Cab Fare Prediction
Case Study

Asheesh Aggarwal

Contents

1. Introduction

1.1 Problem Statement

You are a cab rental start-up company. You have successfully run the pilot project and now want to launch your cab service across the country. You have collected the historical data from your pilot project and now have a requirement to apply analytics for fare prediction. You need to design a system that predicts the fare amount for a cab ride in the city.

1.2 Dataset

Training Dataset-

train_cab

fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count
4.5	2009-06-15 17:26:21 UTC	-73.844311	40.721319	-73.84161	40.712278	1
16.9	2010-01-05 16:52:16 UTC	-74.016048	40.711303	-73.979268	40.782004	1
5.7	2011-08-18 00:35:00 UTC	-73.982738	40.76127	-73.991242	40.750562	2
7.7	2012-04-21 04:30:42 UTC	-73.98713	40.733143	-73.991567	40.758092	1
5.3	2010-03-09 07:51:00 UTC	-73.968095	40.768008	-73.956655	40.783762	1
12.1	2011-01-06 09:50:45 UTC	-74.000964	40.73163	-73.972892	40.758233	1
7.5	2012-11-20 20:35:00 UTC	-73.980002	40.751662	-73.973802	40.764842	1
16.5	2012-01-04 17:22:00 UTC	-73.9513	40.774138	-73.990095	40.751048	1
	2012-12-03 13:10:00 UTC	-74.006462	40.726713	-73.993078	40.731628	1
8.9	2009-09-02 01:11:00 UTC	-73.980658	40.733873	-73.99154	40.758138	2
5.3	2012-04-08 07:30:50 UTC	-73.996335	40.737142	-73.980721	40.733559	1

Testing Dataset -

test

pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count
2015-01-27 13:08:24 UTC	-73.97332001	40.76380539	-73.98143005	40.74383545	1
2015-01-27 13:08:24 UTC	-73.98686218	40.71938324	-73.99888611	40.73920059	1
2011-10-08 11:53:44 UTC	-73.982524	40.75126	-73.979654	40.746139	1
2012-12-01 21:12:12 UTC	-73.98116	40.767807	-73.990448	40.751635	1
2012-12-01 21:12:12 UTC	-73.966046	40.789775	-73.988565	40.744427	1
2012-12-01 21:12:12 UTC	-73.960983	40.765547	-73.979177	40.740053	1
2011-10-06 12:10:20 UTC	-73.949013	40.773204	-73.959622	40.770893	1
2011-10-06 12:10:20 UTC	-73.777282	40.646636	-73.985083	40.759368	1
2011-10-06 12:10:20 UTC	-74.014099	40.709638	-73.995106	40.741365	1
2014-02-18 15:22:20 UTC	-73.969582	40.765519	-73.980686	40.770725	1
2014-02-18 15:22:20 UTC	-73.989374	40.741973	-73.9993	40.722534	1
2014-02-18 15:22:20 UTC	-74.001614	40.740893	-73.956387	40.767437	1
2010-03-29 20:20:32 UTC	-73.991198	40.739937	-73.997166	40.735269	1

Number of attributes:

· pickup_datetime - timestamp value indicating when the cab ride started.
· pickup_longitude - float for longitude coordinate of where the cab ride started.
· pickup_latitude - float for latitude coordinate of where the cab ride started.
· dropoff_longitude - float for longitude coordinate of where the cab ride ended.
· dropoff_latitude - float for latitude coordinate of where the cab ride ended.
· passenger_count - an integer indicating the number of passengers in the cab

Missing Values: Yes

Methodology

2.1 Pre Processing

When we look at statistic summary, we have several discoveries:

```
train.describe()
```

	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count
count	16067.000000	16067.000000	16067.000000	16067.000000	16012.000000
mean	-72.462787	39.914725	-72.462328	39.897906	2.625070
std	10.578384	6.826587	10.575062	6.187087	60.844122
min	-74.438233	-74.006893	-74.429332	-74.006377	0.000000
25%	-73.992156	40.734927	-73.991182	40.734651	1.000000
50%	-73.981698	40.752603	-73.980172	40.753567	1.000000
75%	-73.966838	40.767381	-73.963643	40.768013	2.000000
max	40.766125	401.083332	40.802437	41.366138	5345.000000

The minimum fare amount is negative.
Maximum latitude look sunreal.
Minimum passenger count is 0.
We are going to fix them.
Latitudes only range from -90 to 90
Longitudes only range from -180 to 180

Apart from that we will also perform the below actions :

- Shape the train and test sets

- Check for NaNs and drop them (if any)
- Check for outliers and drop them (if any)
- Type conversion of relevant fields
- Filtering out fare amounts greater than 200 and equal to 0 as that is unrealistic amounts.
- Filter out rows that have passengers greater than 6 and less than 1 as that is not possible in a cab ride

2.1.1 Exploratory Data Analysis

Now, for EDA. We went with the following assumptions.

- Does the number of passengers affect the fare?
- Does the date and time of pickup affect the fare?
- Does the day of the week affect the fare?
- Does the distance travelled affect the fare?

First, let's split the datetime field 'pickup_datetime' to the following -

Using these we shall calculate the day of the week and come to our conclusions about how pickup_location affects the fare. Also, create a new field 'distance' to fetch the distance between the pickup and the drop.

2.1.2 Feature Engineering

We can calulate the distance in a sphere when latitudes and longitudes are given by **Haversine formula**

$\text{haversine}(\theta) = \sin^2(\theta/2)$

Eventually, the formual boils down to the following where ϕ is latitude, λ is longitude, R is earth’s radius (mean radius = 6,371km) to include latitude and longitude coordinates (A and B in this case).

$a = \sin^2((\phi B - \phi A)/2) + \cos \phi A \,.\, \cos \phi B \,.\, \sin^2((\lambda B - \lambda A)/2)$

$c = 2 * \text{atan2}(\sqrt{a}, \sqrt{(1-a)})$

d = R · c

d = Haversine distance

```
def haversine_distance(lat1, long1, lat2, long2):
    data = [train, test]
    for i in data:
        R = 6371  #radius of earth in kilometers
        #R = 3959 #radius of earth in miles
        phi1 = np.radians(i[lat1])
        phi2 = np.radians(i[lat2])

        delta_phi = np.radians(i[lat2]-i[lat1])
        delta_lambda = np.radians(i[long2]-i[long1])

        #a = sin²((φB - φA)/2) + cos φA . cos φB . sin²((λB - λA)/2)
        a = np.sin(delta_phi / 2.0) ** 2 + np.cos(phi1) * np.cos(phi2) * np.sin(delta_lambda / 2.0) ** 2

        #c = 2 * atan2( √a, √(1−a) )
        c = 2 * np.arctan2(np.sqrt(a), np.sqrt(1-a))

        #d = R*c
        d = (R * c) #in kilometers
        i['H_Distance'] = d
    return d
```

Now that we have calculated the distance, we shall create columns for the following -

- year
- month
- date
- hour
- day of week

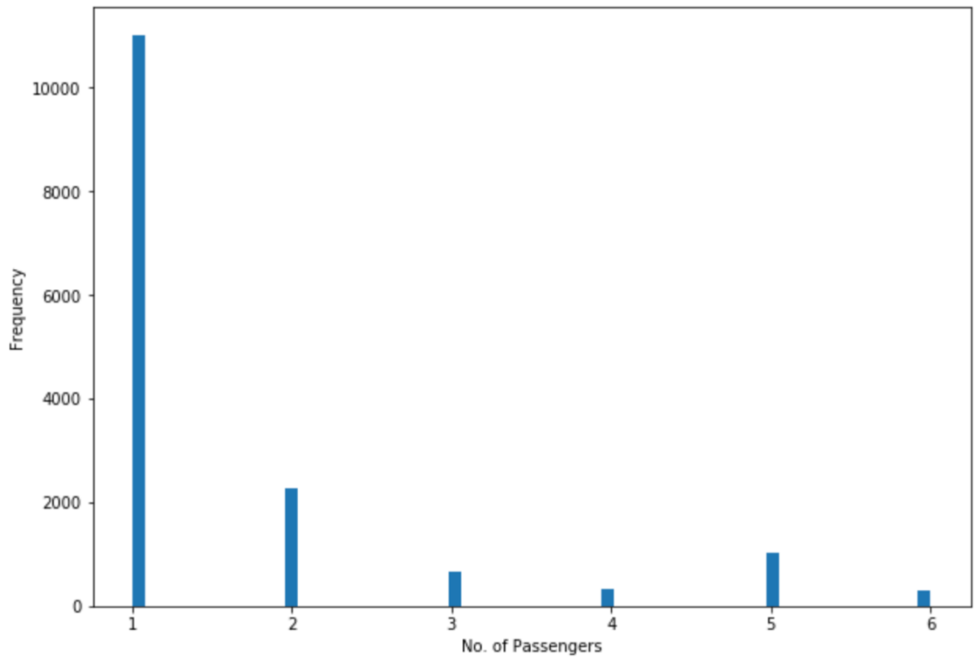
```
data = [train,test]
for i in data:
    i['Year'] = i['pickup_datetime'].dt.year
    i['Month'] = i['pickup_datetime'].dt.month
    i['Date'] = i['pickup_datetime'].dt.day
    i['Day of Week'] = i['pickup_datetime'].dt.dayofweek
    i['Hour'] = i['pickup_datetime'].dt.hour
```

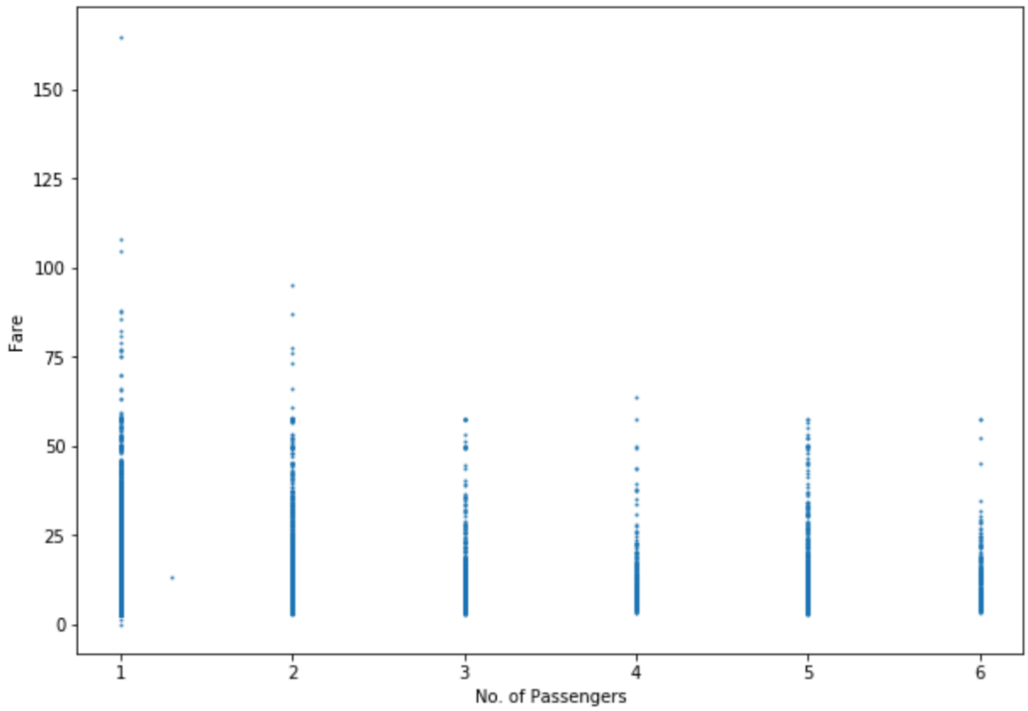
After adding new features for analysis our dataset looks like this:

	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count	H_Distance	Year	Month	Date	Day of Week	Hour
0	2015-01-27 13:08:24	-73.973320	40.763805	-73.981430	40.743835	1	2.323259	2015	1	27	1	13
1	2015-01-27 13:08:24	-73.986862	40.719383	-73.998886	40.739201	1	2.425353	2015	1	27	1	13
2	2011-10-08 11:53:44	-73.982524	40.751260	-73.979654	40.746139	1	0.618628	2011	10	8	5	11
3	2012-12-01 21:12:12	-73.981160	40.767807	-73.990448	40.751635	1	1.961033	2012	12	1	5	21
4	2012-12-01 21:12:12	-73.966046	40.789775	-73.988565	40.744427	1	5.387301	2012	12	1	5	21
5	2012-12-01 21:12:12	-73.960983	40.765547	-73.979177	40.740053	1	3.222549	2012	12	1	5	21
6	2011-10-06 12:10:20	-73.949013	40.773204	-73.959622	40.770893	1	0.929601	2011	10	6	3	12
7	2011-10-06 12:10:20	-73.777282	40.646636	-73.985083	40.759368	1	21.540102	2011	10	6	3	12
8	2011-10-06 12:10:20	-74.014099	40.709638	-73.995106	40.741365	1	3.873962	2011	10	6	3	12
9	2014-02-18 15:22:20	-73.969582	40.765519	-73.980686	40.770725	1	1.099794	2014	2	18	1	15

2.1.3 Visualizations

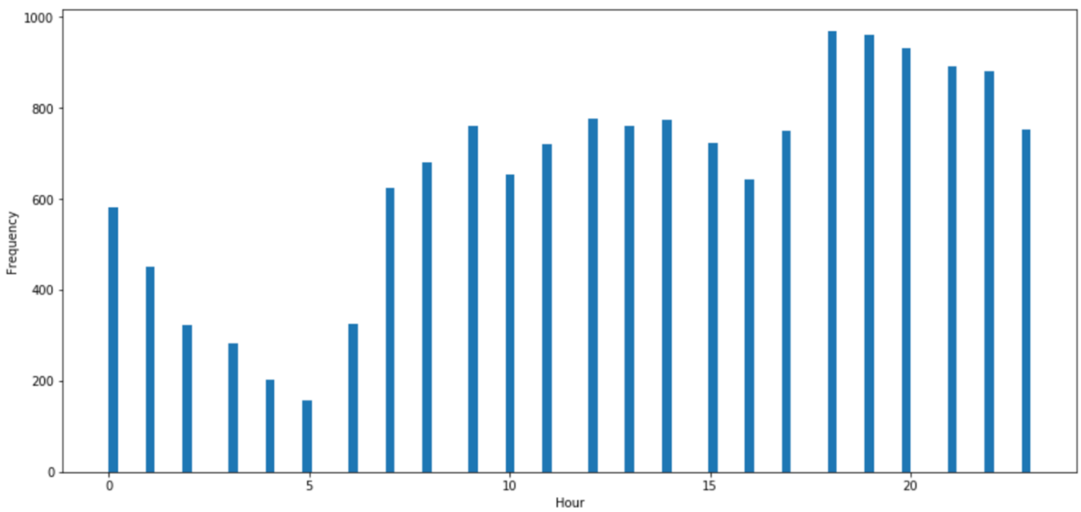
Number of passengers data:





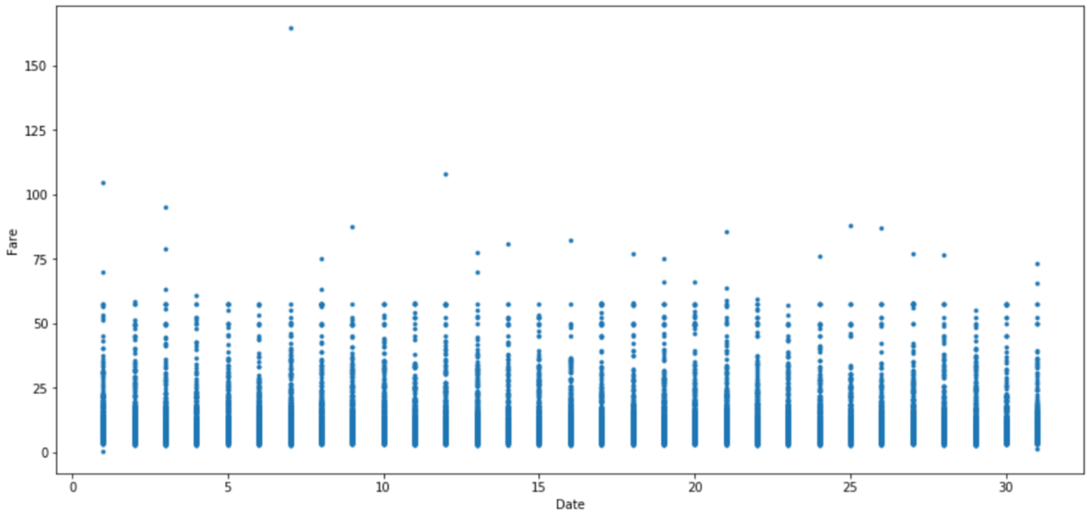
From the above 2 graphs we can see that single passengers are the most frequent travellers, and the highest fare also seems to come from cabs which carry just 1 passenger.

Hours data



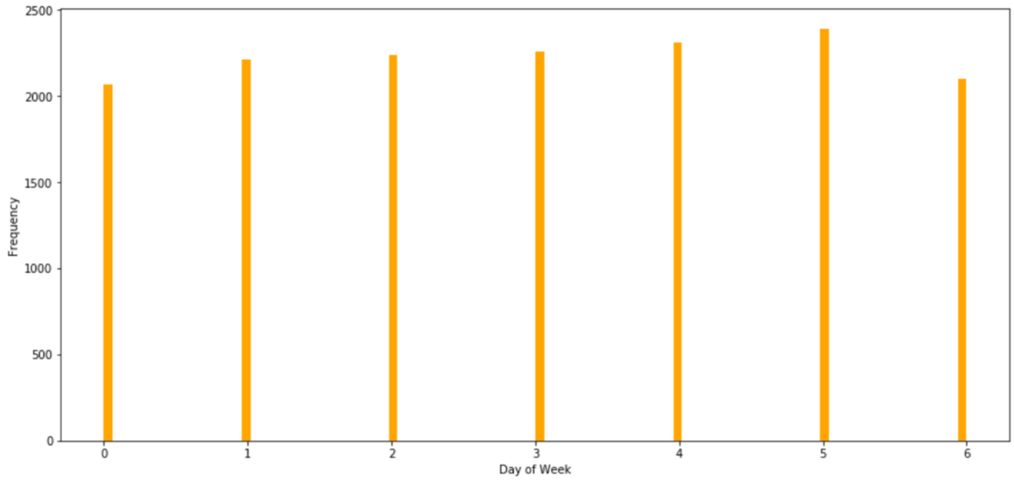
There are some spikes however highest number of rides recorded at 6pm

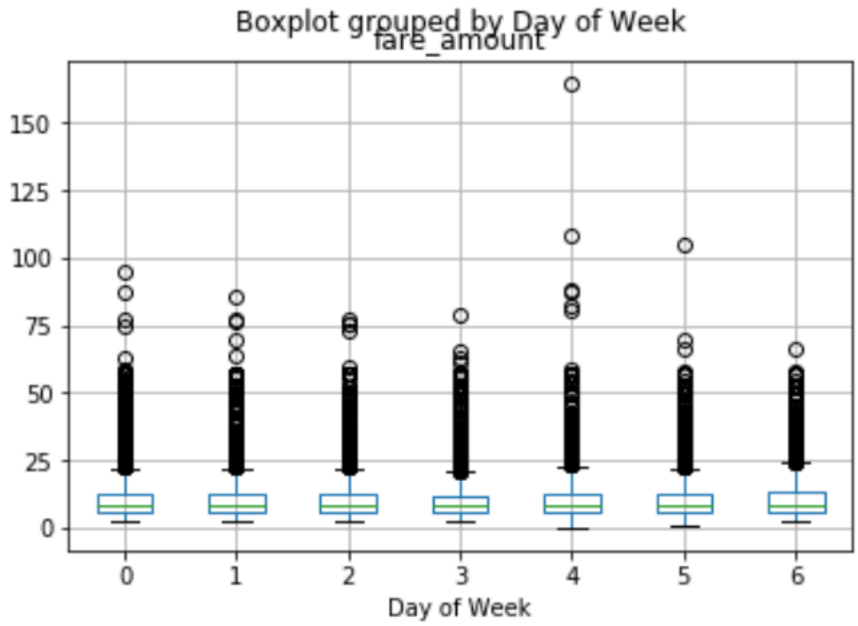
Dates Data



The fares throught the month mostly seem uniform, with the maximum fare received on the 7th

Day of the Week





day of the week doesn't seem to have that much of an influence on the number of cab rides

We go with a little more cleaning of the data after this as we have to limit the distance of the rides to 200km and remove rows with 0 distance.

Data Sample after preprocessing :

fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count	H_Distance	Year	Month	Date	of Week	Hour
4.5	2009-06-15 17:26:21	-73.844311	40.721319	-73.841610	40.712278	1.0	1.030764	2009	6	15	0	17
16.9	2010-01-05 16:52:16	-74.016048	40.711303	-73.979268	40.782004	1.0	8.450134	2010	1	5	1	16
5.7	2011-08-18 00:35:00	-73.982738	40.761270	-73.991242	40.750562	2.0	1.389525	2011	8	18	3	0
7.7	2012-04-21 04:30:42	-73.987130	40.733143	-73.991567	40.758092	1.0	2.799270	2012	4	21	5	4
5.3	2010-03-09 07:51:00	-73.968095	40.768008	-73.956655	40.783762	1.0	1.999157	2010	3	9	1	7

2.2 Model Development

After Data pre-processing the next step is to develop a model using a train or historical data which can perform to predict accurate result on test data or new data. Here we have tried with different models and will choose the model which will provide the most accurate values.

2.2.1 Decision Tree

Decision Tree is a supervised machine learning algorithm, which is used to predict the data for classification and regression. It accepts both continuous and categorical variables. A decision tree is a decision support tool that uses a tree-like graph or model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. Each branch connects nodes with “and” and multiple branches are connected by “or”.
We have prepared a model by using decision tree algorithm and calculate RMSE value and MAE value for our project in R and Python are -

Decision Tree	R	PYTHON
RMSE Test	5.11	6.508
MAE	2.642	2.895

2.2.2 Random Forest

Random Forest is an ensemble technique that consists of many decision trees. The idea behind Random Forest is to build n number of trees to have more accuracy in dataset. It is called random forest as we are building n no. of trees randomly. In other words, to build the decision trees it selects randomly n no of variables and n no of observations. It means to build each decision tree on random forest we are not going to use the same data. The higher no of trees in the random forest will give higher no of accuracy, so in random forest we can go for multiple trees. It can handle large no of independent variables without variable deletion and it will give the estimates that what variables are important.
The RMSE value and MAE value for our project in R and Python are -

Decision Tree	R	PYTHON
RMSE Test	4.22	4.929
MAE	2.04	2.895

2.2.3 Liner Regression

Linear Regression is one of the statistical method of prediction. It is most common predictive analysis algorithm. It uses only for regression, means if the target variable is continuous than we can use linear regression machine learning algorithm. The RMSE value and MAE value for our project in R and Python are -

Decision Tree	R	PYTHON
RMSE Test	7.763	6.530
MAE Test	3.123	3.2935

Since Random forest algorithm gives us the least error rate we go with that algorithm to generate fare amounts for the test data set.