New York Taxi Dataset

Leo Marie Stephen

stephen.leo87@gmail.com

Github: https://github.com/stephenleo87/nyc-taxi

Objective

- Use Google Big Query New York Taxi dataset to gain insights

Data Set structure

- Google Big Query New York Taxi data set has several tables
- Each table is 1 year worth of data
- Each table contains information about individual trips and contains information like pickup/dropoff time, location, distance, fare, etc
- Each row is one trip.
- Inspecting the table, there are both independent and dependent variables as below.

List of Tables

- tlc_yellow_trips_2009
- tlc_yellow_trips_2010
- tlc_yellow_trips_2011
- tlc_yellow_trips_2012
- tlc_yellow_trips_2013
- tlc_yellow_trips_2014
- tlc_yellow_trips_2015
- tlc_yellow_trips_2016

Independent Variables	Dependent Variables
vendor_id	
pickup_datetime	dropoff_datetime
passenger_count	
pickup_longitude	dropoff_longitude
pickup_latitude	dropoff_latitude
rate_code	
store_and_fwd_flag	
payment_type	
	trip_distance
	fare_amount
	extra
	mta_tax
	tip_amount
	tolls_amount
	imp_surcharge
	total_amount

Approach

- To build prediction models for the dependent variables
- Try 1 regression problem and 1 classification problem
- For Regression: Attempt to predict the fare_amount
- For Classification: Attempt to predict tip percentage is high or low, drop off neighborhood

Strategy

- Since data tables are split into different years, use 2015 data as training data set and 2016 data as testing data set
- Query random 100K rows from each table
- Jupyter Notebook: [01_SQL Query.ipynb]

In [2]:
```
# Create and run a SQL query for the taxi_data from 2015 and 2016
query = bq.Query('(SELECT * FROM `bigquery-public-data.new_york.tlc_yellow_trips_2015` LIMIT 100000) union all (SELECT * FROM `bigquery-public-data.new_york.tlc_yellow_trips_2016` LIMIT 100000)')
output_options = bq.QueryOutput.table(use_cache=False)
result = query.execute(output_options=output_options).result()
```

In [3]:
```
# Convert to DataFrame
df = result.to_dataframe()
df.head()
```

Out[3]:

	vendor_id	pickup_datetime	dropoff_datetime	passenger_count	trip_distance	pickup_longitude	pickup_latitude	rate_
0	2	2015-07-18 11:25:58	2015-07-18 11:43:47	1	7.21	-73.862762	40.769028	1.0
1	1	2015-03-15 12:50:01	2015-03-15 13:23:35	1	10.80	-73.870926	40.773727	NaN
2	2	2015-04-30 12:25:44	2015-04-30 13:03:51	1	4.28	-73.978180	40.762341	NaN
3	2	2015-05-28 08:47:56	2015-05-28 09:26:08	1	18.47	-73.776711	40.645302	NaN
4	1	2015-06-20 19:36:17	2015-06-20 20:10:49	1	15.50	-73.777054	40.644947	NaN

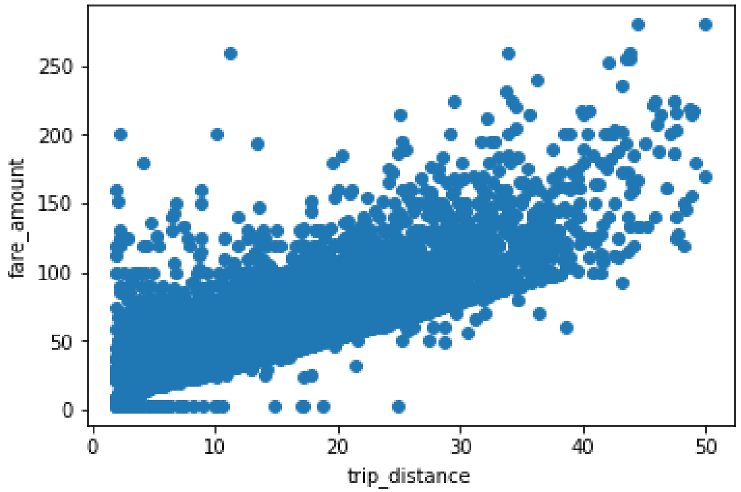
In [4]:
```
# Write to csv
df.to_csv('data.csv')
```

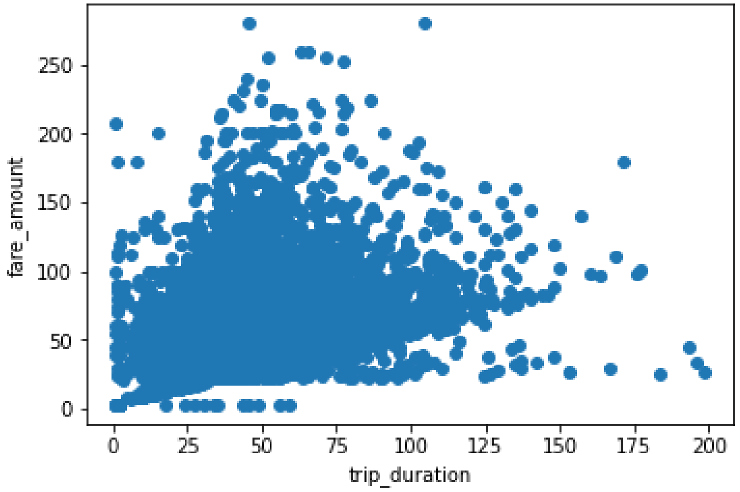
Fare Prediction

Jupyter Notebook: [02_Fare Prediction.ipynb]

Intuition

- Fare would correlate to trip distance and trip duration.
- After some data cleanup, just plotting fare_amount vs trip_distance and calculated trip_duration, we do see this relationship.





Intuition

- Build a simple bivariate linear model to check the predictive power.
- As mentioned, model is built on 2015 data and tested on 2016 data.
- We use RMSE on training and testing set to check the model's predictive power.
- This simple intuitive model can achieve an RMSE of $5.5 on testing data set

```
# Simple bivariate model
lm = LinearRegression().fit(X_train,y_train)
model_results(X_train, y_train, X_test, y_test, lm)
```

```
----Training Data results (2015 data set)----
RMSE: $5.6
R2: 0.79

----Test Data results (2016 data set)----
RMSE: $5.5
R2: 0.81
```

Feature Engineering

- Thinking deeper about the problem
  - Pickup day of the week and hour of the day could play an important role in demand and traffic conditions leading to an impact on fare amount
  - Latitude difference and Longitude difference could be an additional data to augment trip_distance's impact on fare
  - Pickup and dropoff neighbourhoods could have an effect on fare due to additional fees for specific locations?
- Add in features as below
  - Pickup month
  - Pickup day of the week
  - Pickup hour of the day
  - Lattitude difference
  - Longitude difference
  - Geohashed pickup location (idea from internet on how to discretize location)
  - Geohashed dropoff location (idea from internet on how to discretize location)

Feature Engineering

- Through trial and error found that adding dropoff_geohash is leading to overfit so drop this from the features list.
- Lasso confirmed that coefficients for dropoff_geohash were zero as well.
- Multivariate linear model has improved the RMSE to $5.1
- Since the Training Data still has high RMSE, the linear model is suffering from high Bias
- Try Boosting models to improve the Bias

```
# Multivariate Linear Model
lm = LinearRegression().fit(X_train,y_train)
model_results(X_train, y_train, X_test, y_test, lm)
```

```
----Training Data results (2015 data set)----
RMSE: $5.3
R2: 0.81

----Test Data results (2016 data set)----
RMSE: $5.1
R2: 0.83
```

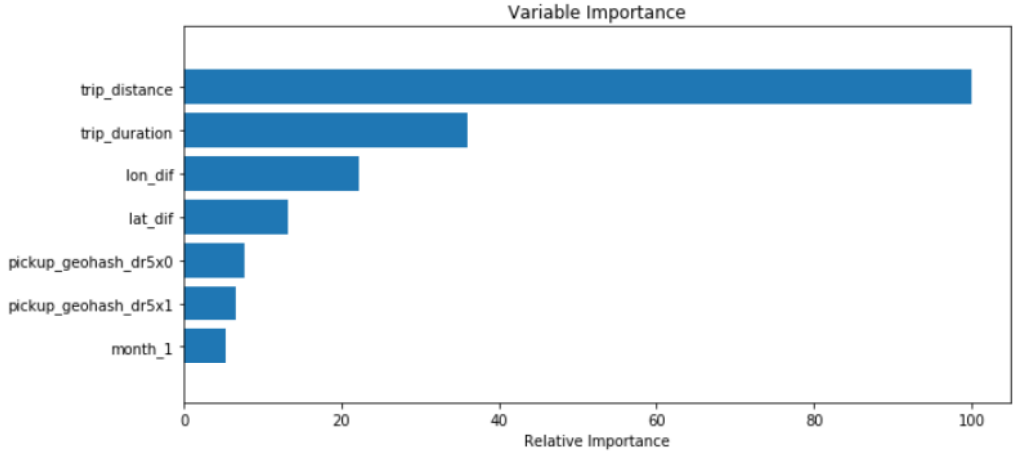
Gradient Boosting

- Trying gradient Boosting to reduce the Bias
- Boosting model has significantly improved the RMSE to $4
- Variable Importance shows that trip_distance and trip_duration are still the top features to predict fare amount and confirms our initial intuition.
- Try Data Augmentation to drive RMSE lower

```
# Gradient boosting regressor to reduce bias
params = {'n_estimators': 500, 'max_depth': 4, 'min_samples_split': 2, 'learning_rate': 0.01, 'loss': 'ls'
}
gbr = ensemble.GradientBoostingRegressor(**params)
gbr.fit(X_train, y_train)
model_results(X_train, y_train, X_test, y_test, gbr)
```

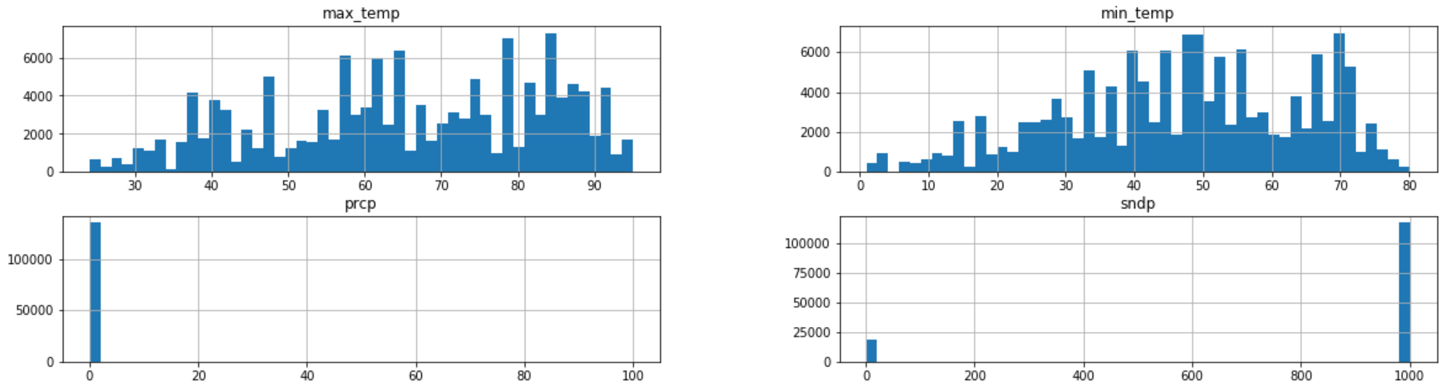
```
----Training Data results (2015 data set)----
RMSE: $3.9
R2: 0.90

----Test Data results (2016 data set)----
RMSE: $4.0
R2: 0.89
```



Weather Data Augmentation

- Does weather play a role in taxi fare?
- Big Query has weather data
- Choose 1 station within New York to augment weather data such as min/max temp, rain and snow to original dataset
- Running the same GBR model did not improve the RMSE further.




```
# Gradient boosting regressor
params = {'n_estimators': 500, 'max_depth': 4, 'min_samples_split': 2, 'learning_rate': 0.01, 'loss': 'ls'
}
gbr = ensemble.GradientBoostingRegressor(**params)
gbr.fit(X_train, y_train)

model_results(X_train, y_train, X_test, y_test, gbr)
```

```
----Training Data results (2015 data set)----
RMSE: $3.9
R2: 0.90

----Test Data results (2016 data set)----
RMSE: $4.0
R2: 0.89
```

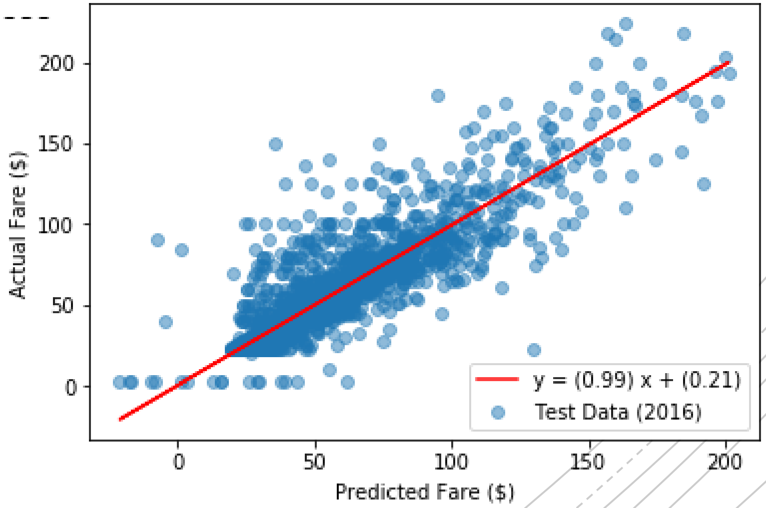
GBR Hyperparameter tuning

- Try tuning hyperparameters to reduce RMSE
- Best parameters reduced the testing data RMSE to $3.5
- Strong correlation seen between Actual Fare and Predicted Fare as expected from the RMSE and R-square

```
# Best parameters from Hyperparameter tuning
params = {
    'learning_rate':0.1,
    'min_samples_split':10,
    'min_samples_leaf':10,
    'max_depth':4,
    'n_estimators':1000}
gbr_tuned = ensemble.GradientBoostingRegressor(**params)
gbr_tuned.fit(X_train, y_train)
model_results(X_train, y_train, X_test, y_test, gbr_tuned)
```

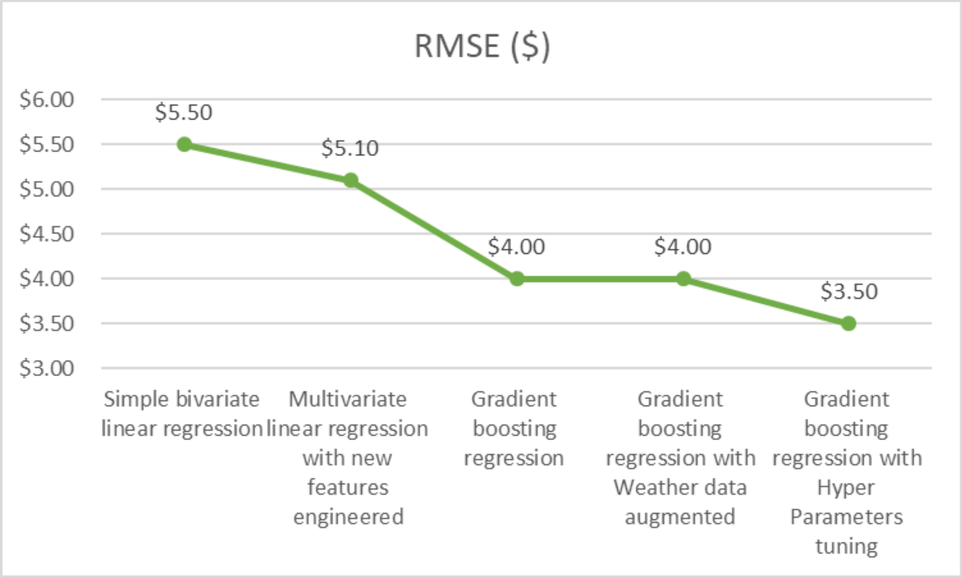
```
----Training Data results (2015 data set)----
RMSE: $2.4
R2: 0.96

----Test Data results (2016 data set)----
RMSE: $3.5
R2: 0.92
```



Conclusion

- Various methods were tried out to predict the fare amount of individual trips
- Data from 2015 was used to build the models and data from 2016 was used to test the model performance
- RMSE (root mean square error) was chosen as a metric to evaluate model performance as it can directly give us the error in dollars
- A gradient boosting regressor with hyperparameter tuning provided the best result of RMSE = $ 3.5 for testing data
- The gradient boosting regressor RMSE is $ 2 better than a simple intuitive modeling approach

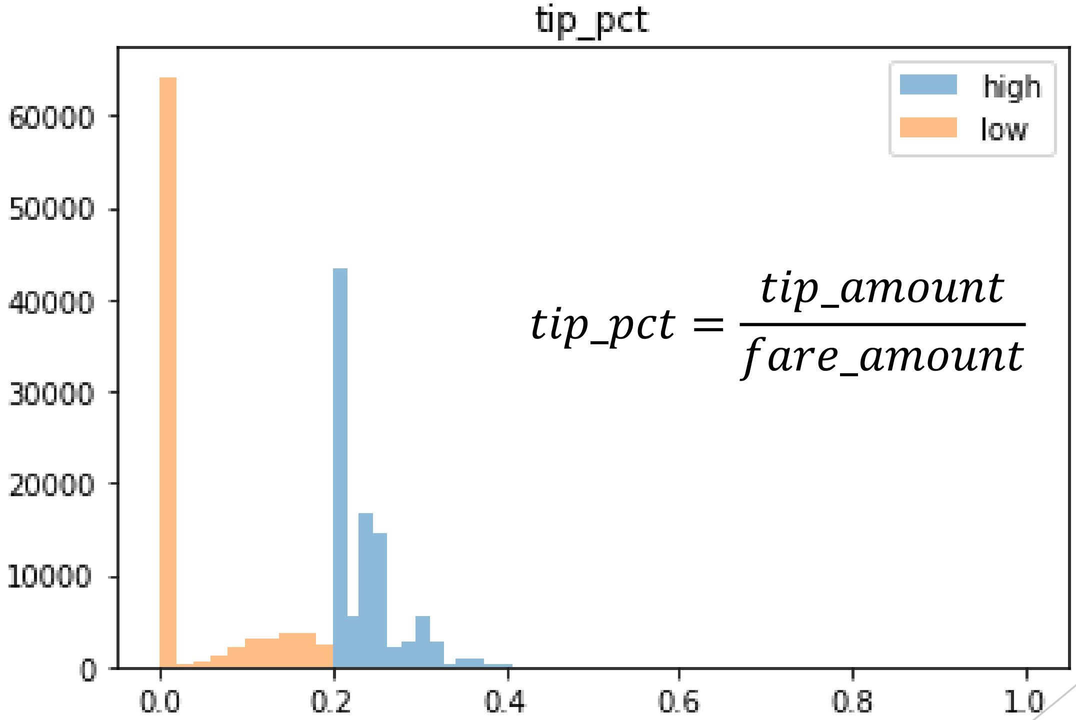


Tip Class Prediction

Jupyter Notebook: 03_Tip Prediction.ipynb

Intuition

- Intuitively, the absolute value of tip_pct maybe very hard to predict without details about the individual's financial status
- However, from the tip_pct distribution, it looks like majority of the tip_pct are clustered around 2 values (0 and 0.2).
- Could it be possible to predict whether a trip can result in low tip_pct of high_tip percentage?



$$tip\_pct = \frac{tip\_amount}{fare\_amount}$$

Feature Engineering

- We use back all the features created during the Fare Prediction exercise and try out multiple classification models.
- The best model without any hyperparameter tuning is a Gradient Boosting model but with only 60% accuracy in testing data set

Model	Training Data		Testing Data	
	Accuracy	AUC	Accuracy	AUC
Logistic Regression	0.51	0.5	0.57	0.5
SVC	0.59	0.59	0.56	0.58
Naïve Bayes	0.51	0.51	0.57	0.51
KNN	0.71	0.71	0.55	0.55
Decision Tree	1	1	0.54	0.54
Random Forest	0.59	0.59	0.57	0.58
Gradient Boosting	0.65	0.65	0.61	0.6
Neural Net	0.59	0.58	0.59	0.58

Weather Data Augmentation

- Could it be possible that weather plays a role in making passengers more generous to tip higher?
- Did not observe any improvement in Accuracy

Model	Training Data		Testing Data	
	Accuracy	AUC	Accuracy	AUC
Logistic Regression	0.51	0.51	0.56	0.5
SVC	0.59	0.59	0.55	0.57
Naïve Bayes	0.54	0.54	0.59	0.54
KNN	0.7	0.7	0.53	0.53
Decision Tree	1	1	0.54	0.54
Random Forest	0.59	0.59	0.57	0.58
Gradient Boosting	0.65	0.65	0.61	0.6
Neural Net	0.58	0.58	0.58	0.58

Conclusion

- All models have low predictive power with the best model only having 60% accuracy
- Possibly because the data set does not contain important information like passenger financial information that could determine whether a passenger tips a high percentage or not.

Destination Prediction

Jupyter Notebook: 04_Destination Prediction.ipynb

Intuition

- Attempting to predict the destination location (geohashed dropoff latitude, longitude) from the available data such as pickup time, pickup location
- Intuitively, the dropoff location could be random
- However, could it be possible that at a given day of the week, hour of the day, pickup location and number of passengers; there is a correlation to the dropoff location?

Feature Engineering

- We use back all the features created during the Fare Prediction exercise and try out multiple classification models.
- The best model without any hyperparameter tuning is a Neural Net model but with only 30% accuracy in testing data set

Model	Training Data Accuracy	Testing Data Accuracy
Decision Tree	0.57	0.21
Random Forest	0.56	0.22
Gradient Boosting	0.13	0.13
Neural Net	0.32	0.3

Conclusion

- All models have low predictive power with the best model only having ~30% accuracy in Test data
- Abandon this approach.

Final Conclusions

Conclusions

- There are 4 Jupyter Notebooks in this repo
- [01_SQL Query.ipynb]
  - All SQL queries are run from Google Cloud's Datalab platform and the data is stored as csv files for use by subsequent notebooks
  - This notebook is included for reference purposes only
- [02_Fare Prediction.ipynb]
  - Attempting to predict the fare amount from the available data such as trip distance, pickup locations, etc
  - TL;DR: Best prediction model achieved RMSE of $3.5 which $2 better than an simple intuitive model.
- [03_Tip Prediction.ipynb]
  - Attempting to predict the tip percentage (tip_amount/fare_amount) from the available data such as trip distance, pickup locations, etc
  - TL;DR: Best prediction model could only achieve accuracy of 60% indicating the available dataset is not sufficient to accurately predict tip percentage
- [04_Destination Prediction.ipynb]
  - Attempting to predict the destination location from the available data such as pickup time, pickup location
  - TL;DR: No good prediction model could be found indicating the available dataset is not sufficient to predict dropoff location

Thank You