A PROJECT REPORT ON

BMTC TRIP TIME PREDICTION

Submitted to

International Institute of Information Technology, Bangalore

Submitted by

Subarna Kanti Samanta MT2019523

Under the guidance of Prof. G Srinivasaraghavan

Problem Statement

In this project we are asked to predict the trip duration of bmtc buses from source to destination .

Dataset Used

Raw Dataset : It consists of six columns of busid, latitude , longitude, direction, speed and time-
stamp.

• Busid: It consists of bus id of around 6000 buses .

• Latitude: It is the latitude of the location at which bus was present at a given timestamp.

• Longitude: It is the longitude of the location at which bus was present at a given timestamp.

• Direction: it is the direction in which bus is moving.

• Speed: speed of bus at that timestamp.

• Timestamp: It consists of time and date of a bus at a particular start latitude and longitude.

NOTE: Raw dataset is 14GB in size and doesn't consist of column names . Hence these names
are provided according to the understanding of data. Columns like direction and speed consist of
almost all null values and hence wasn’t considered for prediction.

#Model links are provided at the end

Test Dataset: There are 100 locations for each Id in the test set including starting and ending point of the journey for which we need to predict the duration of journey. Each location is specified by latitude and longitude separated by ':'.

• Busid: It consists of bus id of around 108 buses .

• Timestamp: It consists of time and date of a bus at a particular start latitude and longitude.

• LATLONG1: It refers to starting point.

• LATLONG100: It refers to ending point of the journey for each Id in the test set.

• LATLONG2 - LATLONG99: These are locations in the path of journey.

Approach

As we have to predict trip time for buses starting from a particular latitude and longitude.

So we are going to use regression model for this purpose.

We will first convert the raw dataset provided to us into required train dataset removing irrelevant columns.

Our train data set is in format of test dataset.

We will do preprocessing and model training.

The whole project is done in various jupyter notebooks saved in different files where various splitted files of our main train data file is saved

Later on all the jupyter notebooks are collabed in one notebook and then after python code of main jupyter notebook is downloaded.

Project was done in

- My laptop configuration 4GB ram, i3 processor which causing much time for operations.

- In order to reduce the time I have used my friends laptop for some parts where much bigger operation are needed like running 1Gb file and doing operations on it.

- I have also taken support Kaggle notebooks, when my freind laptop was not available

Steps Involved

- Importing eseential libraries

In [1]:
```
#Importing necessary packages
import pandas as pd
import numpy as np
import seaborn as sns
sns.set(color_codes =True)
import matplotlib.pyplot as plt
%matplotlib inline
import glob
```

• **Chunking**: As file of 14GB size can’t be opened in the text editor (too large) , as well as computation will take very long time to go through it. So I divided it into small chunks of size **59.2 MB** and around **215 chunks** were created(where each chunk consist of 1000000 rows).

In [3]:
```
#splitting bigger csv files into chunks
chunksize = 10 ** 6
batch_no = 1
for chunk in pd.read_csv('/home/subarna/Documents/BMTC/BMTC/train.csv',chunksize=chunksize):
    chunk.to_csv('training set'+str(batch_no)+'.csv',index=False)
    batch_no += 1
```

- As column headers are not present so appended the Bus_id','Latitude','Longitude','Date and Time'

to top of each csv.

“ data.set_axis(['Bus_id','Latitude','Longitude','Date and Time'], axis=1, inplace=True) “

- **Elemination of rows with defective informations:**

Rows with information like longitude and latitude beyond range as per bangalore location are removed.

Also rows with null values and duplicate values are removed.

In [2]:
```
#removing defective informations
number_of_files = 215

for i in range(1, number_of_files+1):
    data = pd.read_csv("/home/subarna/Documents/BMTC/BMTC/training_sets/training set{}.csv".format(i))
    del data['27.00']
    del data['214.0']
    data.set_axis(['Bus_id','Latitude','Longitude','Date and Time'], axis=1, inplace=True)
    data.drop(data[data['Latitude'] < 12].index, inplace = True)
    data.drop(data[data['Latitude'] >14].index, inplace = True)
    data.drop(data[data['Longitude'] <77].index, inplace = True)
    data.drop(data[data['Longitude'] >78.5].index, inplace = True)
    data.dropna(axis=0, how='any', thresh=None, subset=None, inplace=True)
    data.to_csv('mod_training_set'+str(i)+'.csv',index=False)
```

In [ ]:
```
#removing duplicate cells
number_of_files = 215

for i in range(1, number_of_files+1):
    data = pd.read_csv("/home/subarna/Documents/BMTC/BMTC/mod_data/mod_training_set{}.csv".format(i))
    data['Latitude'] = data['Latitude'].map(str) + ':' +data['Longitude'].map(str)
    data=data.drop(['Longitude'], axis=1).rename(columns={'Latitude':'LATLONG'})
    data.drop_duplicates(subset =["LATLONG",'Bus_id'],keep = 'first', inplace = True)
    data.to_csv('set'+str(i)+'.csv',index=False)
```

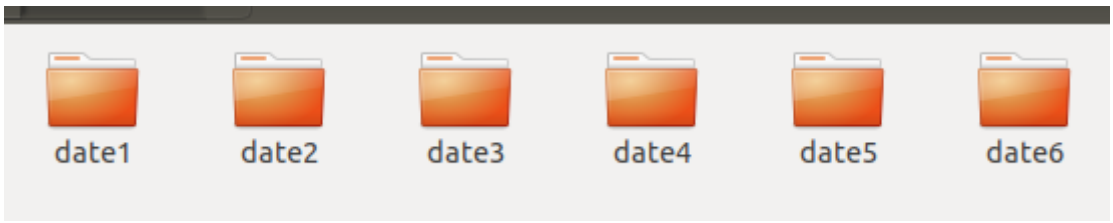
All the above operatipns are performed on all the 215 files by putting in a loop function.

- Splitting Date and Time columns of each file and rearranging them as per there dates and group all the files as per date in seperate folders.

In [ ]:
```
#rename date and time

number_of_files = 37

for i in range(1, number_of_files+1):
    df = pd.read_csv("/home/subarna/Documents/BMTC/new_task/date1/set{}.csv".format(i))
    df.rename(columns={'Date and Time': 'Date_Time'}, inplace=True)
    df[['Date','Time']] = df.Date_Time.str.split(expand=True)
    del df['Date_Time']
    for j, x in df.groupby('Date'):
        x.to_csv(str(j)+"_{}.csv".format(i),index=False)
```

In [ ]:
```
for i in range(39,74):
    df = pd.read_csv("/home/subarna/Documents/BMTC/new_task/date2/set{}.csv".format(i))
    df.rename(columns={'Date and Time': 'Date_Time'}, inplace=True)
    df[['Date','Time']] = df.Date_Time.str.split(expand=True)
    del df['Date_Time']
    for j, x in df.groupby('Date'):
        x.to_csv(str(j)+"_{}.csv".format(i),index=False)
```

In [ ]:
```
for i in range(75,109):
    df = pd.read_csv("/home/subarna/Documents/BMTC/new_task/date3/set{}.csv".format(i))
    df.rename(columns={'Date and Time': 'Date_Time'}, inplace=True)
    df[['Date','Time']] = df.Date_Time.str.split(expand=True)
```



- Merging all the files of each folder into a single file and name them according to date index.

In [ ]:
```
import glob
# get data file names
path ='/home/subarna/Documents/BMTC/new_task/modifie_dates/date1_mod'
filenames = glob.glob(path + "/*.csv")

dfs = []
for filename in filenames:
    dfs.append(pd.read_csv(filename))

# Concatenate all data into one DataFrame
big_frame = pd.concat(dfs, ignore_index=True)
big_frame.to_csv('date1.csv',index=False)
```

We get six folders and each of size around 1 Gb.

- Splitting individual date file into many as per bus id as per the below code:

“ for i,x in df.groupby('Bus_id'):
x.to_csv("{}.csv".format(i),index=False) “

Now we have more than 5500 csv files in each date folder and named as per the bus id.

- **Sampling Process:**

Now I have taken 15 samples from each file in particular date folder with 100 random rows in each dataframe.

Now each dataframe is further modified to a single row with LATLONG 1 to 100 with each LATLONG assigned symbol “:” which means latitude and longitude on either side od it.

The particular row also has “bus id” and “Timestamp” labeled on it.

Each row signifies **Trip** with **Date** on which trip happens also **duration** of trip.

Such 15 trips I get from each bus Id on each six dates.

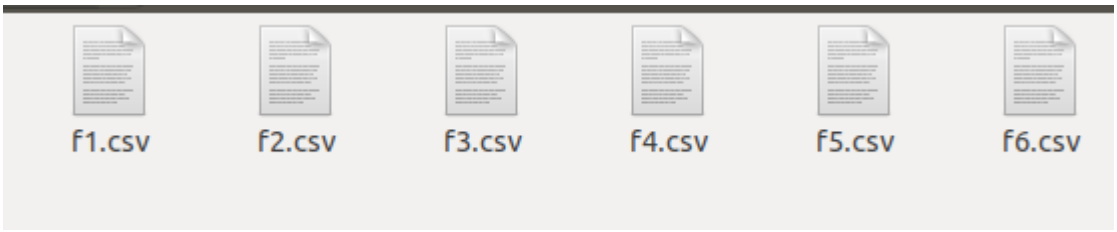
Sampling is done **random** by both **replacement** and **without replacement** as per condition.

In [ ]:
```
path ='/home/subarna/Documents/BMTC/new_task/modifie_dates/date1'
filenames = glob.glob(path + "/*.csv")
i=1
for filename in filenames:
    df=pd.read_csv(filename)
    q=15
    for x in range(1,q+1):
        if df.shape[0]>=100:
            df_sub=df.sample(n=100)
        else:
            df_sub=df.sample(n=100,replace=True)
        df1=df_sub.sort_values(by=['Time'])
        df1['Date'] = df1['Date'].map(str)+' ' +df1['Time'].map(str)
        df1=df1.drop(['Time'], axis=1).rename(columns={'Date':'Time_stamp'})
        df1['Time_stamp']=pd.to_datetime(df1.Time_stamp)
        u=[df1.Time_stamp.max()-df1.Time_stamp.min()]
        df0=df1.groupby('Bus_id')['LATLONG'].apply(' '.join).reset_index()
        df3 = df0['LATLONG'].str.split(n=100, expand=True)
        df3.columns = ['LATLONG{}'.format(x+1) for x in df3.columns]
        df0 = df0.join(df3)
        del df0['LATLONG']
        df0=df0.rename(columns={'Bus_id':'Id'})
        dk = pd.DataFrame({'A':u})
        dk['B'] = (pd.Timestamp('now').normalize() + dk['A']).dt.time
        df0.insert(1,'TimeStamp',dk['B'],True)
        df0.insert(2,'Date','2016-07-01',True)
        df0['TimeStamp'] = df0['Date'].map(str)+' '+df0['TimeStamp'].map(str)
        del df0['Date']
```

```
        df0 = df0.join(df3)
        del df0['LATLONG']
        df0=df0.rename(columns={'Bus_id':'Id'})
        dk = pd.DataFrame({'A':u})
        dk['B'] = (pd.Timestamp('now').normalize() + dk['A']).dt.time
        df0.insert(1,'TimeStamp',dk['B'],True)
        df0.insert(2,'Date','2016-07-01',True)
        df0['TimeStamp'] = df0['Date'].map(str)+' '+df0['TimeStamp'].map(str)
        del df0['Date']
        df0['TimeStamp']=pd.to_datetime(df0.TimeStamp)
        df0.to_csv('set_'+str(i)+'.csv',index=False)
        i+=1
```

- Merging all trips of particular date in individual folder.

```
“# get data file names
path ='/home/subarna/Documents/BMTC/trips'
filenames = glob.glob(path + "/*.csv")

dfs = []
for filename in filenames:
    dfs.append(pd.read_csv(filename))

# Concatenate all data into one DataFrame
big_frame = pd.concat(dfs, ignore_index=True)
big_frame.to_csv('f1.csv',index=False)”//same now for
f2,f3,f4,f5,f6.
```



- **Modifying new files f{}:**

The new 6 files creates are further modified to new six files with columns containing l**atitude and longitude seperated** so we have now **100 latitudes and longitudes for 100** locations arranged in a order as the bus goes and we have only column for **time duration** of each trip without dates mentioned in them (because trip durations **doesn’t depend on dates**).

Also we converted object type columns created into float types (latitude and longitude).

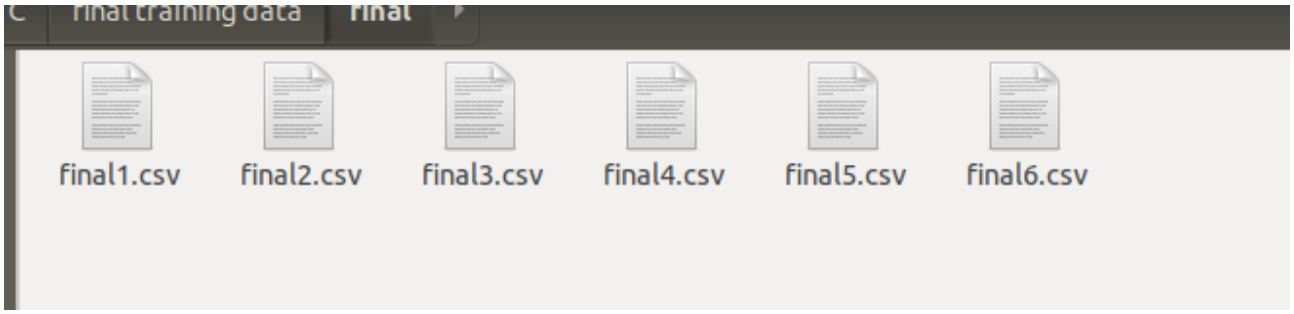
```
for x in range(1,7):
    df=pd.read_csv('f{}.csv'.format(x))
    df[['Date','Time']] = df.TimeStamp.str.split(expand=True)
    del df['TimeStamp']
    del df['Date']
```

```
    df[['HH','MM','SS']] = df.Time.str.split(":",n=3,expand=True)
    del df['Time']
    df.dropna(axis=0, how='any', thresh=None, subset=None,
inplace=True)
    convert_dict = {'HH':int,'MM':int,'SS':int}
    df = df.astype(convert_dict)
    df['duration']=df['HH']*3600+df['MM']*60+df['SS']
    del df['HH']
    del df['MM']
    del df['SS']
    for i in range(1,101):

df[['lat{}'.format(i),'long{}'.format(i)]]=df['LATLONG{}'.format(i
)].str.split(":", n = 1, expand = True)
        del df['LATLONG{}'.format(i)]
    cols = df.select_dtypes(include=['object']).columns
    df[cols] = df[cols].apply(pd.to_numeric, downcast='float',
errors='coerce')
    df.to_csv('final{}.csv'.format(x))
```

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Id	duration	lat1	long1	lat2	long2	lat3	long3	lat4	long4	lat5	long5	lat6
2	150813293	8372	12.78076	77.6342	12.78076	77.6342	12.78076	77.6342	12.78076	77.6342	12.78076	77.6342	12.780
3	150222449	58728	12.917833	77.57273	12.917069	77.585304	12.916972	77.59413	12.916685	77.60728	12.916296	77.61687	12.9164
4	150221099	23409	13.02582	77.63245	13.02582	77.63245	13.02582	77.63245	13.02582	77.63245	13.02582	77.63245	13.025
5	150222117	56832	13.190086	77.80731	13.188541	77.80498	13.186307	77.80216	13.185542	77.80021	13.197987	77.77681	13.2083
6	150220845	61788	13.084229	77.82283	13.078524	77.81098	13.07732	77.79964	13.06612	77.77662	13.059388	77.77099	13.0275
7	150811626	62671	12.852661	77.67702	12.85261	77.67703	12.852606	77.67702	12.847725	77.67551	12.86068	77.66057	12.9077
8	150220591	69092	13.002797	77.68304	12.995186	77.666176	12.997014	77.66915	12.980998	77.6942	12.978893	77.69515	12.9504
9	150221560	63987	12.9169	77.590546	12.916649	77.60479	12.916617	77.60778	12.909934	77.62718	12.894955	77.636375	12.8938
10	150220476	56312	12.919381	77.57036	12.914471	77.56813	12.914016	77.565765	12.914714	77.56552	12.918631	77.56397	12.9192
11	150812906	49408	13.02667	77.63841	13.026653	77.63844	13.026691	77.63844	13.026681	77.63872	13.026663	77.63864	13.0268
12	150218918	63554	12.924759	77.59266	12.928666	77.58539	12.929384	77.583855	12.943917	77.58516	12.94638	77.58978	12.9460
13	150220691	78983	13.019332	77.501686	12.993205	77.504654	12.988268	77.51168	12.98714	77.51998	12.978196	77.54426	12.9628

GG1 | lat94

	GG	GH	GI	GJ	GK	GL	GM	GN	GO	GP	GQ	GR	GS
1	lat94	long94	lat95	long95	lat96	long96	lat97	long97	lat98	long98	lat99	long99	lat100
2	12.780837	77.63474	12.780837	77.63474	12.780837	77.63474	12.780837	77.63474	12.780837	77.63474	12.780837	77.63474	12.780
3	12.916474	77.61376	12.917867	77.62259	12.91647	77.613045	12.916448	77.60843	12.916702	77.60394	12.916649	77.602745	12.916
4	13.0259	77.63239	13.0259	77.63239	13.0259	77.63239	13.0259	77.63239	13.0259	77.63239	13.0259	77.63239	13.0
5	13.187616	77.7974	13.226969	77.73534	13.22775	77.734406	13.233687	77.727036	13.244965	77.71312	13.243318	77.720116	13.209
6	12.986862	77.64769	12.99517	77.66541	13.007913	77.69052	13.072419	77.78702	13.073689	77.78951	13.076095	77.79377	13.077
7	12.945682	77.596886	12.869298	77.65386	12.848412	77.67053	12.853685	77.67526	12.846935	77.67144	12.888914	77.63999	12.914
8	12.922149	77.65929	12.924187	77.651886	12.919907	77.643234	12.916655	77.63041	12.917496	77.62606	12.91733	77.62257	12.921
9	12.823832	77.684204	12.825561	77.68307	12.84415	77.67358	12.864107	77.6579	12.916392	77.61425	12.916496	77.608574	12.917
10	12.957342	77.56745	12.941973	77.565315	12.937328	77.54923	12.928966	77.5452	12.917409	77.56461	12.914793	77.56841	12.919
11	13.017948	77.5571	13.038325	77.55738	13.043396	77.557	13.04377	77.60462	13.040152	77.624626	13.039809	77.62488	13.034
12	12.953567	77.59138	12.941854	77.55486	12.94346	77.559616	12.943415	77.56081	12.947321	77.56742	12.936801	77.58047	12.931
13	13.037241	77.5217	13.039416	77.51926	13.030548	77.51889	13.027616	77.517456	13.0203	77.5081	13.019023	77.50146	13.020
14	12.843016	77.61584	12.841555	77.64772	12.846954	77.670654	12.846988	77.670616	12.84691	77.67067	12.846967	77.67062	12.846
15	12.893895	77.59908	12.866562	77.591995	12.819533	77.584816	12.80247	77.59666	12.797461	77.60902	12.79711	77.61442	12.78



- Merging all the above files in a single **data file:**

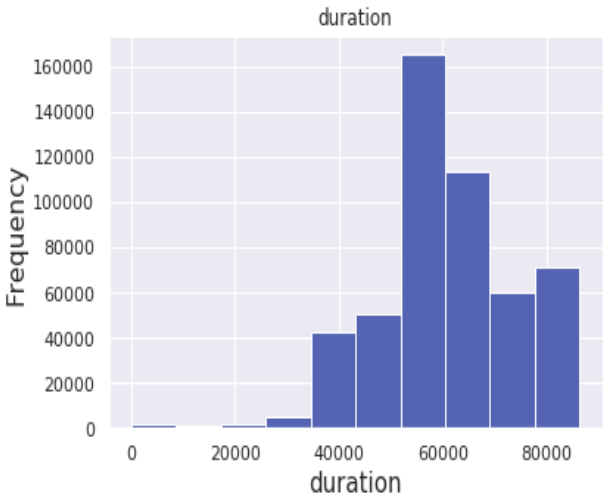
```
#merging all trips
path ='/home/subarna/Documents/BMTC/final training data/final'
filenames = glob.glob(path + "/*.csv")

dfs = []
for filename in filenames:
    dfs.append(pd.read_csv(filename))

# Concatenate all data into one DataFrame
big_frame = pd.concat(dfs, ignore_index=True)
big_frame.to_csv('data.csv',index=False)
```

Now this is our final training data of around 1.2 Gb which will be further processed and reduction will occur.

- Null values and duplicate values are dropped in new final training data.
- Histogram plots are used to analyse the training data.

In [11]:
```
#histogram plots
fig=plt.figure(figsize=(5000,10))
df.hist(column="duration")
plt.xlabel("duration",fontsize=15)
plt.ylabel("Frequency",fontsize=15)
plt.show()
```

```
<Figure size 360000x720 with 0 Axes>
```

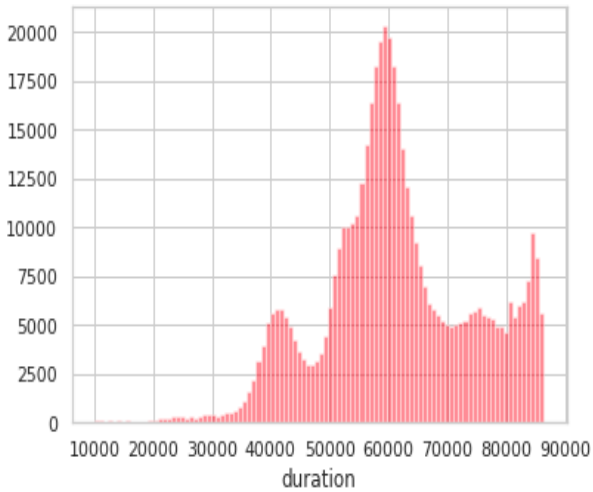


The duration column which is shown is in **seconds.**

- Deleting rows with duration below 10000 seconds becuse it may lead to wrong results as continuity is disturbed.

In [19]:
```
# set the background style of the plot
sns.set_style('whitegrid')
sns.distplot(df['duration'], kde = False, color ='red', bins = 100)
```

Out[19]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1d7af4a358>
```



- Finally training our data using

Random Forest

First using number of estimators =100 which is taking unkowing time reduced to 1 which is taking hardly minute but prediction very low with **~58%**

Then used n=10 which is giving prediction **~90%** the prediction is done on the training set only.

With the last mode of n=10 predictted in test data rms score in kaggle:

9606.61923

- The test data is processed and modified to data as the train data in order to have fit in model.

Future Scope

As we can see that our previous applied models have a very high RMSE . So these models can not be used where error is propagating after each prediction for a single datapoint.

So we think that better models like neural network can help to get better RMSE as it updates its parameters after each iteration.

Links:

model name: “model.pkl”(random forest n=1):

https://iiitborg-my.sharepoint.com/:u:/g/personal/subarnakanti_samanta_iiitb_org/Ed2F6G5-OClJg39suQNBXooBxEiFfh3wG56r_DnjzYbA1A?e=k1hRig

model name: “model2.pkl”(random forest n=10):

https://iiitborg-my.sharepoint.com/:u:/g/personal/subarnakanti_samanta_iiitb_org/Edg_OOU9ZYBHqSGk6xMMtF8B3RUiYUB2FE-2a9v9GBroTA?e=Tkm6dP

model name: “model0.pkl”(random forest n=10, busid == categorical):

model name: “linear.pkl”
https://iiitborg-my.sharepoint.com/:u:/g/personal/subarnakanti_samanta_iiitb_org/Ee5CXvOB741Ej5YT7dnb0I0BGxsyobFcwhsJf_BjGTUBKA?e=VNSTts