

# Predicting NYC Yellow Taxi Fares Using Regression

Group 32

Shadman Sakib

Khalid Hasan




# Outline

- Introduction
- Problem Statement
- Literature Review
- Data Preprocessing
- Model Definition
- Model Tuning
- Model Training
- Prediction
- Evaluation



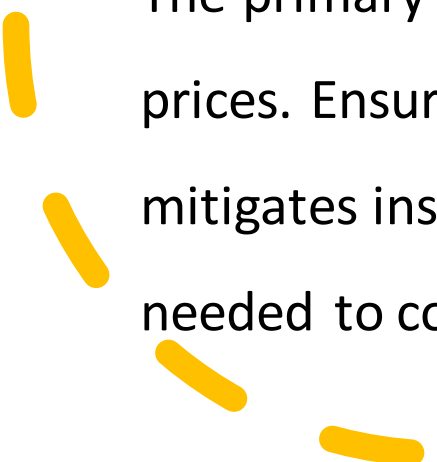
# Introduction

In this research we have developed a fare prediction model for the New York City Yellow Taxi rides. We have used a few regression techniques to predict the continuous value of price based on a handful of features we have extracted from the raw data itself. We have collected our dataset from Kaggle [1] which consists of records collected by two vendors contracted by the NYC TLC over a couple months in 2020.









# Problem Statement

- Due to the increasing number of ride-sharing apps these days, taxi businesses are at a disadvantage. Ride-sharing apps often offer lower prices than traditional taxis due to lower overhead costs. This makes them more attractive to cost-conscious passengers, which can lead to a decline in taxi ridership.
  - The primary challenge encountered by taxi customers revolves around the uncertainty of prices. Ensuring price predictability is crucial, as it facilitates improved cost management, mitigates instances of unfair pricing, and empowers customers with the information needed to compare prices with those of competitors.
- 



# Literature Review

In a comparative study, the effectiveness of gradient boosting using XGBoost was evaluated against a deep learning technique known as multi-layer perceptron (MLP) [2] for predicting trip duration. The findings revealed that XGBoost, demonstrated superior performance over the MLP model when accounting for all variables. Nonetheless, the authors observed that the MLP model could potentially be enhanced through autotuning, albeit with the trade-off of requiring additional time.



# Dataset Summary



Table

|   | summary ▲ | trip_distance ▲   | PULocationID ▲     | DOLocationID ▲    | RateCodeID ▲       | fare_amount ▲      |  |
|---|-----------|-------------------|--------------------|-------------------|--------------------|--------------------|--|
| 1 | count     | 6405008           | 6405008            | 6405008           | 6339567            | 6405008            |  |
| 2 | mean      | 2.929643933309735 | 164.73225778952968 | 162.6626908194338 | 1.0599077192495954 | 12.694108119770615 |  |
| 3 | stddev    | 83.1591059732502  | 65.54373944111758  | 69.91260629496094 | 0.811843207190649  | 12.127295340046553 |  |
| 4 | min       | -30.62            | 1.0                | 1.0               | 1.0                | -1238.0            |  |
| 5 | max       | 210240.07         | 265.0              | 265.0             | 99.0               | 4265.0             |  |

5 rows



# Data Preprocessing

1. Removed noisy data from our desired features (Null, Negatives)
2. Removed outliers by utilizing the Interquartile range of the data

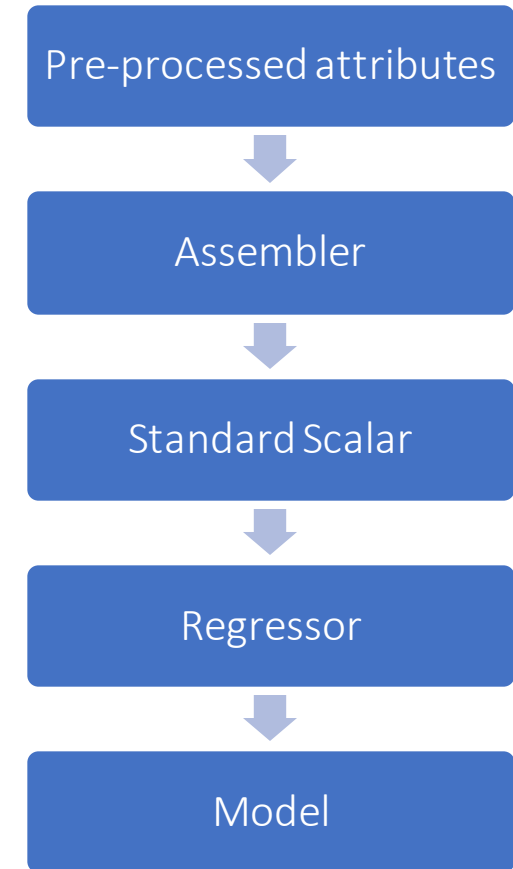
The noisy data was roughly around 5% of our total data that we started off with.



# Model Definition

- Three regression models:
  - Linear Regression
  - Decision Tree Regression
  - Random Forest Tree Regression
- Three Pipelines for each Regression Model.
  - Each pipeline includes three stages: Assembler, Standard scalar, and a regressor.

## Pipeline





# Model Definition

- A Subset of Processed features before model training

| features               | stdScal_cc68611fb632__output  |
|------------------------|---|
| [0.2,141.0,140.0,4.0]  | [0.05268478039847957,2.1563902561084034,2.0038965377967126,7.383538874611791] |
| [0.28,74.0,41.0,1.0]   | [0.0737586925578714,1.1317225457590203,0.5868554146404659,1.8458847186529477] |
| [0.28,107.0,137.0,1.0] | [0.0737586925578714,1.6364096269758805,1.9609558977010688,1.8458847186529477] |
| [0.28,113.0,113.0,1.0] | [0.0737586925578714,1.7281709144698552,1.617430776935918,1.8458847186529477]  |
| [0.28,132.0,132.0,1.0] | [0.0737586925578714,2.0187483248674414,1.889388164208329,1.8458847186529477]  |

# Model Tuning

- To get the optimized model, we have tuned the models with different set of hyper-parameter.
- Spark's **ParmGridBuilder** is used to construct parameter grids for hyperparameter tuning.

## Linear Regressor

- ElasticNetParam: [0.0, 0.5, 1.0]
- RegParam: [0.1, 0.5, 0.9]
- MaxIter: [50, 100, 200]

## Decision Tree Regressor

- MaxDepth: [3, 5, 10]
- MinInfoGain: [0, 0.5]
- MinInstancesPerNode: [1, 2, 5, 10]

## Random Forest Regressor

- NumTrees: [10, 20, 30, 40]
- MaxDepth: [5, 10]

# Model Training

- Used Spark's **TrainValidationSplit** model to split our dataset into a training set and a validation set.
- Set up one **TrainValidationSplit** model for each regressor with the pipeline, the parameter grid, an evaluation metric (Root Mean Squared Error (RMSE) in our case), and the train ratio.
- Fitted the **TrainValidationSplit** models to the training data.

## TrainValidationSplit

- TrainRatio: 0.75
- Parameter Grid
- Estimator:  
Pipeline
- RMSE evaluator

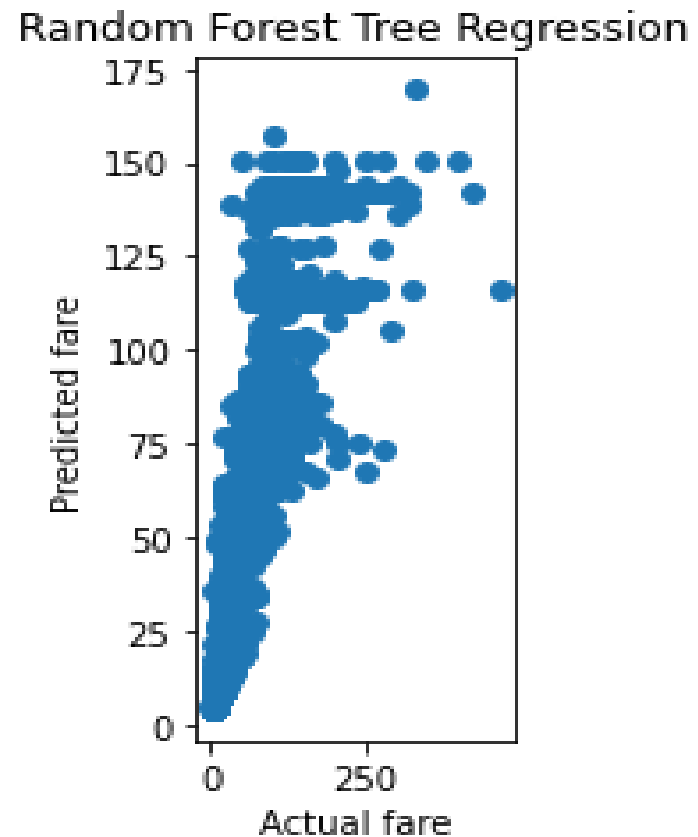
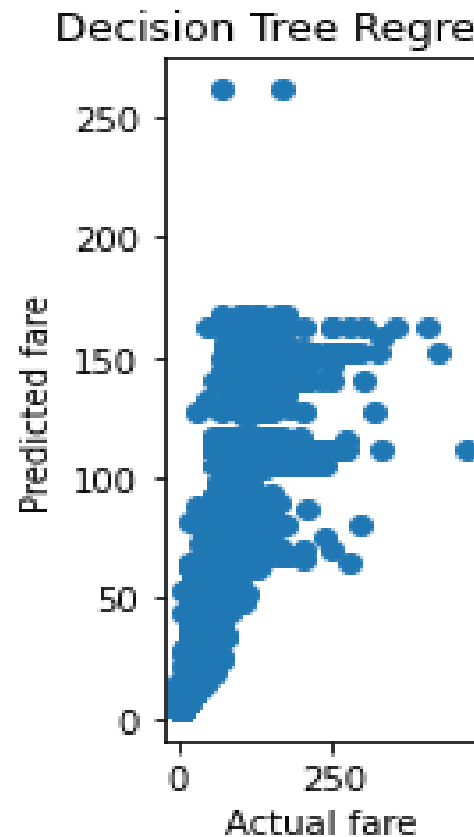
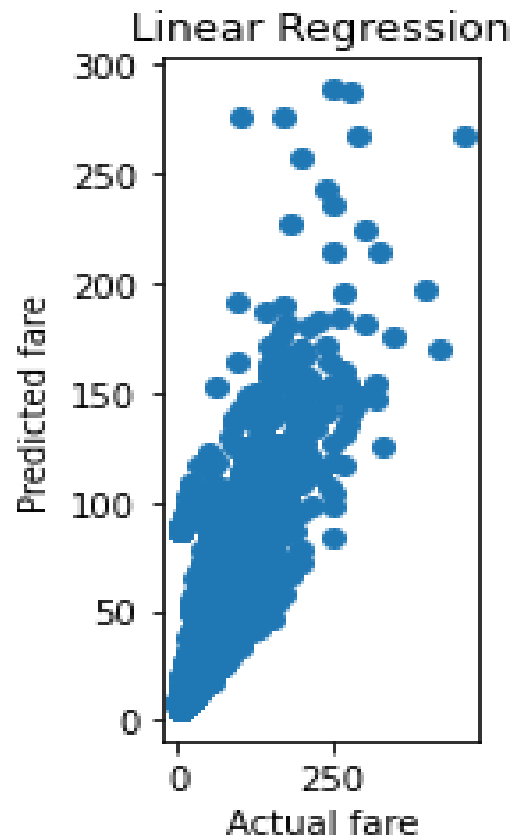
# Model Training (Best Models)

- After finishing the model training stage with the different combinations of hyper-parameters, we find our best trained models with optimized hyper-parameters.

| Model              | Training Time | Parameters and hyper-parameters  |
|--------------------|---------------|--|
| Linear Regression  | 27.54 minutes | numIterations: 7<br>Co-efficients: [2.723277, 4.589777E-4, -2.8002004E-4, 0.825135]<br>Intercept: 3.537987 |
| Decision Tree      | 20.93 minutes | depth=10<br>numNodes=1663  |
| Random Forest Tree | 42.18 minutes | numTrees=30  |

# Prediction

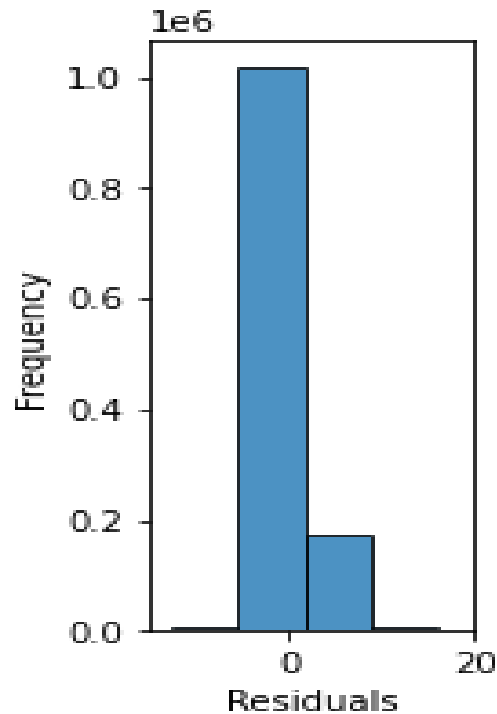
- **Actual fare vs Predicted fare Analysis:** Assess the goodness of fit
- The fewer points deviate from a line, the better the model.
- In our case, the better model in order: Random Forest > Decision Tree > Linear Regression



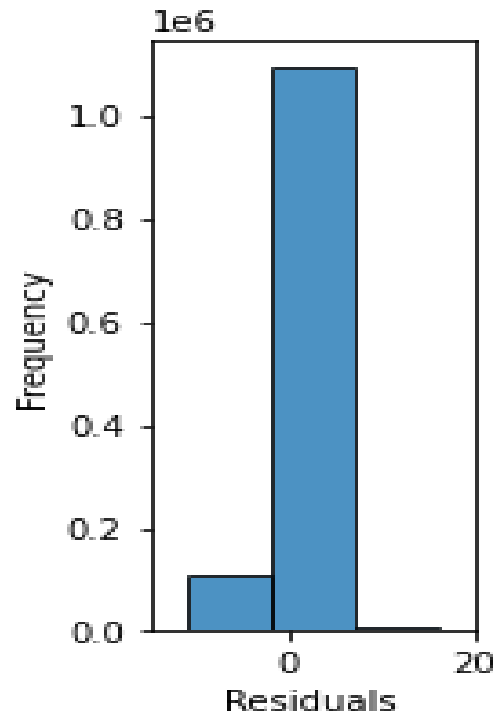
# Prediction

## Residuals Analysis: Actual values – Predicted values

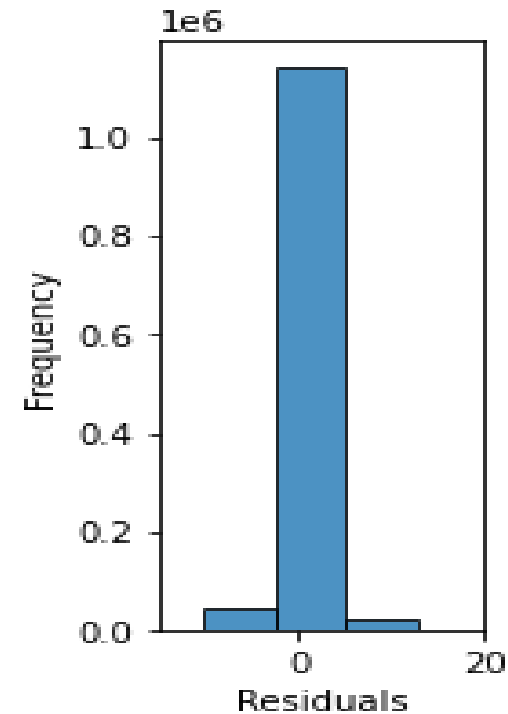
- In an ideal case, residuals should be normally distributed around zero.
- In our case, Random Forest regression shows better normally distributed.



Linear Regression



Decision Tree Regression



Random Forest Tree Regression

# Evaluation (Model Error)

- **Root Mean Square Error (RMSE):** no upper limit, tends to 0 for better model.
- **Mean Absolute Error(MAE):** no upper limit, tends to 0 for better model.
- **R-squared (R2):** range=[0, 1] ; tends to 1 for better model.
- Each regression model shows the excellence but the Random Forest Tree emerges as the best.

| Regression Model ▲ | RMSE ▲   | MAE ▲    | R2 ▲     |
|--------------------|----------|----------|----------|
| Linear             | 2.898132 | 1.565766 | 0.929683 |
| Decision Tree      | 2.416508 | 1.287255 | 0.951112 |
| Random Forest Tree | 2.399815 | 1.303674 | 0.951785 |

# Summary

- Defined various regression models to predict continuous variable like taxi fare.
- Tuned our models with different combinations of hyperparameter values to find the best model.
- Analyzed our predicted results over different regression models in two aspects:  
1) Actual values vs Predicted values Analysis, 2) Residuals Analysis
- Evaluated our models on the basis of several evaluation metrics.
- All of our models have displayed excellence but the random forest tree regression performed best.



# References

1. <https://www.kaggle.com/datasets/microize/newyork-yellow-taxi-trip-data-2020-2019/data>
2. Poongodi, M., Malviya, M., Kumar, C. et al. New York City taxi trip duration prediction using MLP and XGBoost. Int J Syst Assur Eng Manag 13 (Suppl 1), 16–27 (2022). <https://doi.org/10.1007/s13198-021-01130-x> [Accessed 27/12/2022]

Thank You!

