Project Draft/Milestone 3, Project 2

Introduction

Perhaps one of the most common sights, until a few years ago at least, in New York City is their famous yellow Taxi Cabs. The New York City Taxi Fare Prediction competition on Kaggle is a competition hosted by both Google and Coursera. The goal of the competition is to predict the cost of the ride given the starting and ending points (New York City Taxi Fare Prediction, n.d.). With the starting and ending points provided as geolocational coordinates, it is possible to calculate the distance between two points as the crow flies. Using this distance, it is possible to make a prediction on what it might potentially cost to take a taxi. The reality is that in New York City, it is never a straight line to get from point A to point B. New York City can have routes closed off, traffic patterns, and other intricacies on getting around such as rush hour traffic. The question at hand here is if additional data is added, does that impact how well a prediction does? It is possible with the starting and ending points to get this data and add it into machine learning models. This leads to the research question being posed: if we add in the distance and duration of the trip, will that affect the output models?

For the research, the Google Maps Directions API will be used. The directions API is very robust, it allows for multiple input data types. Input types can be addresses or Latitude and Longitude for starting and end points. The response back includes the duration and time length of the trip and full step by step directions. API requests can be made over HTTPS and requires a custom API key (Web Services Directions API, n.d.).

Kaggle competition is a competition platform which hosts competitions in the data science realm. Kaggle has data sets and a wide variety of competition types to the data science community in order to challenge individuals regardless of where they are within their data science careers. Kaggle hosts several competition types, including featured which are what they are best known for, which feature a cash prize. The playground competition type, which the Taxi Fare prediction competition falls into, is a for fun type of competition. These competition types are typically targeted towards newcomers ("How to use Kaggle", n.d).

Background

The problems presented here fall into the type of problem defined as a supervised learning problem. In problems with supervised learning, an algorithm uses labeled data in order to make a prediction. In other words, in a supervised learning problem, you should have some prior knowledge, in the form of samples, of what your output prediction should be. This is in contrast to unsupervised learning. In unsupervised learning, the goal is to ultimately infer a natural structure which is present within the data. Within supervised learning, the goal is to try and find a structure or relationship to the input data that allows the algorithm to correctly predict the outcome of new data (Soni, 2018).

Supervised learning problems typically fall into two categories, either they are a classification type problem, or they are a regression type problem. Classification type problems tend to try and 'classify' the category that the data belongs to. Common examples of classification problems are dog breed detection, determining if a customer will churn, and determining if an email is spam or not. In classification problems, there are generally 2 or more categories in which an item can belong, and the goal is to find the relationship between the data and the category so that an accurate prediction can be made. This is in contrast to regression type problems where the goal is to make a numeric type of prediction. Some of the more common regression problems include stock price prediction, house price predictions, and others (Soni, 2018).

Linear Regression

Linear regression is perhaps one of the oldest algorithms currently still used in machine learning. Linear regression dates itself back to the 18th century. Linear regressions first trivial form was proposed by mathematician Carl Friedrich Gauss. It is generally thought to be one of the most well known and well understood algorithms within the realm of statistics and machine learning (Essentials of Linear Regression in Python, n.d.). A model is based on linear regression attempts to describe the relationship between two or more variables by fitting an equation to the observed data. The formula for linear regression can be expressed very simply with y = a + bx. Y is what is known as the dependent variable, and X is what is known as the explanatory variable. The intercept of the equation is a, and b is the slope (Linear Regression, n.d.). Ultimately, the pros of using linear regression are that it is a simple algorithm. it is easy to explain the results of the coefficients to others. It is a computationally efficient algorithm, it does not eat up a lot of memory, and makes it very fast to make a prediction. Due to its age, it has been incorporated into virtually all statistical software packages. The downsides are that only numerical variables can be used with linear regression. Another big con is that it is extremely sensitive to outliers as well as missing data. If the relationship of the data is not linear, the predictive power is also very poor (Shuster, 2015).

Decision Tree

Decision trees are a tool in which they can be applied both to classification as well as regression. A decision tree is contrasted from other classification algorithms mainly in the approach it takes in order to perform the classification. Most other algorithms perform the classification within a single step. Decision trees use a multi staged hierarchy in order to make a classification, the resulting structure branching out in a root like structure, hence the name decision tree. The tree starts with a root node, as it begins to learn, binary splits are added to the data, which separate some of the classes from the remaining classes. As you trace down the tree, you each terminal node results in a conclusion, which is known as a top down approach. The decision tree at its core aims to use the concept of splitting complex questions into simpler ones until a decision can be reached. When the target of a decision tree is a discrete variable, such as a class attribute, this is known as decision tree classification. When the target variable of the decision tree is continuous, it is known as decision tree regression. When a decision tree regressor is being trained, the samples are used to determine the structure. The algorithm will then break the data down into every possible binary split. It selects the split that moves the data into parts that minimize the sum of the squared deviations from the mean in each of the split parts. The same splitting process is then applied to each of the new branches. This entire process continues on until each node reaches a terminal node. It is important to call out that a decision tree is generated from the training samples that it sees. Decision trees are known to be susceptible to being overfit, meaning that they fit the sample data that they have seen very well, but do poorly when working with data that it has not seen before (Xu, M, 2005). As discussed before, the advantages of using a decision tree are that it is a simple and easy algorithm to understand. It follows a human understandable approach to making decisions, and it can be visualized and explained. The drawbacks include that it is very susceptible to being over fit and not generalizable to new data. The calculations can also become very complex when there are a number of class labels or potential outputs (Rawale, 2018).

Random Forest

The random forest is an ensemble technique which, like decision trees, can be used to perform both classification as well as regression related tasks. The random forest algorithm performs this through a technique called Bootstrap Aggregation. Bootstrap Aggregation, or as it is more commonly known as

bagging, is a technique which involves training and using multiple decision trees (Hewa, 2018). The bagging method combines together the predictions from multiple decision trees, which ultimately result in a better predictive method. The name random forest ultimately refers to the implementation of the bagging method with decision trees. Instead of using a single decision tree, multiple trees are used, hence the name forest. Ultimately, the pros for using a random forest are that its predictive power from the ensemble methods makes it one of the best supervised learning algorithms. It can provide reliable estimates based upon feature importance. Some of the main drawbacks include its ability to be human interpretable, especially when contrasted against just a single decision tree. The training can also be computationally expensive and can use up a lot of memory. The overall speed also plays into that as can create challenges for use cases based on the amount of time it takes to produce a prediction (Jansen, 2018).

K Neighbors Regressor

The K nearest neighbors algorithm is what is known as a non-parametric lazy learning algorithm. When an algorithm is considered non-parametric, it implies that it does not make assumptions based upon the underlying data distribution. When we consider an algorithm lazy, it means that it does not make generalizations based upon the training data. What this means is that in order to use new data, the training data must be retained. K nearest neighbors operates as a feature similarity, which also is where the name comes from, it operates by finding the closest "neighbor" to determine how a point is classified. The K nearest neighbors algorithm, like the others discussed, can be used for both classification as well as regression. Some of the more popular use cases for using the K nearest neighbors algorithm include things like credit ratings, potential political affiliations, and others. The benefits of using the algorithm are that it makes no assumptions about the data, it is very simple and easy to understand, and it is also versatile in use in both classification and regression. Some problems begin to occur with highly dimensional data. K nearest neighbors is computationally expensive, as a result of having to store the training data, the more data and the more complex your data is, the more expensive it is to run the algorithm. This results in high memory requirements, and can often not be the best fit for things which require timeliness of response. The k neighbors algorithm is also sensitive to the scale of the data, and sometimes to irrelevant features (Bronshtein, 2017).

Data Overview

The data used for this project was provided to Kaggle by the competition organizers, Google Cloud. The data is a set of historical taxi rides taken in New York City. There were 3 featured data sets, a train, a test, and a sample submission. The training dataset provided label training data in which to expose the algorithm. The training data featured over 55 million rows of data. There were 7 rows of data that were within the data set. There was a unique key, which uniquely identified each row in both the training and test data sets. The Unique key was a mixture of the pickup datetime and a unique integer. The data included a pickup datetime column. This is a time stamp which indicated when the taxi ride began. In the training data set specifically, there is a column for the dollar amount of the fair cost. Four of the columns contained within both the training and test data sets pertained to the starting and ending location of the taxi rides. Two of the columns were for the pickup location latitude and longitude. The other two were for the drop off latitude and longitude. The final column is the passenger count in which the ride was taken.

Software

The software packages used for this project included the anaconda distribution of the programming language Python and Jupyter Notebook. The project was written in a Jupyter Notebook. The popular version control Git was used to store various different versions. While the project was written in Python, with several imported packages were used as well. Those imported packages include Numpy, Pandas, and SkLearn.

Data Preprocessing

The data preprocessing was and is the most essential part of answering the research question being asked. The data was first loaded into a pandas data frame. From there, all preprocessing steps were applied to generated dataframe. The starting and ending latitude and longitude are the main data points in which data preprocessing was applied to. The coordinates were run through the Google Maps Directions API. The Google Maps Directions API was given the starting and Ending latitude and longitudes as arguments. The results returned: Place ID, type of location, routes of travel, duration of trip, starting address, ending address, total distance, and step by step driving directions. The travel mode was not specified, which defaults to Driving by car. Additionally, the time was not specified, which defaults to the current time. The Google API offers a free tier of requests, which limit to 100,000 monthly requests. The data preprocessing was applied, and then all rows with NA values were dropped. Some of those rows included those with bad coordinate data, ultimately resulting in 129,057 rows of the 55,000,000 rows available. The returned results which were incorporated into the models and data sets included distance and duration of trip. Distance was returned as both a text and numeric based result. The text based result was an Imperial result, for example, 2.3 miles. The numeric based result was a metric result based on meters, 3623 for example. The duration was returned in a similar format with a text and numeric result. The text result, for example, was 12 minutes. The numeric result was 739, which is based on the seconds the of the trip duration. Only the second and metric results were incorporated into the data set as part of the preprocessing. These data preprocessing steps were reapplied to the test data set. The test data set found no errors in the preprocessing, such as invalid starting or ending coordinates, resulting in a perfect test data set.

Model Creation

The training data was split into a test train split of .33 being held back for validation. The random state of the data was set at 42. The models were trained and evaluated against the split of the training data set. The results in the figure below are for the Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Square Error (RSME). While the RMSE is the ultimate metric used as the evaluation criterion for the competition, the MAE and MSE provide other insights into how the models ultimately performed with the testing data set. Unfortunately, because the Test data set true numbers are held back, these statistics will not be available for compare and contrast on how it did with the set of data ultimately used to score.

MAE	MSE	RMSE
Linear Regression		
2.5101570085271114	36.67511222076623	6.055998036720804
KNeighbors Regressor		
2.48725699124187	25.827367838268096	5.082063344574534
Random Forest		
2.581242077650527	27.32729787386641	5.227551804991167
Decision Tree Regressor		

3.206659241175312	40.604707717410015	6.372182335543296

Based upon the initial results of the training data, the KNeighbors Regressor was heavily favored to be the algorithm that was best fit to the data.

Results

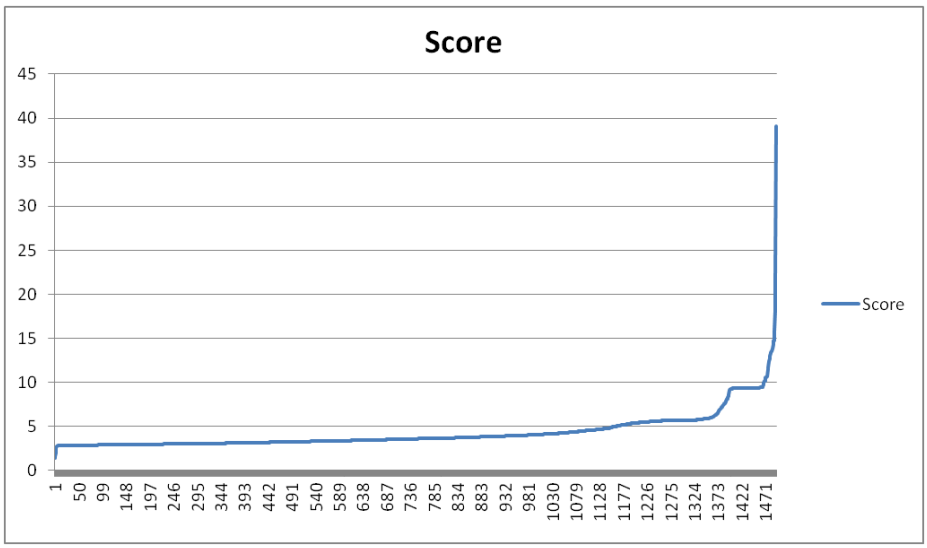
Evaluation Criteria: The competition specified that the goal metric was to reduce the Root mean squared error (RMSE). The RMSE is a scoring role that measures the average magnitude of the error. The RMSE takes the square root of the average difference between the actual observation and the prediction.

After the models were trained, they were exposed to the test data set to make predictions. Predictions were added as their own columns to the original data set. For each of the models trained, a CSV was generated of the predictive results and the unique key in the test data set. Those results were uploaded to the Kaggle Scoring engine. The returned results from Kaggle is what are presented here.

There was a total of 1488 submissions to the competition. While the competition is closed, it is possible to position the highest scoring results as position 1086 on the leader board. The bar the research question posed is clearly indicated in the leader board as 5.74184, which correlates to position 1285 on the leader board. The results of the trained models:

Model	RMSE Score
Decision Tree	5.78530
KN Regressor	4.43719
Random Forest	4.48351
Linear Regression	5.11151

Calling out the rest of the results, we can see the overall score distribution listed below. The top scoring model came in at position 1086. The second highest scoring model came in at 1094, the third came in at 1170. Finally, the weakest model, the decision tree regressor came in at position 1333.



Results Analysis

The KN Regressor had the lowest RMSE, followed by the Random Forest, The Linear Regression, and finally the decision tree. In relation to the other scores, the median score from the leader board was 3.59969 with an average score of 4.312063369 and standard deviation of 2.148519839. The results reflect the scoring based on the validation tested against the training data. When comparing apples to apples, and the linear regression results from the trained models versus the score for a basic linear model provided by the competition, we can see a clear advantage to including the external data. This, however, comes with caveats as there are several points in which bias can be entered. The first is the total number of results analyzed. The models made use of approximately 130 thousand rows of an available 55 million. That equates to the models being trained with 0.23% of the overall data. This speaks to the external data being highly helpful in the scoring. Regardless of the results appearing on the surface to be highly conclusive to the research questions presented, the ultimate comparison is not comparing to equal models. This is because of the limits of the Google Maps API. The results of the Google API were created in sequential order of the rows as they appeared in the training data set. This presents a problem as they are not a random sampling of the data. This leaves the chance that there may be bias in the selected sampling of the data. The second point of bias is the duration and length of the trip. The results used the current time as the default of the trip, which may not have been the same as the traffic at the conditions when the trip was taken. The second point of bias coming in from the Google Maps directions is that the first route was taken, which may not have been the route that was ultimately taken. New York City cab drivers are famously known for being able to navigate the streets and find the quickest and best route to get somewhere. As a result, the shortest route calculated by Google Maps, may not have actually been the route taken.

Other interesting things to note was the difference between the results for the decision tree regression and the random forest regressor. The random forest regressor was the second most accurate model, with only a, 8 position difference on the leader board, whereas the decision tree regressor came in at position 1333. One of the characteristics of decision trees is their proneness to being overfit to the data. By comparing and contrasting the score to the random forest, we can see that it has higher predictive power here as it appears to not have been overfit to the data.

Conclusions

The research topic here proposes that adding additional data such as route and duration increase the results of accuracy for predicting the cost of the fare. As we compare the linear regression model without that data, and the one that did have that data, we are able to safely say that yes, the RMSE is definitely lower with that data. That conclusion comes with those caveats discussed in the analysis of the results. The model was both trained on the same data, but one of them only used 0.23% of the available data. The other caveats are that the additional information has bias, and it is an approximation of the true duration and distance traveled, not necessarily the actual time and duration of the trip.

Future Studies

The conclusions of the study indicate that there is definitely merit to further study here. At this point, it would make sense to propose budget and examine further. The Google Maps Route API provides the first 100,000 calls for free, which is an equivalent of $200 in usage for free. The cost is $5 per 40,000 calls, which would mean that 1,373 bundles of 40,000 calls would be needed in order to fully evaluate the data (Pricing Table | Google Maps Platform | Google Cloud. n.d.). The net cost of continuing this evaluation with the logic that Google Maps provided for the route and duration would be $6,865. There are other open source alternatives, however in order to have uniformity, then it would make sense to start over from scratch with the alternative rather than using the existing API data collected.

References

Bronshtein, A. (2017, April 11). A Quick Introduction to K-Nearest Neighbors Algorithm. Retrieved from https://blog.usejournal.com/a-quick-introduction-to-k-nearest-neighbors-algorithm-62214cea29c7

Essentials of Linear Regression in Python. (n.d.). Retrieved from https://www.datacamp.com/community/tutorials/essentials-linear-regression-python

Linear Regression. (n.d.). Retrieved from http://www.stat.yale.edu/Courses/1997-98/101/linreg.htm

Hewa, K. (2018, November 27). A Beginners Guide to Random Forest Regression. Retrieved from https://medium.com/datadriveninvestor/random-forest-regression-9871bc9a25eb

How to use Kaggle. (n.d.). Retrieved from https://www.kaggle.com/docs/competitions

Jansen, S. (2018, December 31). Hands-On Machine Learning for Algorithmic Trading. Retrieved from https://learning.oreilly.com/library/view/hands-on-machine-learning/9781789346411/

New York City Taxi Fare Prediction. (n.d.). Retrieved from https://www.kaggle.com/c/new-york-city-taxi-fare-prediction/overview

Pal, M. (2005). Random forest classifier for remote sensing classification. *International Journal of Remote Sensing*, *26*(1), 217-22

Pricing Table | Google Maps Platform | Google Cloud. (n.d.). Retrieved from https://cloud.google.com/maps-platform/pricing/sheet/

Rawale, S. (2018, May 30). Understanding Decision Tree, Algorithm, Drawbacks and Advantages. Retrieved from https://medium.com/@sagar.rawale3/understanding-decision-tree-algorithm-drawbacks-and-advantages-4486efa6b8c3

Shetty, B. (2018, December 12). Supervised Machine Learning: Classification. Retrieved from https://towardsdatascience.com/supervised-machine-learning-classification-5e685fe18a6d

Shuster, M. (2015, November 23). Machine Learning Algorithm Series: Linear Regression. Retrieved from https://blog.knowledgent.com/machine-learning-algorithm-series-linear-regression/

Soni, D. (2018, March 22). Supervised vs. Unsupervised Learning. Retrieved from https://towardsdatascience.com/supervised-vs-unsupervised-learning-14f68e32ea8d

Web Services Directions API. (n.d.). Retrieved from https://developers.google.com/maps/documentation/directions/start

Xu, M., Watanachaturaporn, P., Varshney, P. K., & Arora, M. K. (2005). Decision tree regression for soft classification of remote sensing data. *Remote Sensing of Environment*, *97*(3), 322-336.