In [1]:

```
import pandas as pd
import numpy as np
%matplotlib inline
import seaborn as sns
from datetime import datetime
import matplotlib.pyplot as plt
import os
from sklearn.preprocessing import LabelEncoder,StandardScaler, OneHotEncoder
from scipy.sparse import csr_matrix, hstack
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import log_loss
from sklearn.feature_extraction.text import TfidfTransformer,TfidfVectorizer,C
ountVectorizer
from sklearn.cluster import KMeans
from xgboost import XGBClassifier
from sklearn.calibration import CalibratedClassifierCV
import warnings
warnings.filterwarnings("ignore")
```

In [2]:

```
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation,BatchNormalization
from keras.wrappers.scikit_learn import KerasClassifier
from keras import models
from keras.models import load_model
from keras.utils import np_utils
from keras.optimizers import SGD,Adagrad
from keras.layers.advanced_activations import PReLU
from sklearn.model_selection import train_test_split
from keras.callbacks import EarlyStopping,TensorBoard
from statistics import mean
```

```
Using TensorFlow backend.
```

In [3]:

```
gatest = pd.read_csv("gender_age_test.csv",index_col='device_id')
phone=pd.read_csv("phone_brand_device_model.csv")
app_label=pd.read_csv('app_labels.csv')
label_cat=pd.read_csv("label_categories.csv")
app_events=pd.read_csv("app_events.csv", dtype={'is_active':bool})
events = pd.read_csv('events.csv',  parse_dates=['timestamp'],index_col='event
_id')
```

In [4]:

```
#removing duplicate device id's
phone = phone.drop_duplicates('device_id',keep='first').set_index('device_id')
```

In [5]:
```
print(gatest.shape)
print(phone.shape)
print(app_label.shape)
print(label_cat.shape)
print(app_events.shape)
print(events.shape)
```

```
(112071, 0)
(186716, 2)
(459943, 2)
(930, 2)
(32473067, 4)
(3252950, 4)
```

LOADING PICKLE FILES

In [6]:
```
import pickle
with open('brandencoder','rb') as fp:
    brand_encoder = pickle.load(fp)
```

In [7]:
```
import pickle
with open('modelencoder','rb') as fp:
    model_encoder = pickle.load(fp)
```

In [8]:
```
import pickle
with open('appencoder','rb') as fp:
    app_encoder = pickle.load(fp)
```

In [9]:
```
import pickle
with open('labelencoder','rb') as fp:
    label_encoder = pickle.load(fp)
```

In [10]:
```
import pickle
with open('hour_tfidf','rb') as fp:
    hour_tfidf = pickle.load(fp)
```

In [11]:
```
import pickle
with open('hour_bow','rb') as fp:
    hour_bow= pickle.load(fp)
```

In [12]:
```
import pickle
with open('hour_bin_bow','rb') as fp:
    hour_bin_bow = pickle.load(fp)
```

In [13]:
```
import pickle
with open('day_tfidf','rb') as fp:
    day_tfidf = pickle.load(fp)
```

In [14]:

```
import pickle
with open('lat_scaler','rb') as fp:
    lat_scaler = pickle.load(fp)
```

In [15]:

```
import pickle
with open('lon_scaler','rb') as fp:
    lon_scaler = pickle.load(fp)
```

In [16]:

```
import pickle
with open('clustered_features','rb') as fp:
    clustered_features = pickle.load(fp)
```

In [17]:

```
import pickle
with open('isactive_tfidf','rb') as fp:
    isactive_tfidf = pickle.load(fp)
```

In [18]:

```
import pickle
with open('class_columns','rb') as fp:
    classes = pickle.load(fp)
```

In [19]:

```
import pickle
with open('model_onehot','rb') as fp:
    model_onehot = pickle.load(fp)
```

In [20]:

```
import pickle
with open('brand_onehot','rb') as fp:
    brand_onehot = pickle.load(fp)
```

In [21]:

```
import pickle
with open('kmeans_labels','rb') as fp:
    kmeans_labels = pickle.load(fp)
```

FINAL FUNCTION

In [26]:
```
def final_function_3(gatest,phone,app_label,label_cat,app_events,events):
    start=datetime.now()
    mask=np.in1d(gatest.index,events["device_id"].values)
    gatest_events= gatest[mask]
    mask=np.in1d(gatest.index,events["device_id"].values,invert=True)
    gatest_noevents= gatest[mask]
    if(gatest_noevents.shape[0] ==1):
        gatest['testrow'] = np.arange(gatest.shape[0])
        gatest_noevents['testrow']=np.arange(gatest_noevents.shape[0])

        gatest_noevents['model']=phone['device_model']
        gatest_noevents['brand']=phone['phone_brand']

        gatest_noevents['model'] = str(phone['device_model'])
        gatest_noevents['brand'] = str(phone['phone_brand'])
        gatest_noevents_model = model_onehot.transform(gatest_noevents['model'
])
        gatest_noevents_brand= brand_onehot.transform(gatest_noevents['brand'
])
        xtest_noevents=hstack((gatest_noevents_brand, gatest_noevents_model),
format='csr')
        model_list_1=[]
        for i in range(5):
            model=load_model('nn_onehot '+str(i+1))
            model_list_1.append(model)
            avg_pred1=np.zeros((xtest_noevents.shape[0],12))
        for i in range(len(model_list_1)):
            test_pred=model_list_1[i].predict_proba(xtest_noevents)
            avg_pred1+=test_pred
        avg_pred1/=len(model_list_1)
        print(classes)
        print(avg_pred1)

    if (gatest_events.shape[0] == 1):
        gatest['testrow'] = np.arange(gatest.shape[0])
        gatest_events['testrow']=np.arange(gatest_events.shape[0])

        phone['brand'] = brand_encoder.transform(phone['phone_brand'])
        nbrand=len(brand_encoder.classes_)
        m=phone['phone_brand'].str.cat(phone['device_model'])
        phone['model'] = model_encoder.transform(m)
        nmodel=len(model_encoder.classes_)

        app_events['app'] = app_encoder.transform(app_events['app_id'])
        napps = len(app_encoder.classes_)
        deviceapps = (app_events.merge(events[['device_id']], how='left',left_
on='event_id',right_index=True)
                               .groupby(['device_id','app'])['app'].agg(['siz
e'])# grouping by device id and app and finding size of app
                               .merge(gatest_events[['testrow']], how='left',
left_index=True, right_index=True)#finding testrow
                               .reset_index())
        app_label = app_label.loc[app_label.app_id.isin(app_events.app_id.uniq
ue())]
        app_label['app'] = app_encoder.transform(app_label.app_id)
        app_label['label'] = label_encoder.transform(app_label.label_id)
        nlabels = len(label_encoder.classes_)
        devicelabels = (deviceapps[['device_id','app']]
                        .merge(app_label[['app','label']])
                        .groupby(['device_id','label'])['app'].agg(['size'])
                        .merge(gatest_events[['testrow']], how='left', left_in
dex=True, right_index=True)
                        .reset_index())
        events['hour'] = events['timestamp'].map(lambda x:pd.to_datetime(x).ho
ur)
        events['hourbin'] = [1 if ((x>=1)&(x<=6)) else 2 if ((x>=7)&(x<=12)) e
lse 3 if ((x>=13)&(x<=18)) else 4 for x in events['hour']]
        events.hour=events.hour.astype(str)
        events.hourbin=events.hourbin.astype(str)
        hourjoin = events.groupby("device_id")["hour"].apply(lambda x: " ".joi
n('0'+str(s) for s in x))
        hourbinjoin=events.groupby("device_id")["hourbin"].apply(lambda x: " "
.join('0'+str(s) for s in x))
        daysjoin=events['timestamp'].dt.day_name()
        events['day']=daysjoin.map({'Sunday':0,'Monday':1,'Tuesday':2,'Wednesd
ay':3,'Thursday':4,'Friday':5,'Saturday':6})
        daysjoin = events.groupby("device_id")["day"].apply(lambda x: " ".join
("0"+str(s) for s in x))
        median_lat = events.groupby("device_id")["latitude"].agg('median')
        median_lon=events.groupby("device_id")["longitude"].agg('median')
        com=pd.concat([median_lat, median_lon], axis=1)
        clustered_geo_features=pd.Series(kmeans_labels)
        clustered_geo_features.index=median_lon.index
        apps = app_events.groupby("event_id")["is_active"].apply(lambda x: " "
.join(str(s) for s in x))
        events["apps_active"] = events.index.map(apps)
        active_apps_events = events.groupby("device_id")["apps_active"].apply(
lambda x: " ".join(str(s) for s in x if str(s)!='nan'))
        gatest_events['brand']=phone['brand']
        Xte_events_brand = csr_matrix((np.ones(gatest_events.shape[0]), # Numb
er of Rows/Devices
                               (gatest_events.testrow, gatest_events.brand)),s
hape=(gatest_events.shape[0],nbrand))
        gatest_events['model']=phone['model']
        Xte_events_model = csr_matrix((np.ones(gatest_events.shape[0]),
                               (gatest_events.testrow, gatest_events.model)),s
hape=(gatest_events.shape[0],nmodel))
        d = deviceapps.dropna(subset=['testrow'])
        Xte_events_app = csr_matrix((np.ones(d.shape[0]), (d.testrow, d.app)),
                              shape=(gatest_events.shape[0],napps))
        d = devicelabels.dropna(subset=['testrow'])
        Xte_events_labels = csr_matrix((np.ones(d.shape[0]), (d.testrow, d.lab
el)),
                              shape=(gatest_events.shape[0],nlabels))
        gatest_events["hourjoin"]=gatest_events.index.map(hourjoin)
        X_te_hourjoin_tfidf = hour_tfidf.transform(gatest_events['hourjoin'].v
alues)
        gatest_events["hourbinjoin"]=gatest_events.index.map(hourbinjoin)
        X_te_hourbinjoin_onehot = hour_bin_bow.transform(gatest_events['hourbi
njoin'].values)
        gatest_events["daysjoin"]=gatest_events.index.map(daysjoin)
        X_te_daysjoin_tfidf = day_tfidf.transform(gatest_events['daysjoin'].va
lues)
```

```
        gatest_events["latitude"]=gatest_events.index.map(median_lat)
        X_te_event_lat = lat_scaler.transform(gatest_events['latitude'].values.reshape(-1,1))
        gatest_events["longitude"]=gatest_events.index.map(median_lon)
        X_te_event_lon = lon_scaler.transform(gatest_events['longitude'].values.reshape(-1,1))
        gatest_events["locationbin"]=gatest_events.index.map(clustered_geo_features)
        X_te_clus = clustered_features.transform(gatest_events['locationbin'].values.reshape(-1,1))
        gatest_events['apps_active']=gatest_events.index.map(active_apps_events)
        X_te_active = isactive_tfidf.transform(gatest_events['apps_active'].values)

        X_test_events =hstack((Xte_events_brand,Xte_events_model,Xte_events_labels,X_te_hourjoin_tfidf,X_te_hourbinjoin_onehot,X_te_daysjoin_tfidf,X_te_event_lat,X_te_event_lon,Xte_events_app,X_te_active,X_te_clus),format='csr')

        model_list_1=[]
        for i in range(5):
            model=load_model('nn1'+str(i+1))
            model_list_1.append(model)
        avg_pred2=np.zeros((X_test_events.shape[0],12))
        for i in range(len(model_list_1)):
            test_pred=model_list_1[i].predict_proba(X_test_events)
            avg_pred2+=test_pred
        avg_pred2/=len(model_list_1)
        model_list_1=[]
        for i in range(5):
            model=load_model('nn2'+str(i+1))
            model_list_1.append(model)
        avg_pred3=np.zeros((X_test_events.shape[0],12))
        for i in range(len(model_list_1)):
            test_pred=model_list_1[i].predict_proba(X_test_events)
            avg_pred3+=test_pred
        avg_pred3/=len(model_list_1)
        test2=(0.5*avg_pred2)+(0.5*avg_pred3)
        print(classes)
        print(test2)

    if(gatest_events.shape[0] == gatest_noevents.shape[0]):
        print('device id is not present')
```

In [24]:

```
final_function_3(gatest[4:5],phone,app_label,label_cat,app_events,events)
```

```
['F23-' 'F24-26' 'F27-28' 'F29-32' 'F33-42' 'F43+' 'M22-' 'M23-26'
 'M27-28' 'M29-31' 'M32-38' 'M39+']
[[2.0000000e-01 0.0000000e+00 0.0000000e+00 2.0000000e-01 0.0000000e+00
  9.8520795e-17 2.0000000e-01 2.0000000e-01 0.0000000e+00 0.0000000e+00
  2.0000000e-01 0.0000000e+00]]
```

In [29]:
```
final_function_3(gatest[0:1],phone,app_label,label_cat,app_events,events)
```

```
['F23-' 'F24-26' 'F27-28' 'F29-32' 'F33-42' 'F43+' 'M22-' 'M23-26'
 'M27-28' 'M29-31' 'M32-38' 'M39+']
[[3.92903631e-04 1.07207675e-03 2.00781806e-03 7.64651578e-03
  3.69362756e-02 5.11963476e-02 3.27780007e-03 2.34491333e-02
  2.80932087e-02 9.01987899e-02 2.58072273e-01 4.97656864e-01]]
```

In [ ]: