Project Report

Project Title: Tip guesser (Team 10)

Team Members: Terry Shih (1113678) , Narendra Babu Ramisetty (1549690)

Section 1 Introduction

Motivation

Our motivation for this project was mainly based how a tip can be predicted taking into consideration different features rather than using the traditional percentage of total (10% of bill amount) amount used in real life. We decided to go with taxi tip prediction as different factors such as pickup and dropoff location, time of the day, trip length and other factors might play a part in the amount one tips. Uber or Lyft usually suggests a tip amount for the trip which is either fixed or a percentage value as discussed previously.

Objective

Our objective is to design a model that would take into consideration the past information of the tips made by different customers for various taxi trips in New York and extract the features that can be important for the model to make a good prediction. This model when integrated into the taxi meters can suggest a tip amount after a trip is finished by considering the features it was trained on and thereby letting a user know both whether and how much to tip a taxi driver.

Section 2 System Design & Implementation details

Algorithms considered

Random Forest: Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. We considered random forest estimators largely because our data contained many nominal features, and research online led us to using random forests because they are designed to handle a mix of data categories.

Gradient Boosting: Boosting is a method of converting weak learners into strong learners. In boosting, each new tree is a fit on a modified version of the original data set. Gradient boosting is a machine learning technique for regression and classification problems, which produces a

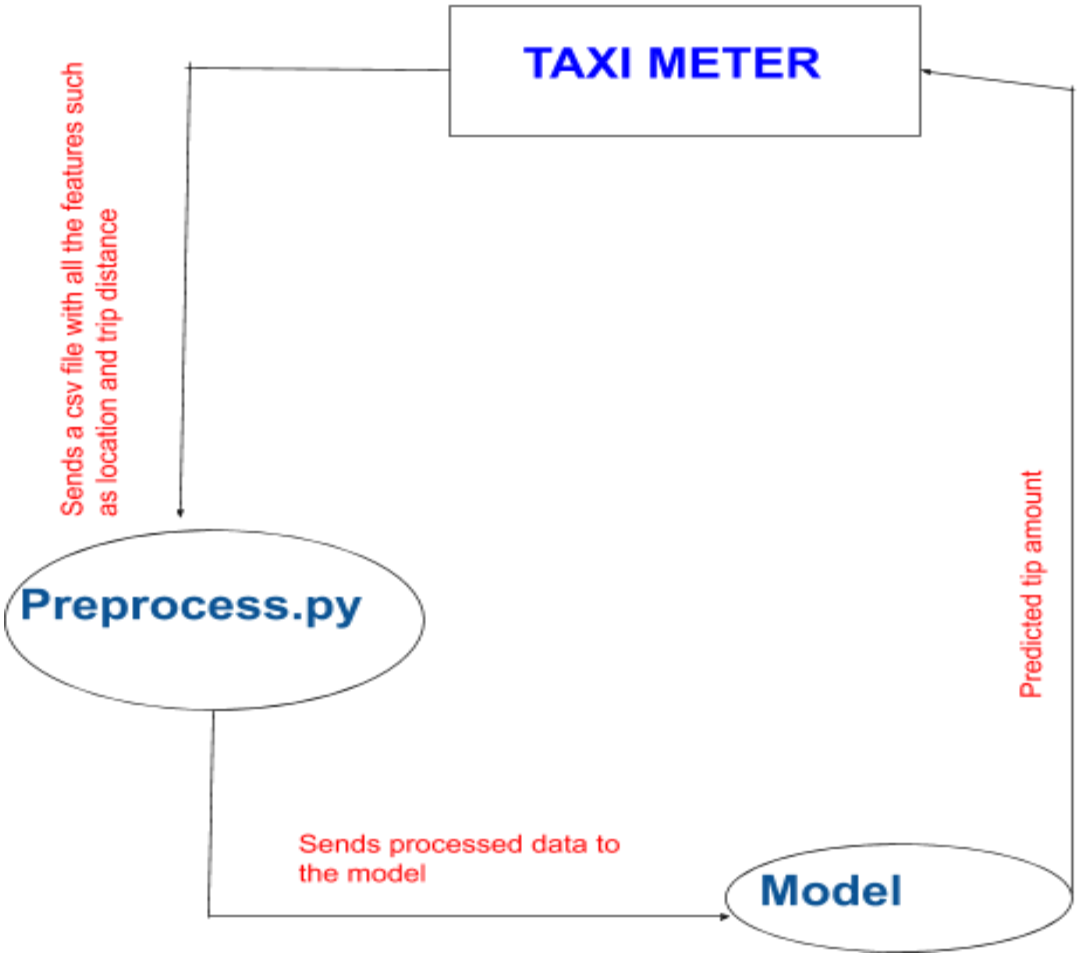
prediction model in the form of an ensemble of weak prediction models, typically decision trees. It builds the model in a stage-wise fashion like other boosting methods do. The main advantage of Gradient Boosting is its optimization of an arbitrary loss function through gradient descent, which allows it to be applied to many cases. Our decision to apply gradient boosting to the problem comes from the fact that the library we used for Random Forest also contained an implementation of Gradient Boosting.

In the end, we decided to try two different approaches to the problem. One approach is to build a classifier / regressor combination model, where the classifier is used to predict whether or not someone will tip, and the regressor is only used to predict how much someone will tip if the classifier says that the person will tip. The other approach is to simply apply a regressor over the training data using the given tip amount as the target variable. For both cases, we tested using both Random Forest and Gradient Boosting.

Technologies & tools used

Python: An interpreted high-level programming language. Contains libraries useful for classification and regression tasks.
HPC server: High Performance Computing server to work and process large data sets.
H2o.ai: Main library used for their implementation of Random forest and Gradient Boosting Machine. Used because of the presence of nominal features in our dataset led us to explore Random forests. One article led us to h2o.ai which seemed like the perfect fit for our problem.

Data flow



Dataflow diagram showing the exchange of data behind the scene

Use cases / GUI

The model would be integrated into taxi meters and when a taxi meter ends (the ride is finished), the meter will generate a csv file with other fields such as time, traffic conditions and send it to a server for processing.
The preprocessing file removes any unnecessary fields or undesired inputs from the csv and sends the processed data to the model, which then predicts the tip amount and sends it to the taxi meter to be displayed to the customer.

This can also be further developed to be used with any application (web or mobile) by running the model on the backend and communicating through the internet, and by estimating fare costs and other parameters through things like Google Maps.

Section 3 Experiments

Dataset used

For this project, we used taxi data provided by New York City. Two different datasets namely Yellow taxi and Green taxi datasets are available for use. We decided to go ahead with the Green Taxi dataset as it has an extra column (trip_type) which is extra information that can be used to better train the model. The dataset can be downloaded from the NYC official website and is separated by month. We trained and tested our model on the data for March 2019, which contained about 600k entries. The features are a mixture of continuous numbers and categorical data.

Different columns present in the dataset and their brief descriptions are as follows:

- VendorID: indicates the provider of the record.
- Lpep_pickup_datetime/lpep_dropoff_datetime: Pickup and dropoff time that meter was engaged and disengaged respectively.
- Passenger_count: Driver entered value for the number of passengers in that vehicle.
- Trip_distance: Elapsed trip distance in miles.
- PULocationID and DOLocationID: Zone ID where the taximeter is engaged and disengaged.
- RateCodeID: Ratecode effective at the end of the trip. Such as Groupride, Negotiated price and so on.
- Store_and_fwd_flag: Represents if the trip data is stored and sent to the server at some time because the vehicle did not have an active connection with the server.
- Payment_type: Indicates the type of payment made by the passenger. Such as cash, credit card etc.
- Fare_amount: Fare calculated by the meter.
- Extra: Miscellaneous charges.
- Mta_tax: Tax amount that is triggered based on metered rate in use.
- Improvement_surcharge: Improvement surcharge assessed on hailed trips.
- Tip_amount: Includes the credit card tips paid by the passenger.
- Tolls_amount: Total amount of all tolls paid in trip.
- Total_amount: Total amount charged to passengers.
- Trip_type: Specifies if it is a street-hail or dispatch.

For preprocessing, we first filtered out the dataset to only include the fares that were paid with a credit card, as the dataset only kept tip_amount for rides which were paid in card. From there, to reduce the size of the dataset and remove features that were unrelated to the passenger, we

dropped the VendorID and store_and_fwd_flag features. In the dataset, some of the surcharge fields like mta_tax and improvement_surcharge had negative values, so we sifted through the set and set the negative values to NaN under the assumption that the negative values were errors.

To make the DateTime stored in the dataset usable for a model, we translated the pickup and dropoff times into Unix timestamps. We also subtracted New Year's Day's timestamp from each of the translated timestamps so that our model could work on any year, not just the year the training dataset is set in. We also added in a new column that calculates the duration of the trip in seconds based on the calculated timestamps and one more column that acted as a flag for whether a given sample tipped or not based on the value of tip_amount.

Methodology followed

As was mentioned previously, we used the dataset for March of 2019 to train and test our model. We split the data set 90% for training, and 10% for testing out the prediction pipeline once a model was chosen. We then split the 90% further into training, validation, and testing for use while generating the model. After the splits, we trained our model on roughly 300K data points. We are training our model with Using h2o.ai's dataframe allowed us to mark which columns were categorical in nature (e.g. PULocationID and DOLocationID).

One great aspect of h2o.ai's implementation of the Random Forest Classifier/Regressor is that it has a built-in feature to stop generating more trees if a new tree fails to improve the training metric, so the proper number of trees to generate is taken care of by the algorithm. So, we generate three different random forests: one for classifying the samples by whether the sample is likely to tip, one for predicting the tip amount using only the samples that did tip, and one for predicting the tip amount using all of the samples. We also generate three similar models using Gradient Boosting, utilizing h2o.ai's hyperparameter grid searching to find what the best values for the hyperparameters. For the classifier and the all samples predictor, max depth of twenty with 30 trees generated the highest f1-score for the classifier and r2 score for the predictor. Forty trees with a max depth of 20 generated the best r2 for the tipping-samples predictor. After generating all 6 models, we then combined the models as described previously. We use the all-samples predictors by themselves, but for the classifier / predictor combination, we tested the dual-layered model for the four different combinations of using Random Forest and Gradient Boost Machine for the classifier and the predictor.

Comparison charts

	tip_amount	RF_RF	RF_GBM	GBM_RF	GBM_GBM	RF	GBM
0	0.00	0.00	0.00	0.00	0.00	0.02	0.57
1	3.69	3.42	3.88	3.42	3.88	2.97	3.57
2	0.00	0.00	0.00	0.00	0.00	0.24	0.06
3	1.70	1.65	1.74	1.65	1.74	1.63	1.69
4	2.16	2.09	2.18	2.09	2.18	2.15	2.14
5	0.00	0.00	0.00	0.00	0.00	0.41	0.06
6	2.00	2.10	2.03	2.10	2.03	2.24	1.99
7	2.82	2.48	2.75	2.48	2.75	2.25	2.75
8	0.00	1.03	0.16	0.00	0.00	0.48	0.07
9	2.15	2.17	2.18	2.17	2.18	2.18	2.13
10	0.96	0.99	1.04	0.99	1.04	0.95	0.99
11	0.00	0.68	0.19	0.00	0.00	0.58	0.86
12	1.56	1.64	1.61	1.64	1.61	1.58	1.56
13	30.74	13.46	16.37	13.46	16.37	18.73	18.62
14	0.00	0.00	0.00	0.00	0.00	0.31	0.06
15	3.26	3.16	3.23	3.16	3.23	2.55	3.16
16	1.36	1.42	1.42	1.42	1.42	1.45	1.37
17	4.51	4.56	4.45	4.56	4.45	4.68	4.42
18	0.00	0.00	0.00	0.00	0.00	0.15	0.06
19	1.66	1.68	1.70	1.68	1.70	1.60	1.65

Sample predictions for various combinations of Random forest and Gradient Boosting algorithm vs the original tip amount

Metric	RF_RF ClassReg	RF_GBM ClassReg	GBM_RF ClassReg	GBM_GBM ClassReg	RF Regress	GBM Regress
R2	0.82054	0.84991	0.83721	0.86655	0.84116	0.89476
MSE	0.78785	0.65891	0.71467	0.58585	0.69735	0.46202
MAE	0.23912	0.17933	0.21509	0.15801	0.29266	0.19436

Evaluation metrics for the algorithms used.

Analysis of results

- The chart displayed above show a sample of the different predicted values of each of the six final potential models, which are (RF = Random Forest, GBM = Gradient Boosting Machine) RF classifier/RF predictor, RF classifier/GBM predictor, GBM classifier/RF predictor, GBM classifier/GBM predictor, RF predictor, GBM predictor. As can be seen in the chart comparing the model's different metrics, the predictors that were fitted over the whole dataset tended to outperform the classifier/regressor combination model. This makes sense as the solo predictors will very, very rarely predict that someone will not tip. A solo predictor will still suggest a couple of cents at the minimum, whereas the combination model will always suggest 0$ for any sample that it predicts will not tip. Another interesting observation is that Gradient Boosting outperforms Random Forest. This can largely be attributed to the capabilities of Gradient Boosting exceeding Random Forest. The RF models storage wise cap out far before the GB models do

Section 4 Discussion & conclusion

Decisions made

- Dropping of columns that would not contribute much for the prediction to speed up the training of the model.
- Sticking to training with just one month to also speed up training

Difficulties faced

- Pickup and dropoff locations are mentioned as Zone IDs in the dataset. It was an issue to map the google maps to produce those Zone IDs for prototype.
- H2o.ai refused to cooperate with the HPC on our end for a solid couple hours.
- Lack of experience with h2o in comparison to sklearn and pandas.

Things that worked

- H2o's ability to handle categorical data

Things that didn't work well

- A working GUI (website) prototype to generate csv files from user inputs as Zone IDs are taxi meter specific and the only solution is to put in fake/ random zone IDs.

Conclusion

- Given the metrics, we decided that the pure GBM predictor would be best suited for the task, though the double GBM predictor is not that far behind, especially with its ability to predict a tip amount of 0.
- Given the opportunity to continue working on this project, analyzing the data over a whole year seems like the next logical step. We could also take steps towards gathering data on cash tips or hoping that NYC comes up with a method to collect that information, which would expand how much data our models could fit over.

Section 5 Project Plan / Task Distribution

1. Project Idea and Dataset search: Assigned and did by Narendra Babu Ramisetty and Terry Shih
2. Data preprocessing: Assigned to Narendra Babu Ramisetty and performed by Terry Shih and Narendra Babu Ramisetty
3. Front end prototype model: Assigned and did by Narendra Babu Ramisetty
4. Training the model: Assigned and did by Terry Shih
5. Report and slides: Assigned and did by Terry Shih and Narendra Babu Ramisetty