Chicago Taxi Tips Prediction

Jeriel Lao
jlao@stevens.edu

Jian Hui Mai
jmai2@stevens.edu

Tianhao Zhao
tzhao21@stevens.edu

***Abstract*—Taxi drivers rely on tips as a percentage of their income, but have no easy way of considering account how much that portion accounts for. In response, we have constructed multiple algorithms with the goal of providing a reliable solution to gauging a tip on a per ride basis. As of the writing of this paper, we have tested and expanded upon multiple iterations of Machine Learning algorithms using K Nearest Neighbor, Support Vector, and Random Forest Regressors. Most of our initial models found improvements with updating feature space and optimizing hyper-parameters, reducing average error by over 50%. While there are no pre-existing solutions for the data we've considered, for problems of a similar focus, we've achieved comparable results in possibly a fraction of the time.**

I. Introduction

A. Problem Statement

With the current economic situation limiting travel, workers dealing in transportation need to be efficient in their rides due to limited frequency. Given this shortage, it is important to maximize every part of the revenue stream for any given driver, which for taxi drivers includes tips. However, there is no way of estimating tip given its financial relevancy in the overall revenue stream of the driver. With this in mind, we have created a machine learning algorithm to help predict the amount of tips a driver should expect in each ride.

B. Experimental Results

In consideration of predicting tips, some situations we are taking into account are the length of the trip in both miles and time and the total cost of the trip. Given the low accuracy of our initial model, we decided to change these situations. In this project, we used several methods to select the best features for model training in order to improve our prediction accuracy, and we tuned the model parameters for a lower Root Mean Square Error (RMSE), which gave us information on the differences of actual values and our predicted values. The change our feature selections for model training made our predictions values much closer to actual values and the accuracy is much higher.

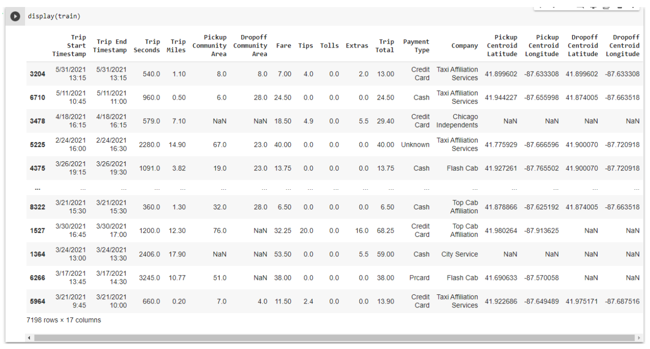
II. Related Work

An existing technique used a New York City taxi data set to predict the amount of tips given by a particular rider. A simple Google search of "Taxi Tip Predictions" yields many results which are all related to New York City. While many options can be referenced, this paper will briefly cover the first two. The first result relates to a Medium article [2]. The Medium article provides a step-by-step guide from pre-processing to training. The data set used comes from the NYC Taxi Commission. Additionally, a subset of the data set was used, namely only November and December 2019. The columns of the data set included the 'VendorID', 'tpep_pickup_datetime', 'tpep_dropoff_datetime', 'passenger_count', 'trip_distance', 'RatecodeID', 'store_and_fwd_flag', 'PULocationID', 'DOLocationID', 'payment_type', 'fare_amount', 'extra', 'mta_tax', 'tip_amount', 'tolls_amount', 'improvement_surcharge', 'total_amount', and 'congestion_surcharge'. The algorithms that were implemented include K Nearest Neighbor, Random Forest and Decision Tree. The independent variables were generated using heat map analysis and include only the 'PULocationID' and 'DOLocationID'. The dependent variable includes only the 'tip_amount'. In comparison, the data set used in this problem consists of different columns and does not have a pick up and drop off location ID; We only have the pickup and drop off community area and latitude and longitude locations. The second result yields a GitHub repository. [3] The data set also comes from the NYC Taxi Commission and includes only 2013 data. The columns of the data set includes 'vendor_id', 'pickup_datetime', 'dropoff_datetime', 'passenger_count', 'trip_distance', 'pickup_longitude', 'pickup_latitude', 'rate_code', 'store_and_fwd_flag', 'dropoff_longitude', 'dropoff_latitude', 'payment_type', 'fare_amount', 'surcharge', 'mta_tax', 'tip_amount', 'tolls_amount', and 'total_amount'. The independent variables include 'medallion', 'hack_license', 'vendor_id', 'pickup_month', 'pickup_weekday', 'pickup_day', 'pickup_time_in_mins', 'pickup_non_working_today', 'pickup_non_working_tomorrow', 'fare_amount', 'surcharge', 'tolls_amount', 'passenger_count', 'trip_time_in_secs', 'trip_distance', 'pickup_longitude', 'pickup_latitude', 'dropoff_longitude', 'dropoff_latitude'. The dependent variable is the amount of tips. The implementation of this model utilizes only Random Forest. Compared to the implementation, we are using only a random subset of the 2021 data and the data set does not contain many of the features like 'passenger_count'. Additionally, we planned to utilize more than one training algorithm to create the best accurate model.

III. Our Solution

A. Description of Data Set

We are using the data of the taxi trips reported to the City of Chicago as the data set [1]. The taxi trips information in the data set is the data source for analyzing and predicting the taxi trip tips. Since there are millions of data entries in the data set, we want to choose a relatively appropriate amount of data for the model training to optimize the time designated for purely training. By considering its size, useful information and computational limitations of our computers during training, we will trim and choose only 1% of the data from the year of 2021.



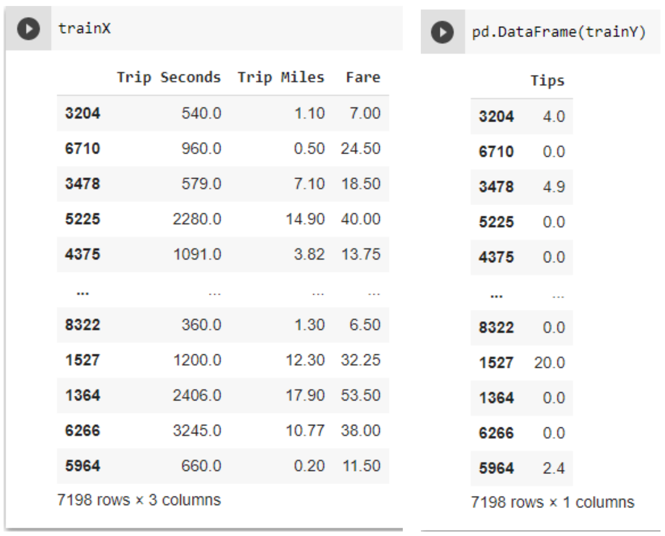
```
display(train)
```

	Trip Start Timestamp	Trip End Timestamp	Trip Seconds	Trip Miles	Pickup Community Area	Dropoff Community Area	Fare	Tips	Tolls	Extras	Trip Total	Payment Type	Company	Pickup Centroid Latitude	Pickup Centroid Longitude	Dropoff Centroid Latitude	Dropoff Centroid Longitude
3204	5/31/2021 13:15	5/31/2021 13:15	540.0	1.10	8.0	8.0	7.00	4.0	0.0	2.0	13.00	Credit Card	Taxi Affiliation Services	41.899602	-87.633308	41.899602	-87.633308
6710	5/11/2021 10:45	5/11/2021 11:00	960.0	0.50	6.0	28.0	24.50	0.0	0.0	0.0	24.50	Cash	Taxi Affiliation Services	41.944227	-87.655998	41.874005	-87.663518
3478	4/18/2021 16:15	4/18/2021 16:15	579.0	7.10	NaN	NaN	18.50	4.9	0.0	5.5	29.40	Credit Card	Chicago Independents	NaN	NaN	NaN	NaN
5225	2/24/2021 16:00	2/24/2021 16:30	2280.0	14.90	67.0	23.0	40.00	0.0	0.0	0.0	40.00	Unknown	Taxi Affiliation Services	41.775929	-87.666596	41.900070	-87.720918
4375	3/26/2021 19:15	3/26/2021 19:30	1091.0	3.82	19.0	23.0	13.75	0.0	0.0	0.0	13.75	Cash	Flash Cab	41.927261	-87.765502	41.900070	-87.720918
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
8322	3/21/2021 15:30	3/21/2021 15:30	360.0	1.30	32.0	28.0	6.50	0.0	0.0	0.0	6.50	Cash	Top Cab Affiliation	41.878866	-87.625192	41.874005	-87.663518
1527	3/30/2021 16:45	3/30/2021 17:00	1200.0	12.30	76.0	NaN	32.25	20.0	0.0	16.0	68.25	Credit Card	Top Cab Affiliation	41.980264	-87.913625	NaN	NaN
1364	3/24/2021 13:00	3/24/2021 13:30	2406.0	17.90	NaN	NaN	53.50	0.0	0.0	5.5	59.00	Cash	City Service	NaN	NaN	NaN	NaN
6266	3/17/2021 13:45	3/17/2021 14:30	3245.0	10.77	51.0	NaN	38.00	0.0	0.0	0.0	38.00	Prcard	Flash Cab	41.690633	-87.570058	NaN	NaN
5964	3/21/2021 9:45	3/21/2021 10:00	660.0	0.20	7.0	4.0	11.50	2.4	0.0	0.0	13.90	Credit Card	Taxi Affiliation Services	41.922686	-87.649489	41.975171	-87.687516

7198 rows × 17 columns

Fig. 1. Sample of the Data Set

As shown in Figure 1, the original data set includes the travel time, travel distance, trip fare, tips, and some other information. We used the travel time, distance, and fare as the major factors of the model training in the initial attempt. We are analyzing the relationship between those factors and tips of the trips. Since the data is inclusive and includes some non-useful trip information, like empty entries, we will remove those entries in the travel time, distance, and fare columns. Then, we split the data set into training data set and validation data set. We will then shuffle the data and use 80 percent of the data for training and 20 percent of the data for testing. We set tips as the dependent variable as shown on Figure 2.



```
trainX
```

	Trip Seconds	Trip Miles	Fare
3204	540.0	1.10	7.00
6710	960.0	0.50	24.50
3478	579.0	7.10	18.50
5225	2280.0	14.90	40.00
4375	1091.0	3.82	13.75
...	...	...	...
8322	360.0	1.30	6.50
1527	1200.0	12.30	32.25
1364	2406.0	17.90	53.50
6266	3245.0	10.77	38.00
5964	660.0	0.20	11.50

7198 rows × 3 columns

```
pd.DataFrame(trainY)
```

	Tips
3204	4.0
6710	0.0
3478	4.9
5225	0.0
4375	0.0
...	...
8322	0.0
1527	20.0
1364	0.0
6266	0.0
5964	2.4

7198 rows × 1 columns

Fig. 2. Sample of X and Y Training Sets

In our training with the initial subset, we received a RMSE of around 3. We assume that noise or faulty data might be the cause. We decided to create a second subset called ”TaxiTrip2021Subset.csv” which removed all empty data from any of the rows. This data set was created using the ”preprocessing.ipynb” script. The resulting subset would also contain only 1% of the original data.

B. *Machine Learning Algorithms*

1) *K Nearest Neighbor Regression:*

K Nearest Neighbor (KNN) works by storing the whole training data. In order to make predictions, the KNN algorithm finds similar data to the unknown data. The new data will then be summarized by its neighbors which then outputs a prediction. In other words, a new piece of data is predicted using the similarity score of the available data. KNN might be appropriate for the problem because KNN is a non-parametric algorithm meaning that there are no assumptions to implement KNN. For example, in Linear Regression, the data set needs to be linearly separable which we could not determine if this data set is. Additionally, the data set is relatively small therefore the performance limitations of KNN should not affect us. The initial values are as follows: n_neighbors = 5, algorithm = ’auto’, leaf_size = 30, p = 2, metric = ’minkowski’, metric_params = ’None’, n_jobs = ’None’.

2) *Support Vector Regression:*

Support Vector Regression (SVR) gives us the prediction on how the data are correlated to each other within a certain range. Like Support Vector Machine algorithm, it is aiming to find a center line between data with a decision boundary, then to consider the data points within decision boundary. We could use this feature to see how much differences are there between prediction results and true values. Therefore, we could improve the prediction accuracy by tuning the parameters throughout testing. By considering how the data is spread out along the variables, the SVR algorithm finds a middle line to separate the data points linearly, and since our data set is relatively small, SVR would also be a good fit for this particular problem. For our initial attempt on the SVR model, we use the default setting of input parameter values: kernel to ’rbf’, degree to 3, gamma to ’scale’, coef0 to 0.0, tol to 0.001, C to 1.0, epsilon to 0.1, shrinking to ’True’, cache_size to 200, verbose to ’False’, and max_iter to -1.

3) *Random Forest Regression:*

Random Forest (RF) Regression is an ensemble learning method of training where it simulates possible outputs through multiple decision trees to predict a line of regression based on the data. The errors are meant to be smoothed out with multiple iterations of decision trees, by averaging their results. RF Regression could prove to be quite useful in this problem as there appears to be a large number of outliers which can introduce an unwanted shift in the predicted data. Additionally, since we have a small data set, it is not as computationally expensive thus a great potential fit.

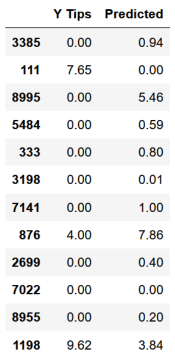
C. *Implementation Details*

1) *K Nearest Neighbor Regression:*

Original Model

Using a random subset called 'TaxiTrip2021SubsetOriginal.csv', a 80% training and 20% testing set was created by utilizing Pandas Data Frames. The training and testing sets would then be divided into train_X, train_Y, test_X and test_Y where the X contain the columns for 'Trip Seconds', 'Trip Miles' and 'Fare' while the Y contains only the 'Tips'. (The columns for X were determined by what we believe were best correlated to the tips at the time). A KNN Regression model was then created using SK-learn and both train_X and train_Y. No parameters were passed resulting in the default values of n_neighbors, weights, algorithm, leaf_size, p, metric, metric_params and n_jobs. Additionally, a pred_Y was created to predict the test_X data. RMSE was used to measure the accuracy of my model by comparing the prediction and true values in test_Y. The result yielded an RMSE of 3.14. Given the high error value, it can be said for certain that the model is not accurate at all.

Additionally, from the chart on Figure 3, we can see that many of the predicted values have a huge difference from the true values. For example, many of the $0 tips are incorrectly predicted with a value of $1 and many of the high tips are predicted with a value a lot lower than the actual. These differences contribute to the high RMSE value found above. Also, the scatter plot which models the Fare as the X value and Tips as the Y show that our model performs terribly on high tip values where all the predicted tips are below $10. Due to this for the next steps of this project, the hyper-parameters will need to be adjusted to facilitate the training of the model and the independent variables will be changed.



	Y Tips	Predicted
3385	0.00	0.94
111	7.65	0.00
8995	0.00	5.46
5484	0.00	0.59
333	0.00	0.80
3198	0.00	0.01
7141	0.00	1.00
876	4.00	7.86
2699	0.00	0.40
7022	0.00	0.00
8955	0.00	0.20
1198	9.62	3.84

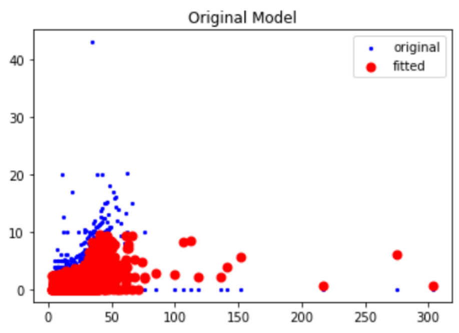


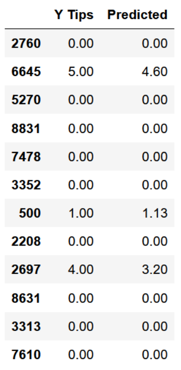
Fig. 3. Sample of X and Y Training Sets Original Model

Feature Tuned Model

Given the model that created above, we wanted to utilize a correlation statistics algorithm to determine the best features to predict the tips. In order to do this, we had decided to utilize LabelEncoder() to convert all categorical values. We also just kept only Trip Start Hour and removed the Trip Start Timestamp and Trip End Timestamp columns. In order to retrieve the most correlated features, we wanted to try three correlation statistics algorithms, Pearson's Correlation Coefficient, Chi-Square and Mutual Info Regression to determine which statistical method would yield me the lowest RMSE. Since my data set contained negative values in the longitude columns, we were not able to use the Chi-Square Test. Additionally, Pearson's did not work because a lot of my data set contained values of 0 which yielded a division by 0 error. Unfortunately at this point, the only available option was to use Mutual Info Regression. The Mutual Info Regression correlation statistics works by measuring the dependency between the variables. The Mutual Information will be equal to 0 if two random variables are independent. Additionally, higher values will signify higher dependency.

For each value returned by Mutual Info, they were added to a dictionary which was then sorted. The four features that had the highest values were then created as the train_X and test_X. Using Mutual Info to find the best correlated features, the model was later trained using the same default parameters in my original model. Afterwards, an RMSE of 1.07 was achieved, which was around 65% lower even with default parameters.

Using a similar chart as seen in the original model, you can see that in Figure 4 many 0 tips are predicting numbers closer to 0. Additionally, with the scatter plot which models fare as x and tips as y, we can see that most of the predicted (fitted) points were closer to the true (original) tip values. Given the couple of outliers here and there, it can be said for certain that this model is more accurate than the original model.



	Y Tips	Predicted
2760	0.00	0.00
6645	5.00	4.60
5270	0.00	0.00
8831	0.00	0.00
7478	0.00	0.00
3352	0.00	0.00
500	1.00	1.13
2208	0.00	0.00
2697	4.00	3.20
8631	0.00	0.00
3313	0.00	0.00
7610	0.00	0.00

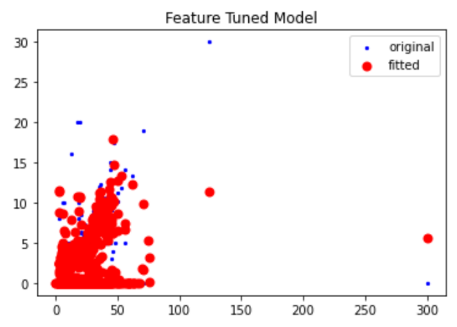
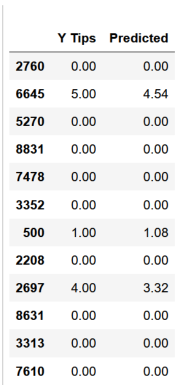


Fig. 4. Sample of X and Y Training Sets Feature Tuned Model

Feature and KNN Parameter Tuned Model

In an effort to further improve the accuracy of my model, GridSearchCV was used to find the best parameters. The parameters of my search include n_neighbors of 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, weights of uniform and distance, and algorithm of auto, ball_tree, kd_tree and brute. The result of my search yielded the best parameters of ball_tree for algorithm, n_neighbors of 7, and weights of distance. Using these parameters, we were able to yield a RMSE of 0.98 which was around 10% lower than the feature tuned model.



	Y Tips	Predicted
2760	0.00	0.00
6645	5.00	4.54
5270	0.00	0.00
8831	0.00	0.00
7478	0.00	0.00
3352	0.00	0.00
500	1.00	1.08
2208	0.00	0.00
2697	4.00	3.32
8631	0.00	0.00
3313	0.00	0.00
7610	0.00	0.00

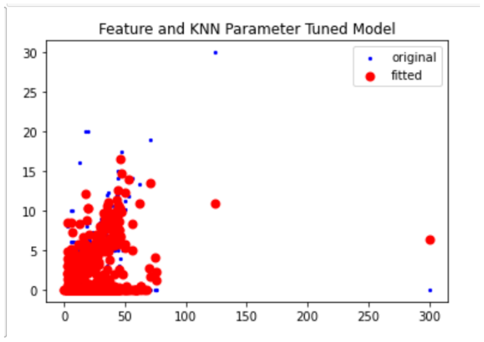


Fig. 5. Sample of Y Tips and Predicted Features and Parameter Tuned Model

As depicted on Figure 5, the predicted tips do not seem far off than the true values. The tips with \$0 yielded a prediction of \$0 and those who tipped are not far off the predicted values. Additionally, the scatter plot looks very similar to the one on figure 3. Since this model yielded me the lowest RMSE value, this would be my final model used to predict tips.

2) *Support Vector Regression:*

Original Model

We got training set and validation set with the same initial features as above algorithm, by using Pandas Data Frame which makes it easier to manipulate and split data. Since the SVR algorithm provides the differences of prediction results by showing error between them and true value, we use Standard Scalar from SK-learn library in order to make the prediction reasonable in comparison. The Standard Scalar library also could scale the prediction result back to the original scale, which makes it easy to compare and analyze for us. By training the model and predicting the testing set, we got the comparison result as shown in Figures 6 and 7.

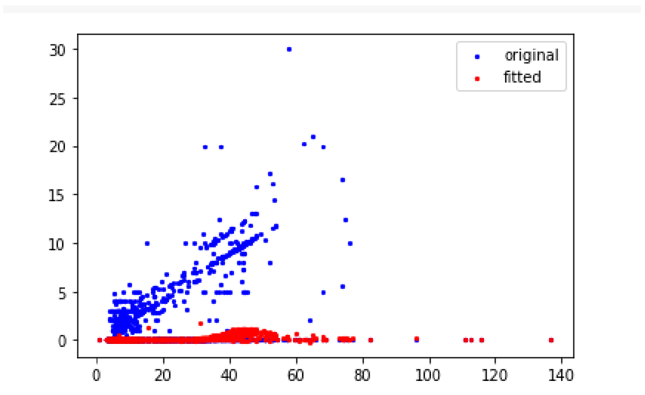
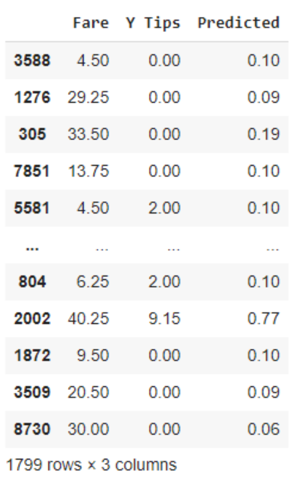


Fig. 6. SVR Prediction vs. True Value.



	Fare	Y Tips	Predicted
3588	4.50	0.00	0.10
1276	29.25	0.00	0.09
305	33.50	0.00	0.19
7851	13.75	0.00	0.10
5581	4.50	2.00	0.10
...	...	...	...
804	6.25	2.00	0.10
2002	40.25	9.15	0.77
1872	9.50	0.00	0.10
3509	20.50	0.00	0.09
8730	30.00	0.00	0.06

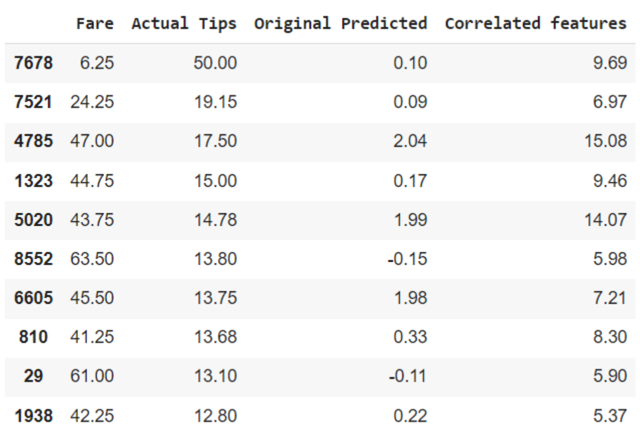
1799 rows × 3 columns

Fig. 7. SVR Prediction Values.

By comparing to the true value, the prediction result is mostly small numbers, possibly due to high occurrence of zero entries for tips from the original data set during training. In order to improve our SVR model and get better prediction result, we need to reconsider our training features.

Feature Tuned Model

Even though the features we selected are the ones that seems more related to the target tips in logical sense, we still need to use different methods to find out a more suitable set of features for our model. First, we tried to compute the correlation matrix of all features provided in the data set. We are trying to select four of the highest scores in the matrix, Trip Miles, Fare, Trip Total and Pickup Longitude. It roughly reduces our RMSE score by half. As shown in Figure 8, the prediction is closer to actual Tips.



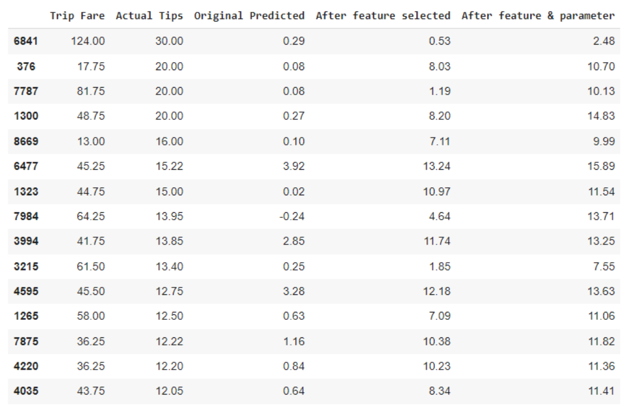
	Fare	Actual Tips	Original Predicted	Correlated features
7678	6.25	50.00	0.10	9.69
7521	24.25	19.15	0.09	6.97
4785	47.00	17.50	2.04	15.08
1323	44.75	15.00	0.17	9.46
5020	43.75	14.78	1.99	14.07
8552	63.50	13.80	-0.15	5.98
6605	45.50	13.75	1.98	7.21
810	41.25	13.68	0.33	8.30
29	61.00	13.10	-0.11	5.90
1938	42.25	12.80	0.22	5.37

Fig. 8. SVR Prediction with features selected by correlation matrix

The problem with correlation matrix is that not all feature columns are able to show the correlation factors, probably because of large amount of missing values or zero entries in the feature. In order to solve the problem and get more accurate prediction result, we came up with using the feature selection functions from the SK-learn libraries. Since we are using regression model, and all data needs to be numerical in order to get regression result, we need to change other types of information into numbers. For time information, we parsed the timestamp value and toke the hour number as time, which we thought it may be a important factor in relevance to tips. For other information, like company name, payment type, we used the LabelEncoder() function to convert it to numbers by numbering them in order of the occurrence. After preprocessing the data, we used the SelectKBest() function from library and set k to 4 to select four best features, and choose to use Mutual Info Regression to derive the feature selection result. Here we got the best four features which are Trip Miles, Fare, Trip Total and Payment Type. By training the SVR model with new features, we got smaller RMSE by comparing to the scores using correlation matrix.

Feature and SVR Parameter Tuned Model

We used the GridSearchCV() function to find out the best parameter value input for SVR() function. By adjusting parameters input set and repeating the GridSearchCV() training, We get a better result with regularization factor C around 20. The new parameters set we got is as following: kernel to 'rbf', degree to 2, gamma to 'scale', coef0 to 0.0, tol to 0.001, C to 20, epsilon to 0.1, shrinking to 'True', cache_size to 200, verbose to 'False', and max_iter to -1.



	Trip Fare	Actual Tips	Original Predicted	After feature selected	After feature & parameter
6841	124.00	30.00	0.29	0.53	2.48
376	17.75	20.00	0.08	8.03	10.70
7787	81.75	20.00	0.08	1.19	10.13
1300	48.75	20.00	0.27	8.20	14.83
8669	13.00	16.00	0.10	7.11	9.99
6477	45.25	15.22	3.92	13.24	15.89
1323	44.75	15.00	0.02	10.97	11.54
7984	64.25	13.95	-0.24	4.64	13.71
3994	41.75	13.85	2.85	11.74	13.25
3215	61.50	13.40	0.25	1.85	7.55
4595	45.50	12.75	3.28	12.18	13.63
1265	58.00	12.50	0.63	7.09	11.06
7875	36.25	12.22	1.16	10.38	11.82
4220	36.25	12.20	0.84	10.23	11.36
4035	43.75	12.05	0.64	8.34	11.41

Fig. 9. SVR Prediction Comparison.

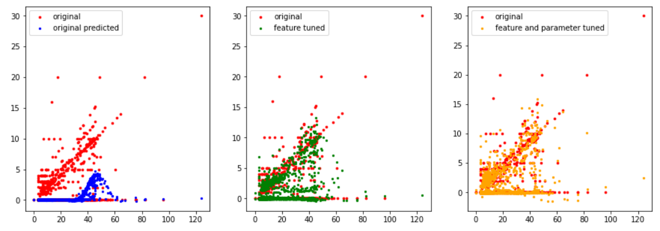


Fig. 10. SVR Prediction Graph Comparison

We gradually refined our model to reduce the RMSE between actual tips and predicted tips value. Figure 9 and 10 shows all the differences between actual tips and predicted tips in our design. It is clear to see the tips predicted from the last one with parameter tuned model are closer to the actual tips.

3) *Random Forest Regression:*

Original Model

Similar to the other two methods, a random subset created by the pre-processing script was separated into am 80% training and a 20% testing set by utilizing Pandas Data Frames. Each section was then filtered such that the independent variables of 'Trip Seconds', 'Trip Miles', and 'Fare' are represented in X, leaving only 'Tips' in Y. Once the data has been processed, an RF Regressor is called and used to fit the data. For this particular regressor, the depth of the decision trees was set to a maximum of 3, to account for all 3 variables, with minimal risk for over fitting the data. For the other variables, they were set to a default as follows: n_estimators to 100, criterion to 'gini', min_samples_split to 2, min_sample_leaf to 1, min_weight_fraction_leaf to 0.0, max_features to 'auto', max_leaf_nodes to "None", min_impurity_decrease to 0.0, bootstrap to "True", oob_score to "False", n_jobs to "None", Random_state to "None", verbose to 0, warm_start to "False", class_weight to "None", ccp_alpha to 0.0, and max_samples to "None".

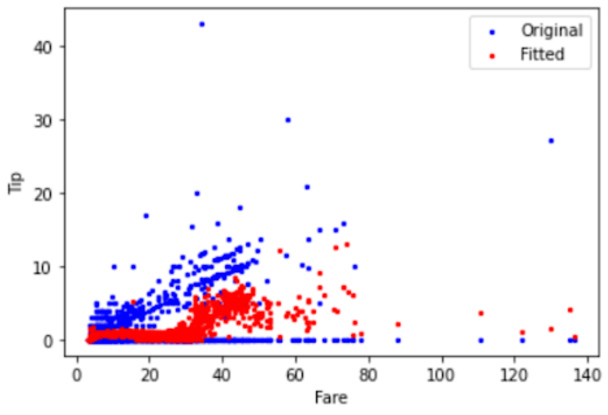


Fig. 11. RFR Prediction vs. True Value.

Classifier Pre-Processing

Since a markedly large percentage of the tips (roughly 72%) are set to 0, a method considered for improving the accuracy of the model was to first label the data by whether or not a tip was offered. By doing so, this could eliminate tips evaluated as $0.00 in the regression which would otherwise skew the predictions. In our implementation however, the accuracy of the combined model were not enough to surpass the original model. This was largely in part due to the high False Negative rate, which predicted many of the trips with a tip as not having one. So despite the many points that were correctly predicted, the error of False Negatives skews the RMSE to a point which is considered worse, even if more points were predicted with 100% accuracy.

IV. Comparison

For the SVR, RF Regression and KNN Regression algorithms described above, we used the RMSE to test the accuracy of our models. This is calculated by taking the square root of the sum of all differences between the true and predicted values squared divided by sample size. We each created an initial model with default parameters and used various different techniques to improve that initial model. In the Support Vector Regression, Tianhao used feature selection and GridSearchCV(). In Random Forest Regression, Jeriel used a Random Forest Classifier to determine if a certain rider is expected to tip. Given that output, he used the Random Forest Regression algorithm to determine the amount of tips the rider is expected to give. In the K Nearest Neighbor Regression algorithm, Jian used feature selection and GridSearchCV(). Jian and Tianhao were able to significantly increase the accuracy of their model while Jeriel's model actually had a decrease in accuracy (his RMSE was 3.54 which was a lot higher than his 2.78 initially). With this in mind, Jian and Tianhao selected their tuned model as their best model while Jeriel used his original model. The RMSE for our final models are as follows: SVR: 0.96, Random Forest: 2.78 and KNN Regression of 0.98. Given these numbers, we can say that the SVR has the best accuracy and would be our final model to predict tips given by Chicago riders.

We are now comparing the computational cost of three models. According to SK-learn library, as we selected 4 best features and choose brute force for KNN algorithm, its computational cost of choosing 7 neighbors is about 28n*log(n), where n is the size of training data. SVR algorithm takes about quadratic number of samples in computational time. Therefore, it is not recommend for running on large data set that is more than 10,000 samples. For Random Forest Regression, we run the classifier on training data with zero entries included, in order to build the tree and run the regression on sub-trees. It roughly takes about 3n*log(n) with 3 features and n samples. By comparing among three algorithms, KNN performs faster than other models, especially when there is a large data set. So in our implementation, SVR is the best model in terms of accuracy and KNN is the best model in terms of computational cost.

In terms of any existing solutions to our problem statement, there has been implementations to predict the tips given by taxi riders but not with our data set. All the implementations to predict tips were using the data sets from the NYC Taxi and Limo Commission. Since our implementations used data from Chicago, many of the columns are different and the data itself could have more errors or noise. Additionally, many of the other implementations used the full data set but in our implementations, we only used subset due to computational limitations of our computers. With this in mind, it is hard say for certain that a certain implementation is better whether that be our implementation or that of another.

V. Future Direction

Given our best and second-best model still yielded a RMSE of around 1, we should for future implementations try train other regression algorithms. For example, some of the other regression algorithms we might consider are Linear Regression, Neural Networks, Ridge Regression and Decision Tree Regression. Additionally, we can increase our training data set size since we only used 1% of the original data set for training. We could include the whole 2021 data set or include prior years and 2021. We would train this increase data set using Microsoft Azure or Amazon Web Services given the computational limitations of our computers. Given the limited time frame for this project, we were not able to implement and create our model using Azure or Amazon Web Services.

VI. Conclusion

Given the length of time allocated to completing the project, we believe that the solutions we have devised are well executed despite the constraints. Our models, over the course of implementation, have seen several iterations, improving by as much as 63% (for the SVR model). Other than higher computing power or more data, we could have tried combining our separate methods of pruning the data, making a model that could be more resilient than the sum of its parts. Otherwise, we could have simply experimented with more algorithms than with just the three we set upon originally. Another alternative would have been to train with all of the features and prune with LDA, though that could only be done with a more powerful computer at our disposal. As an applicable tool for assisting taxi drivers with predicting tips, this model fails on the basis that most of the variables we calculated with are only known at the end of a particular trip, at which point the customer would already be tipping. If nothing else, this model serves as a basis for other jobs within the service industry, which with more time and improvements, can allow for the generalized prediction of tips given initial conditions, and not just in hindsight.

References

[1] Chicago, City of. "Taxi Trips - 2021: City of Chicago: Data Portal." Chicago Data Portal, 10 Nov. 2021, https://data.cityofchicago.org/Transportation/Taxi-Trips-2021/9kgb-ykyt.

[2] Hajjej, Adam. "NYC Taxi&nbsp;: Tip Amount Prediction." Medium, Medium, 15 Apr. 2020, https://medium.com/@adam.hajjej.ah/nyc-taxi-tip-amount-prediction-1bacf9eac920.

[3] Josemazo. "Josemazo/NYC-Taxi-Tip-Predictor: A Case Study for Predicting the Tips in the New York City Taxis." GitHub, https://github.com/josemazo/nyc-taxi-tip-predictor.