NYC Taxi Fare prediction

CIS 731: ANN project

By: Gaurav Misra
Hardik Patil

ANN Project

1. Data Summary
2. Data Exploration and cleaning
3. Deep Neural Network
4. Backpropagation
5. ARIMA Model
6. Testing
7. Improvements

Data Summary

	key	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count
0	2009-06-15 17:26:21.0000001	4.5	2009-06-15 17:26:21+00:00	-73.844311	40.721319	-73.841610	40.712278	1
1	2010-01-05 16:52:16.0000002	16.9	2010-01-05 16:52:16+00:00	-74.016048	40.711303	-73.979268	40.782004	1
2	2011-08-18 00:35:00.00000049	5.7	2011-08-18 00:35:00+00:00	-73.982738	40.761270	-73.991242	40.750562	2
3	2012-04-21 04:30:42.0000001	7.7	2012-04-21 04:30:42+00:00	-73.987130	40.733143	-73.991567	40.758092	1
4	2010-03-09 07:51:00.000000135	5.3	2010-03-09 07:51:00+00:00	-73.968095	40.768008	-73.956655	40.783762	1

Data Exploration and Cleaning

Data features -

- Fare_amount (target)
- Pickup_datetime
- Pickup_longitude
- Pickup_latitude
- Dropoff_longitude
- Dropoff_latitude
- passenger_count

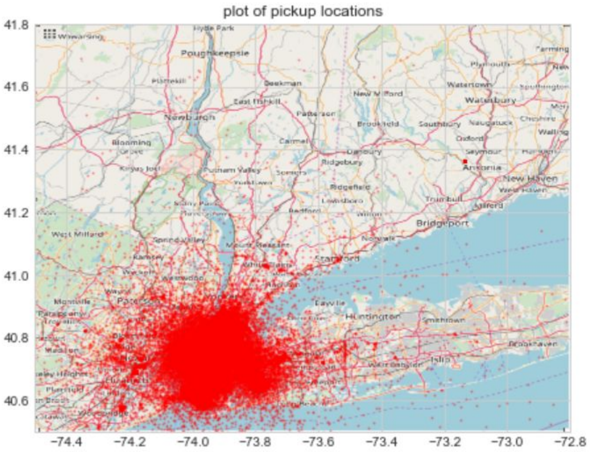
Data Exploration and Cleaning

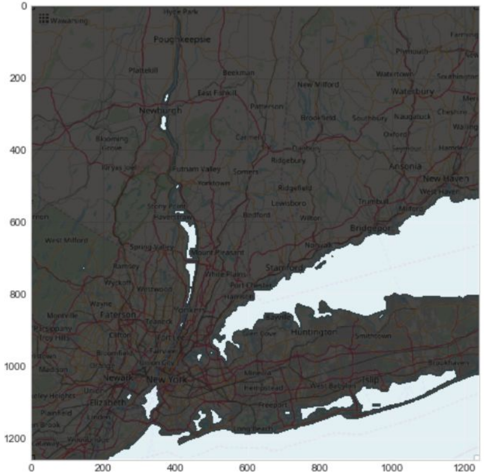
Removing inconsistencies -

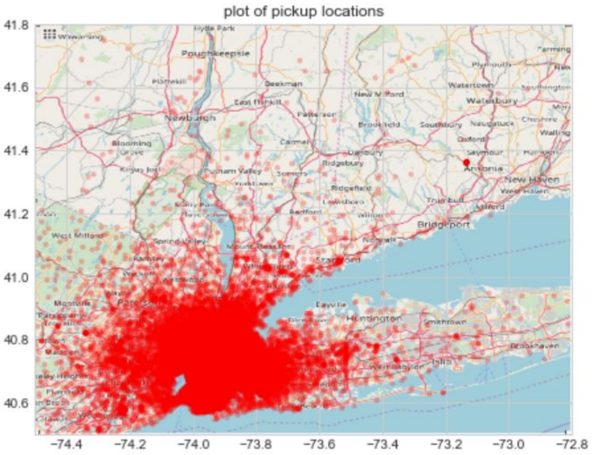
1. Fare amount less than zero
2. Drop Off location missing
3. Trips that either started or finished in water
4. Passenger count was less than zero
5. Trips that started or finished outside NYC

Adding New Features - Trip distance in km

Data Exploration and Cleaning



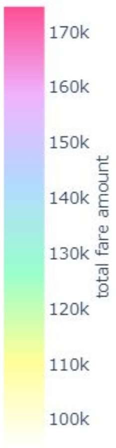




Data Exploration and Cleaning

Total fare amount by month - year .

	2009	2010	2011	2012	2013	2014	2015
JAN	118643.0	124919.0	117739.0	128218.0	156208.0	148024.0	136167.0
FEB	116939.0	94767.0	122696.0	130610.0	146777.0	143348.0	137234.0
MAR	125536.0	110680.0	138564.0	143354.0	168851.0	170265.0	151239.0
APR	123665.0	132362.0	129557.0	140865.0	164346.0	163842.0	153621.0
MAY	131621.0	141142.0	139251.0	144536.0	164190.0	172570.0	157171.0
JUN	124646.0	132744.0	136509.0	141815.0	162133.0	157709.0	149042.0
JUL	120873.0	128220.0	131414.0	132635.0	159098.0	150000.0	nan
AUG	123842.0	110744.0	117543.0	135492.0	141078.0	146172.0	nan
SEP	129113.0	126919.0	137149.0	159596.0	159128.0	153707.0	nan
OCT	140527.0	130815.0	142818.0	162768.0	174593.0	164582.0	nan
NOV	127090.0	127009.0	131768.0	150595.0	160419.0	152692.0	nan
DEC	128658.0	126701.0	136420.0	163520.0	159181.0	151686.0	nan



Deep Neural Network

In DNN we used Sequential Model from keras.

Layers used - Dropout Layer, Dense , BatchNormalization

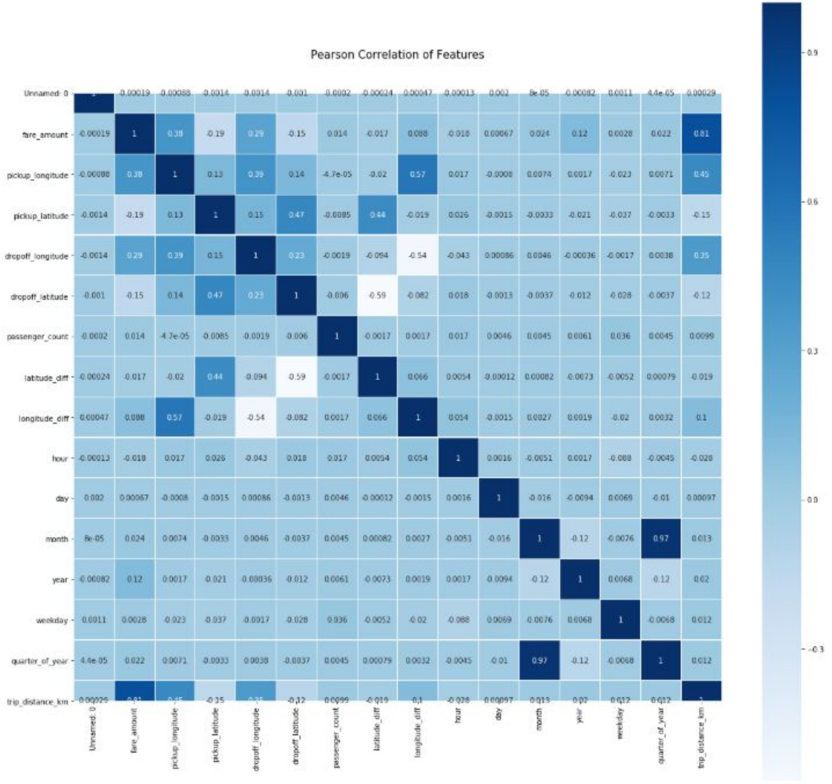
Activation function -relu

Optimizer - Adam and Adamax

Evaluation metric - rmse

Number of epochs - 80

Pearson correlation



fare_amount	1.000000
pickup_longitude	0.381347
pickup_latitude	-0.189143
dropoff_longitude	0.291213
dropoff_latitude	-0.154360
passenger_count	0.014121
latitude_diff	-0.016596
longitude_diff	0.088398
hour	-0.018280
day	0.000666
month	0.024430
year	0.116795
weekday	0.002760
quarter_of_year	0.021874
trip_distance_km	0.807037

Deep Neural Network

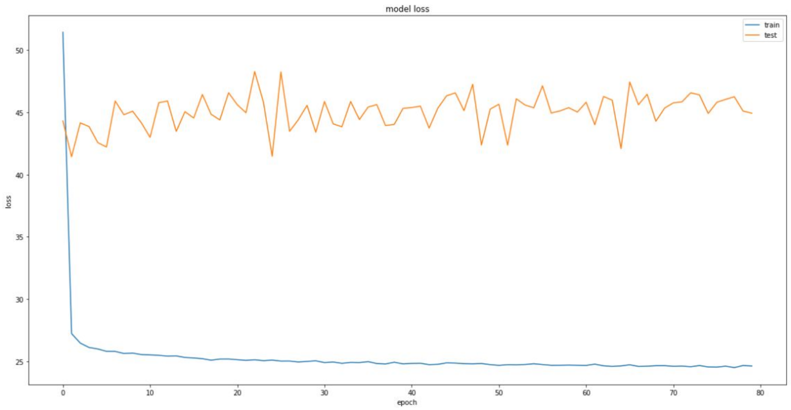
```
model = Sequential()
model.add(Dropout(0.2,input_shape=(para.shape[1],)))
model.add(BatchNormalization())
model.add(Dense(512,activation='relu'))#512 neurons in input layer
model.add(Dropout(0.2))
model.add(BatchNormalization())
model.add(Dense(256,activation='relu')) #256 neurons in hidden layer
model.add(BatchNormalization())
model.add(Dense(128,activation='relu'))  # 128 neurons in hidden layer
model.add(BatchNormalization())
model.add(Dense(64,activation='relu'))   # 64 neurons in hidden layer
model.add(BatchNormalization())
model.add(Dense(32,activation='relu'))   # 32 neurons in hidden layer
model.add(BatchNormalization())
model.add(Dense(16,activation='relu')) # 16 neurons in hidden layer
model.add(BatchNormalization())
model.add(Dense(8,activation='relu')) # 8 neurons in hidden layer
model.add(BatchNormalization())
model.add(Dense(1)) # 1 neuron in output layer
```

The rmse for

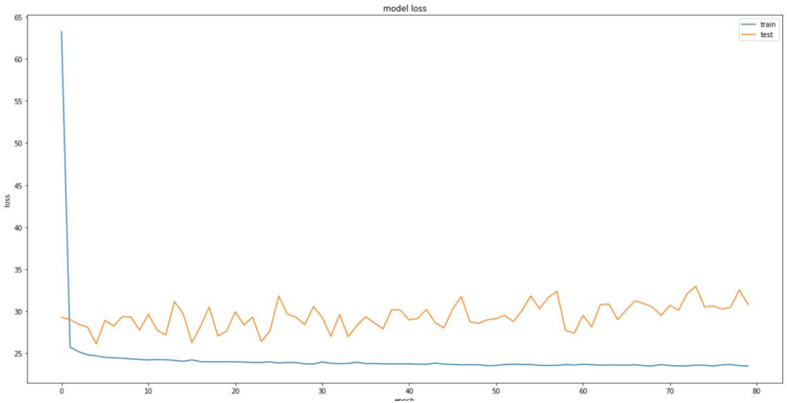
Model 1 - 3.5715

Model 2 - 2.4585

Loss plot for DNN



Model 1



Model 2

Simple Backpropagation

In Backpropagation we used Sequential Model from keras.

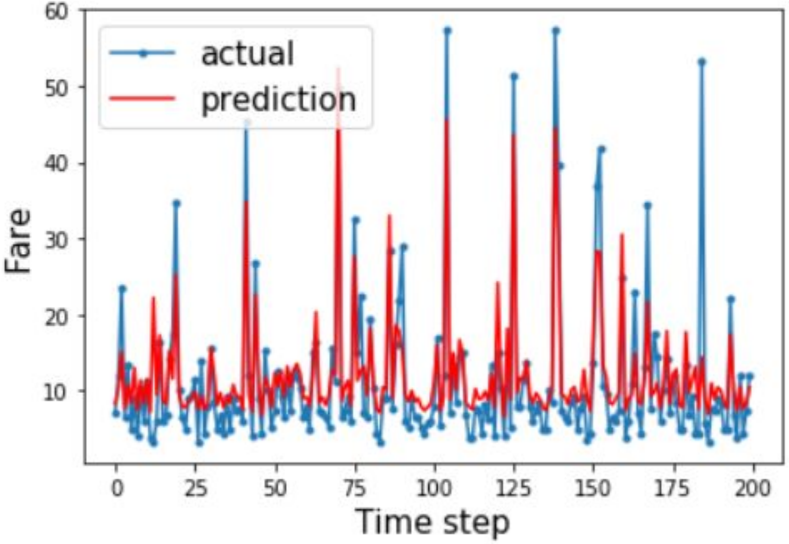
Layers used - Dropout Layer, Dense , BatchNormalization

Activation function - relu

Optimizer - Adam

Evaluation metric - Rmse **5.322**

Number of epochs - 200



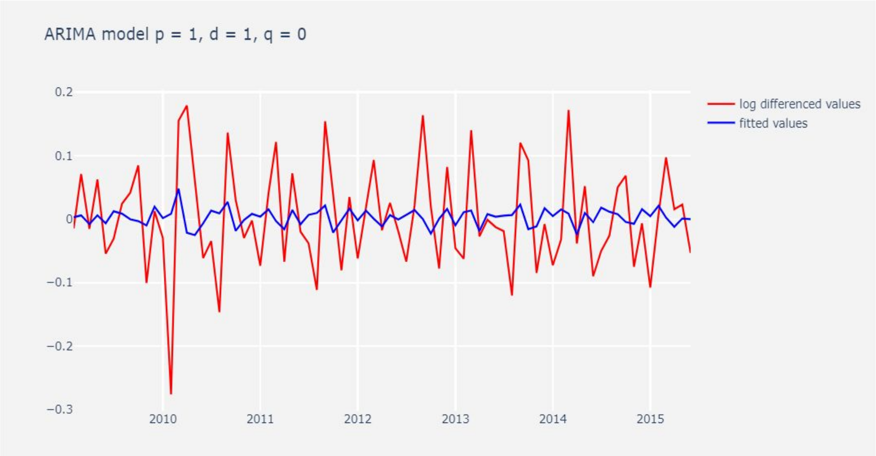
ARIMA Model

- Considered 2 different sets of values for p, d and q.
- Arima Works better than remaining models because the log time series and the exponential averages are slightly increasing as time passes.

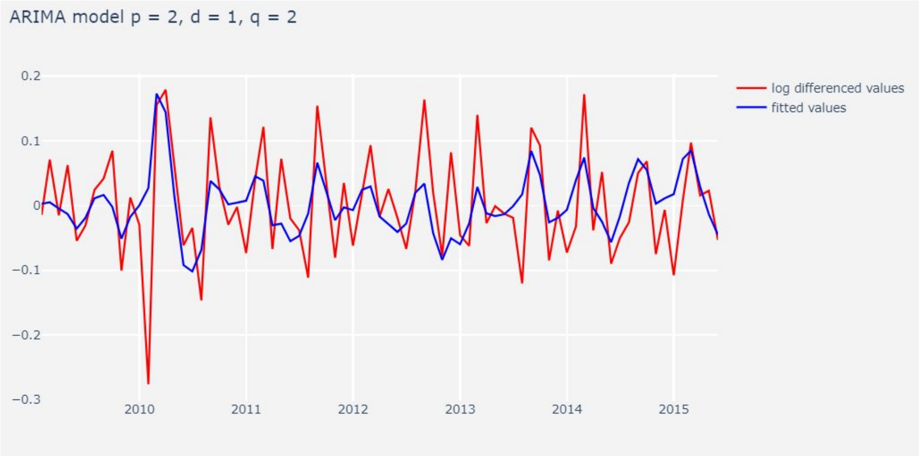
Likelihood of Arima Models:

- Arima (1,1,0)   :: 84.5
- Arima (2,1,2)   :: 98.85

ARIMA Model

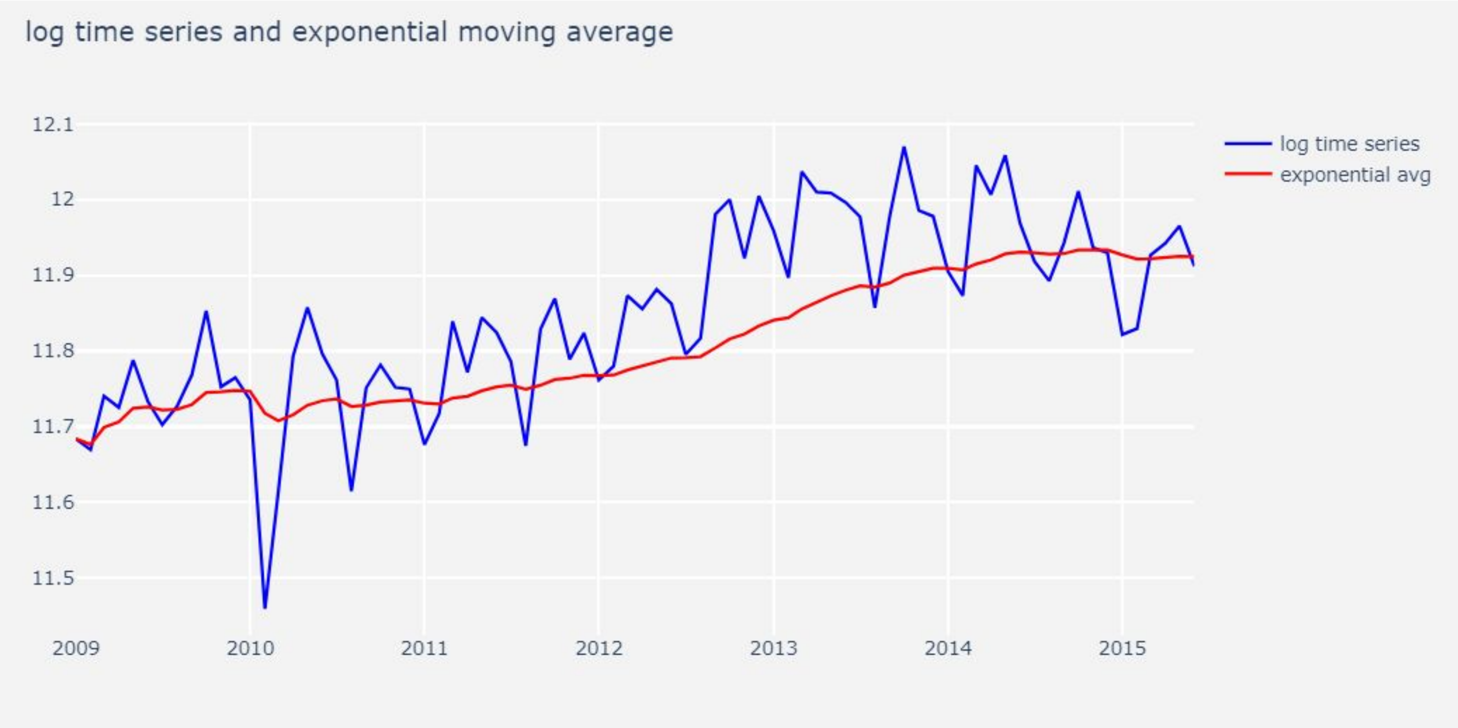



```
                             ARIMA Model Results
==============================================================================
Dep. Variable:           D.fare_amount   No. Observations:                   77
Model:                  ARIMA(1, 1, 0)   Log Likelihood                  84.517
Method:                        css-mle   S.D. of innovations              0.081
Date:                 Fri, 29 Nov 2019   AIC                           -163.034
Time:                         13:29:23   BIC                           -156.002
Sample:                     02-01-2009   HQIC                          -160.221
                          - 06-01-2015
```




```
                             ARIMA Model Results
==============================================================================
Dep. Variable:           D.fare_amount   No. Observations:                   77
Model:                  ARIMA(2, 1, 2)   Log Likelihood                  98.851
Method:                        css-mle   S.D. of innovations              0.066
Date:                 Fri, 29 Nov 2019   AIC                           -185.701
Time:                         13:29:25   BIC                           -171.638
Sample:                     02-01-2009   HQIC                          -180.076
                          - 06-01-2015
```

Why ARIMA works well



Testing

- We are currently testing our data on simple models, for checking how DNN and Arima are better than ML algorithms.
- Algorithms which we are using for testing are : XGBoost, Linear Regression.

RMSE

- XGBoost :: 3.9
- Linear Regression :: 5.5

Improvements

1. We don't have the drop off timestamp so we do not know when the trip ended.
2. Traffic density of the roads.
3. Distance was calculated using manhattan distance but can be improved by using google API.

Thank You!