International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 03 Issue: 08 | August -2019 ISSN: 2590-1892

Implementation of Machine Learning Tools for Taxi Fare Prediction Using Random Forest Algorithm

1. Anusha JM, 2.Chandramouli S, 3.Manasa K C, 4.Manukumar D, 5.Manjunath R Yadawad
Co Author: MAHENDRA KUMAR B

1,2,3,4,5 PG SCHOLORS,DEPT. OF MCA,DSCE
CA-ASST.PROF., DEPT. OF MCA, DSCE

Abstract – This mini project developed under a group of team with the guidance of experts which predicts the fare of taxi. The main theme of our project is to predict the fare of number taxi rides that given brief description about the pickups and drop offs location Latitudes and longitudes, time and date of pickups and number of passengers travelling.
Keywords: Deep learning, Classification, Stacking classifier, Machine learning.

1. INTRODUCTION

Taxi fare problem has been described in many places much better than I am able to, so I'll present a summary: if provided with the start and end points of taxi rides in New York, the date of the ride and the final amount paid, plus several other pieces of data, can we predict the taxi fare for a proposed ride? In more formal terms: this is a typical prediction model usually resolved with a regression model approach.

2. THE DATA

As almost everybody in the BD and AI business knows, one of the most important steps in getting value from the data is to prepare it for analysis. This is particularly important for ML algorithms because we are training our model and we want the data to show an accurate picture of what we have or else it will produce wrong results.

I've obtained a publicly available data set in CSV format and imported it into my laboratory database in two sets: one of two million rows and another with the whole 54 million rows.

	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count	year	month	day	dayOfWeek	hour	ti
0	4.5	-73.844311	40.721319	-73.841610	40.712278	1	1	6	15	2	17	
1	16.9	-74.016048	40.711303	-73.979268	40.782004	1	2	1	5	3	16	
2	5.7	-73.982738	40.761270	-73.991242	40.750562	2	3	8	18	5	0	
3	7.7	-73.987130	40.733143	-73.991567	40.758092	1	4	4	21	7	4	
4	5.3	-73.968085	40.768008	-73.956655	40.783762	1	2	3	9	3	7	

Table-1 CSV Data file with data points format

3. DATA PREPARATION

As I mentioned above, most AI experts say that preparing The data for our model is the key to obtaining the best results. Because the data reflects the world as the model sees it. In our case, if we are analyzing taxi trips in New York City, why would the data include coordinates for Madrid? Or coordinates that make no sense such as longitude values outside of the -180º/180º range?

What comes in handy here is the Oracle Data Miner SQL Developer extension, or more precisely, the "Explore Data" component. This component analyzes the data and provides some basic statistics about each of the columns such as min, max, average or median. This is very useful to spot data issues such as negative fare amounts or coordinates that do not fall in the New York City area. There are also tools to chart the data, extract features and, of course, to create models, among others.

The person or team who knows what the data brings is also very important because they add the most value. While the taxi fare problem is very generic, for more specific business data such as pharmaceutical molecules or tuna migration routes, having someone who really knows the business and understands the data is critical. It is critical for the DBA (or data scientist) to work with this person or team in order to obtain the maximum value out of the model and tune it accordingly.

WHAT TO LOOK FOR?

Back to our taxi rides; we have dates, coordinates, fares and the number of passengers per ride.

New York City coordinates are 40ºN 74ºW, so in order to keep things simple, I've assigned a range of 35º to 45º for latitude and -80º to -70º for longitude as valid data ranges. Every row that refers to coordinates outside of these boundaries has been deleted.

The other easy one is the fare amount. Negative fare? No way and, now that I think of it, a negative number of

passengers? Not a good idea either, so all these rows have been deleted as well.

It is interesting to mention that a person who knows the business may not think of these cases simply because they make no sense, but for those of us who "Love your data", it is obvious that data can have errors, either human or machine induced, and it is our job to take care of them.

Finally, the pickup date and time is an interesting piece of data. In order to improve the model, we can make sure that we know exactly what time range we are talking here so we don't hit any statistical oddities, such as a very long taxi drivers strike or a natural phenomenon that may have altered the behavior of the NYC citizens for a couple of months. And also, we don't want to take into account things like the taxi rides of the early 1930's, so we may want to make sure that we are working only with data that makes sense in time. Due to the process I used to load the public data into my sandbox Oracle database, all the dates that were not actual dates have been automatically removed, which is a necessary data cleansing task. I also made sure that the rides occurred in the last decade, so we should be fine here.

So far, we've done only data cleansing; that is, making sure that the data we have makes sense for our particular business case and that it reflects the world as we want our model to see it.

4. DATA TRANSFORMATIONS

While it is cool to have the tiniest details of a given taxi ride, this is the XXI century and we collect Exabyte's of data every second, it is not so useful to train a model.

A simple example. We have the pickup time down to the minute but does it really matter if a taxi ride started at 2:00 PM or at 2:10 PM? The answer is "it depends." If we are looking for some very detailed analysis of the traffic flows to cross-reference with traffic jams, red lights and accidents, then yes, it is relevant. But for our current needs, no way. Even more, we can fall in the pit of over fitting our model. So in this particular case, I've opted for scaling up the pickup time to weekday and instead of using a date data type, I transformed this to a three-character string because ML algorithms seem to prefer data in strings, or categorized, rather than in continuous values, or non-categorized. This is not my idea but rather a standard way to transform date data for easy categorization by the model.

Next is the coordinate's data. Again, it does not matter much a few arch seconds up or down in the map, but the

General geographical area within the city is still important. This is mainly because of routes with more bridges, which means tolls, routes that start or end at an airport, routes with fixed fares and that kind of stuff. Here again, avoiding the detailed coordinates information both reduces the chances of over fitting the model and improves the accuracy based on neighborhoods that may include special cases. There is, of course, a data transformation technique to deal with these cases. The idea, again, is to get rid of the details in the pickup and drop-off coordinates by grouping them into, in this case, artificially created city areas or squares. This creates a matrix and assigns a code to all the coordinate pairs that pertain to a given square, hence categorizing our pickup and drop-off points.

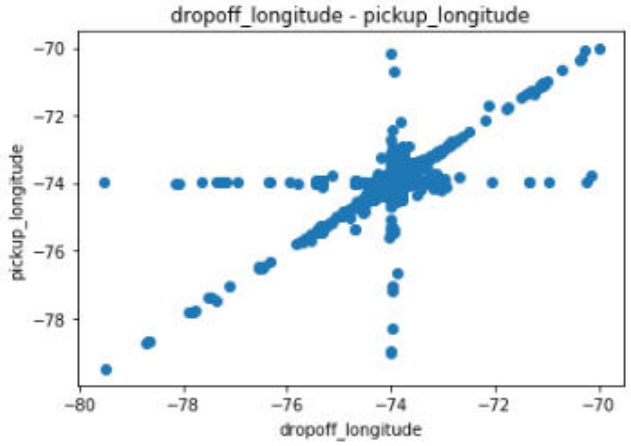
Finally, I added a new column to the data, a new feature, to store the geographical distance between the pickup and drop-off points. This piece of information is important in our model because there are relatively short rides with quite high fares which go against the principle of longer distance and time equal a higher fare. These cases appear due to the fixed fares to and from the airports, so the distance, even the simplest linear calculation used here, is a relevant feature for our model. This again is achieved with an Oracle option called Spatial and Graph. This option includes the SDO GEOMETRY type which allows these kinds of calculations to be simple to code in SQL while remaining accurate.

As a last step for the data preparation, I added a column called CASEID.

In the beginning, there was raw data in our database. For this experiment, I've loaded it from a public source but in a real-world scenario, it can be data that is already in your DWH or DSS or even your production OLTP database.

Once the raw data was analyzed in statistical and more importantly, business logic terms, several cleansing and transforming actions were deemed necessary and executed on it. Out of these activities came the features we will be using in our model.

After cleansing and transforming the data and storing it into a single table, we are ready to create our model.



5. MODEL CREATION

Splitting and validation:

Divide the data into training and test set. Train set contains 80% of the data. Test set contains 20% of the data**.**

Create an object, Train the model using the training sets and Make predictions using the testing set.

```
#Split train set into test and train subsets
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.20,random_state=0)
```

	Actual	Predicted
678229	10.00	10.306121
1611561	24.50	12.737027
731459	6.50	11.604189
174668	20.33	12.814102
1635387	9.70	9.228704
1737463	10.90	10.317578
1464178	5.70	8.939896
760493	7.50	9.360846
590829	8.90	9.156174
1230822	13.00	12.972972

Table-2 Actual and predicted data

Fitting Simple Linear Regression to the Training set:

low values of Scores are a measure of R-squared values and showing that the data points in both train and test data sets are far away from the regression line and hence less percentage only ~17% variance of dependent variable is being accounted by the regression line.

Measuring Error to evaluate model performance on test data set:

Only a slightly low RMSE value is obtained on 5-fold cross validation, 10-fold gave the same results to doing only with 5-folds, to keep efficiency of run-time.

Fitting Decision Tree Regression to the dataset:

Over fitting here, here, on test set 72.5% variance is accounted by its predicted values, while it is showing 100% variance on train observed value which means it is fitting completely on train but fail to generalise the test data.

RMSE for train and test errors difference is large thereby introducing some variance in model causing Over fitting on test data.

It looks like pickup and drop-off locations are highly significant features for predicting fair.

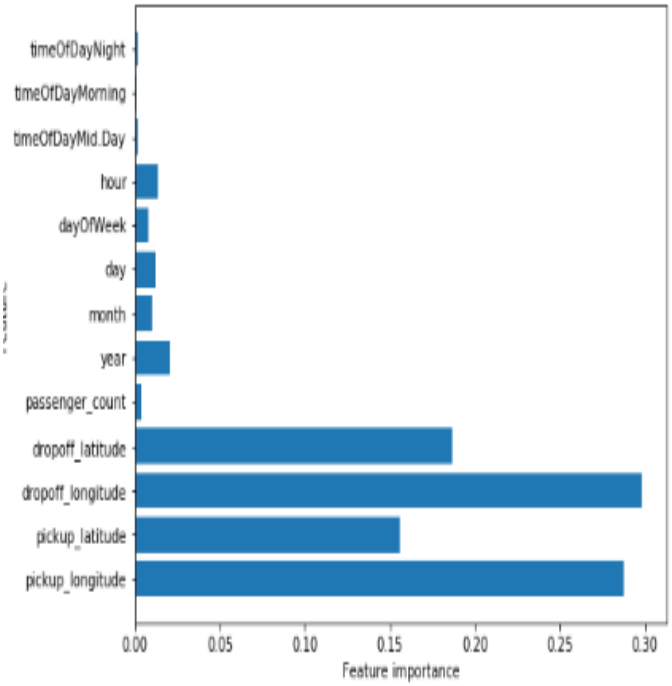


Fig -1: linear regression (Charts)

Fitting Random Forest Regression to the dataset:

from feature importance plot we can see that fairs are mostly affected by the pickup and drop-off locations, so they are highly significant features and very less by the "time of day", we got very low importance values and also considerably by "year" ,"day", "day of week", "month" features.

Tried using Support vector regression on this is data, but the disadvantage for using SVR is that it is only used for data sets containing less than 1000 observations, above that it takes a lot of time or say infinite time which is not efficient way of modeling.

Random forest and decision tree also took long to give output when putted on K-fold Cross validation.

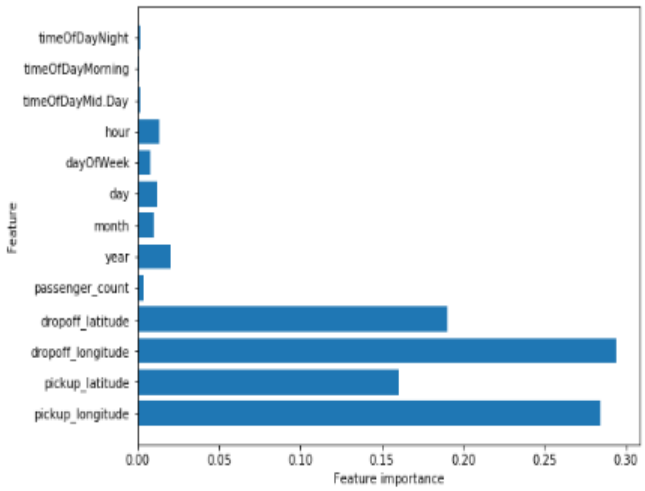


Fig -2: Random forest (Charts)

6. CONCLUSIONS

Finally our Model is able to predict on test data with ~2% Root Mean Squared error on test data, which is quite good.

REFERENCES

1. https://www.researchgate.net
2. https://github.com/Lasagne/Lasagne/wiki/Lasagne-Citation
3. https://www.tripadvisor.com
4. Y. Bengio. Learning deep architectures for AI. Foundations and trends in Machine Learning, 2(1):1–127, 2009
5. https://books.google.co.in