Taxis Cab Analysis

Group 3
Nathan England (nle4bz)
Yihnew Eshetu (yte9pc)
Karyne Williams (kw7me)

Abstract

While many ridesharing apps are rendering taxi cab services obsolete in cities all over the United States, taxi cabs are still highly utilized in New York City. We chose to investigate four months of 2019 data from the NYC Yellow Taxi Trip data with the aim of whether we could predict taxi cab driver's tip amount and whether or not a passenger is going to pay for a given taxi cab ride. We ran a simple linear and logistic regression, multiple linear and logistic regression, gradient-boosted tree regression and decision tree regression models to find our solutions. The gradient boosted and tree regression models showed that trip distance, fare amount, the toll amount, and the congestion surcharge variables had the largest impact on a taxi cab drivers' tip amounts. The gradient boosted model best predicted the tip amount with a RMSE of 1.91, but the relatively high RMSE and low R2 is evidence of limited predictive power of the model. Implementing a logistic regression model with 10 predictors, we were able to predict whether or not a passenger is going to pay for a cab ride with 66.9% accuracy. The more complex logistical regression model with 10 predictors improved the predictive performance over the simpler logistic model substantially, however, predictive performance was still limited.

Data Exploration, Preprocessing, and Methods

New York City Taxi & Limousine Commission provides records for every for-hire yellow and green taxis, and their monthly trip data to the public. For our project, we focused on 2019 yellow taxi trips. The yellow taxi trip records include fields capturing pick-up and drop-off dates and times, pick-up and drop-off locations, trip distances, itemized fares, rate types, payment types, and driver-reported passenger counts.

To download the necessary CSV files and data dictionary, we wrote a python script that was able to retrieve the files stored on Amazon S3. Once the files were downloaded into Rivanna, we used four months of data (May - August) from 2019 to limit the computational strain of using all twelve months. Afterward, we joined all four months into one data frame, joined the taxi zone lookup to the data frame, and added additional columns, such as pick-up and drop-off Borough, pick-up and drop-off time, and pick-up and drop-off service zone. Next, we dropped any duplicate records and defined the column types. We derived features from the preexisting features, such as the day of pick-up and drop-off, the hour of pick-up and drop-off, trip time, and distance. Afterward, we decided to remove timestamp columns and we dropped NULLs and NAs.

Our next step was examining our data and identifying any possible outliers. We noticed that there were payment records with a total amount less than zero for credit card and cash

payment. We also noticed that there were trips with a trip time less than zero and trip distance less than zero. We believed all three conditions were outliers and removed any of those records.

Next, we further explored our response and predictor variables. Our first response variable to predict for our first research question was tip amount. From the data dictionary, we noticed that the tip amount field is automatically populated for credit card tips and does not include cash tips. Thus, we filtered the data frame to include only credit card payment types. As shown in Figure 1, we quickly noticed that the distribution of tip amount is skewed to the left, indicating that most passengers usually tip in the single digits. Furthermore, the mean for the tip was about $3.12 with a standard deviation of $2.98, as shown in Figure 2. Before moving forward to building our models, we noted that total amount variable should not be used as a feature because it includes tip amount.

Figure 1: Histogram of Tip Amount

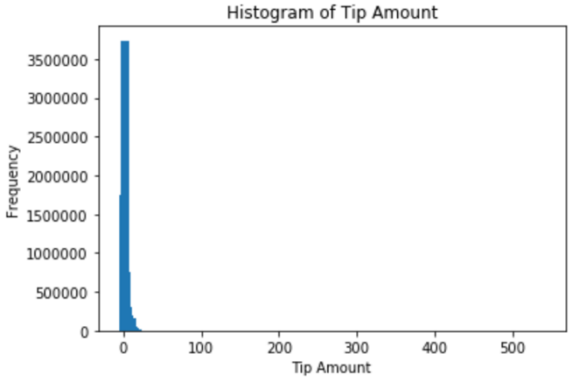
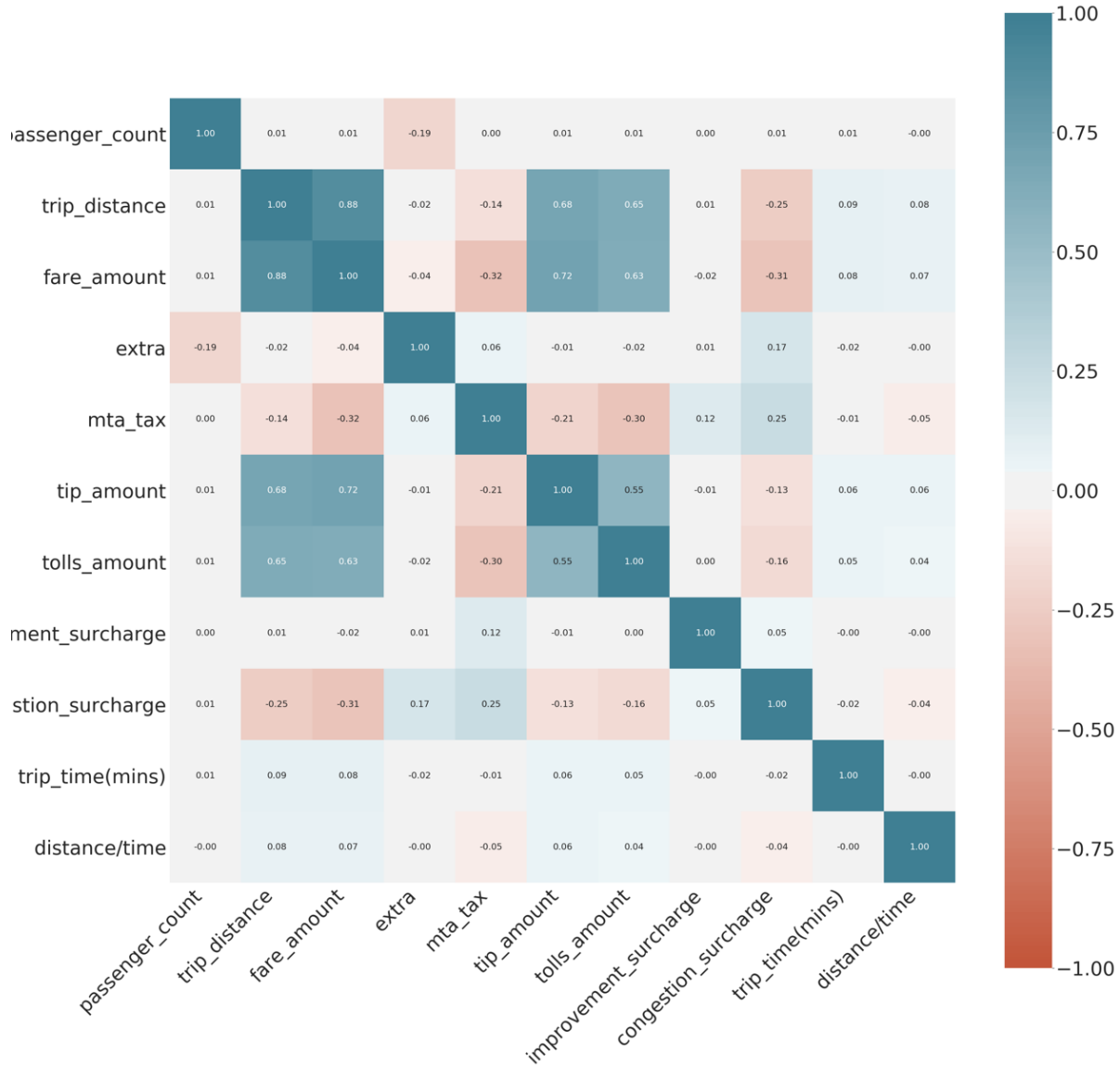


Figure 2: Summary of Tip Amount

summary	tip_amount
count	12696866
mean	3.1224676651878704
stddev	2.9783664721843484
min	0.0
25%	1.75
50%	2.36
75%	3.5
max	537.0

In order to determine what factors impacted a taxis cab driver's tip amount, we plotted a correlation matrix of all useful predictors, Figure 3. From the correlation matrix we noticed that fare amount and trip distance have a high correlation (r= 0.88) between one another. This made sense since the fare amount is calculated based on mileage of the trip. We also noticed that fare amount and trip distance have the highest correlation to tip amount, (r = 0.72 and r = 0.68). Once we had a basic understanding of how each feature related to tip amount, we started building our models to predict tip amount.

Figure 3: Correlation Matrix



Our second response variable was a binary classification label indicating whether or not the passenger paid for a given ride. We created this label by removing voided and unknown trips and then segmented the remaining data based on payment type. Trips where rides were paid for with credit card or cash were labelled with “1” to denote the trip was paid for and trips with no charge or disputed charges were denoted with “0” to indicate the trip was not paid for. Figure 4 displays the resulting split of paid and unpaid trips after applying this labelling convention. It quickly became apparent to us that the data was unbalanced with less than 1% of trips being labelled as unpaid for. We decided that in order to avoid improper training and testing biases with our models, a sampling method would be necessary to better balance the data. Specifically, we down-sampled instances where trips were labelled as paid for by sampling 1% of the positive cases without replacement. At this point the 1% sample of positives was still

larger than the number of negatives (i.e. trips unpaid for) so we up-sampled the instances where trips were unpaid for by sampling with replacement. Figure 5 displays the balanced data after implementing these sampling methods. We were then ready to build our binary classification models.

Figure 4: Original split of classification labels

paid	count
1	17743064
0	141077

Figure 5: Balanced split of classification labels after resampling

paid	count
1	176927
0	176830

Results

Tip Amount

We looked at four different types of models to build the best model possible for determining how much a passenger would tip.

Basic Linear Regression Model (Benchmark)

The benchmark model was a basic linear regression model with fare amount as the only feature. Though fare amount had about the same correlation to tip amount as trip distance, we decided to use fare amount because it would be redundant to use both. After splitting the data into an 80% training and 20% testing sets, we created a pipeline that consisted of a vector assembler, standard scaler, and linear regression (maxIter = 10, regParam = 0.01). We then fitted the training set on the pipeline to build the model and transformed the model on the test set. Our benchmark model had a Root Mean Square Error (RMSE) of 2.121. Which is not great since 75% of tips are between $0 and $3.5. The model's R2 value of 0.502, indicates that just 50% of the sample proportion of variance is explained by the fare amount predictor.

Multiple Linear Regression Model

The multiple linear regression model consisted of 4 features: fare amount, trip distance, toll amount, and congestion surcharge. These features were selected based on their correlation to the tip amount and their correlation to each other. Similar to the benchmark model, the data was split into a 80% training and 20% test set. Then a pipeline was created consisting of 3 stages: vector assembler, standard scaler, and linear regression. Afterwards, we created a paramGrid for regParam and elasticNetParam. Then used a 10 fold CrossValidator for model selection using the pipeline as the estimator, the paramGrid as the estimatorParamMaps, and the RegressionEvaluator RMSE as the evaluator. Then fitted the training set on the

crossValidator. Which returned the best model with the following parameters, regParam = 0.01 elasticNetParam = 0, and, maxIter = 100.
Afterwards the test set was transformed using the best model. The RMSE for the best model was 2.043, again not so great but better than the benchmark model. Since the model had multiple features we calculated the AdjR2 to be 0.537. Thus, adding more features improved RMSE and AdjR2.

Gradient-boosted Tree Regression Model (Champion Model)
The gradient-boosted models consisted of 10 features: fare amount, trip distance, toll amount, congestion surcharge, improvement surcharge, pick-up and drop-off borough, trip time, distance/time, and pick up week day. Like the other models, the data was split into a 80/20 training and test set. We first created a vector assembler that returned all 10 features in a vector to be passed to the vector indexer. The vector indexer would then automatically identify categorical features, and index them. We specified the maxCategories so features with distinct values greater than 7 would be treated as continuous. Afterwards, we created a user defined function (UDF) that passed maxIter and maxDepth to the GBTRegressor. This allowed us to modify and run GBTRegressor using different combinations of maxIter and maxDepth. Within the UDF, a pipeline was created consisting of 3 stages: vector assembler, vector indexer, and GBTRegressor. Then the training set was fitted on the pipeline to create a model, which then was transformed on the test set. The UDF would then print out the model evaluation results. Using the UDF, we were able to run 5 different models.
We started with a maxIter value of 5 and maxDepth value of 3. We noticed that as we increased the values for maxIter and maxDepth the performance of the models improved. Our first model had a RMSE of 1.99 and an AdjR2 of 0.545. It performed better than the benchmark model and multiple linear model but still needed improvement. Our best gradient boosted model consisted of maxIter = 15 and maxDepth = 10, and had a RMSE of 1.91 and an AdjR2 of 0.583.

Decision Tree Regression Model
Two Decision Tree Regression Models were run on the dataset that was split into a 70% training and 30% testing set. The initial baseline model used all ten predictors. Running the model on the training sample yielded an RMSE of 1.961 with a 0.575 R2. The performance suffered slightly when running the model on the testing sample, generating an RMSE of 1.992. The featureImportances() function from the pyspark.ml package, showed that the fare amount had the greatest influence on the model's decisions, with the trip distance, toll amount, and congestion surcharge also showing very modest effects.

The reduced model used the top four predictors suggested by the feature importance values. The model performed slightly worse on the training data compared to the full model, but it had the best performance of both models on the testing sample with an RMSE of 1.98946 and an R^2 of 0.553. Finally, a 3-fold cross validation was used to determine the parameters for the best model using the same four predictors. The parameter grid tested values 2, 5, 8, 10, 20, and 25 for the maximum tree depth, and values 5, 10, 15, 20, 25, 32 for the maximum bins. The cross validation results returned a model with equivalent RMSE to that of the reduced model,

indicating that parameters using 10 as the maximum depth and 24 (or 25) maximum bins yielded the best results.

Figure 6: Summary of Tip Amount Models Performance

	RMSE	R2	AdjR2
Basic Linear **(Bench Mark)**	2.121	0.502	0.502
Linear w/ Model Selection & Tuning	2.043	0.537	0.537
Gradient Boost w/ Model Selection & Tuning **(Champion Model)**	1.910	0.583	0.583
Decision Tree Regression w/ Model Selection & Tuning	1.989	0.553	0.553

Paid Classifier

We built two binary classification models to predict whether or not a passenger would pay for a given taxi cab ride.

Basic Logistic Regression Model (Benchmark)

The benchmark logistic regression consisted of two predictors: fare amount and trip distance. After splitting the data into a 80/20 training and test set, we created a pipeline that consisted of a vector assembler, standard scaler, and logistic regression (maxIter = 10). Then we created a paramGrid for regParam (values used were 0.01 and 0.1) and elasticNetParam (values used were (0, 0.5, 1) and implemented 10 fold CrossValidator for model selection using the pipeline as the estimator and the paramGrid as the estimatorParamMaps. The model selected by this cross validation was then used to make predictions on the 20% hold out split. See Figures 7 and 8 for the summary evaluation metrics and confusion matrix output from this model.

Figure 7: Benchmark Evaluation Metrics

```
Summary Stats
Accuracy = 0.5627411484339537
Precision = 0.5896986824821876
Recall = 0.4193082757839918
F1 Score = 0.4901166156645439
Area under ROC = 0.5630837818460399
```

Figure 8: Benchmark Confusion Matrix

```
Confusion Matrix
[[24856. 10308.]
 [20517. 14815.]]
```

In assessing these statistics, it became apparent that there was significant room for improvement considering the model's low accuracy of 0.5627. Even more notable, was the model's low area under the ROC with a value of 0.5631, suggesting that the model only had slightly more predictive power than random guessing. It was clear that a more advanced model was needed.

Complex Logistic Regression Model (Champion Model)

This new logistic regression model used all of the same training/testing splits, pipeline, paramGrid, and cross validation processes as the benchmark logistic regression model. The only difference was this new model was more complex in that it consisted of ten predictors: passenger count, trip distance, rate code ID, fare amount, tolls amount, improvement surcharge, congestion surcharge, pickup borough, pickup week day, pickup hour. The aforementioned processes were applied and predictions were made on the 20% hold out split. See Figures 9 and 10 for the Confusion matrix and summary evaluation metrics outputted from this model.

Figures 9: Complex Model Evaluation Metrics

```
Summary Stats
Accuracy = 0.6685060145256468
Precision = 0.6131124600518144
Recall = 0.9176384014491112
F1 Score = 0.7350843979912257
Area under ROC = 0.6679108854020668
```

Figure 10: Complex Model Confusion Matrix

```
Confusion Matrix
[[14705. 20459.]
 [ 2910. 32422.]]
```

In assessing these statistics it immediately became more clear that this model outperformed the benchmark model in terms of accuracy with a value of 0.6685 compared to the benchmark's 0.5627, but this model in fact improved across the board for every single evaluation metric. Furthermore, this model's area under the ROC was 0.6679, suggesting that this model had substantially more predictive power than the benchmark model. See Figure 11 for comparison of all metrics.

Figure 11: Summary of Logistic Regression Model Performance

	Accuracy	**Precision**	**Recall**	**F1 Score**	**Area Under ROC**
Basic Logistic Regression	0.5627	0.5897	0.4193	0.4901	0.5631
Complex Logistic Regression	0.6685	0.6131	0.9176	0.7351	0.6679

Conclusions

We were able to build different types of models for predicting tip amount. Our linear regression benchmark model was the worst performing model but it was useful as a basis for comparison. Our multiple linear regression was better than the benchmark model and we noticed how adding a 10 fold crossValidator and paramGrid improved our results. Compared to our regression models, the tree regression models had higher RMSE and AdjR2. This is not surprising since trees support non linearity, where linear regression supports linear solutions. Furthermore, tree regression handles collinearity better than linear regression. Although, gradient boosted and decision tree had similar RMSE, the gradient boosted tree was the best performing tree model. Gradient boosted works well because of its ability to use additive models to add weak learners to minimize the loss function.

In addition to tip amount models, we were able to build two logistic regression models to predict whether or not a passenger was going to pay for a taxi cab ride. The worse performing model was the benchmark model with only two predictors while our more advanced model with ten predictors was the better performing model not only in terms of accuracy, but in every single evaluation metric looked at. This should come as no surprise considering both models had the exact same implementation with the only difference being that the more advanced model had eight more predictors than the benchmark model. It is clear that adding some features improved predictive performance, but what we are unable to discern at this stage in the analysis is which predictors specifically resulted in the improved performance and which predictors did not significantly assist in performance. It is also worth noting that the best performing model also only had an accuracy of 0.6685 and an area under the ROC of 0.6679 so there is still plenty of room for improvement through the use of better model turning and feature selection, or even potentially through the use of a different type of binary classification model.

Future Research

One of the biggest difficulties we had was runtime of tree model training. Running a crossValidator on a tree regression with a large paramGrid, maxDepth greater than 15, and number of folds greater than 3 consumed computing power and time. Nevertheless, increasing parameter value would lead to better results.

Initially we had planned on conducting our project on one year worth of data, but realized that the size of the dataset would be challenging for model selection. Going forward, we would want to use a larger dataset consisting of multiple years. However, this raises the question how well our current model would behave to a 2020 dataset during the middle of a pandemic. Would there be fewer rides and would passengers give smaller tips?

In conclusion a better understanding of how the data was collected as well as access to more predictors could better enhance any further analysis to be conducted in the subject area of taxi cab rides. This could enable further model tuning and optimal feature selection with existing models as well as other models not tried in this paper.