Predicting NYC Yellow Taxi Fares Using Regression

Group 32

Shadman Sakib

Khalid Hasan

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.
- We wish to propose a fare prediction based on the location of pick-up and destination and other key factors such as the rate code, distance. We will be utilizing the powerful and popular open-source big data processing framework, Apache Spark. We will be training a Regression model to predict the continuous values of price.

Introduction

- Insert

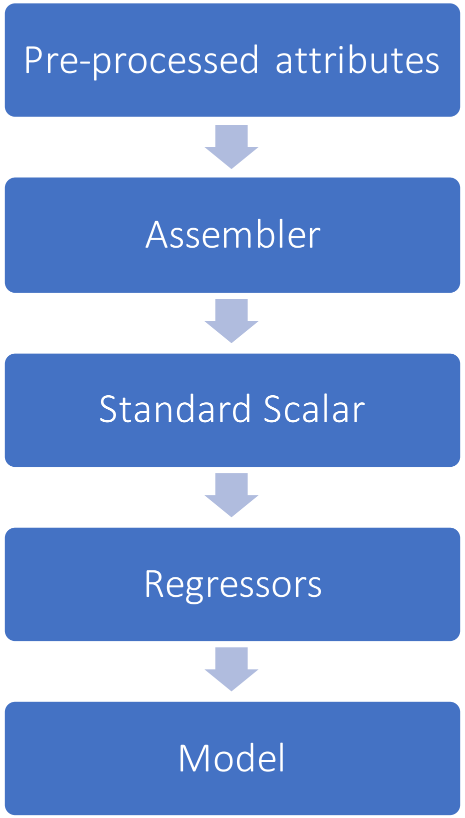
Methodology

- Data Pre-processing
- Model Definition
- Model Tuning
- Evaluation

Model Definition

- Three regression models:
  - Linear Regression
  - Decision Tree Regression
  - Random Forest Tree Regression
- Three Pipelines for each Regression Model

Pipeline



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

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

TrainValidationSplit
• TrainRatio: 0.75 • Parameter Grid • Estimator: Pipeline • RMSE evaluator

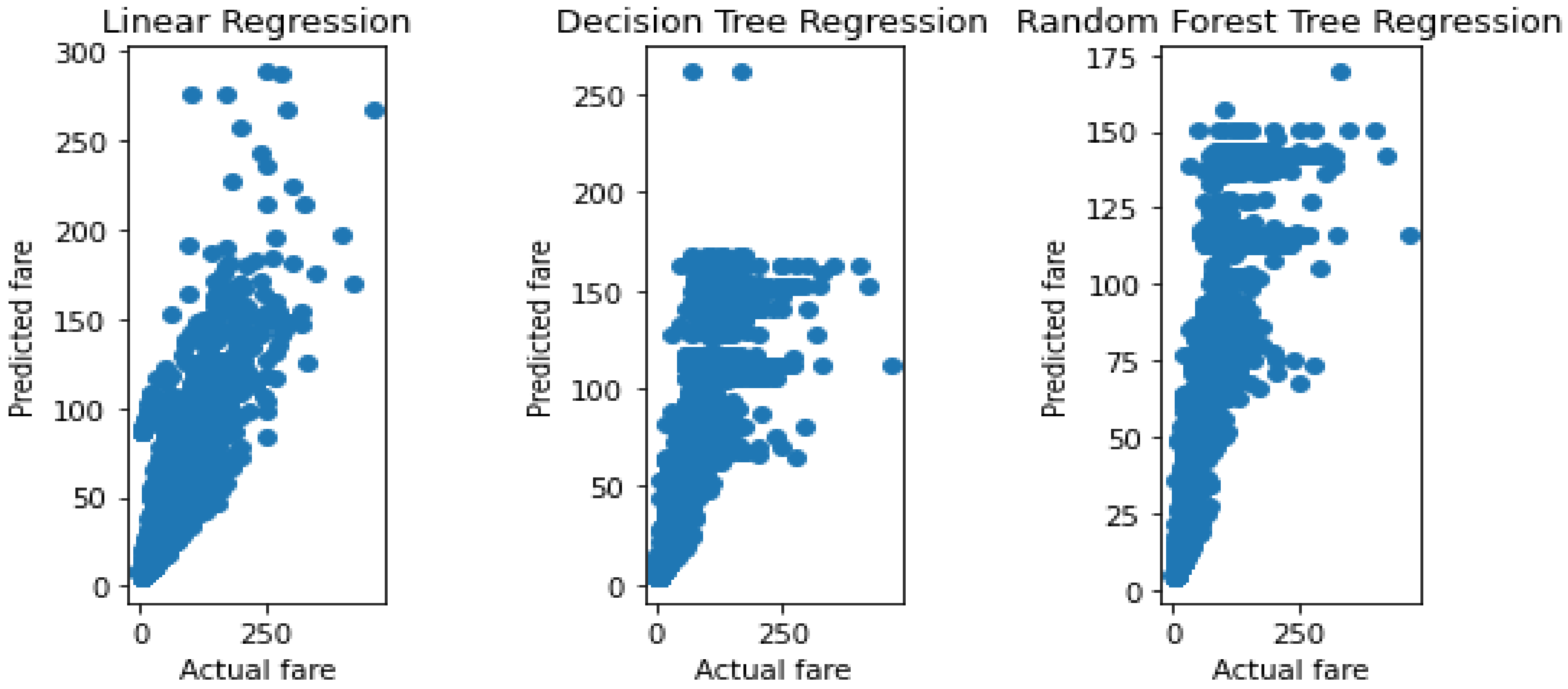
Model Training (Best Models)

- 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 Co-efficients: [2.723277, 4.589777E-4, -2.8002004E-4, 0.825135] Intercept: 3.537987
Decision Tree	20.93 minutes	depth=10 numNodes=1663
Random Forest Tree	42.18 minutes	numTrees=30

Prediction

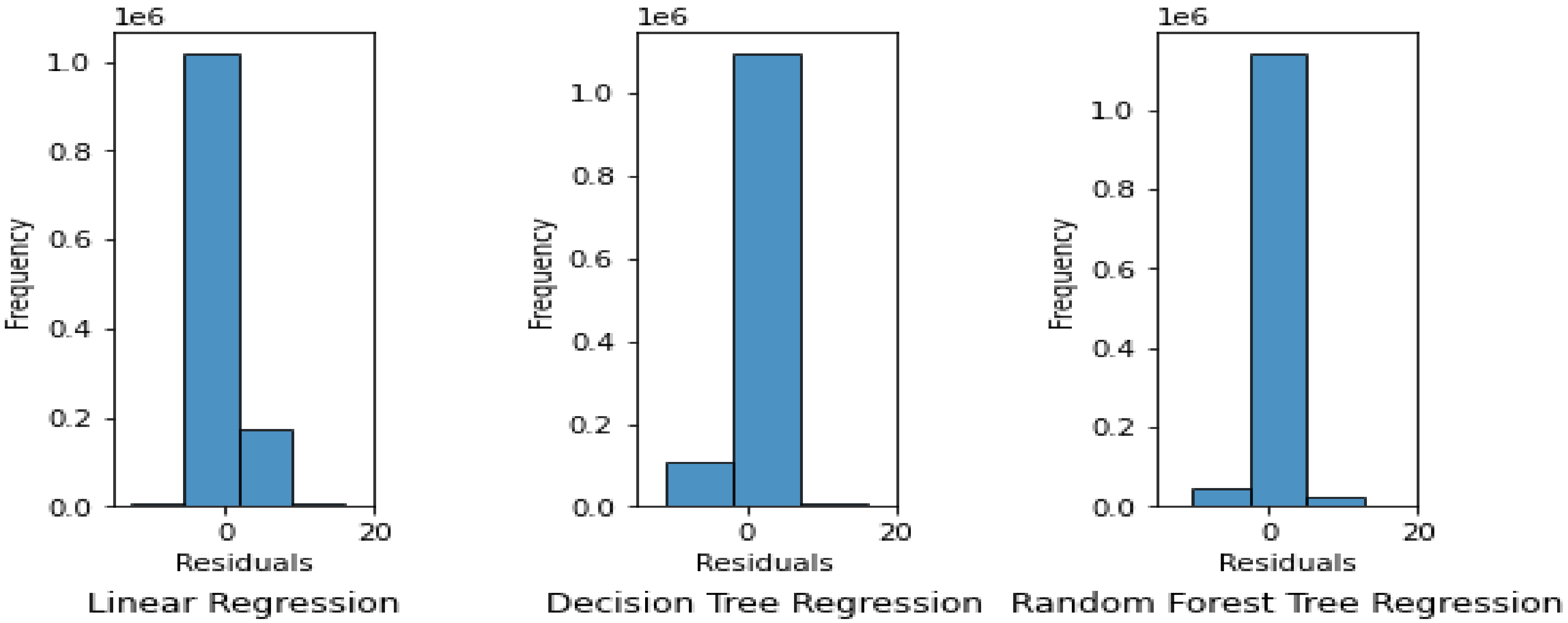
- **Actual fare vs Predicted fare:** Assess the goodness of fit
- The fewer points deviate from a line, the better the model.
- In our case, the better model in order: Random Forest > Decision Tree > Linear Regression



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 distributed.



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	$\boxed{1.287255}$	0.951112
Random Forest Tree	$\boxed{2.399815}$	1.303674	$\boxed{0.951785}$

Methodology

- Insert

Methodology

- Insert

Thank You!