

**Capstone Project Final Report**

**College of Professional Studies, Northeastern University**

**Fall 2023’B**

**ALY6140: Python & Analytics System Technology**

**Prof.** **Daya Rudhramoorthi**

**Done By,**

**Aayush Amit Patil**

**Akash Tirupapuliyur Srikanth**

**INTRODUCTION**

**New York City Yellow Taxi App**

**Taxi and Limousine Commission**

TLC (Taxi and Limousine Commission) (Taxi and Limousine Commission) is an agency in numerous cities that supervises and licenses taxi taxis, for-hire automobiles, and ride-sharing services. TLC oversees taxi and for-hire vehicle regulation and licensing. This entails establishing regulations and standards for drivers, vehicles, and businesses to promote public safety and fair commercial operations. Trip data that has been aggregated and anonymized can be studied to better understand traffic patterns, service demand, and locations with high or low service availability. This data can be useful for urban planning and traffic management. Many cities and regulatory bodies, including TLC, require taxi and ride-sharing companies to submit trip record data regularly. This data typically includes information about each trip, such as:

**VendorID:** 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.

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

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

**Rate CodeID:** The final rate code 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.

**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. Currently this only includes the $0,50 and $1 rush hour and overnight charges.

**MTA\_tax:** $0.50 MTA tax that is automatically triggered based on the metered rate in use.

**Improvement\_surcharge:** $0.30 improvement surcharge assessed trips at the flag drop. The improvement surcharge began being levied in 2015.

**Tip\_amount:** Tip amount- This field is automatically populated for credit card tips. Cash tips are not included.

**Tolls\_amount:** total amount of all tolls paid.

**Total\_amount:** The total amount of charged to passengers. Does not include cash tips.

**Congestion\_Surcharge:** Total amount collected in trip for NYS congestion surcharge.

**Airport\_fee:** $1.25 for pick up only at LaGuardia and John F. Kennedy Airports.

The dataset's aim is to apply regression models to estimate the entire fee of the ride, which can then be used to monitor fare structures and guarantee that passengers are charged fairly for services. It helps to prevent problems like overcharging and ensures pricing transparency. The dataset, which is based on previous taxi trip data, provides critical information for constructing machine learning models targeted at calculating the time required for a ride. Pick-up and drop-off locations, trip time and date, weather conditions, and prior trip data all play important parts in the forecast process. Machine learning models, ranging from linear regression to more complex techniques like gradient boosting or neural networks, are trained on historical data after a process of feature selection, data preprocessing, and feature engineering. To ensure strong generalization of new, unknown data, the trained models are validated and tested using distinct datasets. Once validated, these models can be used to make real-time forecasts, assisting both passengers and service providers in optimizing their operations based on predicted travel times.

We also have a supporting dataset which gives us the names of the zones and boroughs for each location ID. The dataset consists of the following row:

**LocationID:** The Location ID number.

**Borough:** The Borough name.

**Zone:** The Zone name.

**Service\_zone:** Zones of the service.

**Questions to Investigate**

**Business Question 1** – How Regression models can be used to predict the total fare of the ride.

**MODULES**

**Pandas:** Module used to get data into tables to work on it through different commands.

**Numpy:** Module used to perform mathematical operations with ease.

**Sodapy:** Module used for getting data into our system through API’s.

**Matplotlib:** Module used for plotting data into charts.

**Seaborn:** Module used for creating visualizations.

**Scikit-learn:** Module used for machine learning built on top of SciPy and is distributed under the 3-Clause BSD license.

**Xgboost:** It is an open-source software library that implements optimized distributed gradient boosting machine learning algorithms under the Gradient Boosting framework.

**Time:** It is a module used to track time in python.

**Tkinter:** It is a module used to make Graphical user interface as a frontend for our system.

**Geopy:** It is a module used to generate geographical data from names of locations.

**EXPLORATORY DATA ANALYSIS**

A graph of a trip distance

Description automatically generated

**Explanation:**

The histogram indicates that the majority of taxi rides have relatively short distances, with a peak around 2-5 miles. However, there are likely longer-distance trips as well, represented by the right side of the distribution. There is a clear peak around 2-5 miles, it suggests that a significant number of trips cover short distances. A right-skewed distribution with a long tail towards higher distances may indicate that while short trips are common, there are also occasional longer journeys. There are spikes or clusters at round numbers such as 5 miles, 10 miles, it indicates that common routes or fixed distances within the dataset.

A graph of a number of black and white lines

Description automatically generated

**Explanation:**

From the above boxplot, the median fare amount come somewhere between $8 to $12. The majority of the fare falls withing range of $5 to $20. There are many outliers beyond the whiskers, it indicates some extreme values in the fare amounts. There are lot of outliers in the above boxplot like fare amount beyond $50 till $500. This indicates that, a person might have hired the taxi for more than one day.

A graph with green dots

Description automatically generated

There is a positive correlation, which suggests that greater trip distances have higher fare amounts on average. Outliers might represent unusual cases where the fare amount is exceptionally high for a given trip distance. Around 75 percent of the data points lie between 50-mile distance. The relationship between trip distance and fare amount appears to be linear

A graph of a number of locations

Description automatically generated

**Explanation**:

The chart provides a visual representation of the top 10 locations where taxi pickups are most frequent. Each bar represents a unique pickup location, identified by its location name.The height of each bar corresponds to the number of pickups at the respective location. Higher bars indicate locations with a higher frequency of taxi pickups, while lower bars represent less frequently used pickup locations. By examining the chart, you can quickly identify the specific locations that are the most popular for taxi pickups. These locations are crucial for taxi services and may correspond to busy transportation hubs, popular neighborhoods, or areas with high demand.

A graph of blue bars with black text

Description automatically generated

The chart provides a visual representation of the top 10 locations where taxi drop off locations which are most frequent. Each bar represents a unique drop off location, identified by its location name.The height of each bar corresponds to the number of drop off at the respective location. Higher bars indicate locations with a higher frequency of taxi drop offs, while lower bars represent less frequently used drop off locations. By examining the chart, you can quickly identify the specific locations that are the most popular for taxi drop off. These locations are crucial for taxi services and may correspond to busy transportation hubs, popular neighborhoods, or areas with high demand.

A graph of a graph of a graph

Description automatically generated with medium confidence

**Explanation**:

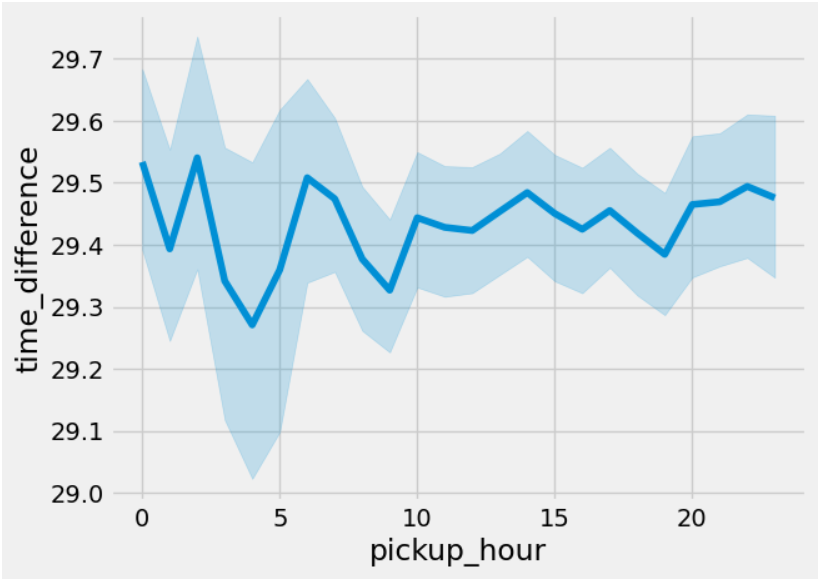
The chart above shows the distribution of tip amounts and fare amounts. From the data we can see a normal distribution of fare amounts. Tip amounts are right skewed which was expected. Median fare amount is 9 dollars which can be seen and the fare amounts vary between 0 to 15 dollars.

A graph of different colored squares

Description automatically generated

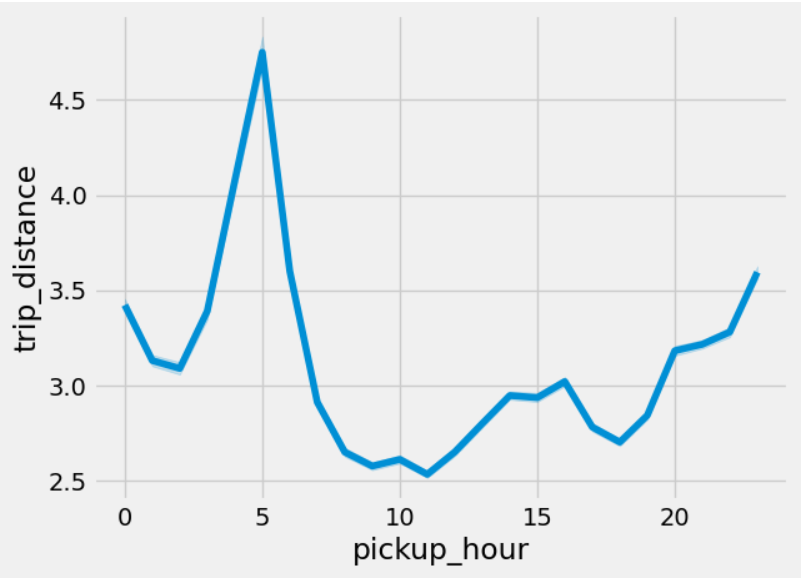
**Explanation**:

Here we can see the bar plot of the trip duration for each passenger volume. As we can see, the median for each passenger volume is nearly the same, which indicates that it is not a major factor affecting the trip duration.



**Explanation**:

The chart represents the distribution of trip durations over time of the day. It shows that trip durations are way lower from 3am to 5am. This can be attributed to lesser traffic on the streets during this time. Also, we can see the trip durations increase from 6am to 8am as they are the peak office travel hours.



**Explanation**:

Like the previous chart this graph represent Trip distances over the time of the day. One key observation is that from 3am to 5 am we see a longer trip distances as these are the times when people travel across the city or to the airports for their flights.

A graph of blue dots

Description automatically generated

**Explanation**:

Finally, we see the bubble plot of trip distance vs trip duration. We can see that there is a near uniform distribution. Which makes sure that trip duration and trip distance are not linearly related and hence cannot be used to derive relation.

**Predictive Models**

We have used five predictive models to predict the fare of the taxis.

1. Linear Regression Model
2. Decision Tree Regressor Model
3. AdaBoost Regressor Model
4. Gradient Boost Regressor Model
5. XGB Regressor Model

What is Linear Regression Model?

Linear regression is a statistical method used for modeling the relationship between a dependent variable and one or more independent variables. In simple linear regression, there is only one independent variable, while in multiple linear regression, there are two or more independent variables. The goal of linear regression is to find the best-fit line that represents the linear relationship between the variables. In a simple linear regression model, the relationship between the dependent variable (Y) and the independent variable (X) is represented by the equation:

Y = B0 + B1X + E

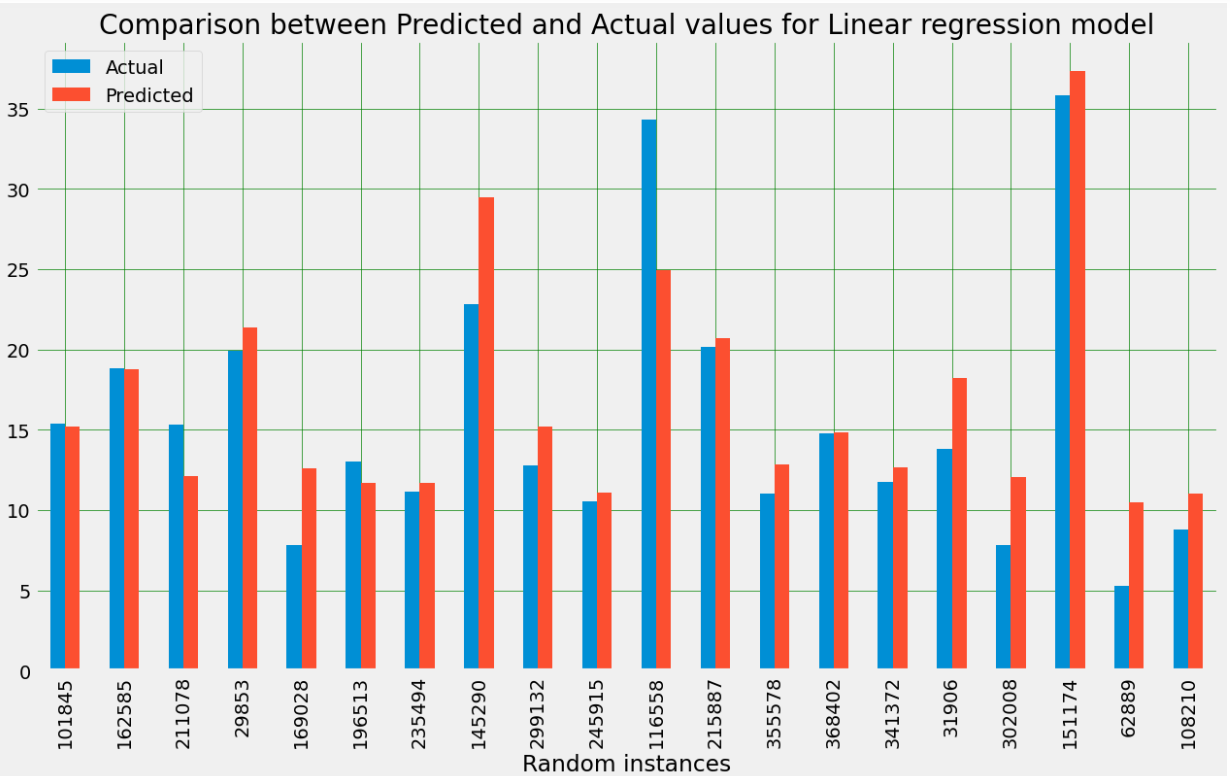
Y = Dependent Variable

X = Independent Variable

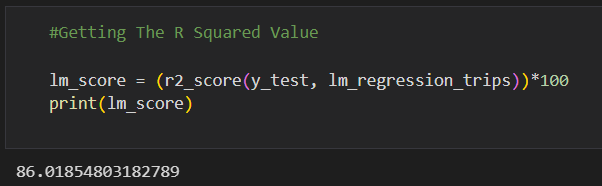
B0 = Intercept

B1 = Slope

E = Error Term.



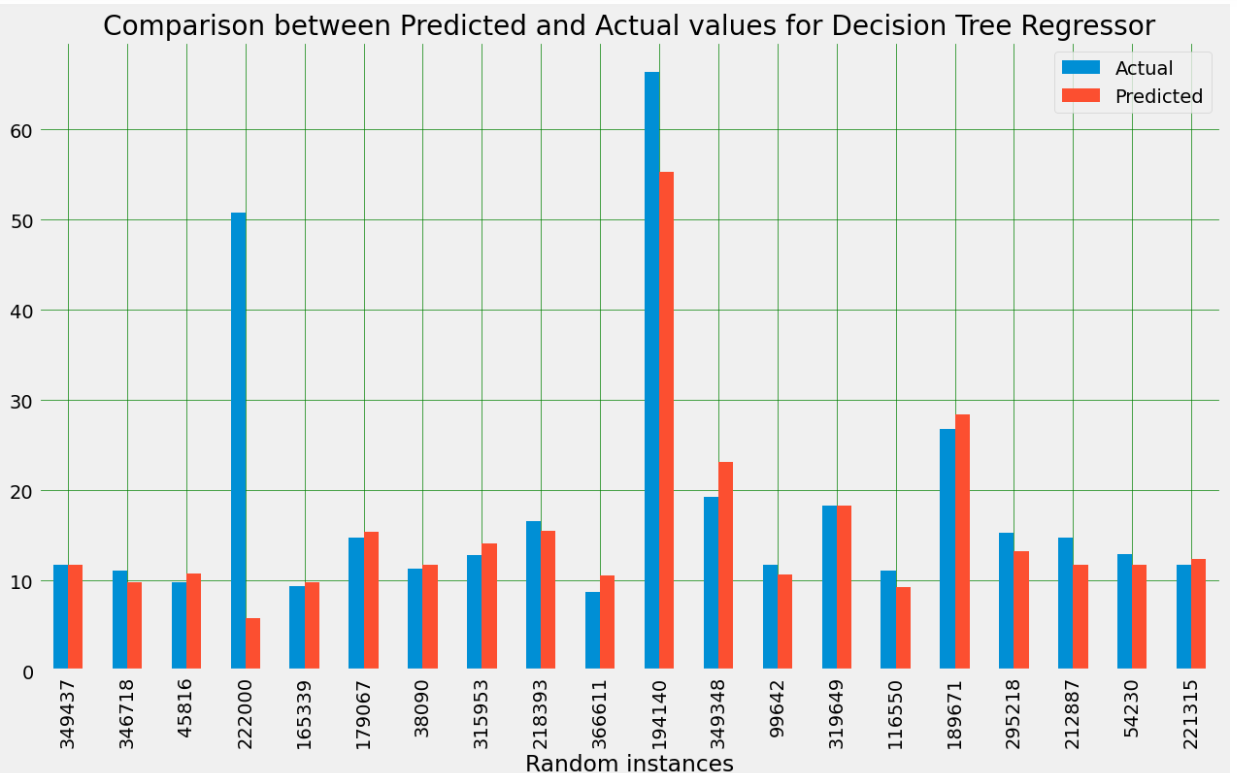
The bar plot visually compares the actual and predicted values for a random sample of 20 instances from the test set, as generated by the Linear Regression model. Each bar in the plot represents an individual instance, with the actual taxi fare depicted alongside the corresponding predicted value. The comparison aims to provide a qualitative assessment of how well the model is capturing the variability in taxi fares. A close alignment between the actual and predicted values suggests good predictive performance, while discrepancies may indicate areas where the model can be further refined. It's important to note that this visual inspection serves as an initial overview, and a more comprehensive evaluation should include quantitative metrics such as Mean Squared Error (MSE) or R-squared which is included in code file which will provide a better understanding of the model's accuracy and generalization on the entire test set.



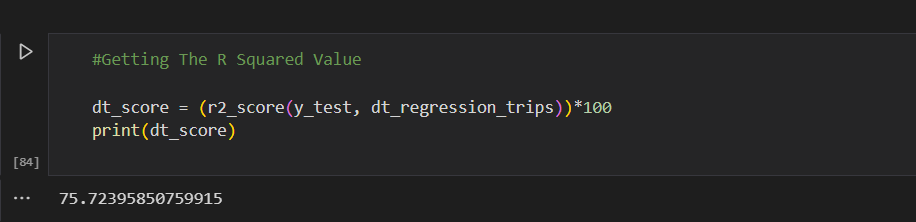
The provided code calculates the R-squared value as a percentage to assess the goodness-of-fit of the Linear Regression model on the test set. R-squared measures the proportion of the variance in the target variable (taxi fares) that is explained by the model. This code uses scikit-learn's R square function to compute the R-squared value. The result is then multiplied by 100 to express it as a percentage. The R-squared value ranges from 0% to 100%, where a higher value indicates a better fit of the model to the data. It provides insights into how well the Linear Regression model is explaining the variability in taxi fares on the test set.

What is Decision Tree Regressor Model?

A Decision Tree Regressor is a machine learning algorithm used for regression tasks, aiming to predict continuous numerical values. The algorithm builds a tree structure, where each internal node represents a decision based on a specific feature, and each leaf node contains the predicted value for the target variable. Nodes are split recursively based on features and thresholds to minimize variance within subsets. Decision Trees capture non-linear relationships in the data, offer interpretability, and are prone to overfitting, mitigated by techniques like pruning. The interpretability and ability to handle complex patterns make Decision Trees valuable for understanding feature importance in regression problems, while ensemble methods such as Random Forests enhance performance and reduce overfitting.



The bar plot, derived from a random sample of 20 instances, visually contrasts the actual taxi fare values with the predictions made by the Decision Tree Regressor model. Each bar in the plot signifies a specific instance, showcasing the side-by-side comparison of actual and predicted values. The plot serves as a quick and intuitive representation, offering insights into how well the model aligns with the true values for these sampled instances. Notably, instances with comparable actual and predicted values suggest the model's effectiveness in capturing the underlying patterns in the data. However, any notable disparities between actual and predicted values highlight potential areas for model refinement or further investigation.



The code snippet calculates the R-squared value as a percentage to assess the goodness-of-fit of the Decision Tree Regressor model on the test set. R-squared measures the proportion of the variance in the target variable (taxi fares, in this case) that is explained by the model. This code uses scikit-learn’ s r2 function to compute the R-squared value. The result is then multiplied by 100 to express it as a percentage. The R-squared value ranges from 0 to 1, where a higher value indicates a better fit of the model to the data. It provides insights into how well the Decision Tree Regressor model is explaining the variability in taxi fares on the test set.

**What is AdaBoost Regressor Model?**

AdaBoost (Adaptive Boosting) Regressor is a machine learning algorithm that belongs to the family of ensemble learning methods. It is specifically designed for regression tasks, where the goal is to predict a continuous numerical value. AdaBoost Regressor combines the predictions from multiple weak learners (typically shallow decision trees) to create a strong, accurate predictive model.

Here are the key characteristics of the AdaBoost Regressor:

Ensemble Learning:

AdaBoost is an ensemble learning method that builds a strong model by combining the predictions of multiple weak learners. Each weak learner is typically a simple decision tree.

Weighted Training:

During the training process, instances that are mis predicted by the current weak learner are assigned higher weights, making them more influential in the subsequent training of the next weak learner. This adaptive weighting helps the model focus on the instances that are more challenging.

Sequential Training:

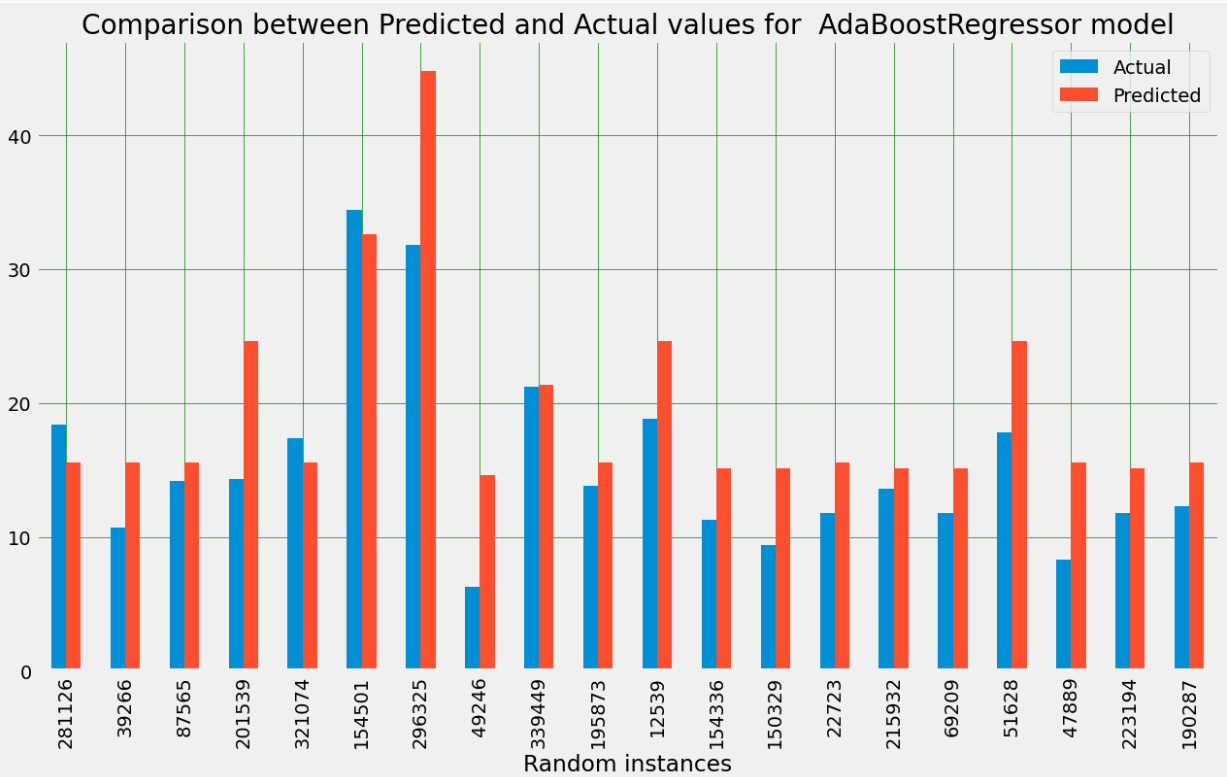
Weak learners are trained sequentially, and each subsequent learner corrects the errors made by its predecessor. The process continues until a predefined number of weak learners are trained or no further improvement can be achieved.

Combining Weak Learners:

Predictions from each weak learner are combined using weighted majority voting. The final prediction is the weighted sum of the individual weak learner predictions.

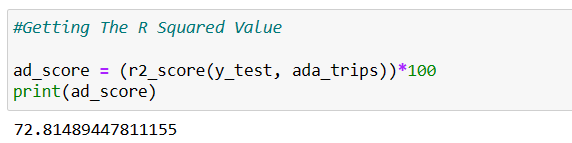
Robustness:

AdaBoost is robust and less prone to overfitting compared to individual weak learners. It can adapt to complex patterns in the data and improve performance.



The generated bar plot visually contrasts the actual and predicted values for a random sample of 20 instances from the test set, as predicted by the AdaBoostRegressor model. Each bar in the plot represents an individual instance, with the 'Actual' and 'Predicted' values depicted side by side. The comparison offers a qualitative assessment of how well the model is capturing the variation in the target variable. A close alignment between the actual and predicted values suggests good predictive performance, while discrepancies may indicate areas for further investigation or model improvement. It's important to note that this visual inspection serves as an initial overview, and a more comprehensive evaluation should include quantitative metrics such as Mean Squared Error (MSE) or R-squared for a nuanced understanding of the model's accuracy and generalization on the entire test set.

**Evaluation of the Model**



The provided code calculates the R-squared value as a percentage to evaluate the goodness-of-fit of the AdaBoost Regressor model on the test set. R-squared measures the proportion of the variance in the target variable (in this case, taxi fares) that is explained by the model. This code uses scikit-learn’ s r square function to compute the R-squared value. The result is then multiplied by 100 to express it as a percentage. The R-squared value ranges from 0% to 100%, where a higher value indicates a better fit of the model to the data. It provides insights into how well the AdaBoost Regressor model is explaining the variability in taxi fares on the test set.

**What is Gradient Boost Regressor Model?**

Gradient Boosting Regressor is a powerful machine learning algorithm that falls under the ensemble learning category. Specifically, it belongs to the family of boosting algorithms, which build a strong predictive model by combining the outputs of multiple weak learners (typically decision trees). Gradient Boosting focuses on minimizing the errors of the model by adding new weak learners sequentially, each correcting the errors made by the previous ones. Here are key characteristics of the Gradient Boosting Regressor:

Sequential Weak Learners:

Weak learners (often shallow decision trees) are added to the model sequentially. Each new tree corrects the errors of the combined model up to that point.

Gradient Descent:

Gradient Boosting minimizes the errors of the model by using gradient descent optimization. It optimizes a loss function by moving in the direction of steepest decrease in the loss.

Residuals-Based Learning:

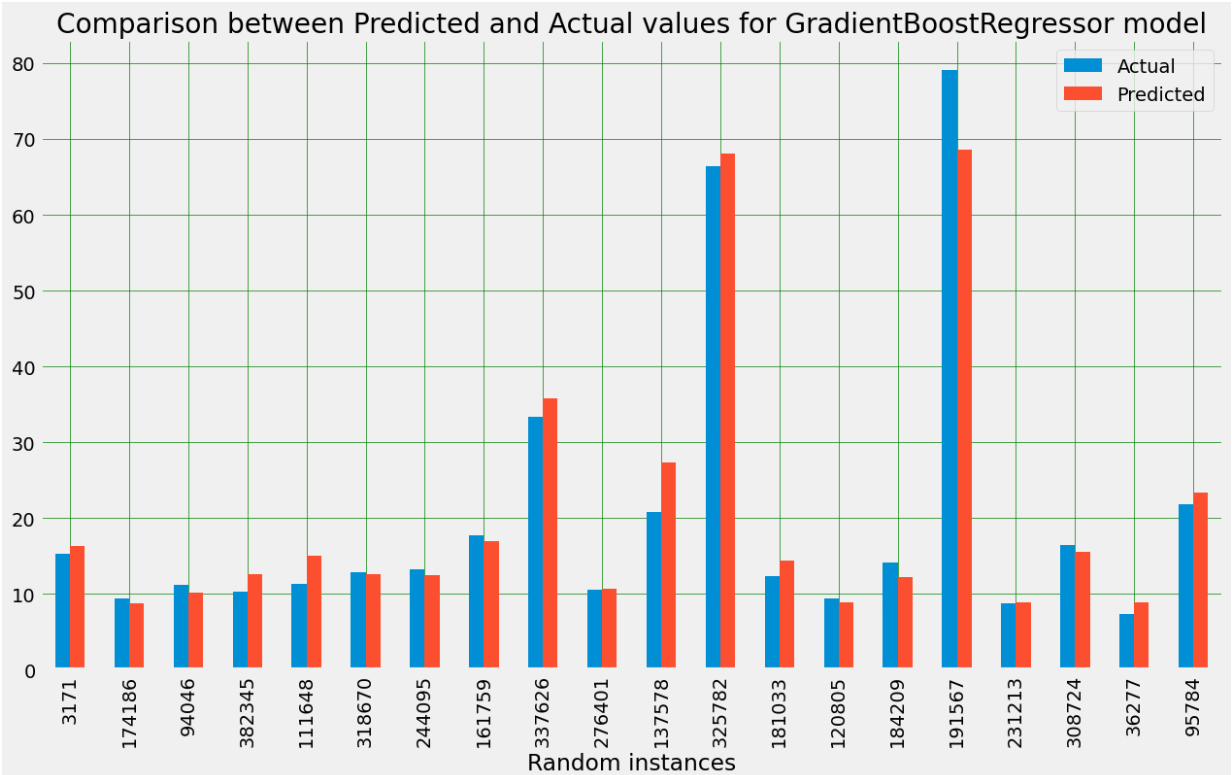
Each new weak learner is trained to predict the residuals (the differences between the actual and predicted values) of the combined model. This approach allows the model to focus on areas where it performs poorly.

Learning Rate:

The learning rate is a hyperparameter that controls the contribution of each weak learner to the overall model. A lower learning rate often requires more weak learners but may lead to better generalization.

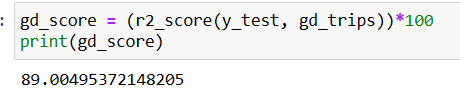
Shrinkage:

Shrinkage is another term for the learning rate. It shrinks the contribution of each weak learner, which can improve the model's generalization.



The bar plot visually compares the actual and predicted taxi fares for a random sample of 20 instances from the test set, as predicted by the GradientBoostingRegressor model. Each bar represents an individual instance, with the 'Actual' and 'Predicted' values side by side. The plot provides an immediate visual assessment of how well the model is capturing the true values, showcasing areas of alignment or discrepancy. A close match between the heights of the 'Actual' and 'Predicted' bars suggests accurate predictions, while noticeable differences indicate instances where the model may be less accurate. The random sampling ensures a diverse representation of instances, avoiding bias. Gridlines enhance readability, and the title provides context. While the visual inspection is insightful, a comprehensive evaluation should involve quantitative metrics for a more nuanced understanding of the GradientBoostingRegressor model's overall performance.

**Evaluation of the Model**



The provided code calculates the R-squared value as a percentage to evaluate the goodness-of-fit of the GradientBoostingRegressor model on the test set. R-squared measures the proportion of the variance in the target variable (in this case, taxi fares) that is explained by the model. This code uses scikit-learn’ s R square function to compute the R-squared value. The result is then multiplied by 100 to express it as a percentage. The R-squared value ranges from 0% to 100%, where a higher value indicates a better fit of the model to the data. It provides insights into how well the GradientBoostingRegressor model is explaining the variability in taxi fares on the test set.

What is XG Boost machine learning algorithm?

XGBoost, which stands for eXtreme Gradient Boosting, is a popular and powerful machine learning algorithm that belongs to the gradient boosting family. It is designed for both classification and regression tasks and has gained widespread popularity for its efficiency and effectiveness in various machine learning competitions. XGBoost is an implementation of gradient boosting that is optimized for speed and performance.

Here are key features and characteristics of XGBoost:

Gradient Boosting Framework:

XGBoost is based on the gradient boosting framework, where weak learners (typically decision trees) are combined sequentially to create a strong predictive model.

Regularization:

XGBoost includes L1 (LASSO) and L2 (Ridge) regularization terms to prevent overfitting, making it more robust to noisy data.

Parallel Processing:

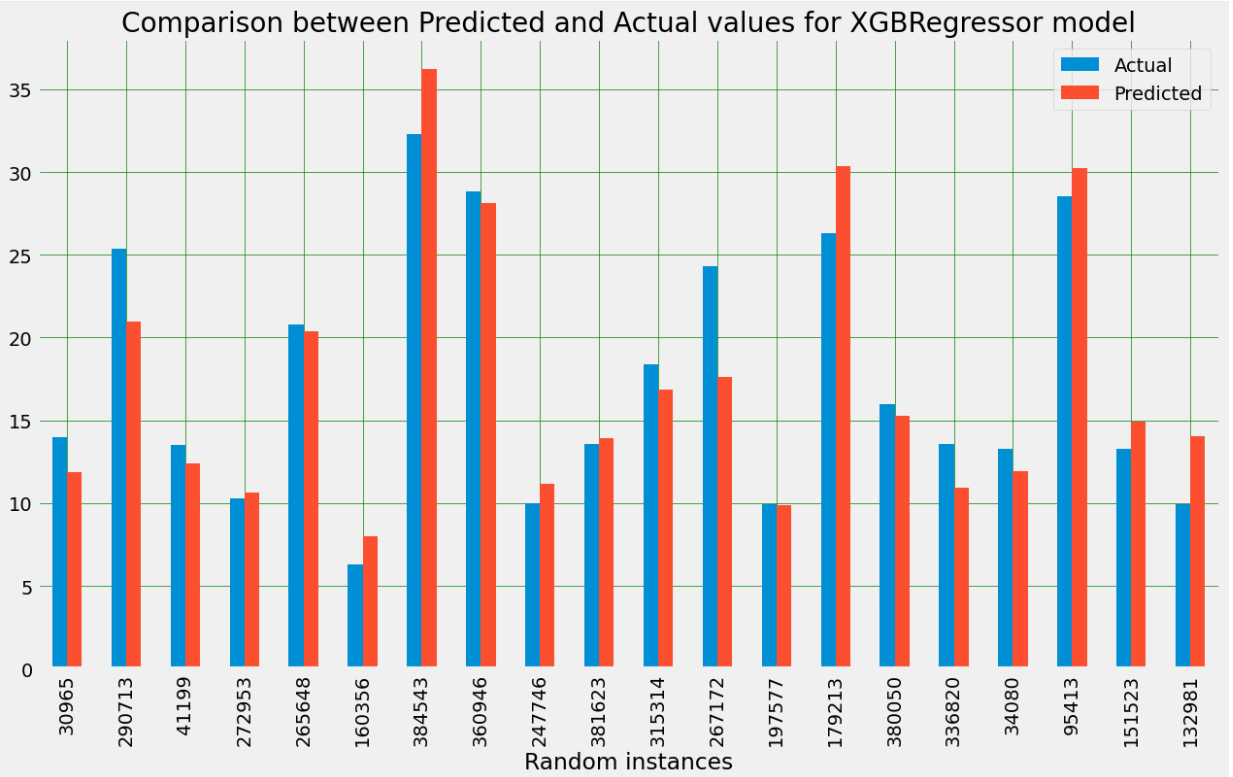
XGBoost is designed to be highly efficient and scalable. It supports parallel processing, making it faster than traditional gradient boosting implementations.

Tree Pruning:

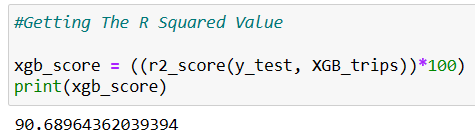
XGBoost includes a pruning algorithm that removes splits that do not contribute significantly to improving the model's performance, reducing the complexity of the trees.

Cross-Validation:

The algorithm has built-in support for cross-validation, allowing users to assess model performance during training.

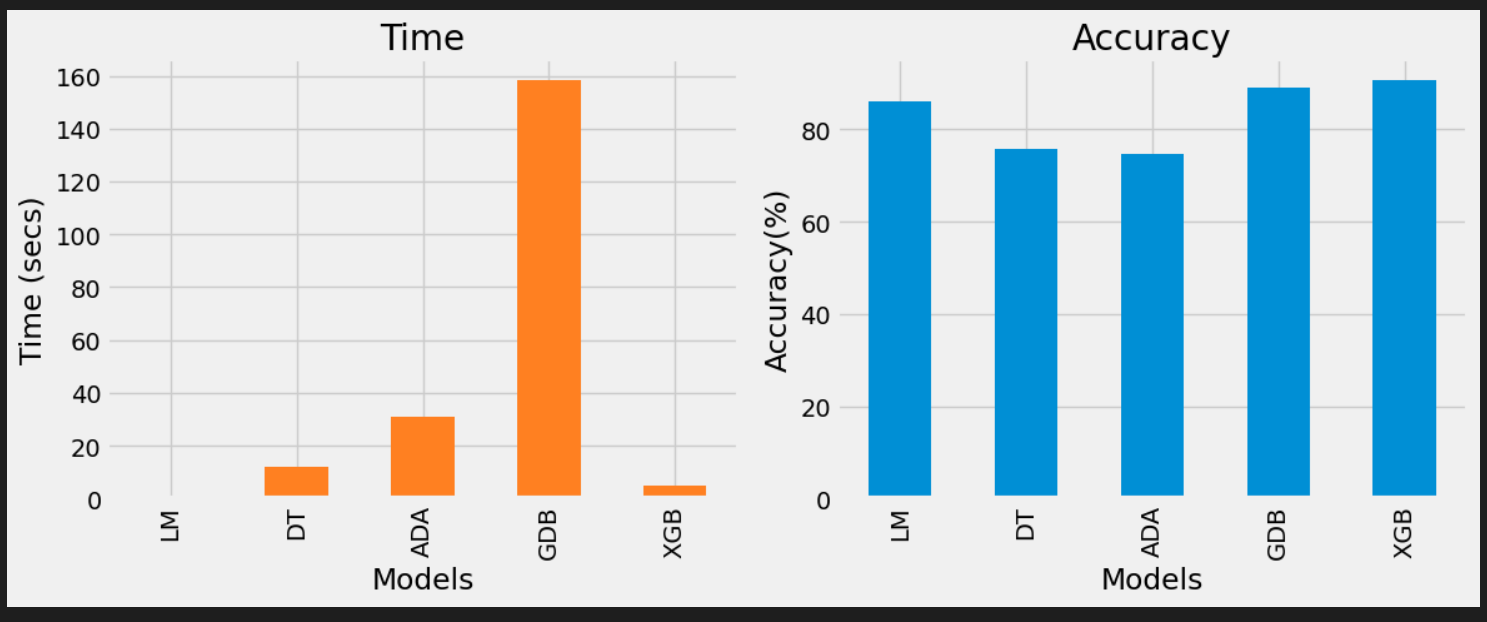


The bar plot visually compares the actual and predicted taxi fares for a random sample of 20 instances from the test set, as predicted by the XGBRegressor model. Each bar represents an individual instance, with the 'Actual' and 'Predicted' values side by side. The plot provides an immediate visual assessment of how well the model is capturing the true values, showcasing areas of alignment or discrepancy. A close match between the heights of the 'Actual' and 'Predicted' bars suggests accurate predictions, while noticeable differences indicate instances where the model may be less accurate. The random sampling ensures a diverse representation of instances, avoiding bias.



The provided code calculates the R-squared value as a percentage to evaluate the goodness-of-fit of the XGBRegressor model on the test set. R-squared measures the proportion of the variance in the target variable (in this case, taxi fares) that is explained by the model. This code uses scikit-learn’ s R square function to compute the R-squared value. The result is then multiplied by 100 to express it as a percentage. The R-squared value ranges from 0% to 100%, where a higher value indicates a better fit of the model to the data. It provides insights into how well the XGBRegressor model is explaining the variability in taxi fares on the test set.

**INTERPRETIVE & COCLUSION**

****

In comparing the performance of various regression models—Linear Regression, Decision Tree, AdaBoost, Gradient Boosting, and XGBoost—for predicting taxi fares based on given features, several insights emerge. Linear Regression, although a basic model, provides a reasonable baseline with an R-squared score of 86 percent. Decision Tree and AdaBoost exhibit 75.72 percent and 74.63 percent R-squared scores, respectively, indicating their effectiveness. However, Gradient Boosting and XGBoost outperform the others, achieving higher R-squared scores of 89 percent and 90.68 percent. This suggests that the ensemble methods, especially XGBoost, demonstrate superior predictive capabilities for taxi fare estimation. The bar plot visualizations further affirm these findings, showcasing the alignment between predicted and actual values. In conclusion, the XGBoost model stands out as the most robust and accurate choice among the evaluated models for predicting taxi fares, demonstrating its effectiveness in capturing the underlying patterns in the data.

**GRAPICAL USER INTERFACE**

**A screenshot of a black and yellow screen

Description automatically generatedA screenshot of a black and yellow screen

Description automatically generated**

The GUI shows us the layout of our app. We ask the user to input Pickup Location, Drop off Location, Day of the week and Time. Based on these inputs we generate a fare prediction running our model. The prediction appears on the GUI for the user.

**REFERENCES**

Geeksforgeeks. (n.d.). *GradientBoosting vs AdaBoost vs XGBoost vs CatBoost vs LightGBM*. Retrieved from Geeksforgeeks: <https://www.geeksforgeeks.org/gradientboosting-vs-adaboost-vs-xgboost-vs-catboost-vs-lightgbm/>

Geeksforgeeks. (n.d.). *GradientBoosting vs AdaBoost vs XGBoost vs CatBoost vs LightGBM*. Retrieved from Geeksforgeeks: <https://www.geeksforgeeks.org/gradientboosting-vs-adaboost-vs-xgboost-vs-catboost-vs-lightgbm/>