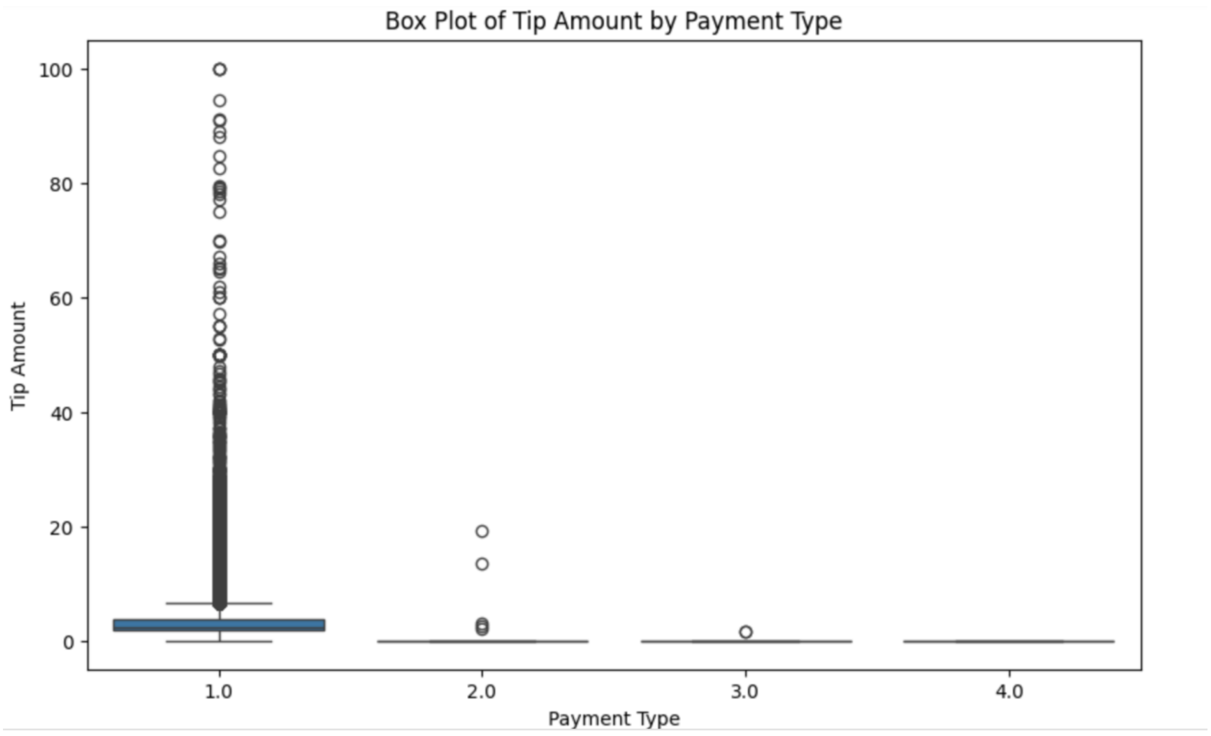
vendorid	1.00	0.12	0.02	-0.07	-0.01	-0.00	-0.03	0.02	-0.86	-0.01	0.03	0.01	-0.00	0.03	0.01	0.03
passenger_count	0.12	1.00	0.04	-0.02	-0.02	-0.01	0.02	0.05	-0.10	-0.02	0.02	0.04	0.00	0.05	0.00	0.02
trip_distance	0.02	0.04	1.00	0.06	-0.12	-0.09	-0.03	0.82	0.01	-0.09	0.52	0.57	0.01	0.82	-0.30	0.61
ratecodeid	-0.07	-0.02	0.06	1.00	-0.03	-0.02	-0.03	0.14	-0.04	-0.05	-0.00	0.08	-0.01	0.12	-0.16	0.01
pulocationid	-0.01	-0.02	-0.12	-0.03	1.00	0.09	-0.03	-0.12	0.01	0.01	-0.07	-0.08	-0.00	-0.12	0.15	-0.15
dolocationid	-0.00	-0.01	-0.09	-0.02	0.09	1.00	-0.04	-0.10	0.02	0.05	-0.04	-0.07	-0.01	-0.09	0.14	-0.06
payment_type	-0.03	0.02	-0.03	-0.03	-0.03	-0.04	1.00	-0.03	-0.00	0.00	-0.45	-0.03	-0.00	-0.11	-0.09	-0.01
fare_amount	0.02	0.05	0.82	0.14	-0.12	-0.10	-0.03	1.00	-0.02	-0.30	0.57	0.61	-0.03	0.98	-0.38	0.57
extra	-0.86	-0.10	0.01	-0.04	0.01	0.02	-0.00	-0.02	1.00	0.06	0.02	0.01	0.01	0.01	0.15	0.01
mta_tax	-0.01	-0.02	-0.09	-0.05	0.01	0.05	0.00	-0.30	0.06	1.00	-0.13	-0.27	-0.02	-0.28	0.20	-0.01
tip_amount	0.03	0.02	0.52	-0.00	-0.07	-0.04	-0.45	0.57	0.02	-0.13	1.00	0.42	-0.01	0.69	-0.14	0.40
tolls_amount	0.01	0.04	0.57	0.08	-0.08	-0.07	-0.03	0.61	0.01	-0.27	0.42	1.00	0.00	0.69	-0.18	0.41

After checking the correlation coefficients between different predictors and the response variable “tip_amount”, we can say two predictors “fare_amount” and “payment_type” have correlation coefficients of “0.57” and “-0.45”.

<u>Data Visualization:</u>

We have visualized the above analysis by plotting a scatter plot between **tip_amount** vs **fare_amount** and a box plot is plotted between **tip_amount** vs **payment_type**

```python
import seaborn as sns
import matplotlib.pyplot as plt

# Set the figure size
plt.figure(figsize=(10, 6))

# Create the box plot
sns.boxplot(x='payment_type', y='tip_amount', data=Data_df)

# Add labels and title
plt.title('Box Plot of Tip Amount by Payment Type')
plt.xlabel('Payment Type')
plt.ylabel('Tip Amount')

# Show the plot
plt.show()
```

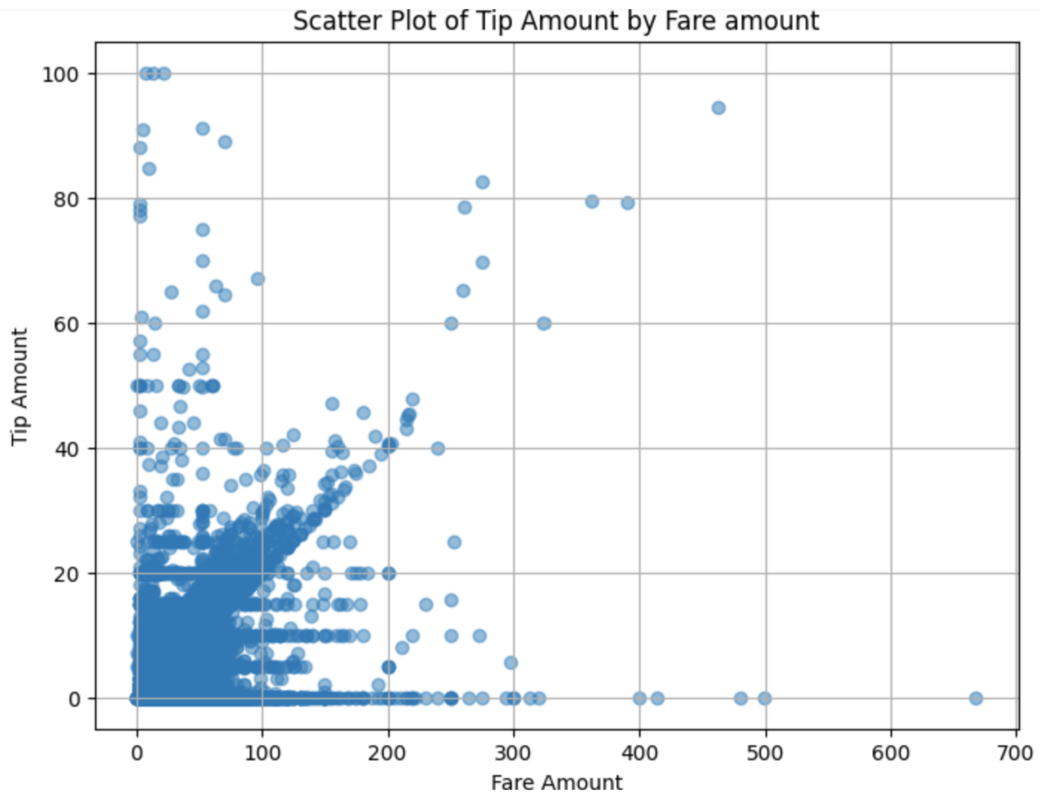


As we can observe there is a negative relationship between **tip_amount** and **payment_type.** Similarly, there is a positive relationship between **tip_amount** and **fare_amount**.

Positive relation with Fare Amount: The tip amount shows a moderately positive correlation with the fare amount, indicating that higher fare amounts tend to correspond to higher tip amounts. This suggests that passengers may be more inclined to tip generously for longer or more expensive rides.

Negative relation with Payment Type: The tip amount exhibits a moderate negative correlation with the payment type, implying that certain payment methods may be associated with lower tip amounts compared to others. This could reflect differences in tipping behavior among passengers using different payment methods.

```python
import matplotlib.pyplot as plt

# Scatter plot between 'tip_amount' and 'fare_amount'
plt.figure(figsize=(8, 6))
plt.scatter(Data_df['fare_amount'], Data_df['tip_amount'], alpha=0.5)
plt.title('Scatter Plot of Tip Amount by Fare amount')
plt.xlabel('Fare Amount')
plt.ylabel('Tip Amount')
plt.grid(True)
plt.show()
```

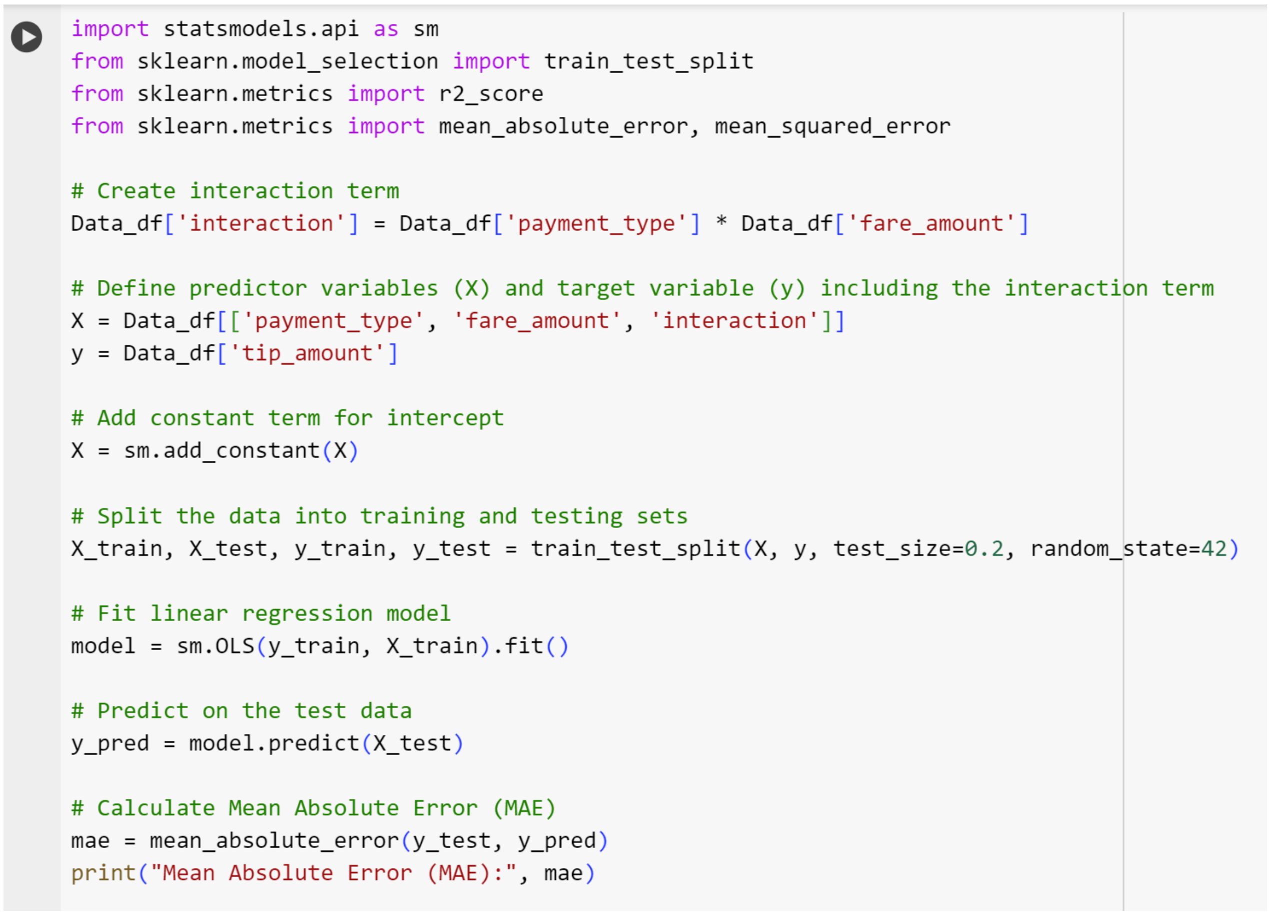


Initial Modeling:

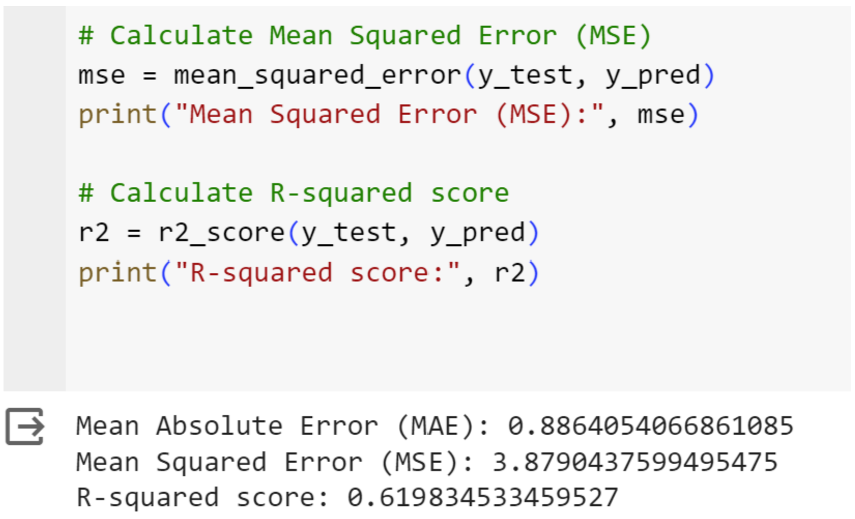
Firstly we have separated the data **“Data_df”** into training and testing sets in 80:20 ratio. Later we performed linear regression on the training set and predicted the values for the testing set.

We got the values of MAE, MSE, and R-squared as follows:

Mean Absolute Error (MAE): 0.8864054066861085
Mean Squared Error (MSE): 3.8790437599495475
R-squared score: 0.619834533459527



```
import statsmodels.api as sm
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score
from sklearn.metrics import mean_absolute_error, mean_squared_error

# Create interaction term
Data_df['interaction'] = Data_df['payment_type'] * Data_df['fare_amount']

# Define predictor variables (X) and target variable (y) including the interaction term
X = Data_df[['payment_type', 'fare_amount', 'interaction']]
y = Data_df['tip_amount']

# Add constant term for intercept
X = sm.add_constant(X)

# 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)

# Fit linear regression model
model = sm.OLS(y_train, X_train).fit()

# Predict on the test data
y_pred = model.predict(X_test)

# Calculate Mean Absolute Error (MAE)
mae = mean_absolute_error(y_test, y_pred)
print("Mean Absolute Error (MAE):", mae)
```



```
# Calculate Mean Squared Error (MSE)
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error (MSE):", mse)

# Calculate R-squared score
r2 = r2_score(y_test, y_pred)
print("R-squared score:", r2)
```

```
Mean Absolute Error (MAE): 0.8864054066861085
Mean Squared Error (MSE): 3.8790437599495475
R-squared score: 0.619834533459527
```

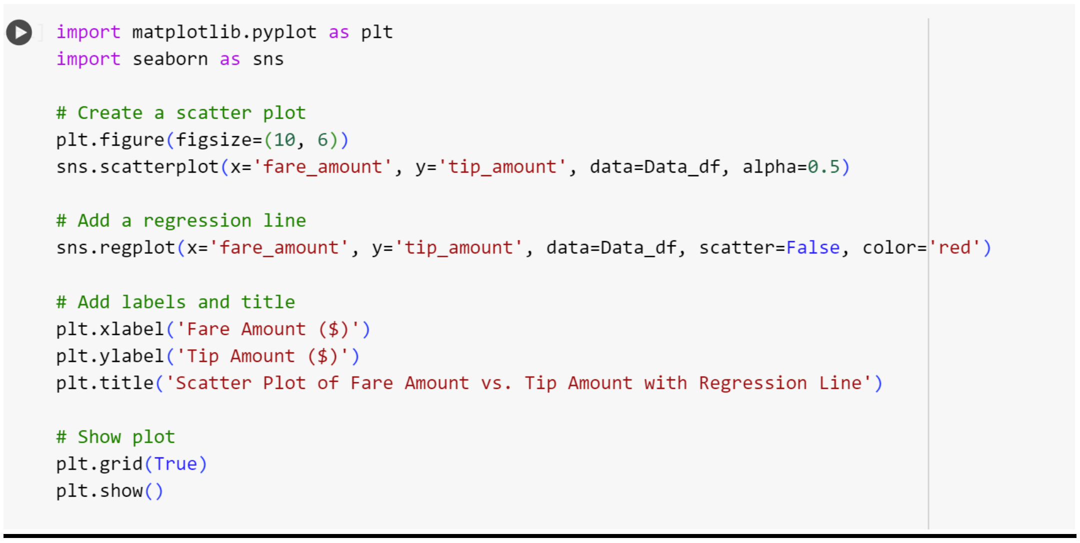
The Mean Absolute Error (MAE) of 0.8864 suggests that, on average, the model's predictions differ by approximately $0.8864 from the actual tip amounts. The tip amount is around $3.8790 since the actual and expected tip amounts differ by a squared amount on average.The independent factors account for around 61.98% of the variation in tip amounts, according to the R-squared score of 0.6198.

Visualizations that showcase model performance:

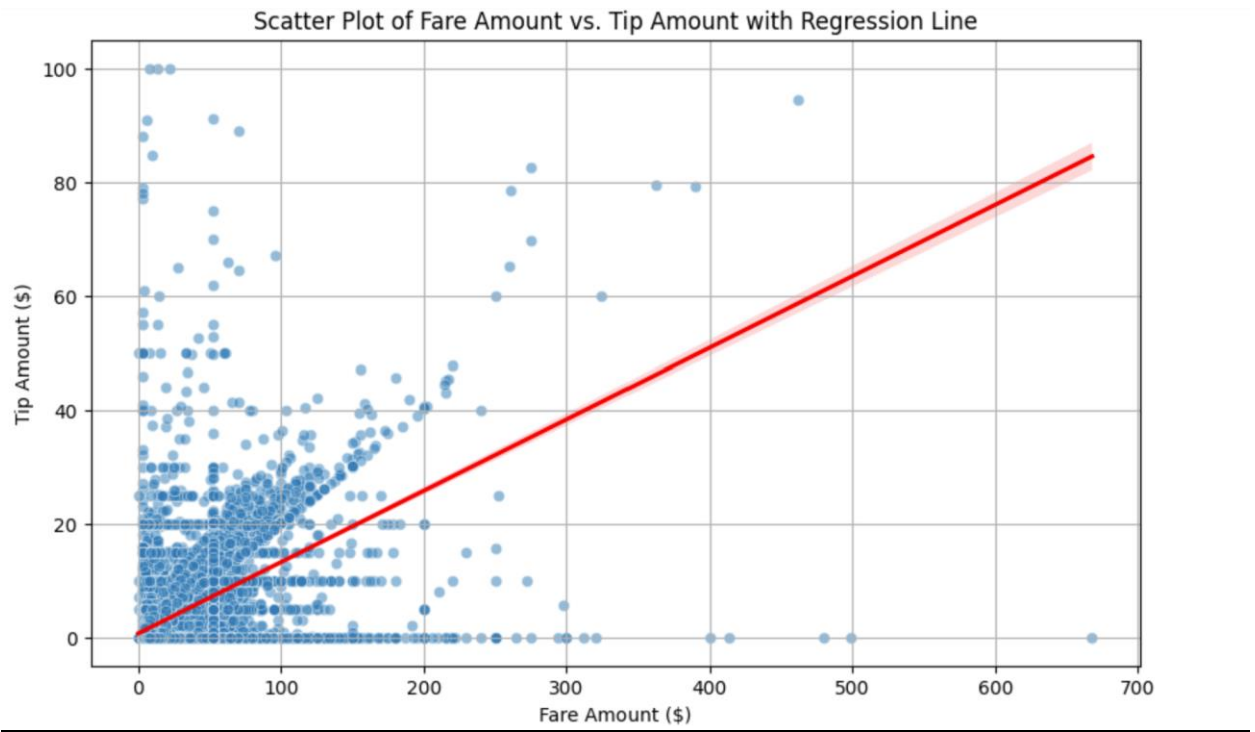
1. Fitted plot between predictor and response variable:

fare_amount vs tip_amount :

Here we have fitted the plot with a regression line corresponding to fare_amount vs tip_amount



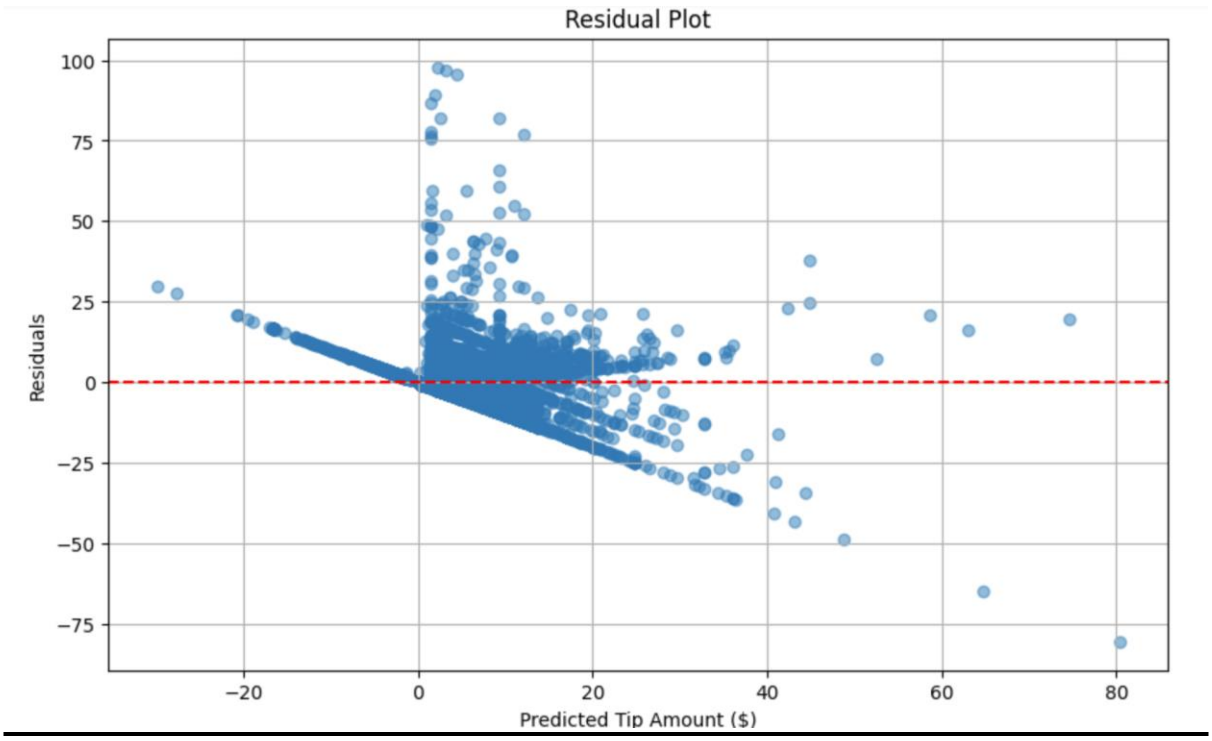
```
import matplotlib.pyplot as plt
import seaborn as sns

# Create a scatter plot
plt.figure(figsize=(10, 6))
sns.scatterplot(x='fare_amount', y='tip_amount', data=Data_df, alpha=0.5)

# Add a regression line
sns.regplot(x='fare_amount', y='tip_amount', data=Data_df, scatter=False, color='red')

# Add labels and title
plt.xlabel('Fare Amount ($)')
plt.ylabel('Tip Amount ($)')
plt.title('Scatter Plot of Fare Amount vs. Tip Amount with Regression Line')

# Show plot
plt.grid(True)
plt.show()
```



2. Residual Plot:

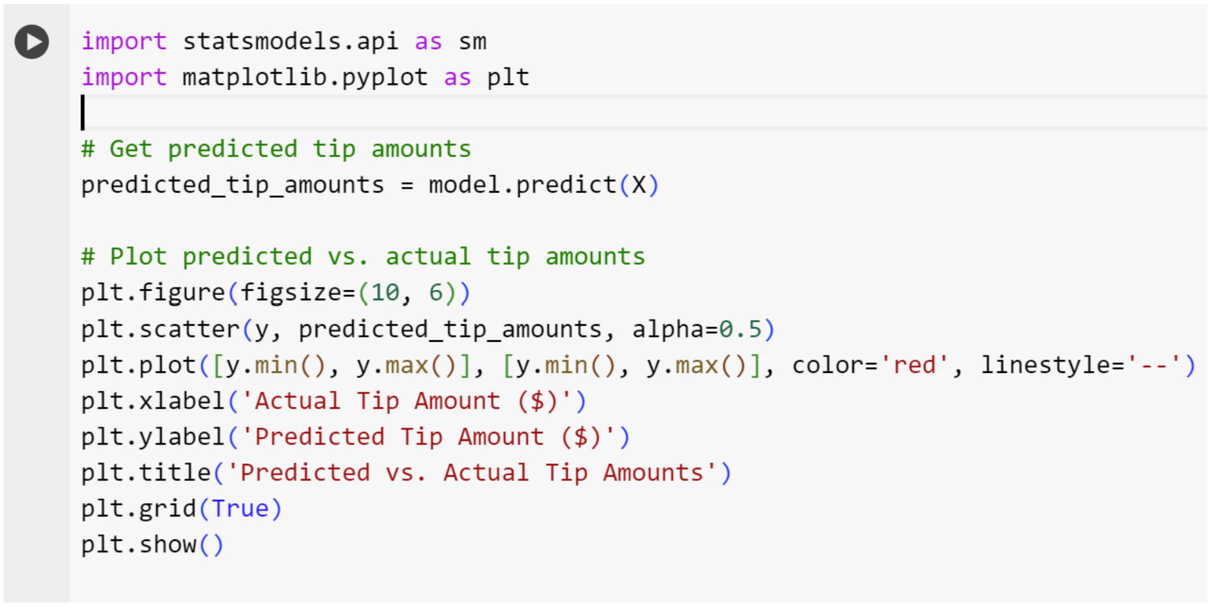
Our fitted model contains most of the residuals evenly distributed around the horizontal line at y=0, indicating that the model adequately captures the variation in the data.

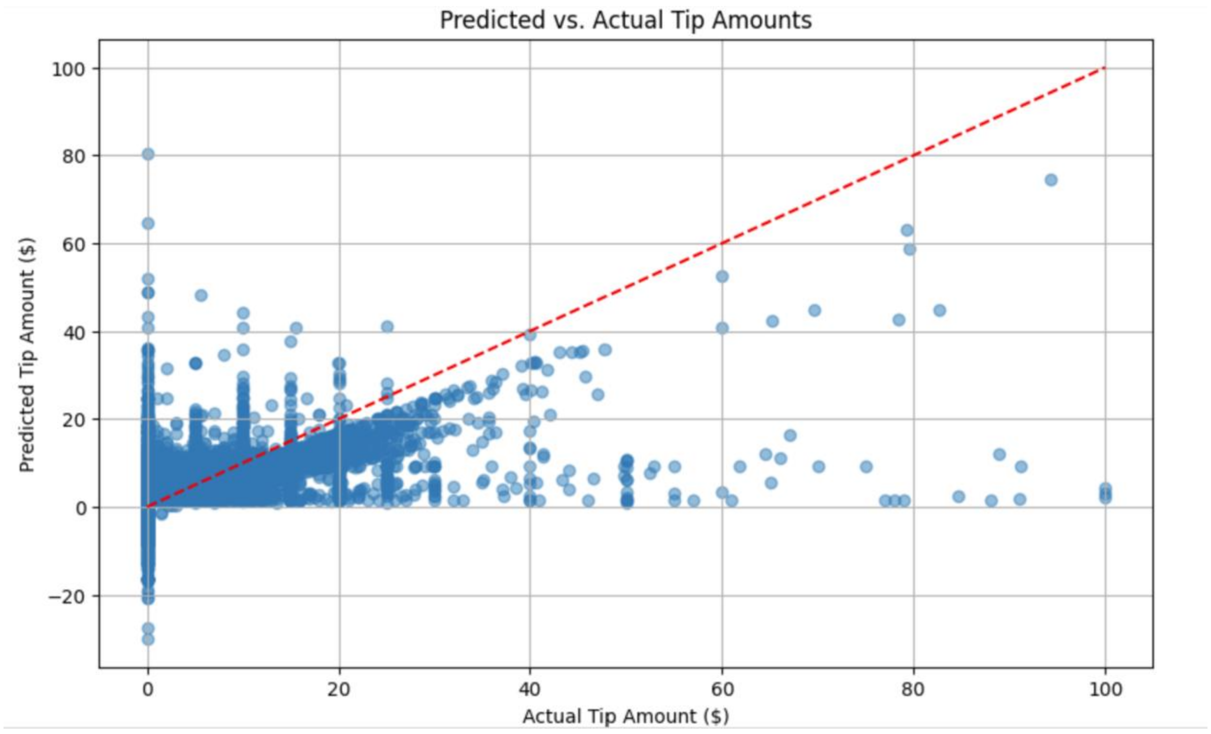
```
[455] import statsmodels.api as sm
      import matplotlib.pyplot as plt

      # Calculate residuals
      residuals = model.resid

      # Plot residuals against predicted tip amounts
      plt.figure(figsize=(10, 6))
      plt.scatter(model.fittedvalues, residuals, alpha=0.5)
      plt.xlabel('Predicted Tip Amount ($)')
      plt.ylabel('Residuals')
      plt.title('Residual Plot')
      plt.grid(True)
      plt.axhline(y=0, color='red', linestyle='--')  # Add horizontal line at y=0
      plt.show()
```



3. Actual Vs Predicted Plot:

Our fitted model has points that closely follow the diagonal line (y = x), indicating that the model's predictions closely match the actual values. Also, some values are deviated from this diagonal line suggesting that there are some discrepancies between the predicted and actual tip amounts.

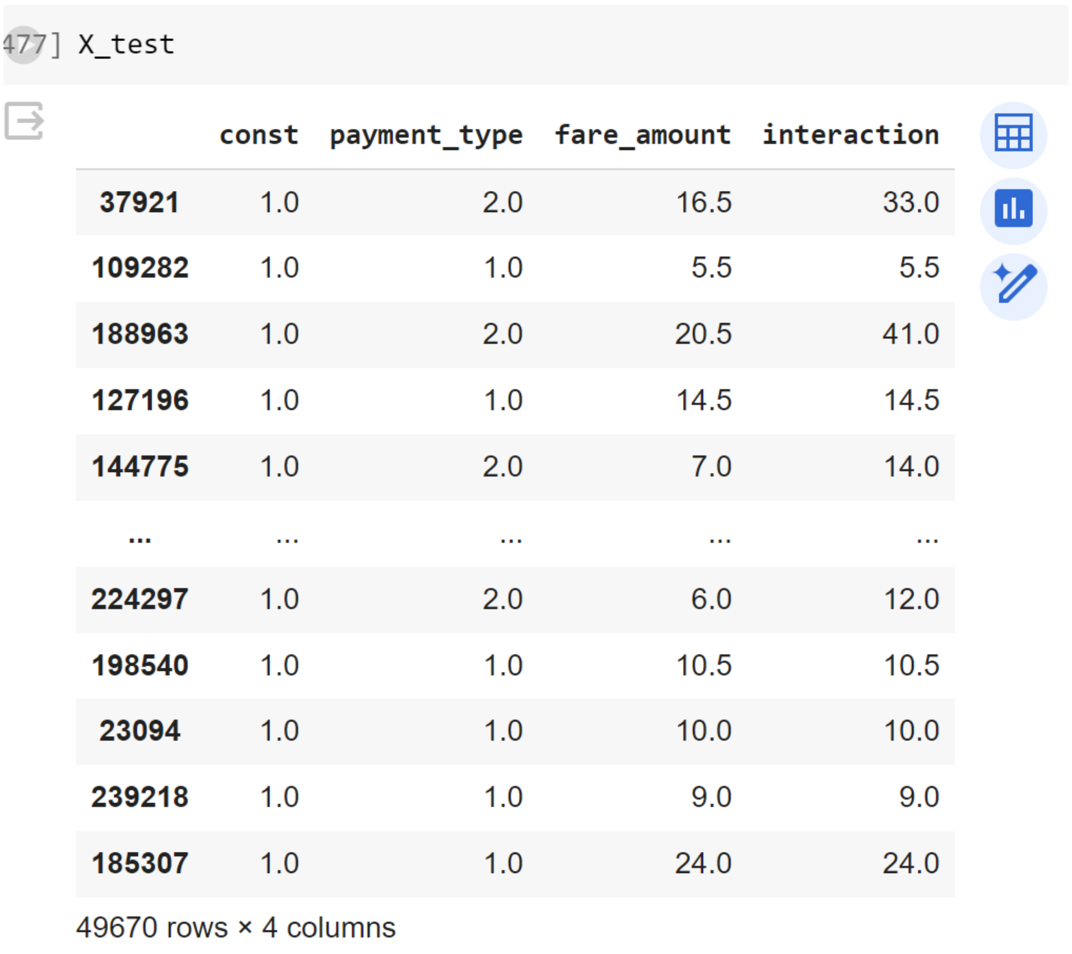

```
import statsmodels.api as sm
import matplotlib.pyplot as plt

# Get predicted tip amounts
predicted_tip_amounts = model.predict(X)

# Plot predicted vs. actual tip amounts
plt.figure(figsize=(10, 6))
plt.scatter(y, predicted_tip_amounts, alpha=0.5)
plt.plot([y.min(), y.max()], [y.min(), y.max()], color='red', linestyle='--')
plt.xlabel('Actual Tip Amount ($)')
plt.ylabel('Predicted Tip Amount ($)')
plt.title('Predicted vs. Actual Tip Amounts')
plt.grid(True)
plt.show()
```



Using the model to predict tip amounts in test data:

Here is the actual test data set we have considered:

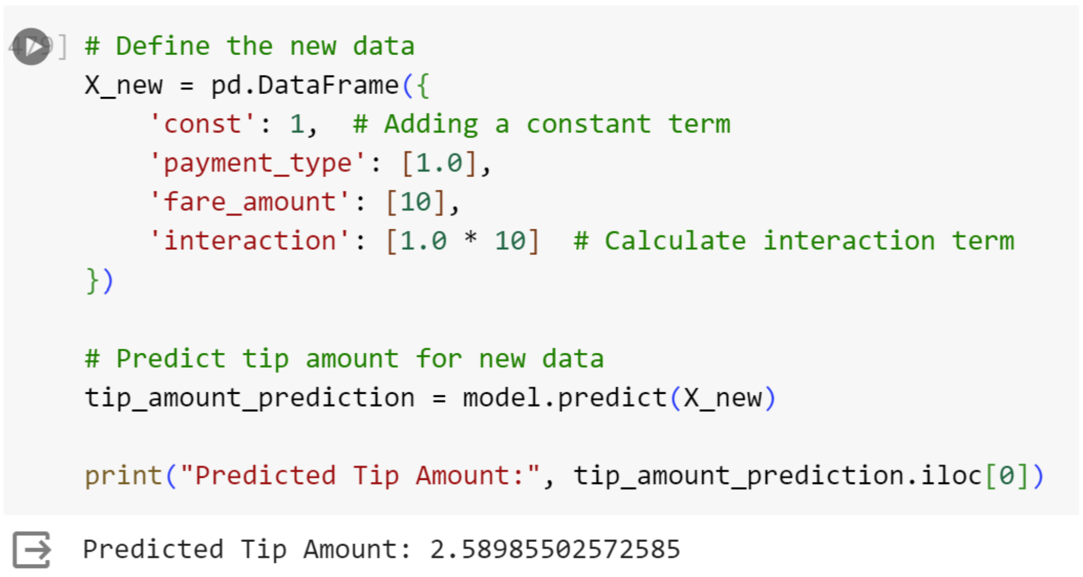
```
478] y_test

     37921      0.00
     109282     2.32
     188963     0.00
     127196     1.00
     144775     0.00
                ...
     224297     0.00
     198540     1.00
     23094      2.66
     239218     3.32
     185307     6.95
     Name: tip_amount, Length: 49670, dtype: float64
```



```
[477] X_test
```

	const	payment_type	fare_amount	interaction
37921	1.0	2.0	16.5	33.0
109282	1.0	1.0	5.5	5.5
188963	1.0	2.0	20.5	41.0
127196	1.0	1.0	14.5	14.5
144775	1.0	2.0	7.0	14.0
...	...	...	...	...
224297	1.0	2.0	6.0	12.0
198540	1.0	1.0	10.5	10.5
23094	1.0	1.0	10.0	10.0
239218	1.0	1.0	9.0	9.0
185307	1.0	1.0	24.0	24.0

49670 rows × 4 columns



```
# Define the new data
X_new = pd.DataFrame({
    'const': 1,  # Adding a constant term
    'payment_type': [1.0],
    'fare_amount': [10],
    'interaction': [1.0 * 10]  # Calculate interaction term
})

# Predict tip amount for new data
tip_amount_prediction = model.predict(X_new)

print("Predicted Tip Amount:", tip_amount_prediction.iloc[0])
```

```
Predicted Tip Amount: 2.58985502572585
```

Here we have taken the values from the test data set as payment_type: 1.0, fare_amount: 10, and predicted tip_amount as 2.58$. The actual value for this row values in the test data is 2.66$.

Results of initial modeling:

Overall, while the initial modeling effort provided valuable insights into the relationship between independent variables and tip amounts, there is potential for improvement to achieve more accurate predictions. Further iterations of modeling and analysis may lead to a better understanding of tipping behavior in taxi rides and improve the model's predictive performance.

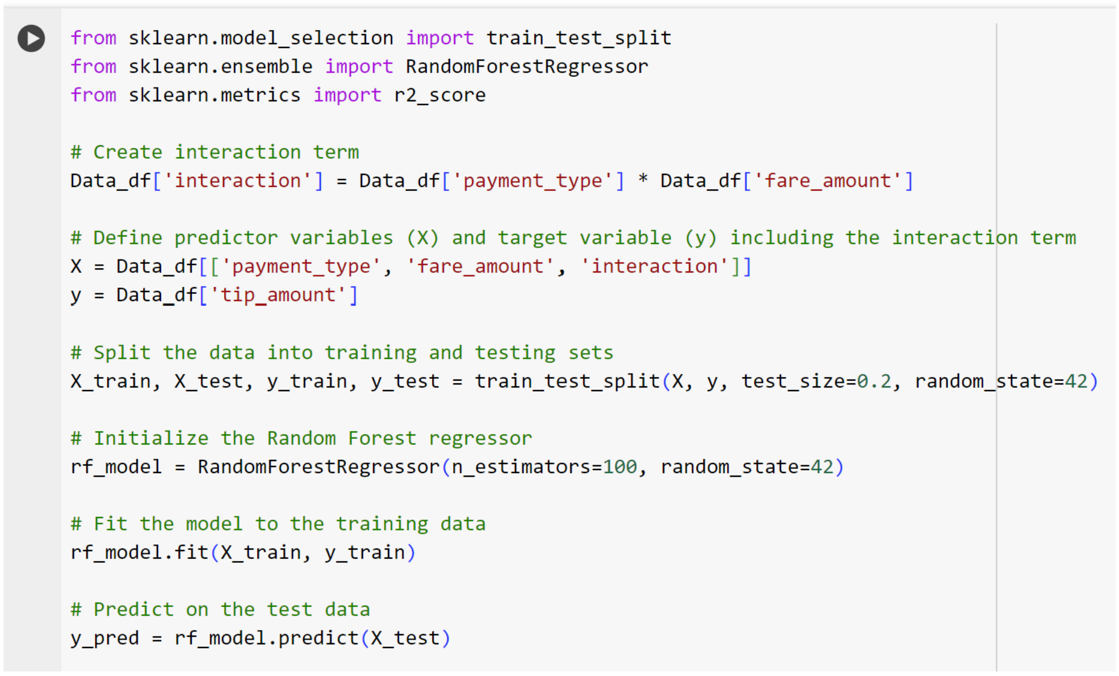
Final Modeling:

Random Forest model has been used to predict the tip_amount based on the predictors fare_amount and payment_type.

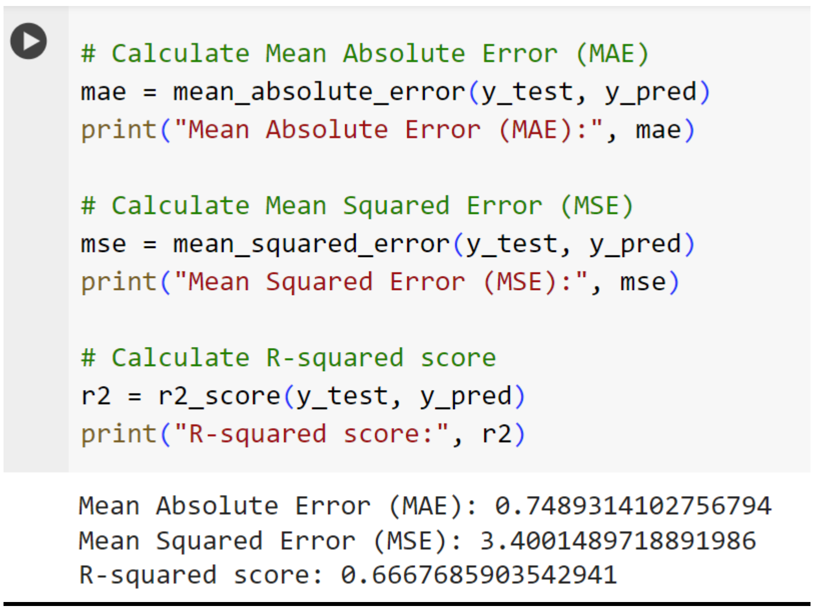
Similar to linear regression we have separated the data **"Data_df"** into training and testing sets in an 80:20 ratio. Later we performed Random Forest regression on 80% training data and predicted the values for the 20% testing set.

Also, we included the interaction term between payment_type and fare_amount for prediction.

Random Forest Model:



```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import r2_score

# Create interaction term
Data_df['interaction'] = Data_df['payment_type'] * Data_df['fare_amount']

# Define predictor variables (X) and target variable (y) including the interaction term
X = Data_df[['payment_type', 'fare_amount', 'interaction']]
y = Data_df['tip_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)

# Initialize the Random Forest regressor
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)

# Fit the model to the training data
rf_model.fit(X_train, y_train)

# Predict on the test data
y_pred = rf_model.predict(X_test)
```



```python
# Calculate Mean Absolute Error (MAE)
mae = mean_absolute_error(y_test, y_pred)
print("Mean Absolute Error (MAE):", mae)

# Calculate Mean Squared Error (MSE)
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error (MSE):", mse)

# Calculate R-squared score
r2 = r2_score(y_test, y_pred)
print("R-squared score:", r2)
```

```
Mean Absolute Error (MAE): 0.7489314102756794
Mean Squared Error (MSE): 3.4001489718891986
R-squared score: 0.6667685903542941
```

The Mean Absolute Error (MAE) of 0.7489 suggests that, on average, the model's predictions differ by approximately $0.7489 from the actual tip amounts. With an average squared discrepancy between expected and actual tip amounts of around $3.4001, the Mean Squared Error (MSE) of 3.4001 is reported. It can be inferred from the R-squared score of 0.6667 that the independent variables account for about 66.67% of the variation in tip amounts.

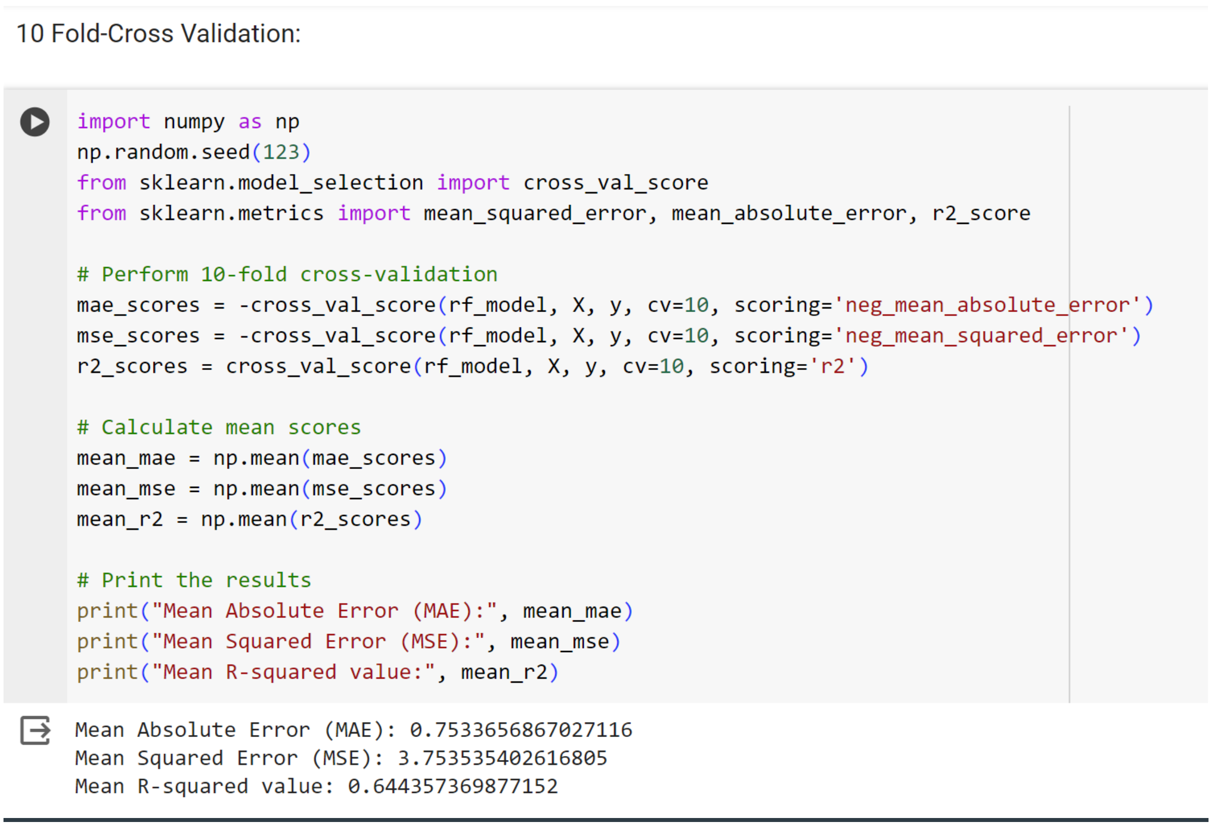
Model quality:

Cross Validation:

A Random Forest regression model's performance is assessed by 10-fold Cross-validation. Rather of dividing the data into training and testing sets only once, it divides the data into ten distinct folds, or subsets. It then repeats this procedure ten times, each time holding out a different fold for testing, training the model on nine folds and evaluating it on the last fold.

Metrics:

Overall the resultant values of **MAE: 0.7533**, **MSE:3.7535** and **R-squared: 0.644** after performing 10-fold cross-validation are similar to the random forest model performed one fold suggesting that the model performance is good and we can predict tip amounts based on the predictors behavior.



10 Fold-Cross Validation:

```python
import numpy as np
np.random.seed(123)
from sklearn.model_selection import cross_val_score
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

# Perform 10-fold cross-validation
mae_scores = -cross_val_score(rf_model, X, y, cv=10, scoring='neg_mean_absolute_error')
mse_scores = -cross_val_score(rf_model, X, y, cv=10, scoring='neg_mean_squared_error')
r2_scores = cross_val_score(rf_model, X, y, cv=10, scoring='r2')

# Calculate mean scores
mean_mae = np.mean(mae_scores)
mean_mse = np.mean(mse_scores)
mean_r2 = np.mean(r2_scores)

# Print the results
print("Mean Absolute Error (MAE):", mean_mae)
print("Mean Squared Error (MSE):", mean_mse)
print("Mean R-squared value:", mean_r2)
```

```
Mean Absolute Error (MAE): 0.7533656867027116
Mean Squared Error (MSE): 3.753535402616805
Mean R-squared value: 0.644357369877152
```

Suggestions for future work and improvements:

Enhanced Feature Engineering:
Explore additional features or transformations that may capture more information relevant to tip amount prediction, such as time-of-day effects, weather conditions, or driver behavior metrics.

Model Ensemble Techniques:
Investigate the use of ensemble methods like stacking or boosting to combine multiple models for improved prediction accuracy and robustness.

Incorporate External Data:
Integrate the external datasets, such as economic indicators or demographic information, to enrich the predictive model and capture additional factors influencing tip amounts.

Conclusions of the experiment:

In conclusion, the experiment focused on understanding the behavior of the tip amount response variable about the predictors fare_amount and payment_type.

The correlation between the tip amount and fare amount is moderately positive, implying that as the fare amount increases, the tip amount also tends to rise. This indicates that passengers are likely to offer more generous tips for longer or pricier rides.

The tip amount exhibits a moderate negative correlation with the payment type in our analysis, where passengers paying with credit cards are more likely to give tips when compared to passengers who are paying with cash or other types.

Overall, this analysis will help the taxi companies to plan the fare amounts for the rides and also offer some special discounts for passengers who are paying using credit cards. This will ultimately result in customer satisfaction and improve the chance of getting more tips.

Appendix A: Statement of goals achieved by the team

As part of our data science project, our team met several important goals. We carefully gathered and processed data to make sure it was good enough to use for research. Our research using advanced statistical methods and machine learning algorithms helped us learn a lot about what makes taxi customers tip, especially how much they pay and how they pay it. Our in-depth exploratory data analysis and regression modeling gave us suggestions for how to improve customer service and make business plans work better. By using compelling visualizations and reports to share our findings, we can give stakeholders the information they need to make smart choices. Our joint approach also encouraged a culture of new ideas and constant improvement, which made the project a success and paved the way for more data-driven projects in the future.

Appendix B: Statement of goals achieved by each person individually

Saran Kumar: Saran played a crucial role in data collection and preprocessing, ensuring the dataset's integrity and quality. He also led the exploratory data analysis, uncovering valuable insights into fare amounts and payment methods impact on tipping behavior.

Ramya Alluri: Ramya led the visualization analysis, examining the relationships between fare amounts, payment methods, and tip amounts. She identified significant predictors that contributed to the project by conducting correlation analysis.

Gowthami: Gowthami focused on developing machine learning models to predict tip amounts based on fare amounts and payment methods. She fine-tuned the models for optimal performance and conducted rigorous evaluations to assess their accuracy and effectiveness

Google Colab link of our analysis:

https://colab.research.google.com/drive/1E6uqlW4mazOb8xSrgJmLpkVVqMUYanwG?usp=sharing