In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [2]:
```
df_initial =pd.read_csv('train_revised.csv', low_memory=False)
```

In [3]:
```
df_initial.head()
```

Out[3]:

	ride_id	seat_number	payment_method	payment_receipt	travel_date	travel_time	travel_from	travel_to	car_
0	1442	15A	Mpesa	UZUEHCBUSO	17-10-17	7:15	Migori	Nairobi	Bus
1	5437	14A	Mpesa	TIHLBUSGTE	19-11-17	7:12	Migori	Nairobi	Bus
2	5710	8B	Mpesa	EQX8QSQ19O	26-11-17	7:05	Keroka	Nairobi	Bus
3	5777	19A	Mpesa	SGP18CL0ME	27-11-17	7:10	Homa Bay	Nairobi	Bus
4	5778	11A	Mpesa	8M97HFRGL9	27-11-17	7:12	Migori	Nairobi	Bus

Create the target variables

In [4]:
```
ride_id_dict = {}
for ride_id in df_initial["ride_id"]:
    if not ride_id in ride_id_dict:
        ride_id_dict[ride_id] = 1
    else:
        ride_id_dict[ride_id] += 1
```

In [5]:
```
df_new = df_initial.drop(['seat_number', 'payment_method', 'payment_receipt', 'travel_to'], axis=1)
```

Drop duplicates

In [6]:
```
df_new.drop_duplicates(inplace=True)
df_new.reset_index(drop= True, inplace=True)
```

In [7]:
```
df_new["number_of_tickets"]= np.zeros(len(df_new))
```

In [8]:
```
for i in range(len(df_new)):
    ride_id = df_new.loc[i]["ride_id"]
    df_new.at[i,"number_of_tickets"] = ride_id_dict[ride_id]
```

In [9]:
```
df_new.head()
```

Out[9]:

	ride_id	travel_date	travel_time	travel_from	car_type	max_capacity	number_of_tickets
0	1442	17-10-17	7:15	Migori	Bus	49	1.0
1	5437	19-11-17	7:12	Migori	Bus	49	1.0
2	5710	26-11-17	7:05	Keroka	Bus	49	1.0
3	5777	27-11-17	7:10	Homa Bay	Bus	49	5.0
4	5778	27-11-17	7:12	Migori	Bus	49	31.0

In [10]:
```
df_new.to_csv('train_aggregated.csv', index=False)
```

In [11]:
```
df_train_set = pd.read_csv('train_aggregated.csv', low_memory=False)
```

In [12]:
```
df_train_set.drop(['ride_id'], axis=1, inplace=True)
```

Pre-process Travel_Date

In [13]:
```
df_train_set["travel_date"] = pd.to_datetime(df_train_set["travel_date"],infer_datetime_format=True)
df_train_set["days of week"] = df_train_set["travel_date"].dt.dayofweek #change the full date to day of week
df_train_set["year"] = df_train_set["travel_date"].dt.year
df_train_set["minute"] = df_train_set["travel_date"].dt.month
```

In [14]:
```
df_train_set.head()
```

Out[14]:

	travel_date	travel_time	travel_from	car_type	max_capacity	number_of_tickets	days of week	year	minute
0	2017-10-17	7:15	Migori	Bus	49	1.0	1	2017	10
1	2017-11-19	7:12	Migori	Bus	49	1.0	6	2017	11
2	2017-11-26	7:05	Keroka	Bus	49	1.0	6	2017	11
3	2017-11-27	7:10	Homa Bay	Bus	49	5.0	0	2017	11
4	2017-11-27	7:12	Migori	Bus	49	31.0	0	2017	11

In [15]:
```
xternaldata = [df_train_set]
d = {'Kisii':305.5, 'Migori':371.4, 'Rodi':349.5, 'Mbita':393.0, 'Keumbu':294.6, 'Kehancha':378.0,
     'Nyachenge':324.5, 'Keroka':280.4, 'Homa Bay':377.4,
     'Nongo':330.9, 'Kisauni':276, 'Oyugis':331.7, 'Awendo':349.0, 'Ndhiwa':370.2, 'Sirare':392.0,
     'Kendu Bay':345.1, 'Sori':275}
for dataset in xternaldata:
    dataset['avg_distance'] = dataset['travel_from'].map(d)
```

Pre-process Car_Type

In [16]:
```
df_train_set["car_type"] = pd.Categorical(df_train_set["car_type"])
car_type_categories = df_train_set.car_type.cat.categories
df_train_set["car_type"] = df_train_set.car_type.cat.codes
```

In [17]:
```
df_train_set["car_type"] = df_train_set["car_type"].replace(to_replace=0, value= 49)
```

In [18]:
```
df_train_set["car_type"] = df_train_set["car_type"].replace(to_replace=1, value= 11)
```

In [19]:
```
df_train_set.head()
```

Out[19]:

	travel_date	travel_time	travel_from	car_type	max_capacity	number_of_tickets	days of week	year	minute	avg_dis
0	2017-10-17	7:15	Migori	49	49	1.0	1	2017	10	371.4
1	2017-11-19	7:12	Migori	49	49	1.0	6	2017	11	371.4
2	2017-11-26	7:05	Keroka	49	49	1.0	6	2017	11	280.4
3	2017-11-27	7:10	Homa Bay	49	49	5.0	0	2017	11	377.4
4	2017-11-27	7:12	Migori	49	49	31.0	0	2017	11	371.4

Pre-Process Travel_from

Convert to categorical variables

In [20]:
```
df_train_set["travel_from"] = pd.Categorical(df_train_set["travel_from"])
travel_from_categories = df_train_set.travel_from.cat.categories
df_train_set["travel_from"] = df_train_set.travel_from.cat.codes
```

In [21]:
```
df_train_set.head()
```

Out[21]:

	travel_date	travel_time	travel_from	car_type	max_capacity	number_of_tickets	days of week	year	minute	avg_dis
0	2017-10-17	7:15	9	49	49	1.0	1	2017	10	371.4
1	2017-11-19	7:12	9	49	49	1.0	6	2017	11	371.4
2	2017-11-26	7:05	4	49	49	1.0	6	2017	11	280.4
3	2017-11-27	7:10	1	49	49	5.0	0	2017	11	377.4
4	2017-11-27	7:12	9	49	49	31.0	0	2017	11	371.4

Pre-Process Travel_time

Convert Travel_time to minutes

In [22]:
```
df_train_set["travel_time"] = df_train_set["travel_time"].str.split(':').apply(lambda x: int(x[0]) *
 60 + int(x[1]))
```

In [23]:
```
df_train_set.head()
```

Out[23]:

	travel_date	travel_time	travel_from	car_type	max_capacity	number_of_tickets	days of week	year	minute	avg_dis
0	2017-10-17	435	9	49	49	1.0	1	2017	10	371.4
1	2017-11-19	432	9	49	49	1.0	6	2017	11	371.4
2	2017-11-26	425	4	49	49	1.0	6	2017	11	280.4
3	2017-11-27	430	1	49	49	5.0	0	2017	11	377.4
4	2017-11-27	432	9	49	49	31.0	0	2017	11	371.4

In [24]:
```
df_train_set.drop('travel_date', axis=1, inplace=True)
```

Train Dataset

In [25]:
```
Xtrain = df_train_set.drop(["number_of_tickets"], axis=1)
Ytrain = df_train_set.number_of_tickets
```

In [26]:
```
from sklearn.ensemble import RandomForestRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_absolute_error
```

In [27]:
```
#tree = DecisionTreeRegressor(max_depth=3, random_state=0) #max_depth=3, random_state=0, parameter t
uning
#tree.fit(Xtrain, Ytrain)
```

In [28]:
```
#predict_train_set = tree.predict(Xtrain)
```

In [29]:
```
#print (mean_absolute_error(predict_train_set,Ytrain))
```

Random Forest

In [30]:
```
#model = RandomForestRegressor(n_estimators=200, criterion="mae", n_jobs=-1)
```

In [31]:
```
model= RandomForestRegressor(bootstrap=True, criterion='mae', n_jobs=-1,random_state=10,n_estimators
=500,max_features=0.8, verbose=1, min_samples_leaf=9,max_depth=10, min_samples_split=45, oob_score=T
rue)
```

In [32]:
```
#model = RandomForestRegressor(n_estimators = 200, oob_score =True,criterion='mae', n_jobs=-1, rando
m_state =10, max_features =0.9, min_samples_leaf =1, max_depth=32, verbose= 1, min_samples_split=10)
```

In [33]:
```
#rfr = RandomForestRegressor(verbose=1,max_features=0.8,random_state=7,min_samples_leaf=8,min_sample
s_split=45, criterion='mae',n_estimators=100,bootstrap=True, n_jobs=-1,max_depth = 10,oob_score =Tru
e)
```

In [34]:
```
model.fit(Xtrain,Ytrain)
```

```
[Parallel(n_jobs=-1)]: Using backend ThreadingBackend with 4 concurrent workers.
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:    8.5s
[Parallel(n_jobs=-1)]: Done 192 tasks      | elapsed:   39.3s
[Parallel(n_jobs=-1)]: Done 442 tasks      | elapsed:  1.5min
[Parallel(n_jobs=-1)]: Done 500 out of 500 | elapsed:  1.7min finished
```

Out[34]:
```
RandomForestRegressor(bootstrap=True, criterion='mae', max_depth=10,
           max_features=0.8, max_leaf_nodes=None,
           min_impurity_decrease=0.0, min_impurity_split=None,
           min_samples_leaf=8, min_samples_split=45,
           min_weight_fraction_leaf=0.0, n_estimators=500, n_jobs=-1,
           oob_score=True, random_state=10, verbose=1, warm_start=False)
```

In [35]:
```
predictt_train_set = model.predict(Xtrain)
```

```
[Parallel(n_jobs=4)]: Using backend ThreadingBackend with 4 concurrent workers.
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    0.0s
[Parallel(n_jobs=4)]: Done 192 tasks      | elapsed:    0.1s
[Parallel(n_jobs=4)]: Done 442 tasks      | elapsed:    0.2s
[Parallel(n_jobs=4)]: Done 500 out of 500 | elapsed:    0.2s finished
```

In [36]:
```
print (mean_absolute_error(predictt_train_set,Ytrain))
```

```
3.2380813329817277
```

Test Set Pre-processing

Import Datasets

In [37]:
```
test_set = pd.read_csv('test_questions.csv', low_memory=False)
```

In [38]:
```
xternaldata = [df_test_set]
d = {'Kisii':305.5, 'Migori':371.4, 'Rodi':349.5, 'Mbita':393.0, 'Keumbu':294.6, 'Kehancha':378.0,
     'Nyachenge':324.5, 'Keroka':280.4, 'Homa Bay':377.4,
     'Nongo':330.9, 'Kisauni':276, 'Oyugis':331.7, 'Awendo':349.0, 'Ndhiwa':370.2, 'Sirare':392.0,
     'Kendu Bay':345.1, 'Sori':275}
for dataset in xternaldata:
    dataset['avg_distance'] = dataset['travel_from'].map(d)
```

travel_to is not important, so drop

In [40]:
```
test_set.drop(['travel_to'], axis=1, inplace=True)
```

Pre-process Travel_Date

In [41]:
```
test_set["travel_date"] = pd.to_datetime(df_test_set["travel_date"],infer_datetime_format=True)
test_set["days of week"] = test_set["travel_date"].dt.dayofweek #change the full date to day of week
test_set["year"] = test_set["travel_date"].dt.year
test_set["minute"] = test_set["travel_date"].dt.month
```

#Pre-process Car_type

In [42]:
```
test_set["car_type"] = pd.Categorical(test_set["car_type"])
car_type_categories = test_set.car_type.cat.categories
test_set["car_type"] = test_set.car_type.cat.codes
test_set["car_type"] = test_set["car_type"].replace(to_replace=0, value= 49)
test_set["car_type"] = test_set["car_type"].replace(to_replace=1, value= 11)
```

#Pre-process travel_from

In [43]:
```
test_set["travel_from"] = pd.Categorical(test_set["travel_from"], categories=travel_from_categories)
test_set["travel_from"] = test_set.travel_from.cat.codes
```

#Pre-process travel_time

In [44]:
```
test_set["travel_time"] = test_set["travel_time"].str.split(':').apply(lambda x: int(x[0]) * 60 + in
t(x[1]))
```

In [45]:
```
test_set.drop('travel_date', axis=1, inplace=True)
```

#Predict on test

In [46]:
```
X_test = test_set.drop(['ride_id'], axis=1)
test_set_predictions = model.predict(X_test)
```

```
[Parallel(n_jobs=4)]: Using backend ThreadingBackend with 4 concurrent workers.
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    0.0s
[Parallel(n_jobs=4)]: Done 192 tasks      | elapsed:    0.0s
[Parallel(n_jobs=4)]: Done 442 tasks      | elapsed:    0.0s
[Parallel(n_jobs=4)]: Done 500 out of 500 | elapsed:    0.1s finished
```

In [47]:
```
d = {'ride_id': test_set["ride_id"], 'number_of_tickets': test_set_predictions.round()}
pred = pd.DataFrame(data=d)
pred = pred[['ride_id','number_of_tickets']]
```

In [48]:
```
pred.to_csv('Bentryfinal.csv', index=False)
```

#Gradient Boosting

In [ ]:
```
#from sklearn.ensemble import GradientBoostingRegressor
#gbr=GradientBoostingRegressor(learning_rate=0.3, max_depth=32, loss='lad',min_samples_split=50,min_
samples_leaf=50, n_estimators=100)
#gbr.fit(Xtrain, Ytrain.ravel())
```

In [ ]:
```
#gbrpred=gbr.predict(Xtrain)
```