Predicting New York City Taxi Trips

SENG 550 Final Project Report

Group 6

Benjamin Nielsen	Trevor Brown	Will Kerr	Ahmed Farazi	Sultan Arafat
30063158	30063456	30064999	30063552	30057064

Abstract

The purpose of this report is to familiarize the reader with problems revolving around taxi trip durations and various factors that influence and impact commute times. The idea behind this project is to analyze previous trends and data of taxi trip times in New York City, and propose a solution for predicting ride durations that builds upon previous solutions. The project will be considered a success if historical data is applied into machine learning algorithms, and accurate results are obtained. Our primary goal is to use the pick up time, location, and drop off location of a user, and use historical data to predict an estimated time that they would be dropped off.

1. Introduction

1.1 Problem Statement and Significance:

In the world of ride-sharing and technology-based route-mapping, there are numerous factors to account for which can cause potential delays, such as traffic conditions, road blocks, construction, etc. Because of these factors, there is quite a bit of uncertainty in when a passenger may arrive at their desired destination. As a result of this uncertainty, passengers may have to adjust their commute times in order to ensure they arrive at their destinations at or before their desired times. It is of utmost importance that commuters are given transparent communication that addresses possible delays, as well as offered essential information such as estimated trip durations to influence their choice of transport and when they use said transport.

1.2 Previous Solutions:

A pre-existing solution that we considered for inspiration was the Deerfoot Commute Time [1] problem addressed in class. While this solution does account for weather for the commute times, its model is limited to the commute time between 2 fixed points, whereas our model proposes to calculate transport time between any random 2 points. Our goal is to ensure that our machine learning model is able to take in a unique starting point, ending point and distance within NYC boundaries and be able to determine the commute time from these parameters.

The proposed Kaggle problem [2] has records of previous machine-learning approaches for its particular problem, in which people used the same provided data and applied various models to calculate trip durations. We will take inspiration from past solutions for our machine learning models.

To produce stronger results from the above described solutions, we will be proposing several ML algorithms that will seek to interpret and validate our dataset and determine which performs best for a real-life application.

1.3 Our Data Analysis Questions

When performing data analysis, the most relevant information to consider is ride trip duration, ride stop and start locations (by coordinate), weather conditions, and the date/time the ride begins and ends.

When considering how we should approach data analytics, we posed several questions.

RQ1. Can we transform our pre-existing data to produce a more useful and efficient result?
RQ2. How do we plan to determine the weight of each parameter of the machine learning model?
RQ3. How do we plan to represent our output of the results?
RQ4. What type of machine learning model would produce the best results?
RQ5. Do we have a margin of error and if so what is acceptable?

These questions are inplace to help investigate the effectiveness and accuracy of our algorithms with respect to estimated ride sharing times given test data.

1.4 Proposal:

Since most ride-sharing and transportation information is logged, historical trends can be mapped and analyzed. We propose a project in which its primary goal is to predict NYC taxi trip times based on historical data and trends which is provided via Kaggle[2].

Our goal for approaching this problem is to traverse the provided data, and use mapreduce algorithms to apply machine learning techniques on historical data trends to then use to estimate how long individual trips may take.

To take into account more factors that could affect trip times, we have fetched weather data[3] from the same period as the data provided by Kaggle[2]. This data will allow us to take into account the weather of that day, and we can determine how much of an impact these conditions have on trip time.

1.5 Main Findings:

It was discovered that the best performing model was Gradient Boosted Trees Model and the weakest performing model was Linear Regression. Additionally, it was found that all models still produced weak outcomes, and would need to be further trained & tuned to produce stronger results.

2. Background and Related Work

2.1 Technical background:

Determining the duration of New York City taxi rides is a difficult task because of complex and unpredictable traffic patterns. One approach to predict the duration of a taxi trip is using real time information to make a short term prediction. These short term predictions can be made based on the current conditions of the road. An example of this is the EasyTracker [4] by utilizing real time datasets from pre-existing transit systems that exist in the area.
While this approach is useful for predicting the best route in real time, this approach is limited when predicting future routes. Google Maps [5] has introduced the need for planning journeys ahead of time, so that they are able to predict travel times into the future.

The second approach is desirable as it simply needs an input of the current starting point and ending point to generate a prediction, and does not actively require real time data. Therefore we will take the approach of trying to determine the taxi duration without real time data, and simply by examining historical taxi trip data.

2.2 Related work:

There have been many researchers who have led efforts within GPS and machine learning fields that have used New York Taxi duration as their problem. In the solution created by [6] they have used the K-means++ model and used parameter calibrations from a XGBoost model to estimate duration. As well as work by [7] which questions the quality of the data that it has. Since data from an urban city has many different factors that lead to the duration, they argue the necessity for data cleaning. They show how outliers or even data that has pickup and dropoff zones within the middle of bodies of water have drastic effects on the quality of results.

The problems we are tackling have a real world impact as outlined in [8] where in this paper they analyze the Machine Learning within Uber's trip prediction. They articulate how important real time predictions for trip duration are crucial and that the historical data is the significant factor in creating more dynamic predictions.

3. Methodology

3.1 Input Data Set:

The data set provided via kaggle [2] for trip data contains a few identifiers for trips and trip providers, pick up times, drop off times, the number of passengers, the longitude and latitude of meter engagement/disengagement, the trip duration, as well as a flag on whether the trip was held in vehicle memory.

We also used weather data provided via kaggle [3] for weather data in New York during the same time period as the data set above. This data set contains information for max temperature, min temperature, average temperature, precipitation, snow fall, snow depth. These 2 data sets will be combined and used for training and testing.

3.2 Experiment setup

In order to set up this experiment, we transferred the csv data into spark dataframes, filtering extreme outliers from trip duration to vastly improve the accuracy of our models. Next, we formatted the dates used in trip data to be identical to those of weather data so that they could be combined into a single dataset for training. After joining the two data sets, the new parsed set was transformed into LabelledPoints with trip duration and all possible factors so that it could be used for training models. With our data in a usable format, it was split into training and test sets with 75% of the data in the training set and 25% of the data in the testing set.

3.3 Experiment Factors:

The following algorithms will be considered for evaluating average commute time(s):
Linear Regression Model: examines if a predictor variable predicts an outcome accurately on a dependent variable, and determines which variable(s) are the most significant in predicting an accurate outcome. [9]
Random Forest: fits a number of classifying decision trees on various sub samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. [10]
Gradient Boosted Trees Model: gives a prediction model in the form of an ensemble of decision trees. [11] This model is expected to outperform the random forest model as the gradient as it has much more predictive accuracy when parameters are tuned correctly and is used to control under-fitting.

We decided to tune 3 parameters from each model:
Linear Regression Model: regParam, elasticNetParam and maxIter
Random Forest Model: maxDepth, maxBins and numTrees
Gradient Boosted Trees Model: maxDepth, maxBins and maxIter

3.4 Experiment Process:

First, we trained a linear regression model with generic parameters and generated some initial metrics and a graph to visualize its accuracy. Attempting to find a more accurate model, we used a crossvalidator to tune the hyperparameters of the model to be more accurate using a number of possibilities. A second set of metrics were generated to compare the results. This process was repeated for both the random forest and gradient boosted trees model, though we were disappointed to find that the time required to cross validate a gradient boosted trees model in particular significantly exceeded the capabilities of a free databricks account. Thus, the amount of tuning on the cross validated gradient boosted trees model is extremely limited. To clearly display the differences between each model, a final graph was generated to simultaneously display all three.

3.5 Performance Metrics:

A few metrics to be considered when considering the overall quality of each machine learning algorithm that we have selected for testing. We will not be using the accuracy, precision, recall, f-score as these metrics are used for measuring binary scores. We will be using the following regression metrics:

Mean Squared Error (MSE): This metric is the average of the squared difference between the original and predicted values in the data set. [12] The lower the MSE, the more accurate the model is.
Root Mean Squared Error (RMSE): This metric is able to report the average difference between the true values and the predicted values by the model [13]. The values are squared so that positive and negative differences would not cancel eachother out. The lower the RMSE value, the more accurate the model is.
Mean Absolute Error (MAE): For this metric, the residual for every data point is calculated. Only the absolute value of each residual is taken, such that the negative and positive residuals do not cancel out. The average of all these residuals are then taken. This metric usually describes the typical magnitudes of the residuals [14]. The lower the MAE, the more accurate the model is.
Coefficient of Determination (R^2): This metric is used to determine the correlation between two features of the dataset [15]. The closer to 1 the R^2 is, the better the model is
Explained Variance: This metric is used to measure the difference or variance between a model predicted data and actual data [16]. The closer to 1 the explained variance is, the better the model is.

4. Results

Table 1: Results of Validation Metrics for all Models

Model	Cross Validation	MSE	RMSE	MAE	R^2	Explained Variance
Linear Regression	No	101017.93	317.83	3472243.88	0.460	-0.245
	Yes	100959.19	317.74	3467216.00	0.460	-0.232
Random Forest	No	79235.49	281.49	3061563.51	0.580	0.131
	Yes	83732.85	289.37	3170612.00	0.564	-0.085
Gradient Boosted Trees	No	69722.28	264.05	2819859.56	0.627	0.381
	Yes	68786.26	262.27	2787111.00	0.632	0.410

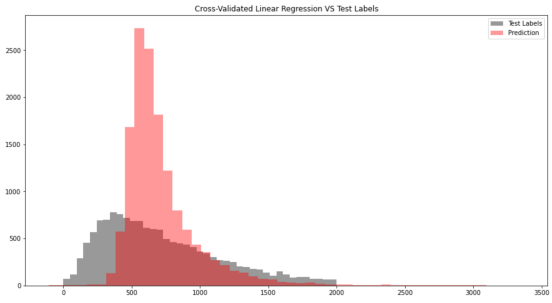

Figure 1. Linear Regression Results

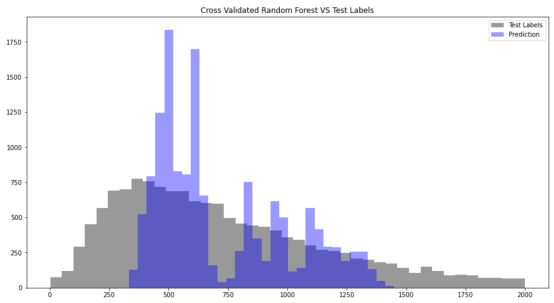

Figure 2. Random Forest Results

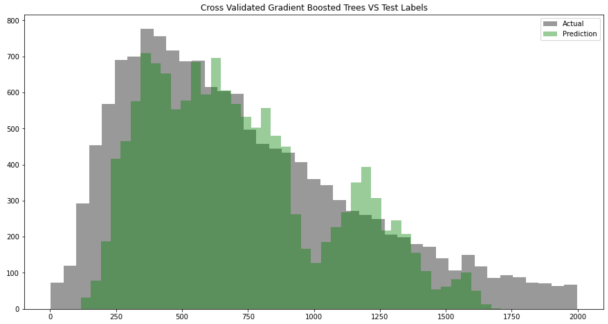

Figure 3. Gradient Boosted Tree Results

RQ1. Can we transform our pre-existing data to produce a more useful and efficient result?
By eliminating outliers in our training data set, along with tuning hyperparameters as described in **3.3**, we were able to produce stronger results as shown in **Table 1**.

RQ2. How do we plan to determine the weight of each parameter of the machine learning model?
For each machine learning model, we created a list of possible parameters within a parameter grid. The parameter grid and the machine learning model were given to the cross validator, which created a model for every possible parameter combination and then trained each model using the training data. The model that performs the best within the cross validator is then the output of the cross validator. This model has the weights of each parameter tuned to improve the performance of the model.

RQ3. How do we plan to represent our output of the results?
We represented the outputs of the results using all of the performance metrics, as well as a double histogram that showcased the frequency of results against the actual test labels. These representations made clear the quality of each model's performance.

RQ4. What type of machine learning model would produce the best results?
Of the three models, it is clear that the best performing model used was the gradient boosted trees model. This is a useful model because when tuned, it performs well, but it also performs well against under fitted data. Additionally, Linear Regression was clearly the weakest of the 3 models used. This is due to linear regression being sensitive to outliers and weak data, which greatly skew training results and affect the overall outcome.

RQ5. Do we have a margin of error and if so what is acceptable?
An agreed upon acceptable margin of error for this product would be 2 minutes or 120 seconds, as trip estimates outside of this range would lead to uncertainty, and not be considered trustworthy. The best performing model, Gradient Boosted Trees, had a RMSE of 262 seconds or a bit over 240 seconds. While this model falls outside the margin of error criteria, we were still impressed with our results considering how little tuning was required. With more parameter tuning, this model's performance would improve greatly and likely fall inside of the margin of error.

5. Conclusion and Future work

Conclusion: Using the historical data provided along with weather data from the same period, we were able to train multiple Machine Learning algorithms to predict trip duration given a set of factors for an arbitrary trip. This fulfilled our original goal. However, while the models we created were generally accurate to overall trends, they were not accurate enough to be satisfactory for use in an end user application. It would require more fine tuning on the data set, along with exploring more relevant machine learning models to produce stronger results that could be considered in a real-life application.

In next iterations, we would like to consider aspects of traffic that require live-data, such as construction, shut-down roads, average number of accidents on major roadways, etc. This data would likely prove useful in making live estimations for drivers on roads and their commute times. Additionally, it would be interesting to see what kind of results arise from using Deep Neural Networking techniques on our data set, and how accurate their results end up being.

Appendix

A. Group contributions

All group members contributed equally and fairly. All group members attended every group meeting.

Benjamin Nielsen:
- General contributions to coding and report
- Edited, commented and finalized code and report as a team
- Obtained, filtered, manipulated and combined datasets into usable format
- Created Performance Metrics for each Model
- Wrote Conclusion and Methodology Sections, along with References

Trevor Brown:
- General contributions to coding and report
- Edited, commented and finalized code and report as a team
- Created Linear Regression Model
- Wrote Introduction Section, Abstract and Results Sections

Will Kerr:
- General contributions to coding and report
- Edited, commented and finalized code and report as a team
- Created Cross-Validation for each Model
- Commented the Codebase
- Wrote Background Section and parts of Methodology Sections

Ahmed Farazi:
- General contributions to coding and report
- Edited, commented and finalized code and report as a team
- Created Random Forest Model
- Wrote parts of Methodology Section, References

Sultan Arafat:
- General contributions to coding and report
- Edited, commented and finalized code and report as a team
- Created Gradient Boosted Trees Model
- Created Graphs for each Model
- Wrote parts of the Introduction.

B. References

[1] J. Reyhaneh *Spark-ML*, 2021, Accessed on Nov 20, 2021. [Streaming video]. Available: https://yuja.ucalgary.ca/V/Video?v=220320&node=893179&a=1909362518&autoplay=1

[2] "New York City Taxi Trip Duration," *kaggle.com*. https://www.kaggle.com/c/nyc-taxi-trip-duration/data (accessed Dec. 19, 2021).

[3] "Weather data in New York City - 2016," *kaggle.com*. https://www.kaggle.com/mathijs/weather-data-in-new-york-city-2016/version/3 (accessed Dec. 19, 2021).

[4] J. Biagioni, T. Gerlich, T. Merrifield, and J. Eriksson, "EasyTracker: Automatic Transit Tracking, Mapping, and Arrival Time Prediction Using Smartphones." [Online]. Available: https://www.cs.uic.edu/~jakob/papers/easytracker-sensys11.pdf.

[5] "Predicting the Future with Google Maps APIs," *Google Maps Platform*. https://mapsplatform.googleblog.com/2015/11/predicting-future-with-google-maps-apis.html.

[6] A. M. Nagy and V. Simon, "Improving traffic prediction using congestion propagation patterns in smart cities," *Advanced Engineering Informatics*, vol. 50, p. 101343, Oct. 2021, doi: 10.1016/j.aei.2021.101343.

[7] N. Ferreira, J. Poco, H. T. Vo, J. Freire, and C. T. Silva, "Visual Exploration of Big Spatio-Temporal Urban Data: A Study of New York City Taxi Trips," *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 12, pp. 2149–2158, Dec. 2013, doi: 10.1109/tvcg.2013.226.

[8] P. Rahul, “Analysing Uber Trips using PySpark,” 2021.
https://iopscience.iop.org/article/10.1088/1757-899X/1119/1/012013/pdf (accessed Dec. 19, 2021).
[9] “What is Linear Regression?,” *Statistics Solutions*.
https://www.statisticssolutions.com/free-resources/directory-of-statistical-analyses/what-is-linear-regression/.
[10] “3.2.4.3.2. sklearn.ensemble.RandomForestRegressor — scikit-learn 0.20.3 documentation,” *Scikit-learn.org*, 2018.
https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html.
[11] Wikipedia Contributors, “Gradient boosting,” *Wikipedia*, May 30, 2019. https://en.wikipedia.org/wiki/Gradient_boosting.
[12] A. Chugh, “MAE, MSE, RMSE, Coefficient of Determination, Adjusted R Squared — Which Metric is Better?,” *Medium*, Dec. 08, 2020.
https://medium.com/analytics-vidhya/mae-mse-rmse-coefficient-of-determination-adjusted-r-squared-which-metric-is-better-cd0326a5697e.
[13] “What are the differences between MSE and RMSE,” *i2tutorials*, Nov. 13, 2019.
https://www.i2tutorials.com/differences-between-mse-and-rmse/.
[14] Tutorial: Understanding Regression Error Metrics in Python
https://www.dataquest.io/blog/understanding-regression-error-metrics/
[15] “R-Squared” https://www.investopedia.com/terms/r/r-squared.asp
[16] “What is Explained Variance?” https://www.statisticshowto.com/explained-variance-variation/