Univerisity of Illinois

Springfield, Illinois

Final Project

A Machine Learning Approach to Classifying Uber Driver Tips

By

Koushik Anumalla

Rutuja Bhairawakar

Contents

Abstract

This project explores a machine learning approach to predict whether the uber driver will get a Tip or not. The dataset comprises metrics such as Vendor ID, trip distance, passenger count, fare amount, tolls amount and some more related attributes, which were preprocessed to handle missing values, duplicates, statistical and Data Analysis, outliers, and feature selection for optimal model performance. Classification algorithms such as Logistic Regression, neural networks, SVM and KNN are used to predict if Tip is granted or not. The results demonstrate the effectiveness of the proposed model in providing the output we want. Future work may explore accuracy, precisions of the models.

1. Introduction

1.1 Problem Definition and Project Goals

Tipping is a way to appreciate the driver's service, especially if they provided a comfortable, safe, and timely ride. It recognizes the effort they put into navigating traffic, ensuring passenger comfort, and sometimes assisting with luggage. Moreover, driving a cab involves long hours, navigating stressful traffic conditions, and dealing with diverse passengers. Tips provide a small morale boost and acknowledgment of these challenges. Thus, we decided to work on this tipping system and determine, if a cab driver will get a tip or not.

For the cab data, we're using uber data here. The dataset was downloaded from github website.
Here is the link to access it:
https://github.com/darshilparmar/uber-etl-pipeline-data-engineering-project/tree/main/data
The main reason and plan behind selecting this data set is because of the features available in this dataset which is essential to classify driver's tip.

1.2 Related Work

Below is the paper found with the similar topic. Here is the link to access it. *https://www.atlantis-press.com/proceedings/pesd-23/125997022*

The dataset they're using is New York taxi data and it has a total of 271,519 records. The author of that paper has used two step model, in which first he has used classification model to predict if the tip will be given or not and regression is used to determine, the amount of tip.

While handling for the classification model, the author has done preprocessing, handling missing data and has shown in graphical format the impact of each variable on the tipping. Further, the author has done, Normalization, fine-tuned the hyperparameters and prediction. The same is followed for the Regression model, in which the author again has done data processing, impact of variables on tipping shown in graphical format, One-hot encoding, tuning the hyperparameters and then have done the prediction.

The author has concluded the paper, with a 98.3 % accuracy, using classification model and using the regression model the has achieved the goal of predicting the tip amount.

2. Data Exploration and Preprocessing

2.1 Data Exploration

The dataset *uber_Data* has been loaded successfully, with no missing values or empty strings. Initial inspection using the *str()* function revealed the structure and data types of the variables, while the *summary()* function provided descriptive statistics.

The initial dataset uber_Data consists of 100,000 observations and 19 variables. These include VendorID, tpep_pickup_datetime, tpep_dropoff_datetime, passenger_count, and trip_distance. Other variables are pickup_longitude, pickup_latitude, RatecodeID, store_and_fwd_flag, dropoff_longitude, and dropoff_latitude. Additional details include payment_type, fare_amount, extra, mta_tax, tip_amount, tolls_amount, improvement_surcharge, and total_amount. These are the available variables for analysis before performing any feature engineering.

Nominal Variables	Ordinal Variables	Numeric Variables
VendorID	pickup_date	total_amount
RatecodeID	dropoff_date	fare_amount
pickup_day_of_week	pickup_time_bin	extra
dropoff_day_of_week		mta_tax
tip_given		tip_amount
payment_type		improvement_surcharge
store_and_fwd_flag		passenger_count
		trip_distance
		ride_duration

2.2 Preprocessing Steps

2.2.1 Data Cleaning

Splitting date and time into separate variables.

Data cleaning begin with splitting date and time variable into mulitiple usefull separate variable. The dataset is enhanced by adding several new features. A binary variable, tip_given, is created to indicate whether a tip was provided based on the tip_amount. The pickup_date and dropoff_date are extracted from the respective timestamps, while the day of the week is captured for both pickup_day_of_week and dropoff_day_of_week. The duration of each ride is calculated in minutes and stored in the ride_duration variable. Additionally, the pickup times are categorized into four periods: "Early Morning," "Morning," "Afternoon," and "Evening," based on the hour of the day in the pickup_time_bin variable.

Removing unnecessary variables and uninformative data.

The dataset is further refined by removing certain variables to reduce complexity and prevent overfitting. The geographical information, including pickup_longitude, pickup_latitude, dropoff_longitude, and dropoff_latitude, is removed as it is not necessary for the analysis. Additionally, the *RatecodeID* variable is filtered to remove records with a value of 6, as it is considered noise and uninformative due to having only one unique entry.

Handling missing values.

Then, the dataset is checked for missing values using the *colSums* function, which identifies any NA values or empty strings. The result confirms that there are no missing values or empty strings in the dataset.

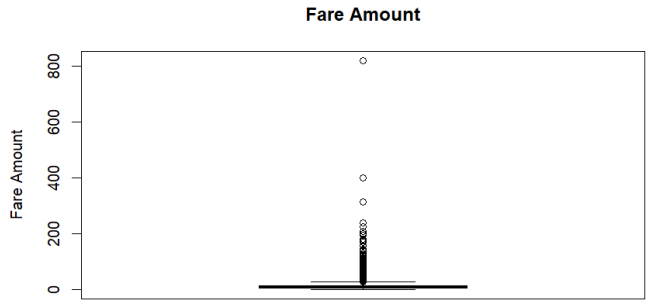
Removing negative fares and duplicates.

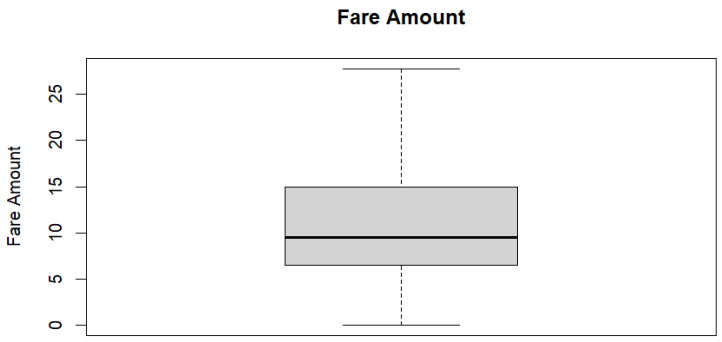
After, the dataset is checked for duplicate records using the *duplicated* function. Any duplicates are identified, and the number of duplicate rows is displayed. These duplicates are then removed from the dataset, and the updated number of rows is printed. Additionally, rows with a negative fare_amount are filtered out.

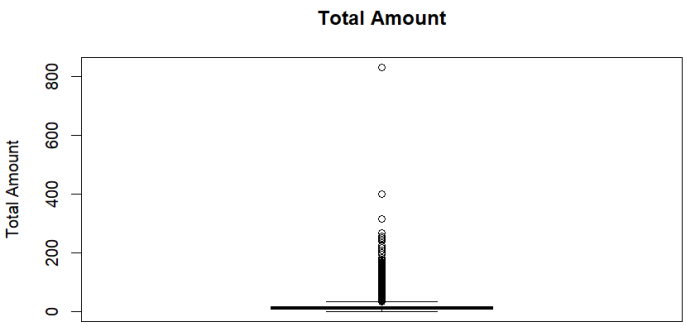
2.2.2 Outlier Detection and Removal

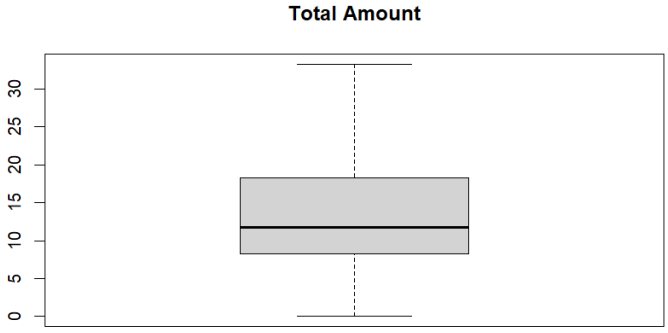
Thirty percent of the dataset is used for testing, while the remaining seventy percent is used for training. In the training set, the distribution of the target variable, tip_given, is examined and displayed as percentages. The interquartile range (IQR) approach is used to cap values in columns such as fare_amount, total_amount, ride_duration, trip_distance, tip_amount, and tolls_amount in order to handle outliers. The training and testing sets are both subjected to the computed boundaries. Following this, boxplots are made to visually evaluate the distribution of these variables in the training set, and summary statistics are examined to determine the effect of eliminating outliers.

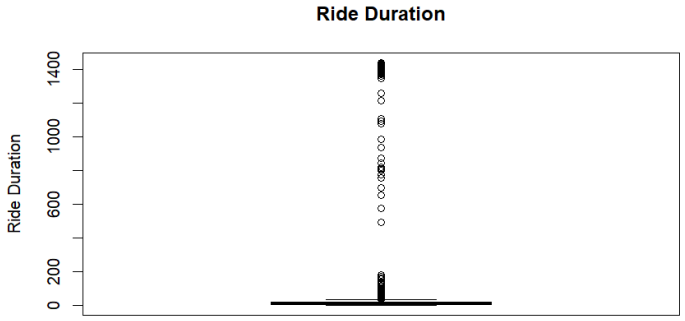
The tolls_amount variable, having bounds of 0 to 0, is removed from both the training and testing datasets using the subset function. Here are graphs before and after removing outliers.

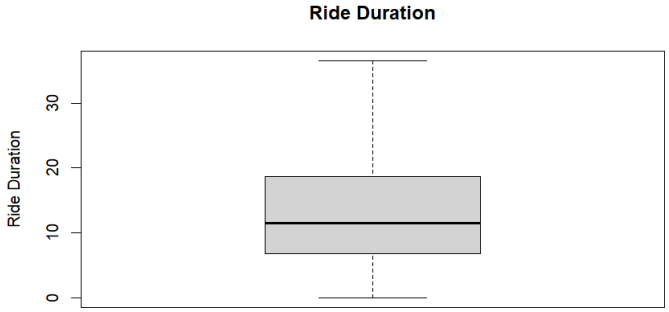


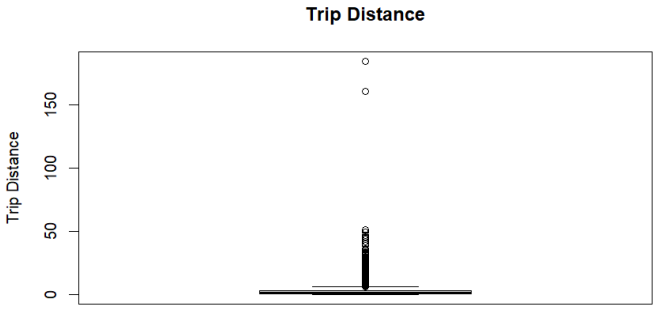


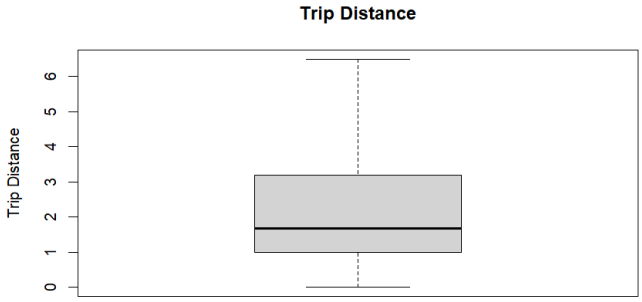


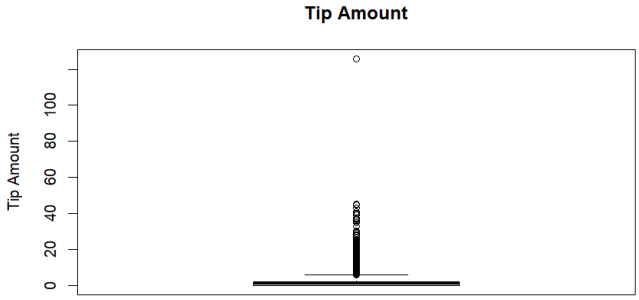


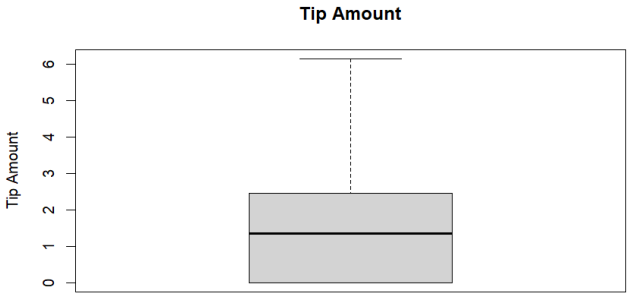












2.2.3 Handling Categorical Variables

Converted pickup_date, dropoff_date, and pickup_time_bin columns into ordered factors for both the training and test datasets. It sets predefined levels for these variables to ensure they maintain their correct order (e.g., dates and times). In this way we can improve how the model interprets and handles these variables during training.

2.2.4 Data Transformation

Performing one-hot encoding separately on train and test datasets for nominal attributes.

In this step, categorical variables such as VendorID, RatecodeID, store_and_fwd_flag, payment_type, pickup_day_of_week, and dropoff_day_of_week are one-hot encoded. In this way the nominal variables are converted to numeric format by creating binary columns for each unique level in the variable. A custom function, one_hot_encode, is used to transform these variables by creating new columns for each category, where each column is marked with a 1

or 0 to indicate the presence or absence of a category. This encoding is applied to both the training and test datasets. By one-hot encoding these categorical features, the data is converted into a format that can be used further in the process.

Performing normalization on train and test datasets.

The numeric variables in the training and test datasets are normalized to have a mean of 0 and a standard deviation of 1. The normalization formula applied is:

Formula: (D - x) / y)

where D is the original value, ‘x’ mean of the training data, and ‘y’ is the standard deviation of the training data. First, the means and standard deviations of the numeric columns in the training dataset are calculated. A custom normalize function is then applied to both the training and test datasets using these values. This normalization process standardizes the numeric features, making them suitable for ML models.

2.2.5 Addressing Class Imbalance

In this step, under-sampling is applied to the majority class to address the imbalance in the target variable. The dataset is first split into two subsets based on the tip_given, one for low tips (tip_given == "0") and one for high tips (tip_given == "1"). Then, the high subset is randomly sampled to match the size of the low tip subset, ensuring a balanced distribution. Finally, the two subsets are combined to create a balanced training dataset, improving the model's ability to learn from both classes equally. This method is choosed to avoid overfitting.

3. Data Analysis and Experimental Results

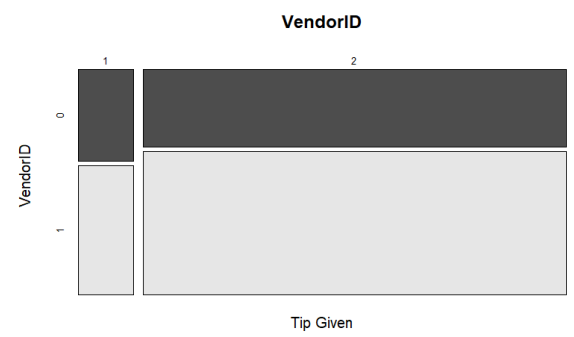
3.1 Statistical Analysis

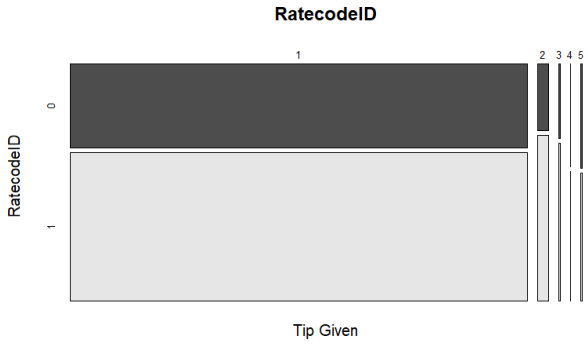
Categorical nominal variables such as VendorID, RatecodeID, store_and_fwd_flag, payment_type, pickup_day_of_week, and dropoff_day_of_week for their relationship with the target variable, tip_given, using Chi-Square tests. Contingency tables and mosaic plots help identify significant associations, retaining variables with p-values below 0.05 for further analysis.

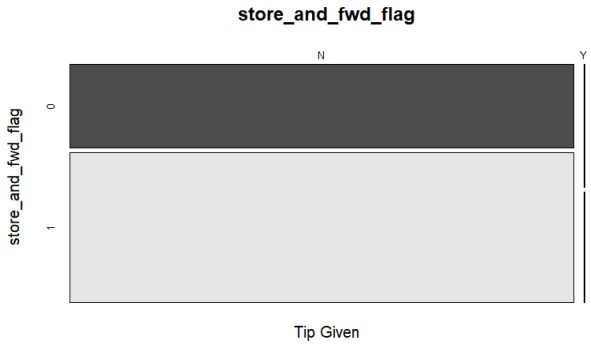
Next, for categorical ordinal variables like pickup_time_bin, pickup_date, and dropoff_date, the Kruskal-Wallis test is applied to explore differences in tipping patterns across levels of these variables, supported by boxplots for visual comparison. Significant variables, based on a p-value threshold of 0.05, are kept. Numeric variables such as fare_amount, trip_distance, tip_amount, total_amount, ride_duration, passenger_count, extra, mta_tax, and improvement_surcharge are analyzed using t-tests to compare means between tipping and non-tipping groups, with boxplots ilustrating the results.

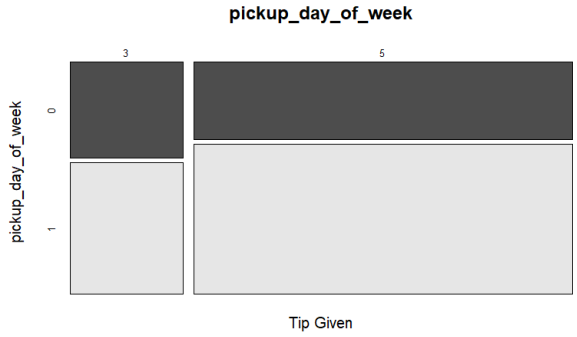
An additional ANOVA test assesses the relationship between fare_amount and payment_type, with visualizations highlighting fare distribution by payment type. The findings are summarized, and decisions based on p-values are presented through bar plots and dot charts.

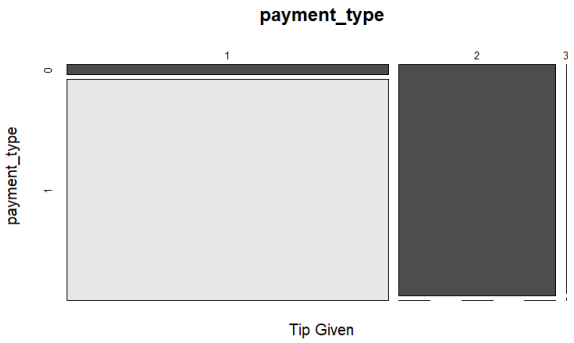
Visulization of variable.

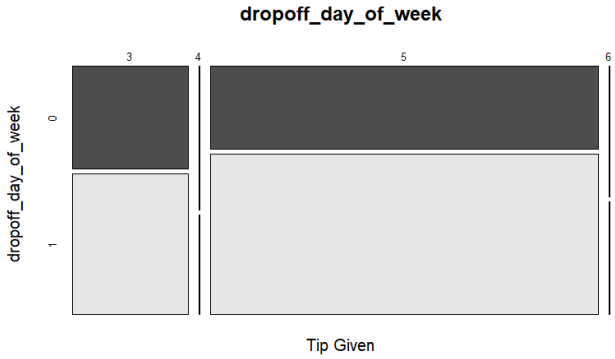


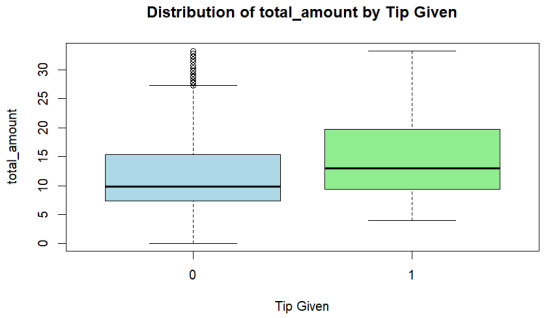


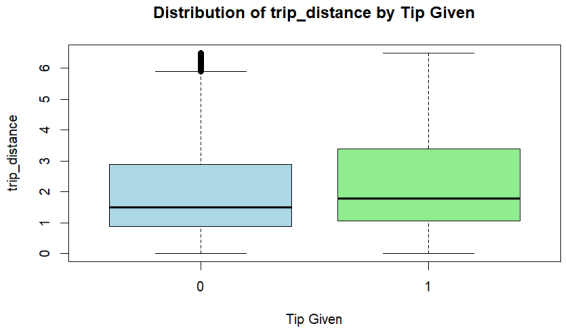


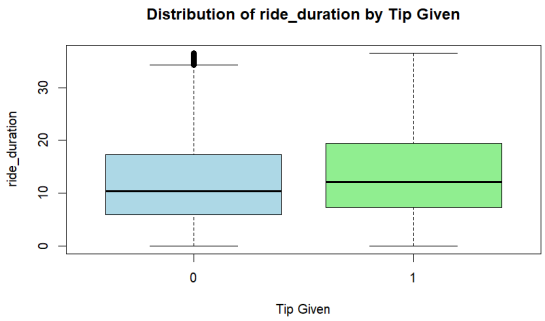


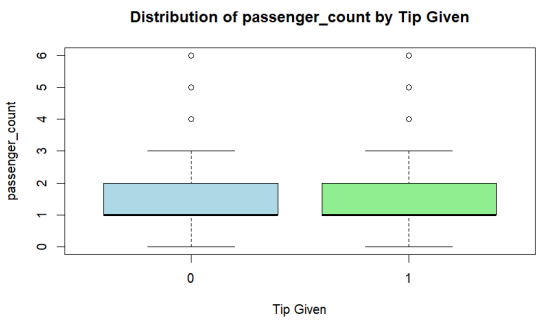


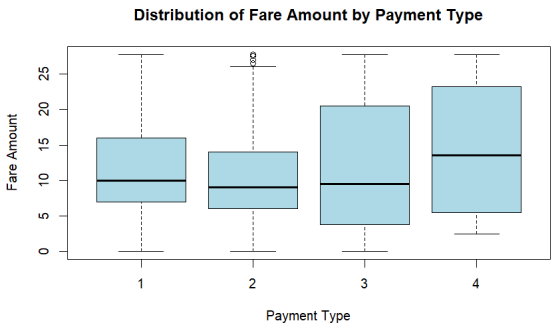


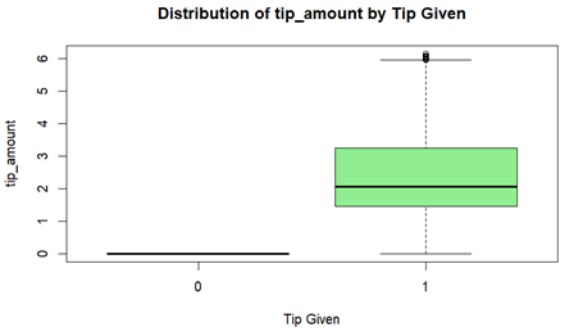


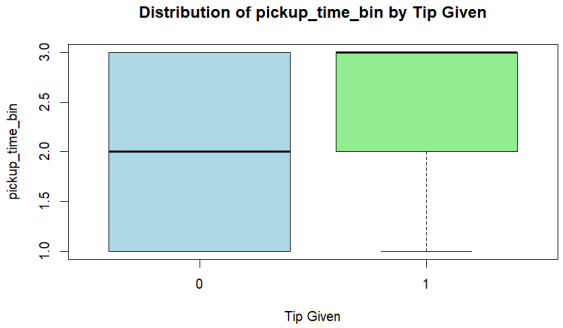


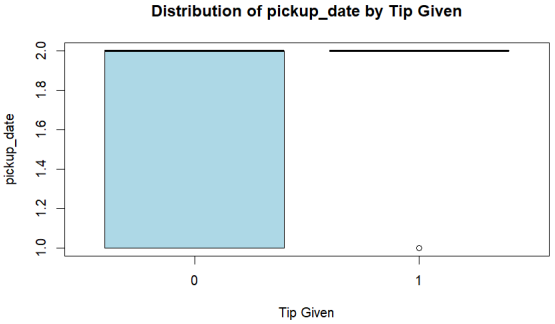

Distribution of ride_duration by Tip Given
ride_duration
0
10
20
30
0
1
Tip Given

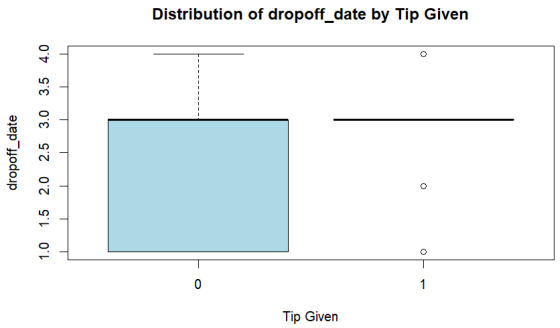

Distribution of passenger_count by Tip Given
passenger_count
0
1
2
3
4
5
6
0
1
Tip Given

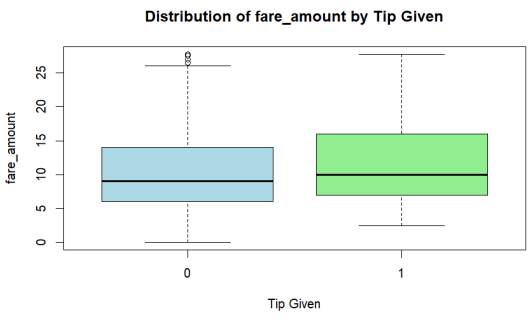

Distribution of Fare Amount by Payment Type
Fare Amount
0
5
10
15
20
25
1
2
3
4
Payment Type


Distribution of tip_amount by Tip Given
tip_amount
0
1
2
3
4
5
6
0
1
Tip Given


Distribution of pickup_time_bin by Tip Given
pickup_time_bin
1.0
1.5
2.0
2.5
3.0
0
1
Tip Given


Distribution of pickup_date by Tip Given
pickup_date
1.0
1.2
1.4
1.6
1.8
2.0
0
1
Tip Given


Distribution of dropoff_date by Tip Given
dropoff_date
1.0
1.5
2.0
2.5
3.0
3.5
4.0
0
1
Tip Given


Distribution of fare_amount by Tip Given
fare_amount
0
5
10
15
20
25
0
1
Tip Given

3.2 Correlation Analysis

Performed hypothesis testing for various variables in the dataset to identify their relevance to the target variable, tip_given. Chi-Square tests were applied to categorical variables, Kruskal-Wallis tests for ordinal variables, and T-tests for numeric variables. All tests showed highly significant p-values (below 0.05), leading to the decision to keep all variables, indicating their importance in predicting tip amounts and minimizing redundancy by retaining relevant features.

3.3 Model Training and Testing

Logistic Regression Model

For binary classification, the logistic_regression_model function makes use of logistic regression with L2 regularization (ridge regression) through the glmnet package. Its inputs are train_data, test_data, and target_column, where target_column indicates the binary target variable and train_data and test_data are the datasets. The function transforms the feature data into a training matrix, separates the target column from the features, and changes the target to binary values. The model is used to estimate probabilities on the test data after being trained with cross-validation and L2 regularization (alpha = 0). A threshold of 0.5 is used to translate predicted probabilities into binary labels. Metrics like accuracy, precision, recall, F1 score, and AUC-ROC are used to assess performance and give information about the model's capacity for categorization.LR

LR Results

Confusion Matrix		
	Predicted	
Actual	0	1
0	10390	493
1	2266	16565

Metric	**Value**
Accuracy	0.907148
Precision	0.879667
Recall	0.971099
F1-Score	0.923124
AUC	0.989094

K Nearest Neighbor Model

The knn_model function applies the k-Nearest Neighbors (KNN) algorithm to classify data, focusing on tuning the hyperparameter k, which determines the number of nearest neighbors to consider for making predictions. The function starts by preparing the training and test data, separating the target variable from the features. It then iterates over a range of k values, training and testing the model for each, calculating the accuracy for each configuration. The results are stored in a metrics dataframe, from which the optimal k value is selected based on the highest accuracy. The model is then retrained with this best k to make predictions on the test set.

In addition to determining the optimal k, the function computes various performance metrics, including the confusion matrix, accuracy, precision, recall, F1 score, and AUC. The confusion matrix helps assess the model's classification performance, while accuracy, precision, and recall provide insights into the model's ability to correctly identify positive and negative instances. The F1 score balances precision and recall, and the AUC score evaluates the model's

ability to distinguish between classes. The ROC curve is also generated to visualize the trade-off between sensitivity and specificity. This approach to tuning k ensures that the model achieves the best performance while avoiding overfitting or underfitting.

KNN Results

Confusion Matrix		
	Predicted	
Actual	0	1
0	10743	140
1	3850	14981

Metric	Value
Accuracy	0.86572
Precision	0.79555
Recall	0.990741
F1-Score	0.882481
AUC	0.891343

Multi-Layer Neural Networks

The neural_network_model function applies a deep learning approach using a neural network built with Keras. The model architecture is constructed sequentially, beginning with a dense layer that has 512 units and a ReLU activation function. This is followed by additional dense layers with 256, 128, and 32 units, each using the ReLU activation function, creating a progressively smaller representation of the data. The final layer is a dense layer with a single unit and a sigmoid activation function, which is suitable for binary classification tasks. The function also defines the optimizer as Adam with a learning rate of 0.0001 and uses binary cross-entropy as the loss function.

The function trains the model using the fit function, specifying 20 epochs and a batch size of 16, with a 20% validation split to monitor the model's performance during training. Once trained, the model makes predictions on the test data, which are thresholded at 0.5 to classify the results into binary outcomes. The model's performance is evaluated through several metrics, including accuracy, precision, recall, F1 score, and AUC, with a confusion matrix providing insights into true positives, false positives, and other classification results. The use of multiple hidden layers with decreasing unit sizes allows the model to learn complex patterns, while the final sigmoid output ensures binary classification is handled effectively.

NN Results

Confusion Matrix		
	Predicted	
Actual	0	1
0	10883	0
1	55	18776

Metric	Value
Accuracy	0.998149
Precision	0.997079
Recall	0.991099
F1-Score	0.998538
AUC	0.999942

Support Vector Machine

In the svm_model function, we use a Support Vector Machine to perform classification, choosing the "C-classification" type with a radial basis function kernel. We train the SVM on the provided training data, separating the independent variables from the target variable. The radial kernel is used to handle non-linear decision boundaries, which allows the model to capture more complex relationships in the data. To ensure that the model performs well, we set the cost parameter to 1, which controls the penalty for misclassifying data points, and use scale = TRUE to normalize the features, ensuring they have zero mean and unit variance. After

training, we use the model to make predictions on the test set, and evaluate its performance using various metrics such as accuracy, precision, recall, F1 score, and AUC. The radial kernel helps us map the data into a higher-dimensional space, making it easier to find an optimal separating hyperplane, which is crucial for handling complex datasets.

SVM Results

Confusion Matrix		
	Predicted	
Actual	0	1
0	10882	1
1	512	18319

Metric	Value
Accuracy	0.9827354
Precision	0.9728108
Recall	0.9999454
F1-Score	0.9861915
AUC	0.9863595

4. Conclusion

The results show that the dataset is imbalanced, meaning there are more examples of one class than the other. This affects how well the models perform, even though results show good accuracy, precision, and recall. Since the models are mostly detecting the majority class correctly, they may not be doing as well with the minority class, which they struggle to identify properly.

Even though accuracy and AUC scores are high, meaning the models are good at predicting the majority class, the recall for the minority class varies. This shows that some models are better at identifying the minority class than others. We used downsampling to address the imbalance, but further techniques like different performance metrics or adjustments to the sampling methods could help improve results. In the future, efforts should focus on improving how well the models recognize the minority class.

5. References

- "Machine Learning with R" by Brett Lantz, 4th edition.
- "Prediction of New York taxi tip behavior based on machine learning classification and regression methods" by Hejingyu Huang.
- Referred https://stackoverflow.com/ for error resolutions.

6. Contributions

Koushik focused on feature engineering, one-hot encoding, and addressing data imbalance. He also worked on building the NN and Logistic Regression models. Rutuja handled calculating and removing outliers, performing statistical analysis, normalizing the data and building KNN and SVM models. Both have contributed to report and PPT generation.