Grab Safty Challenge

Intro

No one likes jerky rides, we’ve been helping our driver-partners understand how they drive so they can become better drivers.

By collecting GPS, gyroscope and accelerometer data from our app during Grab trips, we are able to provide our driver-partners with weekly telematics reports on their driving patterns like speeding, acceleration and braking, so they know where they can do better.

Problem Appears to Sequential Problem But good Feature Engineering will turn this problem to Structure Data Problem.

NOTE: This is only proof of concept, Actual Model will be implemented using Apache BEAM, will be update later in this Repo

Engineered Features:

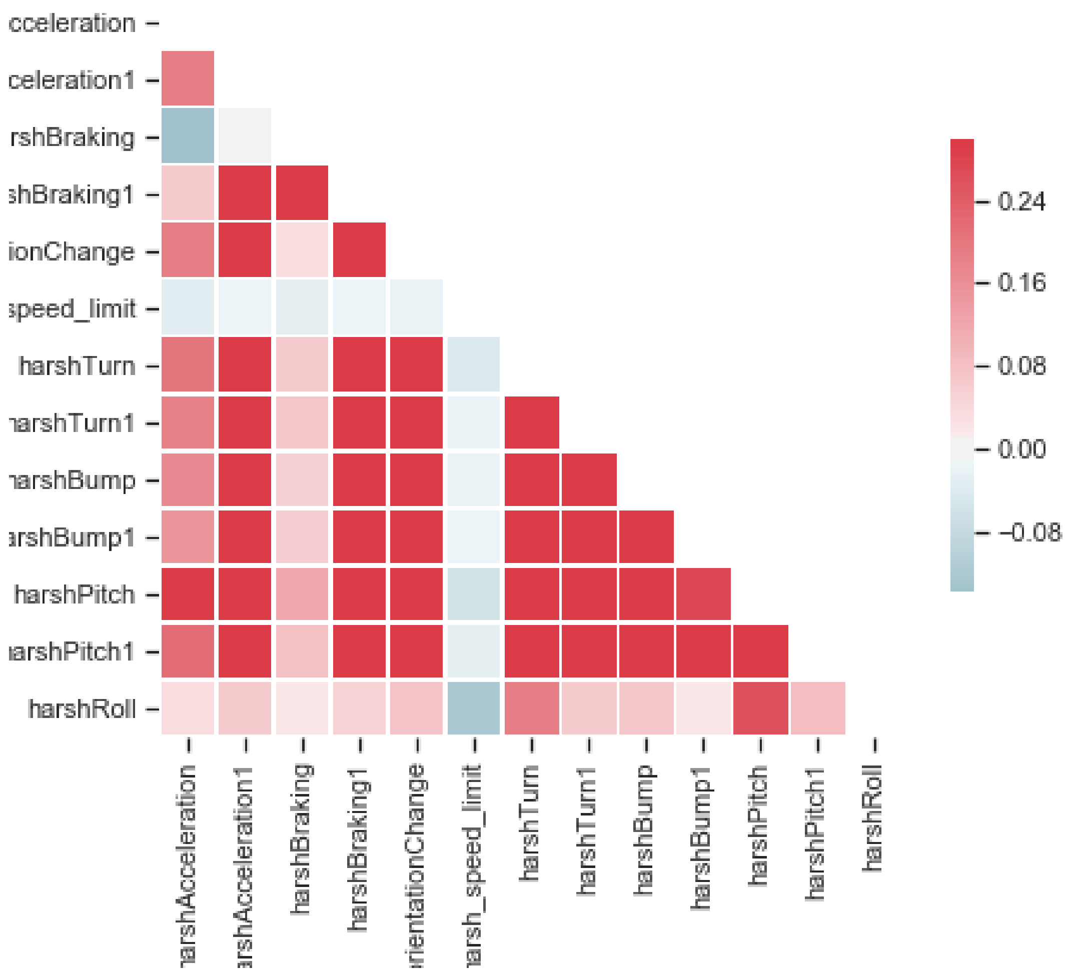
Features are decided upon primarily through domain knowledge and analysis of correlation.

Engineered Features (Datapoint):	Description:
Harsh Accleration	Harsh Accleration is Determined from acceleometer Z axis, if the accelertion is above threshold Value in m/s2.
Harsh Braking	Harsh braking is sudden Decccleration that is Determined from acceleometer Z axis, if the accelertion is below threshold Value in m/s2
Harsh Turns	Harsh Turn is Determined from acceleometer x axis, if the accelertion is above threshold Value m/s2 in Both Direction
Harsh Bumps	Harsh Bumps is Determined from acceleometer y axis, if the accelertion is above threshold Value m/s2 in Both Direction
Change in Phone Orentation	Change in Phone orentation is determined from the change in gyro values
Harsh Pitch	Harsh Pitch is determined from difference in pitch mean value of the bookingID + pitch threshold value,

pitch = atan2((- acceleration_x) , sqrt(acceleration_y *acceleration_y* + *acceleration_z* acceleration_z)) *57.3 |Harsh Roll |Harsh Roll is determined from difference in roll mean value of the bookingID + roll hreshold value, roll = atan2(y_Buff , z_Buff)* 57.3 |Harsh Speeding(Exceding Speed Limit) |Speed above Treshold Values |Distance |Distance based on Time * Average Speed, Distance is used to penalize the above value based on Distance of the BookingID

Statical Feature mean, average and std deviation were added to increase the accuracy. But didn't Help.

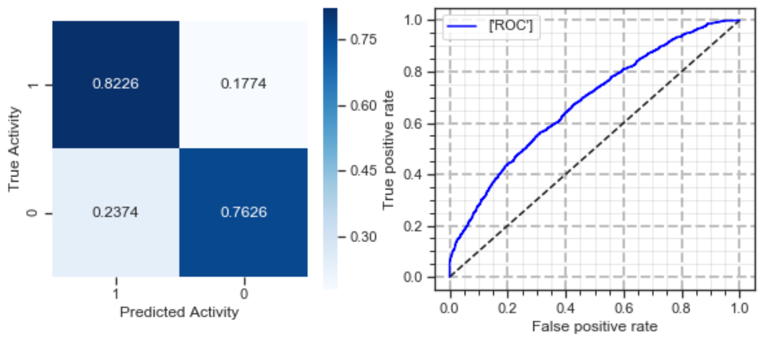
Feature Correlation Map



FeatureData Features (Datapoint):	Description:
Pitch	Correlation With Acceleration, Braking and Bumps
Roll	Turns
Braking	Don't Corelate with Acceleration and correlate with Pitch
Phone Orientation	Correlate with Pitch Roll, Acceleration, Braking and Turns

Did feature bucketing To increase the accuracy.

Final Model accuracy



Conclusion

Best model is gradient boosted decision tree, need more data to increase accuracy, try Deep Neural Network like LinearClassification, LinearDNNClassifier, Custom Estimators. Still we cannot increase the accuracy.

Even consider bucketizing the features and added second order polynomials features, still cannot increase the accuracy. I need more data to increase accuracy and prove my hypothesis.

Implemention

In [2]:
```
import pandas as pd
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import seaborn as sns
import numpy as np
import shutil
import tensorflow as tf
import glob
from sklearn.metrics import accuracy_score, confusion_matrix, roc_auc_score, r
oc_curve
%matplotlib inline
print(tf.__version__)
pd.options.display.float_format = '{:.5f}'.format
```

```
1.12.0
```

In [3]:
```
### Disable PLot.
enableGraph=False
```

Exploratory data analysis

dfRaw will contain all event types, and df will contain only events relevant for analysis, such as behavioral events (harsh turns, accelerations etc.)

To read the data from file:

In [4]:
```
def read_dataset(path_feature, path_label):
    all_files_feature = glob.glob(path_feature + "/*.csv")
    all_files_label = glob.glob(path_label + "/*.csv")
    li=[]
    for filename in all_files_feature:
        df = pd.read_csv(filename, index_col=None, header=0)
        li.append(df)
    df_features = pd.concat(li, axis=0, ignore_index=True)
    li=[]
    for filename in all_files_label:
        df = pd.read_csv(filename, index_col=None, header=0)
        li.append(df)
    df_labels = pd.concat(li, axis=0, ignore_index=True)
    dfRaw= (pd.merge(df_features, df_labels, left_on='bookingID', right_on='bo
okingID', how='left'))
    return dfRaw, df_labels
dfRaw, df_labels = read_dataset(r'C:\Users\sekaranh\Documents\Python Scripts\s
afty\dataset\features', r'C:\Users\sekaranh\Documents\Python Scripts\safty\dat
aset\labels' )
```

In [5]:
```
dfRaw.head()
```

Out[5]:

	bookingID	Accuracy	Bearing	acceleration_x	acceleration_y	acceleration_z	gyro_x
0	1202590843006	3.00000	353.00000	1.22887	8.90010	3.98697	0.00822
1	274877907034	9.29300	17.00000	0.03277	8.65993	4.73730	0.02463
2	884763263056	3.00000	189.00000	1.13967	9.54597	1.95133	-0.00690
3	1073741824054	3.90000	126.00000	3.87154	10.38636	-0.13647	0.00134
4	1056561954943	3.90000	50.00000	-0.11288	10.55096	-1.56011	0.13057

Cleaning Dataset

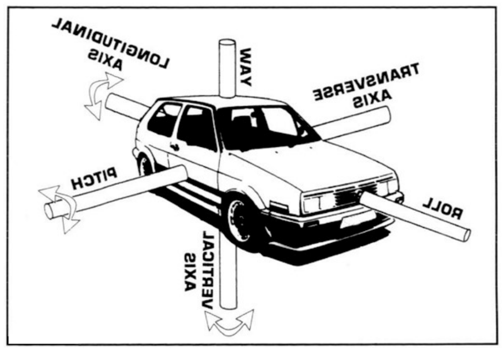
Removing NA , Clamping Speed to 200km, Seconds to 24 hours Max. Dropping off label which toggle between 1 and 0 because of uncertainty

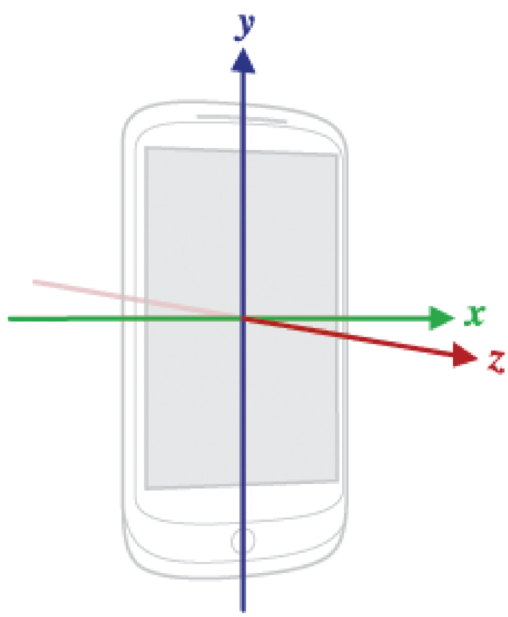
In [222]:
```
RELEVANT_EVENTS = ['bookingID','acceleration_x', 'acceleration_y', 'accelerati
on_z','gyro_x','gyro_y','gyro_z',
        'second', 'Bearing', 'Speed', 'label']

def prepData(dfPrepData, minRecordsPerSubscriber = 1):
    dfPrepData.reset_index(inplace=True)
    print("*** Starting data prep. Length:",len(dfPrepData),"***")


    #Remove NAs
    dfPrepData = dfPrepData.dropna()
    print("Removed NAs. Length:",len(dfPrepData))

    #Remove Negative Speed
    #dfPrepData['Speed']=dfPrepData.Speed.clip(0,10000)
    print("Changing Speed to abs Speed")
    #dfPrepData['Speed']=abs(dfPrepData['Speed'])
    dfPrepData.drop(dfPrepData[dfPrepData['Speed'] >  100].index, inplace=True
)
    dfPrepData.drop(dfPrepData[dfPrepData['Speed'] <  0].index, inplace=True)

    #Filter out second more then 86400 second, 24H
    dfPrepData.drop(dfPrepData[dfPrepData['second'] >  86400].index, inplace=T
rue)
    print("Filter out second more than 86400 second, 24H")

    dfPrepData.drop(dfPrepData[dfPrepData['Accuracy'] >  50].index, inplace=Tr
ue)
    print("Drop coloum accuracy less then 50")

    # Filter out unwanted events
    drop_these = list(set(list(dfPrepData)) - set(RELEVANT_EVENTS))
    df =  dfPrepData.drop(drop_these, axis = 1)
    print("Keeping only events that are relevant for modeling. Length:",len(df
))
    eventCountPerDriver = df.groupby('bookingID')['bookingID'].agg('count')
    driversWithManyRecords = eventCountPerDriver[eventCountPerDriver > minReco
rdsPerSubscriber]
    driversWithManyRecords.keys()
    df = df[df.bookingID.isin(driversWithManyRecords.keys())]
    print("Filtering users with too few samples. Length:",len(df))
    return df
df = prepData(dfRaw)
```

```
*** Starting data prep. Length: 16154418 ***
Removed NAs. Length: 16154418
Changing Speed to abs Speed
Filter out second more than 86400 second, 24H
Drop coloum accuracy less then 50
Keeping only events that are relevant for modeling. Length: 15894164
Filtering users with too few samples. Length: 15894163
```

Feature Engineering

Creating New Feature , Since raw Sensor Information is diffcult to correlate with Safty of the ride . so Creating New Feature Like Harsh Acceleration, Harsh Braking, Harsh Turns and Harsh Bumps, Ride Distance and Change in Phone Orrenataion, Hash Pitch and Hash Roll Based On research paper https://www.researchgate.net/publication/260543538_Safe_Driving_Using_Mobile_Phones (https://www.researchgate.net/publication/260543538_Safe_Driving_Using_Mobile_Phones) Pitch and Roll is determine from Acclerometer https://wiki.dfrobot.com/How_to_Use_a_Three-Axis_Accelerometer_for_Tilt_Sensing (https://wiki.dfrobot.com/How_to_Use_a_Three-Axis_Accelerometer_for_Tilt_Sensing)





pitch = atan2((- acceleration_x) , sqrt(acceleration_y *acceleration_y + acceleration_z* acceleration_z)) * 57.3;

roll = atan2(y_Buff , z_Buff) * 57.3;

In [7]:
```
## reference https://wiki.dfrobot.com/How_to_Use_a_Three-Axis_Accelerometer_for_Tilt_Sensing
#pitch = atan2((- x_Buff) , sqrt(y_Buff * y_Buff + z_Buff * z_Buff)) * 57.3;
#df['pitch'] = np.arctan2(df['acceleration_y']/np.sqrt(np.square(df['acceleration_x'])+np.square(df['acceleration_z'])))
df['f_pitch'] = np.arctan2(-df['acceleration_y'],np.sqrt(np.square(df['acceleration_z'])+np.square(df['acceleration_x'])))*57.3
#roll = atan2(y_Buff , z_Buff) * 57.3;
df['f_roll'] = np.arctan2(df['acceleration_z'],df['acceleration_x'])*57.3

df['f_gyro'] = abs(df['gyro_x']) + abs(df['gyro_y']) + abs(df['gyro_z'])
df['f_accel'] = abs(df['acceleration_x']) + abs(df['acceleration_y']) + abs(df['acceleration_z'])
df['f_bearing'] = df['Bearing']
df['f_speed'] =  df['Speed']
```

There Are scenarios in Data, where phone is lay flat, where phone don't have orientation , axis is determine from gravity and previous orientation

In [8]:
```
print("finding orentation and swaping acceleration")
m = abs(df['acceleration_y']) < abs(df['acceleration_z'])
df.loc[m, ['acceleration_y', 'acceleration_z']] = df.loc[m, ['acceleration_z', 'acceleration_y']].values
```

```
finding orentation and swaping accelromater
```

Remove Gravity Interference from the acclerometer for better isloation and adding positive values to z-axis to say car isn't in stationery

In [9]:
```
print("finding acceleration mean and offseting the values")
df['acceleration_z']= df.groupby('bookingID')['acceleration_z'].transform(lambda x: (x - x.mean()/2))#/x.std())
df['acceleration_x']= df.groupby('bookingID')['acceleration_x'].transform(lambda x: (x - x.mean()))#/x.std())
df['acceleration_y']= df.groupby('bookingID')['acceleration_y'].transform(lambda x: (x - x.mean()))#/x.std())
df['f_diffPitchMean']= df.groupby('bookingID')['f_pitch'].transform(lambda x: (x - x.mean()))
df['f_diffRollMean']= df.groupby('bookingID')['f_roll'].transform(lambda x: (x - x.mean()))
```

```
finding acceleration mean and offseting the values
```

In [10]:
```
#parameter turning values
hash_acceleration_value = 3 #m/s
hash_braking_value = -2
hash_turn_value = 2
hash_bump_value = 3
hash_pitch_value =10
hash_roll_value =10
max_speed_value = 100
```

In [11]:
```
df.head()
```

Out[11]:

	bookingID	Bearing	acceleration_x	acceleration_y	acceleration_z	gyro_x	gyro_y	
0	1202590843006	353.00000	-0.01336	-0.07823	1.98769	0.00822	0.00227	-
1	274877907034	17.00000	-0.31496	-0.32090	2.67161	0.02463	0.00403	-
2	884763263056	189.00000	0.68389	0.10154	0.63521	-0.00690	-0.01508	
3	1073741824054	126.00000	3.17059	0.62018	-0.31855	0.00134	-0.33960	-
4	1056561954943	50.00000	-0.05995	0.81488	-1.04793	0.13057	-0.06170	

In [12]:
```
# featureDerivative = {
#                         'hashAcceleration' : ['acceleration_z', hash_accelerati
on_value],
#                         'hashBraking' : ['acceleration_z', hash_braking_value],
#                         'hashTurn' : ['acceleration_x', hash_braking_value] ,
#                         'hashPitch' : ['diffPitchMean', hash_pitch_value],
#                         'hashRoll':['diffRollMean', hash_roll_value]
#                       }
# for feature_name, col_parameter in featureDerivative.items():
#     if col_parameter[1] > 0 :
#         df[feature_name] = df[col_parameter[0]].apply(lambda x: 1 if x > col
_parameter[1] else 0)
#     else:
#         df[feature_name] = df[col_parameter[0]].apply(lambda x: 1 if x < col
_parameter[1] else 0)
```

determine the Harsh Acceleraion , Harsh Braking, Harsh Turns and Harsh Bump , Have Wider Mode for Following feature. refer [https://www.researchgate.net/publication/260543538_Safe_Driving_Using_Mobile_Phones (https://www.researchgate.net/publication/260543538_Safe_Driving_Using_Mobile_Phones)](https://www.researchgate.net/publication/260543538_Safe_Driving_Using_Mobile_Phones)

In [13]:
```
def create_newFeature(df_feature):
    df_feature['speed_km/h'] = df_feature['Speed']*3.6
    df_feature['harshAcceleration'] = df_feature['acceleration_z'].apply(lambda x: 1 if (x > hash_acceleration_value and x < hash_acceleration_value+2)   else 0)
    df_feature['harshAcceleration1'] = df_feature['acceleration_z'].apply(lambda x: 1 if (x > hash_acceleration_value+2)   else 0)
    df_feature['harshBraking'] = df_feature['acceleration_z'].apply(lambda x: 1 if (x < hash_braking_value and x > hash_braking_value-2 ) else 0)
    df_feature['harshBraking1'] = df_feature['acceleration_z'].apply(lambda x: 1 if (x < hash_braking_value-2 ) else 0)
    df_feature['harsh_orientationChange'] = df_feature['f_gyro'].apply(lambda x: 1 if abs(x) > 1 else 0)
    df_feature['harsh_speed_limit'] = df_feature['speed_km/h'].apply(lambda x: 1 if x > max_speed_value else 0)
    df_feature['harshTurn'] = df_feature['acceleration_x'].apply(lambda x: 1 if (abs(x) > hash_turn_value and abs(x) < hash_turn_value+2 ) else 0)
    df_feature['harshTurn1'] = df_feature['acceleration_x'].apply(lambda x: 1 if (abs(x) > hash_turn_value+2 ) else 0)
    df_feature['harshBump'] = df_feature['acceleration_y'].apply(lambda x: 1 if (abs(x) > hash_bump_value and abs(x) < hash_bump_value+2) else 0)
    df_feature['harshBump1'] = df_feature['acceleration_y'].apply(lambda x: 1 if (abs(x) > hash_bump_value+2) else 0)
    df_feature['harshPitch'] = df_feature['f_diffPitchMean'].apply(lambda x: 1 if (abs(x) > hash_pitch_value and abs(x)< hash_pitch_value+10) else 0)
    df_feature['harshPitch1'] = df_feature['f_diffPitchMean'].apply(lambda x: 1 if (abs(x) > hash_pitch_value+10) else 0)
    df_feature['harshRoll'] = df_feature['f_diffRollMean'].apply(lambda x: 1 if (abs(x) > hash_roll_value and abs(x)< hash_roll_value+10) else 0)
    df_feature['harshRoll1'] = df_feature['f_diffRollMean'].apply(lambda x: 1 if (abs(x) > hash_pitch_value) else 0)
    return df_feature
df=create_newFeature(df)
df.head(10)
```

Out[13]:

	bookingID	Bearing	acceleration_x	acceleration_y	acceleration_z	gyro_x	gyro_y	
0	1202590843006	353.00000	-0.01336	-0.07823	1.98769	0.00822	0.00227	-
1	274877907034	17.00000	-0.31496	-0.32090	2.67161	0.02463	0.00403	-
2	884763263056	189.00000	0.68389	0.10154	0.63521	-0.00690	-0.01508	
3	1073741824054	126.00000	3.17059	0.62018	-0.31855	0.00134	-0.33960	-
4	1056561954943	50.00000	-0.05995	0.81488	-1.04793	0.13057	-0.06170	
5	1185410973787	178.00000	0.29140	0.20354	0.97906	-0.05710	-0.04355	
6	163208757379	262.18442	0.17645	0.11611	-1.21214	0.02677	-0.03069	-
7	884763262976	48.00000	-0.16815	0.12047	-0.08033	-0.00070	-0.00190	
8	841813590178	44.04170	-0.07583	0.01636	-2.68655	0.01377	-0.01708	
9	300647710810	165.00000	0.05394	-0.02938	2.09363	0.02136	0.00161	

10 rows × 34 columns

Clustering the Following Feature

In [15]:
```
def cluster_Feature(df):
    prefixes = ["harsh", "speed",'orientation']
    prefixes_f = ['f_']
    df_safty = pd.DataFrame()
    bookingIDs= df.bookingID.unique()
    df_safty['bookingID'] = bookingIDs
    for df_columns_name in df.columns :
        if df_columns_name.startswith(tuple(prefixes)):
                df_safty1 =df.groupby('bookingID')[df_columns_name].sum().rese
t_index()
                df_safty = pd.merge(df_safty,df_safty1, left_on='bookingID', r
ight_on='bookingID', how='left')
        if df_columns_name.startswith(tuple(prefixes_f)):
                df_safty1 =df.groupby('bookingID')[df_columns_name].mean().res
et_index()
                df_safty1.columns = ['bookingID', df_columns_name+"_mean"]
                df_safty = pd.merge(df_safty,df_safty1, left_on='bookingID', r
ight_on='bookingID', how='left')
                df_safty1 =df.groupby('bookingID')[df_columns_name].max().rese
t_index()
                df_safty1.columns = ['bookingID', df_columns_name+"_max"]
                df_safty = pd.merge(df_safty,df_safty1, left_on='bookingID', r
ight_on='bookingID', how='left')
                df_safty1 =df.groupby('bookingID')[df_columns_name].std().rese
t_index()
                df_safty1.columns = ['bookingID', df_columns_name+"_std"]
                df_safty = pd.merge(df_safty,df_safty1, left_on='bookingID', r
ight_on='bookingID', how='left')

    return df_safty
df_safty=cluster_Feature(df)
df_safty.head(10)
```

Out[15]:

	bookingID	f_pitch_mean	f_pitch_max	f_pitch_std	f_roll_mean	f_roll_max	f_roll_std	f_
0	1202590843006	-64.83775	-44.53816	5.42919	72.58033	107.29085	8.64316	
1	274877907034	-65.08203	-46.83193	3.02289	85.14521	121.67448	6.56035	
2	884763263056	-73.74806	-44.13417	4.21105	80.55469	155.63778	16.37695	
3	1073741824054	-82.15158	-50.66321	4.91920	19.93231	178.38816	60.72369	
4	1056561954943	-80.86812	-30.16102	5.32966	-62.94602	179.13977	79.72628	
5	1185410973787	-65.36784	-31.68921	7.29994	80.97380	167.51633	23.28477	
6	163208757379	73.69823	85.11406	3.61053	-80.04927	-21.81032	13.84089	
7	884763262976	-85.09385	-51.75023	4.08665	-32.01335	179.84508	70.58515	
8	841813590178	49.77278	64.17592	4.40196	-77.08473	-43.62290	7.43668	
9	300647710810	-53.64412	-32.85000	5.19549	81.67475	154.91335	8.06805	

10 rows × 40 columns

Looking at the absolute number of the events is wrong. Instead we'll normalize the number of events per booking based on booking Distance. Distance is calculated based on time and average speed. Since some data is missing in sequential time instead max seconds, I took seconds count.

In [16]:
```
def total_distance(oneBooking):
    return (oneBooking.second.count()* (abs(oneBooking.Speed.mean()) if oneBoo
king.Speed.mean() > 5.5 else 5.5))#
def calculate_overall_distance_travelled(dfRaw):
    dfDistancePerBooking = dfRaw.groupby('bookingID').apply(total_distance).re
set_index(name='Distance')
    return dfDistancePerBooking
distancePerBooking = calculate_overall_distance_travelled(dfRaw)
distancePerBooking.head(10)
```

Out[16]:

	bookingID	Distance
0	0	9030.80122
1	1	6707.23123
2	2	1072.50000
3	4	6729.19001
4	6	6022.50000
5	7	11555.39035
6	8	2128.50000
7	10	3263.26000
8	11	1498.28032
9	13	38121.83062

In [17]:
```
def create_feature_set(df, distancePerBooking):
    dfEventMatrix = df.merge(distancePerBooking, how = 'inner',on='bookingID')
    dfEventMatrix.set_index('bookingID', inplace=True)
    featureCols = [col for col in dfEventMatrix if col.startswith('harsh')]
    dfEventMatrix[featureCols] = dfEventMatrix[featureCols].div(dfEventMatrix[
'Distance'], axis=0)
    dfFeatureSet = dfEventMatrix[featureCols]
    return dfFeatureSet
features_h = create_feature_set(df_safty,distancePerBooking)
features_h.head(10)
```

Out[17]:

	harshAcceleration	harshAcceleration1	harshBraking	harshBraking1	harsh_orient
bookingID					
1202590843006	0.02196	0.00047	0.00000	0.00000	
274877907034	0.00313	0.00000	0.00000	0.00000	
884763263056	0.00157	0.00004	0.00000	0.00000	
1073741824054	0.00129	0.00029	0.00459	0.00043	
1056561954943	0.00021	0.00000	0.00459	0.00027	
1185410973787	0.01692	0.00224	0.00084	0.00009	
163208757379	0.00000	0.00000	0.01355	0.00000	
884763262976	0.00042	0.00000	0.00291	0.00021	
841813590178	0.00000	0.00000	0.07009	0.00235	
300647710810	0.03474	0.00017	0.00000	0.00000	

To Increase the Accuracy adding the static Model also

In [18]:
```
def create_feature_set_stats(df):
    dfEventMatrix = df.merge(distancePerBooking, how = 'inner',on='bookingID')
    dfEventMatrix.set_index('bookingID', inplace=True)
    featureCols = [col for col in dfEventMatrix if col.startswith('f_')]
    dfEventMatrix[featureCols] = dfEventMatrix[featureCols]
    dfFeatureSet = dfEventMatrix[featureCols]
    return dfFeatureSet
features_f = create_feature_set_stats(df_safty)
features_f.head(10)
```

Out[18]:

	f_pitch_mean	f_pitch_max	f_pitch_std	f_roll_mean	f_roll_max	f_roll_std	f_gyr
bookingID							
1202590843006	-64.83775	-44.53816	5.42919	72.58033	107.29085	8.64316	
274877907034	-65.08203	-46.83193	3.02289	85.14521	121.67448	6.56035	
884763263056	-73.74806	-44.13417	4.21105	80.55469	155.63778	16.37695	
1073741824054	-82.15158	-50.66321	4.91920	19.93231	178.38816	60.72369	
1056561954943	-80.86812	-30.16102	5.32966	-62.94602	179.13977	79.72628	
1185410973787	-65.36784	-31.68921	7.29994	80.97380	167.51633	23.28477	
163208757379	73.69823	85.11406	3.61053	-80.04927	-21.81032	13.84089	
884763262976	-85.09385	-51.75023	4.08665	-32.01335	179.84508	70.58515	
841813590178	49.77278	64.17592	4.40196	-77.08473	-43.62290	7.43668	
300647710810	-53.64412	-32.85000	5.19549	81.67475	154.91335	8.06805	

10 rows × 24 columns

In [19]:
```
features = pd.merge(features_h,features_f, left_on='bookingID', right_on='book
ingID', how='left')
features.head()
```

Out[19]:

	harshAcceleration	harshAcceleration1	harshBraking	harshBraking1	harsh_orient
bookingID					
1202590843006	0.02196	0.00047	0.00000	0.00000	
274877907034	0.00313	0.00000	0.00000	0.00000	
884763263056	0.00157	0.00004	0.00000	0.00000	
1073741824054	0.00129	0.00029	0.00459	0.00043	
1056561954943	0.00021	0.00000	0.00459	0.00027	

5 rows × 38 columns

In [20]:
```
# features_DNN= (pd.merge(features, df_labels, left_on='bookingID', right_on
='bookingID', how='left'))
# msk = np.random.rand(len(features_DNN)) < 0.8
# traindf = features_DNN[msk]
# evaldf = features_DNN[~msk]
# traindf.to_csv('train.csv',index=False,header=False)
# evaldf.to_csv('eval.csv' ,index=False,header=False)
```

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

Out[21]:

	harshAcceleration	harshAcceleration1	harshBraking	harshBraking1	harsh_orient
bookingID					
1202590843006	0.02196	0.00047	0.00000	0.00000	
274877907034	0.00313	0.00000	0.00000	0.00000	
884763263056	0.00157	0.00004	0.00000	0.00000	
1073741824054	0.00129	0.00029	0.00459	0.00043	
1056561954943	0.00021	0.00000	0.00459	0.00027	

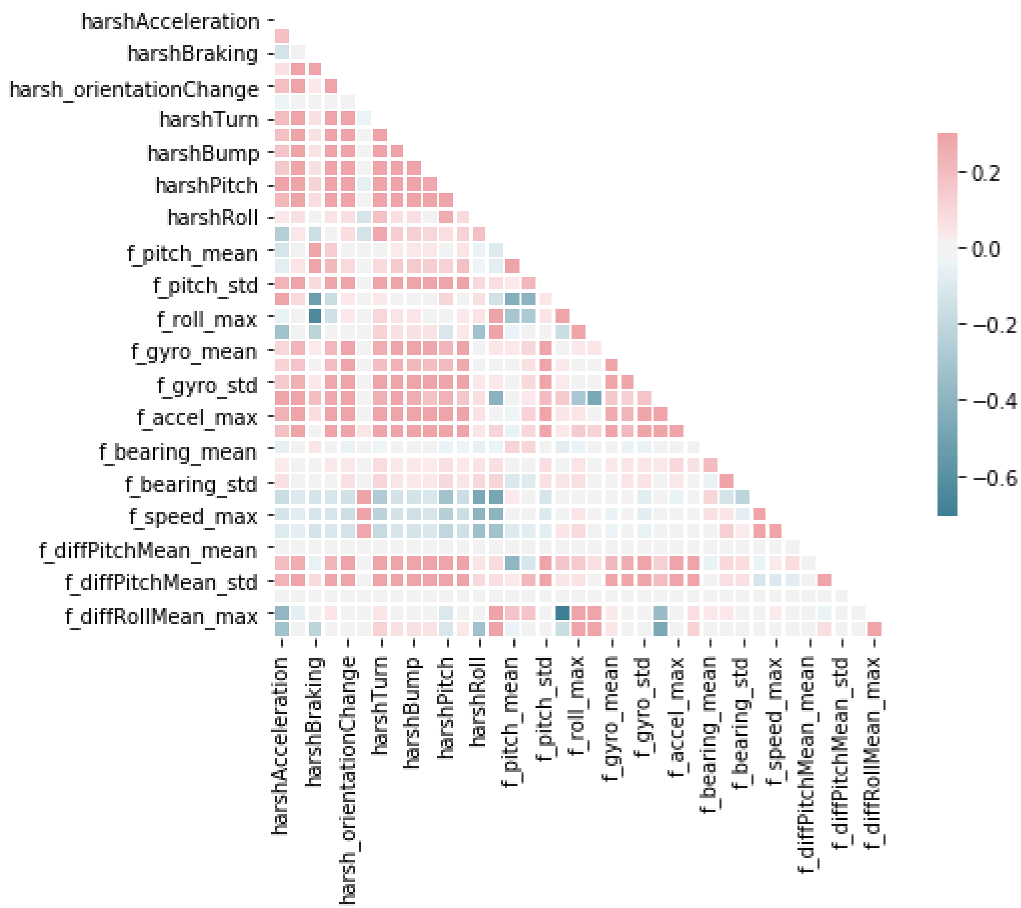
5 rows × 38 columns

Correlations Between Events Are accurate , pitch is coreleationg with Acceleration role with turn, braking and acclerion are in oppsite Direction

In [214]:
```
import seaborn as sns

def create_heat_map_plt(features):
    if True:
        corr = features.corr()

        # Generate a mask for the upper triangle
        mask = np.zeros_like(corr, dtype=np.bool)
        mask[np.triu_indices_from(mask)] = True
        fig = plt.figure()

        # Set up the matplotlib figure
        f, ax = plt.subplots(figsize=(7, 7))

        # Generate a custom diverging colormap
        cmap = sns.diverging_palette(220, 10, as_cmap=True)

        # Draw the heatmap with the mask and correct aspect ratio
        sns.heatmap(corr, mask=mask, cmap=cmap, vmax=.3, center=0,
                    square=True, linewidths=.5, cbar_kws={"shrink": .5})
        plt.savefig('heatmap')
```

In [23]:
```
create_heat_map_plt(features)
```

<Figure size 432x288 with 0 Axes>



Modeling Hypotheseis

Our Assumpation is BookingID with Less Events will drive safly and BookingID with more Events is Unsafe. Events are harsh braking, harsh acceleration, harsh turns, harsh bumps, harsh pitch and harsh roll.

In [24]:
```
import seaborn as sns
def create_scatterplot(features):
    if enableGraph:
        sns.set(style="ticks",font_scale=1.1)
        g = sns.PairGrid(features)
        g = g.map_upper(plt.scatter, edgecolor="w")
        g = g.map_lower(sns.kdeplot)
        g = g.map_diag(plt.hist, edgecolor="w")
```

All features are skewed to the right with a long tail. On the diagnoal, we see a histogram of all features. On the upper triangle we see a scatterplot of each pair of features, and on the bottom triangle we see a KDE (Kernel Density Estimation) of each pair of features

In [25]:
```
import scipy.stats as st

def transform_to_normal_boxcox(x,min_max_transform = False):
    xt = np.zeros(len(x))
    if np.count_nonzero(x) == 0:
        print("only zero valued values found")
        return x

    valueGreaterThanZero = np.where(x<=0,0,1)
    positives = x[valueGreaterThanZero == 1]
    if(len(positives)> 0):
        xt[valueGreaterThanZero == 1],_ = st.boxcox(positives+1)
    if min_max_transform:
        xt = (xt - np.min(xt)) / (np.max(xt)-np.min(xt))
    return xt

transFeatures = features.apply(lambda x: (transform_to_normal_boxcox(x,min_max
_transform = True)))
transFeatures.head()
create_scatterplot(transFeatures)
```

Outliers handling¶ We wish to remove/adjust outliers as they affect many statistical approaches. In order to remove these, we'll transform the features to normal (using a box-cox transformation) and remove based on mean + kstd's* rule. A second approach could be to truncate the tail using some constant, but it will be more difficult to find this threshold than the standard deviation rule. A third option is to remove outliers on all three dimensions (using a multivariate normal distribution, for example).

This code performs the first option, box-cox transformation:

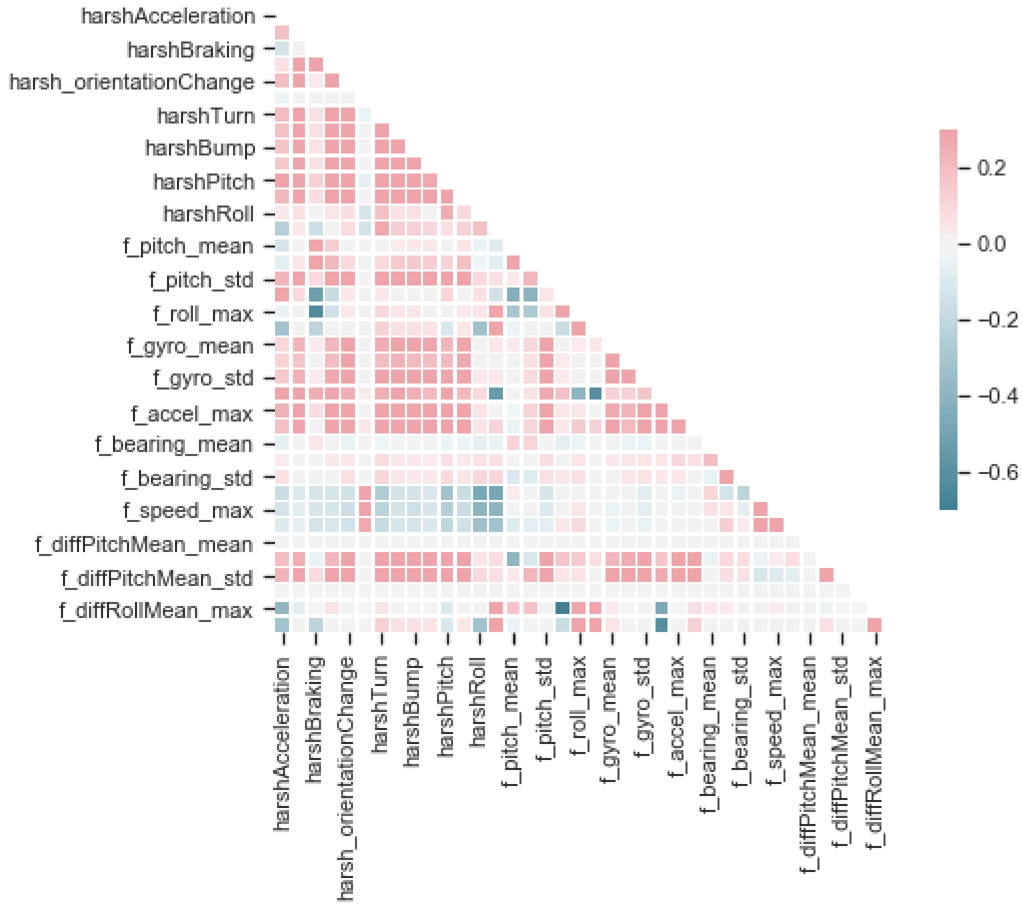
In [26]:
```
### remove outliers
import seaborn as sns
sns.set(style="ticks")

def replace_outliers_with_limit(x, stdFactor =10, normalize = False):
    print(x.name)
    x = x.values
    xt = np.zeros(len(x))
    if np.count_nonzero(x) == 0:
        print("only zero valued values found")
        return x

    xt = transform_to_normal_boxcox(x)

    xMean, xStd = np.mean(xt), np.std(xt)
    outliers = np.where(xt > xMean + stdFactor*xStd)[0]
    inliers = np.where(xt <= xMean + stdFactor*xStd)[0]
    if len(outliers) > 0:
        print("found outlier with factor: "+str(stdFactor)+" : "+str(outliers
))
        xinline = x[inliers]
        maxInRange = np.max(xinline)
        print("replacing outliers {} with max={}".format(outliers,maxInRange))
        vals = x.copy()
        vals[outliers] = maxInRange
        x= pd.Series(vals)
    else:
        print("No outliers found")
    if normalize:
        #Normalize to [0,1]
        x = (x - np.min(x)) / (np.max(x)-np.min(x))
    return x

cleanFeatures = features.apply(lambda x: (replace_outliers_with_limit(x)))
cleanFeatures.head(6)
create_scatterplot(cleanFeatures)
```

```
harshAcceleration
No outliers found
harshAcceleration
No outliers found
harshAcceleration1
No outliers found
harshBraking
No outliers found
harshBraking1
No outliers found
harsh_orientationChange
No outliers found
harsh_speed_limit
No outliers found
harshTurn
No outliers found
harshTurn1
No outliers found
harshBump
No outliers found
harshBump1
No outliers found
harshPitch
No outliers found
harshPitch1
No outliers found
harshRoll
No outliers found
harshRoll1
No outliers found
f_pitch_mean
No outliers found
f_pitch_max
No outliers found
f_pitch_std
No outliers found
f_roll_mean
No outliers found
f_roll_max
No outliers found
f_roll_std
No outliers found
f_gyro_mean
No outliers found
f_gyro_max
No outliers found
f_gyro_std
No outliers found
f_accel_mean
found outlier with factor: 10 : [ 1121  2117  2312  2510  4065  5622  7205  7
715  8062  8606  8801  9282
 12144 14030 15778 16252 17639 17876 18339]
replacing outliers [ 1121  2117  2312  2510  4065  5622  7205  7715  8062  86
06  8801  9282
 12144 14030 15778 16252 17639 17876 18339] with max=25.947339537232953
f_accel_max
No outliers found
```

```
f_accel_std
No outliers found
f_bearing_mean
No outliers found
f_bearing_max
No outliers found
f_bearing_std
No outliers found
f_speed_mean
No outliers found
f_speed_max
found outlier with factor: 10 : [8078]
replacing outliers [8078] with max=56.698967
f_speed_std
No outliers found
f_diffPitchMean_mean
No outliers found
f_diffPitchMean_max
No outliers found
f_diffPitchMean_std
No outliers found
f_diffRollMean_mean
No outliers found
f_diffRollMean_max
No outliers found
f_diffRollMean_std
No outliers found
```

In [27]:
```
create_heat_map_plt(cleanFeatures)
```

<Figure size 432x288 with 0 Axes>



In [28]:
```
## Pre step: Normalize features
minPerFeature = cleanFeatures.min()
maxPerFeature = cleanFeatures.max()

print("Min and Max values per column before normalization")
for col in range(0,len(cleanFeatures.columns)):
    print("{} range:[{},{}]".format(cleanFeatures.columns[col],minPerFeature[c
ol],maxPerFeature[col]))

normalizedFeatures = (cleanFeatures-cleanFeatures.min())/(cleanFeatures.max()-
cleanFeatures.min())
normalizedFeatures.head()


## Standardize features after box-cox as well.
transFeaturesScaled = (transFeatures - transFeatures.mean())/transFeatures.std
()
print("")
print("Mean and STD before standardization")
for col in range(0,len(transFeatures.columns)):
    print("{} range:[{},{}]".format(transFeatures.columns[col],transFeatures.m
ean()[col],transFeatures.std()[col]))


normalizedFeatures.head()
```

```
Min and Max values per column before normalization
harshAcceleration range:[0.0,0.18181818181818182]
harshAcceleration1 range:[0.0,0.17575757575757575]
harshBraking range:[0.0,0.17835497835497835]
harshBraking1 range:[0.0,0.11337579617834395]
harsh_orientationChange range:[0.0,0.1771343410687673]
harsh_speed_limit range:[0.0,0.028425504807336185]
harshTurn range:[0.0,0.1290267492723159]
harshTurn1 range:[0.0,0.12371730018788843]
harshBump range:[0.0,0.12093023255813953]
harshBump1 range:[0.0,0.18071958253227136]
harshPitch range:[0.0,0.1420213697670345]
harshPitch1 range:[0.0,0.18126888217522658]
harshRoll range:[0.0,0.11841491841491841]
harshRoll1 range:[0.0,0.18181818181818182]
f_pitch_mean range:[-87.61548492994265,87.30925356031214]
f_pitch_max range:[-80.31136341003527,89.9984816934055]
f_pitch_std range:[0.2136042774211785,81.24312603733985]
f_roll_mean range:[-155.1084827397622,165.30348447481762]
f_roll_max range:[-130.3066939592289,180.01325905069513]
f_roll_std range:[0.26218814967231896,176.2257062805007]
f_gyro_mean range:[0.004842484009621755,19.345739832304137]
f_gyro_max range:[0.014061678200000002,105.7168355]
f_gyro_std range:[0.0,7.48347992242825]
f_accel_mean range:[1.0790281931559245,25.947339537232953]
f_accel_max range:[1.1289520263671875,193.98544921874998]
f_accel_std range:[0.019663643436282272,13.810623698965541]
f_bearing_mean range:[0.0,357.66163855658635]
f_bearing_max range:[0.0,359.9994812011719]
f_bearing_std range:[0.0,176.73293066520628]
f_speed_mean range:[0.0,29.557195254734584]
f_speed_max range:[0.0,56.698967]
f_speed_std range:[0.0,13.16957955914649]
f_diffPitchMean_mean range:[-8.25044416068597e-13,5.364325416392679e-13]
f_diffPitchMean_max range:[0.5016110402368668,163.85432206941744]
f_diffPitchMean_std range:[0.21360427742117893,81.24312603733982]
f_diffRollMean_mean range:[-5.233685004541641e-13,6.480465546903563e-13]
f_diffRollMean_max range:[1.4084417925201933,334.0836624301625]
f_diffRollMean_std range:[0.2621881496723206,176.2257062805007]

Mean and STD before standardization
harshAcceleration range:[0.254861078570673,0.3096421312285254]
harshAcceleration1 range:[0.1255974056617588,0.22563929708139402]
harshBraking range:[0.25863873829266004,0.3028363887805988]
harshBraking1 range:[0.13493519820634875,0.24343557035233196]
harsh_orientationChange range:[0.13772623679823515,0.24582073121969056]
harsh_speed_limit range:[0.03215388892169278,0.1386786313968253]
harshTurn range:[0.44423077373704795,0.23466238347847268]
harshTurn1 range:[0.1332531053746124,0.22582458411070488]
harshBump range:[0.21405682836504197,0.24448781285160368]
harshBump1 range:[0.0433380891696689,0.15548181004512465]
harshPitch range:[0.4537465185792488,0.23391068728177514]
harshPitch1 range:[0.15611479175276188,0.23703862829571462]
harshRoll range:[0.4140659071979146,0.20451495749601964]
harshRoll1 range:[0.513505220520513,0.263474047409563]
f_pitch_mean range:[0.16020546725508986,0.3038531596654795]
f_pitch_max range:[0.22810781241662048,0.3947605920926835]
```

```
f_pitch_std range:[0.8742642536440363,0.04164794127472464]
f_roll_mean range:[0.20222531451105905,0.19493117744480698]
f_roll_max range:[0.6769168384980498,0.37038467988885637]
f_roll_std range:[0.5173521720037819,0.20295818418424738]
f_gyro_mean range:[0.5542983308535332,0.17152022687177856]
f_gyro_max range:[0.6772574452831828,0.14607846294550728]
f_gyro_std range:[0.6213930140953929,0.15860128603932408]
f_accel_mean range:[0.4291589020424909,0.034154790582501836]
f_accel_max range:[0.6060100410991862,0.04707320993508985]
f_accel_std range:[0.690507130862181,0.07643159669981447]
f_bearing_mean range:[0.5468161862420347,0.13270388754583798]
f_bearing_max range:[0.7684333133802017,0.3035061423530885]
f_bearing_std range:[0.44726540690611316,0.16569784982983124]
f_speed_mean range:[0.5724394703455924,0.1408208953987324]
f_speed_max range:[0.20386328160514433,0.055078980985192635]
f_speed_std range:[0.4801218901978804,0.1414987662397188]
f_diffPitchMean_mean range:[0.2025113735459646,0.2680506383926675]
f_diffPitchMean_max range:[0.6012272481543263,0.12109185238964117]
f_diffPitchMean_std range:[0.8742642536440363,0.04164794127472467]
f_diffRollMean_mean range:[0.2013002621089397,0.27544716951940146]
f_diffRollMean_max range:[0.6069095242085663,0.19285379957653387]
f_diffRollMean_std range:[0.5173521720037819,0.20295818418424738]
```

Out[28]:

	harshAcceleration	harshAcceleration1	harshBraking	harshBraking1	harsh_orient
bookingID					
1202590843006	0.12076	0.00270	0.00000	0.00000	
274877907034	0.01720	0.00000	0.00000	0.00000	
884763263056	0.00865	0.00026	0.00000	0.00000	
1073741824054	0.00709	0.00163	0.02571	0.00379	
1056561954943	0.00113	0.00000	0.02571	0.00242	

5 rows × 38 columns

In [29]:

```
normalizedFeatures= (pd.merge(normalizedFeatures, df_labels, left_on='bookingI
D', right_on='bookingID', how='left'))
temp=normalizedFeatures.set_index('bookingID')
```

In [30]:

```
x = normalizedFeatures.drop(columns=['label'])
x = x.drop(columns=['bookingID'])
y= normalizedFeatures['label']
```

In [31]:

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, random_state=0)
x_train.shape, y_train.shape, x_test.shape, y_test.shape
```

Out[31]: ((14991, 38), (14991,), (4998, 38), (4998,))

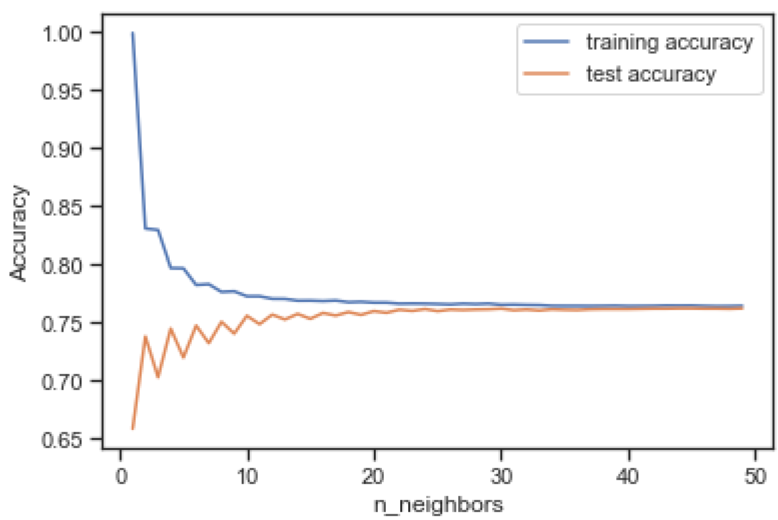
In [32]:
```
msk = np.random.rand(len(normalizedFeatures)) < 0.8
traindf = normalizedFeatures[msk]
evaldf = normalizedFeatures[~msk]
traindf.to_csv('train.csv',index=False,header=False)
evaldf.to_csv('eval.csv' ,index=False,header=False)
```

In [37]:
```
from sklearn.metrics import accuracy_score, confusion_matrix, roc_auc_score, r
oc_curve
from matplotlib import pyplot
def plot_confusion_matrix(model):
    labels_predict = model.predict(x_test)
    cm = confusion_matrix(labels_predict, y_test, labels=normalizedFeatures.la
bel.unique())
    cm_normalized = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
    fig, ax = plt.subplots(figsize=(5,4))
    sns.heatmap(cm_normalized, annot=True, fmt=".4f",
            cmap='Blues', square=True,
            xticklabels=normalizedFeatures.label.unique(),
            yticklabels=normalizedFeatures.label.unique())
    ax.set_xlabel('Predicted Activity')
    ax.set_ylabel('True Activity', )
    plt.tight_layout()
    ac = accuracy_score(y_test, labels_predict)
    rc = roc_auc_score(y_test, labels_predict)
    print("\nAccuracy {0} ROC {1}".format(ac, rc))
    #plt.savefig('confusionmatrix.pdf')
```

In [ ]:
```
from sklearn.metrics import accuracy_score, confusion_matrix, roc_auc_score, r
oc_curve
from matplotlib import pyplot
def plot_acc_curve(model):
    labels_predict = model.predict(x_test)
    cm = confusion_matrix(labels_predict, y_test, labels=normalizedFeatures.la
bel.unique())
    cm_normalized = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
    fpr, tpr, __ = roc_curve(y_test,labels_predict)
    fig= plt.figure(figsize=(15,8))
    ax1=fig.add_subplot(1,3,1)
    sns.heatmap(cm_normalized, annot=True, fmt=".4f",
            cmap='Blues', square=True,
            xticklabels=normalizedFeatures.label.unique(),
            yticklabels=normalizedFeatures.label.unique())
    ax1.set_xlabel('Predicted Activity')
    ax1.set_ylabel('True Activity', )
    ax2=fig.add_subplot(1,3,2)
    ax2.plot([0, 1], [0, 1], 'k--')
    ax2.plot(fpr,tpr ,label=['ROC'],color='blue')
    ax2.grid(True, lw = 2, ls = '--', c = '.75')
    ax2.minorticks_on()
    ax2.grid(b=True, which='minor', color='#999999', linestyle='-', alpha=0.2)
    ax2.set_xlabel('False positive rate')
    ax2.set_ylabel('True positive rate')
    #ax2.title('Test ROC evaluation')
    ax2.legend(loc='best')
    plt.show()
   # plt.tight_layout()
```

In [89]:
```
def plot_feature_importances(model):
    plt.figure(figsize=(8,6))
    n_features = 50
    plt.barh(range(n_features), model.feature_importances_, align='center')
    plt.yticks(np.arange(n_features), cleanFeatures)
    plt.xlabel("Feature importance")
    plt.ylabel("Feature")
    plt.ylim(-1, n_features)
```

In [64]:
```
from sklearn.neighbors import KNeighborsClassifier

training_accuracy = []
test_accuracy = []
# try n_neighbors from 1 to 10
neighbors_settings = range(1, 50)

for n_neighbors in neighbors_settings:
    # build the model
    knn = KNeighborsClassifier(n_neighbors=n_neighbors)
    knn.fit(x_train, y_train)
    # record training set accuracy
    training_accuracy.append(knn.score(x_train, y_train))
    # record test set accuracy
    test_accuracy.append(knn.score(x_test, y_test))

plt.plot(neighbors_settings, training_accuracy, label="training accuracy")
plt.plot(neighbors_settings, test_accuracy, label="test accuracy")
plt.ylabel("Accuracy")
plt.xlabel("n_neighbors")
plt.legend()
plt.savefig('knn_compare_model')
```

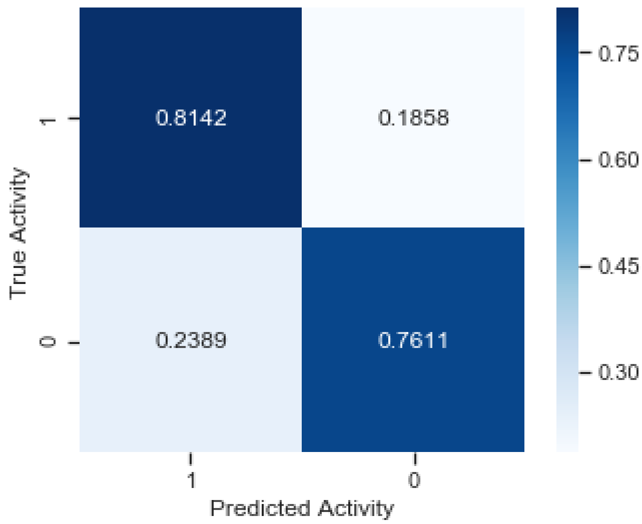


In [65]:
```
knn = KNeighborsClassifier(n_neighbors=51)
knn.fit(x_train, y_train)

print('Accuracy of K-NN classifier on training set: {:.2f}'.format(knn.score(x
_train, y_train)))
print('Accuracy of K-NN classifier on test set: {:.2f}'.format(knn.score(x_tes
t, y_test)))
y_pred = knn.predict(x_test)
```

```
Accuracy of K-NN classifier on training set: 0.76
Accuracy of K-NN classifier on test set: 0.76
```

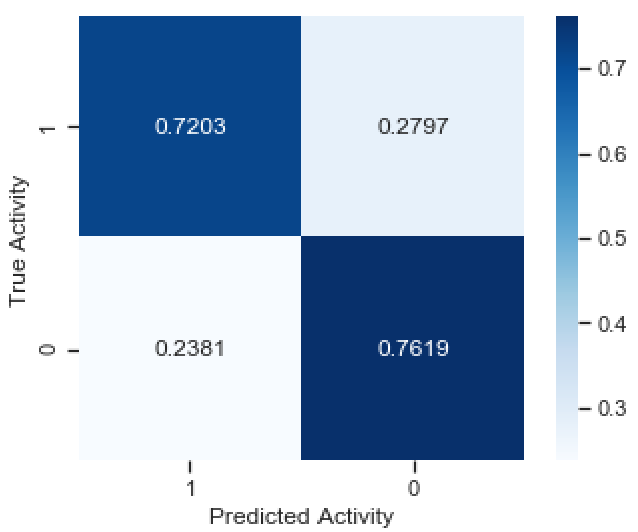
In [74]:
```
plot_confusion_matrix(knn)
```

```
Accuracy 0.7623049219687875 ROC 0.5337286965780056
```



In [67]:
```
from sklearn.linear_model import LogisticRegression

logreg = LogisticRegression().fit(x_train, y_train)
print("Training set accuracy: {:.3f}".format(logreg.score(x_train, y_train)))
print("Test set accuracy: {:.3f}".format(logreg.score(x_test, y_test)))
plot_confusion_matrix(logreg)
```

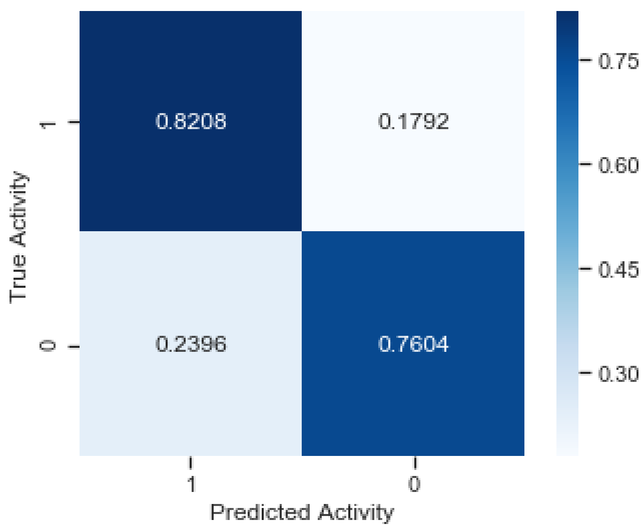
```
C:\Users\sekaranh\AppData\Local\Continuum\anaconda3\lib\site-packages\sklearn
\linear_model\logistic.py:433: FutureWarning: Default solver will be changed
to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)

Training set accuracy: 0.763
Test set accuracy: 0.761

Accuracy 0.7607042817126851 ROC 0.5355564567369553
```



In [68]:

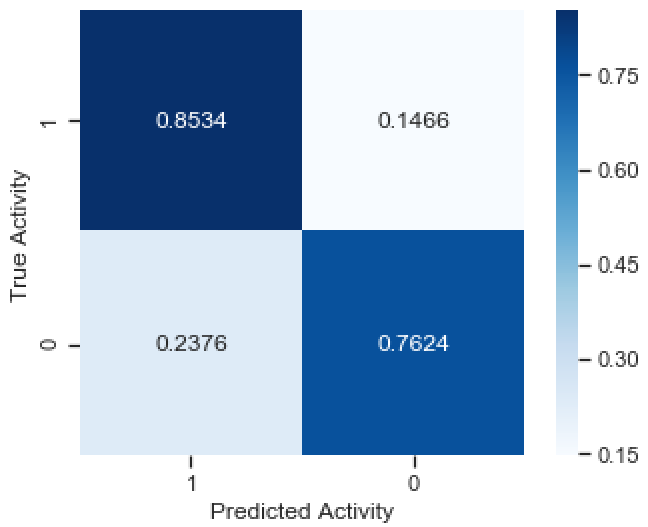
```
from sklearn.tree import DecisionTreeClassifier

tree = tree = DecisionTreeClassifier(max_depth=5, random_state=0)
tree.fit(x_train, y_train)
print("Accuracy on training set: {:.3f}".format(tree.score(x_train, y_train)))
print("Accuracy on test set: {:.3f}".format(tree.score(x_test, y_test)))
plot_confusion_matrix(tree)
```

```
Accuracy on training set: 0.768
Accuracy on test set: 0.762

Accuracy 0.7617046818727491 ROC 0.5320104448293229
```



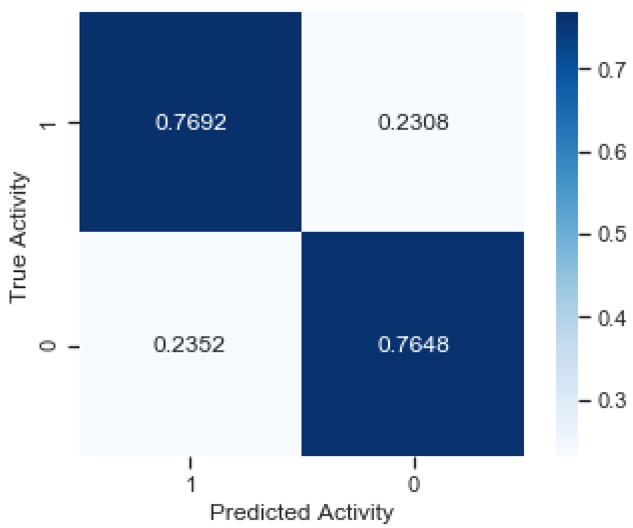
In [92]:
```
rf1 = RandomForestClassifier(max_depth=11, n_estimators=1000, random_state=0)
rf1.fit(x_train, y_train)
print("Accuracy on training set: {:.3f}".format(rf1.score(x_train, y_train)))
print("Accuracy on test set: {:.3f}".format(rf1.score(x_test, y_test)))
plot_confusion_matrix(rf1)
```

```
Accuracy on training set: 0.790
Accuracy on test set: 0.765

Accuracy 0.7645058023209284 ROC 0.5370435830726976
```



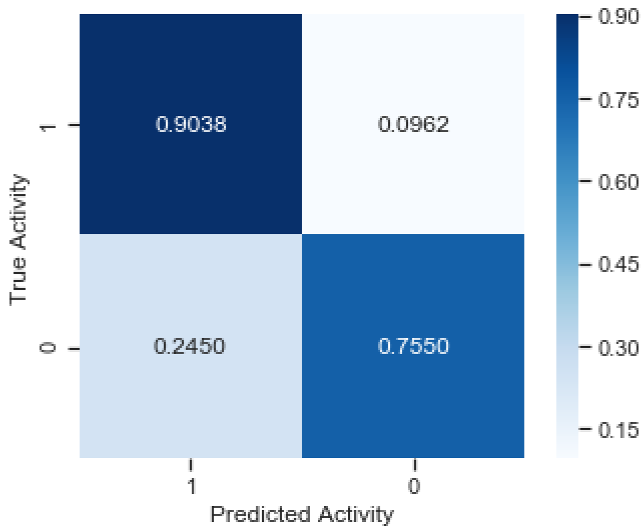
In [71]:
```
from sklearn.ensemble import GradientBoostingClassifier

gb = GradientBoostingClassifier(random_state=0)
gb.fit(x_train, y_train)

print("Accuracy on training set: {:.3f}".format(gb.score(x_train, y_train)))
print("Accuracy on test set: {:.3f}".format(gb.score(x_test, y_test)))
plot_confusion_matrix(gb)
```

```
Accuracy on training set: 0.777
Accuracy on test set: 0.765

Accuracy 0.764905962384954 ROC 0.5428427491093281
```



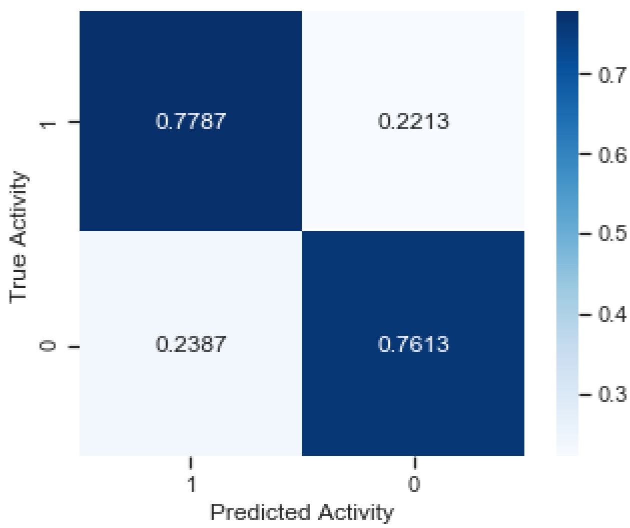
In [204]:
```
from sklearn.svm import SVC

svc = SVC()
svc.fit(x_train, y_train)

print("Accuracy on training set: {:.2f}".format(svc.score(x_train, y_train)))
print("Accuracy on test set: {:.2f}".format(svc.score(x_test, y_test)))

plot_confusion_matrix(svc)
```

```
C:\Users\sekaranh\AppData\Local\Continuum\anaconda3\lib\site-packages\sklearn
\svm\base.py:196: FutureWarning: The default value of gamma will change from
'auto' to 'scale' in version 0.22 to account better for unscaled features. Se
t gamma explicitly to 'auto' or 'scale' to avoid this warning.
  "avoid this warning.", FutureWarning)

Accuracy on training set: 0.76
Accuracy on test set: 0.76

Accuracy 0.7565026010404161 ROC 0.5179969796497048
```



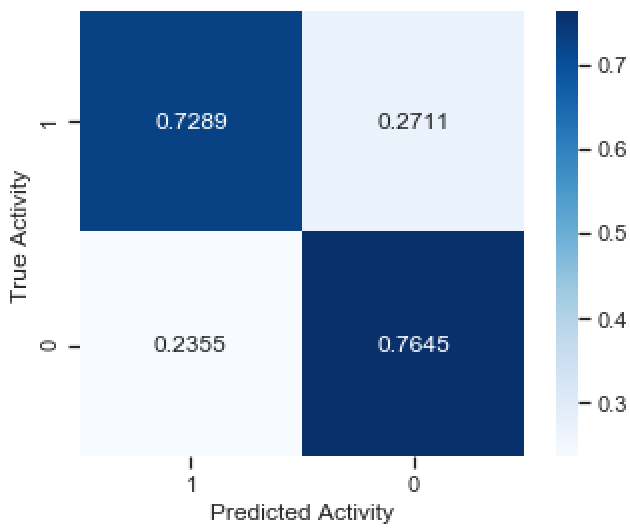
In [72]:
```
gb1 = GradientBoostingClassifier(random_state=0, max_depth=1)
gb1.fit(x_train, y_train)

print("Accuracy on training set: {:.3f}".format(gb1.score(x_train, y_train)))
print("Accuracy on test set: {:.3f}".format(gb1.score(x_test, y_test)))
plot_confusion_matrix(gb1)
```

```
Accuracy on training set: 0.766
Accuracy on test set: 0.762

Accuracy 0.7617046818727491 ROC 0.5341177647708364
```



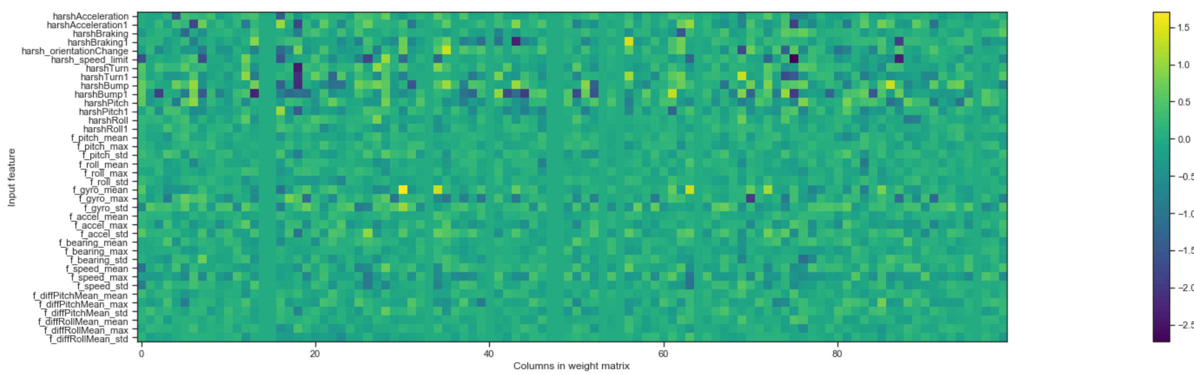
In [177]:
```
from sklearn.neural_network import MLPClassifier

mlp = MLPClassifier(random_state=42)
mlp.fit(x_train, y_train)

print("Accuracy on training set: {:.2f}".format(mlp.score(x_train, y_train)))
print("Accuracy on test set: {:.2f}".format(mlp.score(x_test, y_test)))
plot_confusion_matrix(mlp)
```

```
C:\Users\sekaranh\AppData\Local\Continuum\anaconda3\lib\site-packages\sklearn
\neural_network\multilayer_perceptron.py:562: ConvergenceWarning: Stochastic
Optimizer: Maximum iterations (200) reached and the optimization hasn't conve
rged yet.
  % self.max_iter, ConvergenceWarning)

Accuracy on training set: 0.77
Accuracy on test set: 0.76

Accuracy 0.7633053221288515 ROC 0.542036359341386
```



In [104]:
```
bookingID_features=[ 'harshAcceleration', 'harshAcceleration1', 'harshBraking'
,
       'harshBraking1', 'harsh_orientationChange', 'harsh_speed_limit',
       'harshTurn', 'harshTurn1', 'harshBump', 'harshBump1', 'harshPitch',
       'harshPitch1', 'harshRoll', 'harshRoll1', 'f_pitch_mean', 'f_pitch_max'
,
       'f_pitch_std', 'f_roll_mean', 'f_roll_max', 'f_roll_std', 'f_gyro_mean'
,
       'f_gyro_max', 'f_gyro_std', 'f_accel_mean', 'f_accel_max',
       'f_accel_std', 'f_bearing_mean', 'f_bearing_max', 'f_bearing_std',
       'f_speed_mean', 'f_speed_max', 'f_speed_std', 'f_diffPitchMean_mean',
       'f_diffPitchMean_max', 'f_diffPitchMean_std', 'f_diffRollMean_mean',
       'f_diffRollMean_max', 'f_diffRollMean_std']

plt.figure(figsize=(60, 7))
plt.imshow(mlp.coefs_[0], interpolation='none', cmap='viridis')
plt.yticks(range(38), bookingID_features)
plt.xlabel("Columns in weight matrix")
plt.ylabel("Input feature")
plt.colorbar()
```

Out[104]: <matplotlib.colorbar.Colorbar at 0x1f002743278>



In [93]:
```
from xgboost import XGBClassifier
from sklearn.model_selection import  GridSearchCV
params_grid = [
    {
        'eta':[0.01,0.05,0.1],
        'min_child_weight':[1,10,100],
        'max_depth':[3,5],
        'subsample':[0.5,0.7,0.9],
        'lambda':[0.01,0.1,1],
        'objective':['binary:logistic'],
        'eval_metric':['auc'],
        'seed':[42]
    }
]
xgb = XGBClassifier()
grid_search = GridSearchCV(xgb, params_grid, cv = 10, n_jobs=-1, verbose=1)
grid_search.fit(x_train,y_train)
```

```
Fitting 10 folds for each of 162 candidates, totalling 1620 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  26 tasks      | elapsed:   31.9s
[Parallel(n_jobs=-1)]: Done 176 tasks      | elapsed:  2.7min
[Parallel(n_jobs=-1)]: Done 426 tasks      | elapsed:  6.0min
[Parallel(n_jobs=-1)]: Done 776 tasks      | elapsed: 10.9min
[Parallel(n_jobs=-1)]: Done 1226 tasks      | elapsed: 17.4min
[Parallel(n_jobs=-1)]: Done 1620 out of 1620 | elapsed: 22.9min finished
```

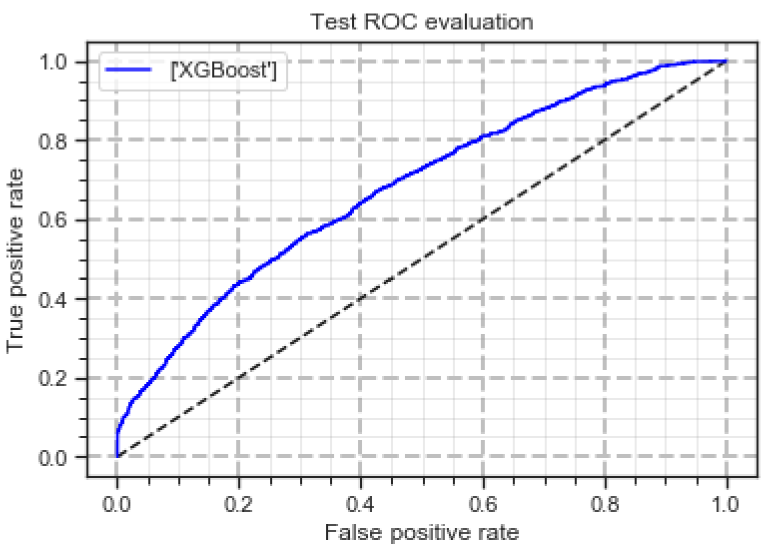
Out[93]:
```
GridSearchCV(cv=10, error_score='raise-deprecating',
       estimator=XGBClassifier(base_score=0.5, booster='gbtree', colsample_by
level=1,
       colsample_bynode=1, colsample_bytree=1, gamma=0, learning_rate=0.1,
       max_delta_step=0, max_depth=3, min_child_weight=1, missing=None,
       n_estimators=100, n_jobs=1, nthread=None,
       objective='binary:logistic', random_state=0, reg_alpha=0,
       reg_lambda=1, scale_pos_weight=1, seed=None, silent=None,
       subsample=1, verbosity=1),
       fit_params=None, iid='warn', n_jobs=-1,
       param_grid=[{'eta': [0.01, 0.05, 0.1], 'min_child_weight': [1, 10, 10
0], 'max_depth': [3, 5], 'subsample': [0.5, 0.7, 0.9], 'lambda': [0.01, 0.1,
1], 'objective': ['binary:logistic'], 'eval_metric': ['auc'], 'seed': [42]}],
       pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
       scoring=None, verbose=1)
```

In [105]:
```
grid_search.best_score_
```

Out[105]:
```
0.7660596357814689
```

In [106]:
```
best_model = grid_search.best_estimator_
prob = best_model.predict_proba(x_test)[:,1]

print('The validation AUC is :', roc_auc_score(y_test,prob))
fpr, tpr, __ = roc_curve(y_test,prob)
plt.figure()
plt.plot([0, 1], [0, 1], 'k--')

plt.plot(fpr,tpr ,label=['XGBoost'],color='blue')
plt.grid(True, lw = 2, ls = '--', c = '.75')
plt.minorticks_on()
plt.grid(b=True, which='minor', color='#999999', linestyle='-', alpha=0.2)
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('Test ROC evaluation')
plt.legend(loc='best')
plt.show()
```

```
The validation AUC is : 0.6790687685200391
```



This Data is not Enough we are able to to acthive accuracy of only 76%

In [103]:
```
normalizedFeatures.columns
```

Out[103]:
```
Index(['bookingID', 'harshAcceleration', 'harshAcceleration1', 'harshBrakin
g',
       'harshBraking1', 'harsh_orientationChange', 'harsh_speed_limit',
       'harshTurn', 'harshTurn1', 'harshBump', 'harshBump1', 'harshPitch',
       'harshPitch1', 'harshRoll', 'harshRoll1', 'f_pitch_mean', 'f_pitch_ma
x',
       'f_pitch_std', 'f_roll_mean', 'f_roll_max', 'f_roll_std', 'f_gyro_mea
n',
       'f_gyro_max', 'f_gyro_std', 'f_accel_mean', 'f_accel_max',
       'f_accel_std', 'f_bearing_mean', 'f_bearing_max', 'f_bearing_std',
       'f_speed_mean', 'f_speed_max', 'f_speed_std', 'f_diffPitchMean_mean',
       'f_diffPitchMean_max', 'f_diffPitchMean_std', 'f_diffRollMean_mean',
       'f_diffRollMean_max', 'f_diffRollMean_std', 'label'],
      dtype='object')
```

DNN modle to increase Accuracy

In [107]:
```
## Define path data
COLUMNS = ['bookingID', 'harshAcceleration', 'harshAcceleration1', 'harshBraki
ng',
       'harshBraking1', 'harsh_orientationChange', 'harsh_speed_limit',
       'harshTurn', 'harshTurn1', 'harshBump', 'harshBump1', 'harshPitch',
       'harshPitch1', 'harshRoll', 'harshRoll1', 'f_pitch_mean', 'f_pitch_max'
,
       'f_pitch_std', 'f_roll_mean', 'f_roll_max', 'f_roll_std', 'f_gyro_mean'
,
       'f_gyro_max', 'f_gyro_std', 'f_accel_mean', 'f_accel_max',
       'f_accel_std', 'f_bearing_mean', 'f_bearing_max', 'f_bearing_std',
       'f_speed_mean', 'f_speed_max', 'f_speed_std', 'f_diffPitchMean_mean',
       'f_diffPitchMean_max', 'f_diffPitchMean_std', 'f_diffRollMean_mean',
       'f_diffRollMean_max', 'f_diffRollMean_std', 'label']
PATH = 'train.csv'
PATH_test = 'eval.csv'
```

In [108]:
```
df_train = pd.read_csv(PATH, skipinitialspace=True, names = COLUMNS, index_col
=False)
df_test = pd.read_csv(PATH_test,skiprows = 1, skipinitialspace=True, names = C
OLUMNS, index_col=False)
```

In [109]:
```
print(df_train.shape, df_test.shape)
```

```
(16083, 40) (3905, 40)
```

In [110]:
```
print(df_train.dtypes)
```

```
bookingID                     int64
harshAcceleration           float64
harshAcceleration1          float64
harshBraking                float64
harshBraking1               float64
harsh_orientationChange     float64
harsh_speed_limit           float64
harshTurn                   float64
harshTurn1                  float64
harshBump                   float64
harshBump1                  float64
harshPitch                  float64
harshPitch1                 float64
harshRoll                   float64
harshRoll1                  float64
f_pitch_mean                float64
f_pitch_max                 float64
f_pitch_std                 float64
f_roll_mean                 float64
f_roll_max                  float64
f_roll_std                  float64
f_gyro_mean                 float64
f_gyro_max                  float64
f_gyro_std                  float64
f_accel_mean                float64
f_accel_max                 float64
f_accel_std                 float64
f_bearing_mean              float64
f_bearing_max               float64
f_bearing_std               float64
f_speed_mean                float64
f_speed_max                 float64
f_speed_std                 float64
f_diffPitchMean_mean        float64
f_diffPitchMean_max         float64
f_diffPitchMean_std         float64
f_diffRollMean_mean         float64
f_diffRollMean_max          float64
f_diffRollMean_std          float64
label                         int64
dtype: object
```

In [112]:
```
COLUMNS_FEATURE = ['harshAcceleration', 'harshAcceleration1', 'harshBraking',
       'harshBraking1', 'harsh_orientationChange', 'harsh_speed_limit',
       'harshTurn', 'harshTurn1', 'harshBump', 'harshBump1', 'harshPitch',
       'harshPitch1', 'harshRoll', 'harshRoll1', 'f_pitch_mean', 'f_pitch_max'
,
       'f_pitch_std', 'f_roll_mean', 'f_roll_max', 'f_roll_std', 'f_gyro_mean'
,
       'f_gyro_max', 'f_gyro_std', 'f_accel_mean', 'f_accel_max',
       'f_accel_std', 'f_bearing_mean', 'f_bearing_max', 'f_bearing_std',
       'f_speed_mean', 'f_speed_max', 'f_speed_std', 'f_diffPitchMean_mean',
       'f_diffPitchMean_max', 'f_diffPitchMean_std', 'f_diffRollMean_mean',
       'f_diffRollMean_max', 'f_diffRollMean_std']
continuous_features = [tf.feature_column.numeric_column(k) for k in COLUMNS_FE
ATURE]
```

In [113]:
```
model = tf.estimator.DNNLinearCombinedClassifier(
    n_classes=2,
    model_dir="ongoing/train15",
    linear_optimizer=tf.train.FtrlOptimizer(learning_rate=0.001,l1_regularizat
ion_strength=0.7,l2_regularization_strength=5),
    dnn_feature_columns=continuous_features,
    dnn_hidden_units=[256,256, 50],
    dnn_optimizer=tf.train.AdagradOptimizer(learning_rate=0.01))
```

```
INFO:tensorflow:Using default config.
INFO:tensorflow:Using config: {'_model_dir': 'ongoing/train15', '_tf_random_s
eed': None, '_save_summary_steps': 100, '_save_checkpoints_steps': None, '_sa
ve_checkpoints_secs': 600, '_session_config': allow_soft_placement: true
graph_options {
  rewrite_options {
    meta_optimizer_iterations: ONE
  }
}
, '_keep_checkpoint_max': 5, '_keep_checkpoint_every_n_hours': 10000, '_log_s
tep_count_steps': 100, '_train_distribute': None, '_device_fn': None, '_proto
col': None, '_eval_distribute': None, '_experimental_distribute': None, '_ser
vice': None, '_cluster_spec': <tensorflow.python.training.server_lib.ClusterS
pec object at 0x000001F003CA7438>, '_task_type': 'worker', '_task_id': 0, '_g
lobal_id_in_cluster': 0, '_master': '', '_evaluation_master': '', '_is_chie
f': True, '_num_ps_replicas': 0, '_num_worker_replicas': 1}
```

In [114]:
```
LABEL= 'label'
def get_input_fn(data_set, num_epochs=None, n_batch = 128, shuffle=True):
    return tf.estimator.inputs.pandas_input_fn(
       x=pd.DataFrame({k: data_set[k].values for k in COLUMNS_FEATURE}),
       y = pd.Series(data_set[LABEL].values),
       batch_size=n_batch,
       num_epochs=num_epochs,
       shuffle=shuffle)
```

In [115]:
```
model.train(input_fn=get_input_fn(df_train,
                                      num_epochs=None,
                                      n_batch = 128,
                                      shuffle=False),
                                      steps=2000)
```

```
WARNING:tensorflow:From C:\Users\sekaranh\AppData\Local\Continuum\anaconda3\l
ib\site-packages\tensorflow\python\estimator\inputs\queues\feeding_queue_runn
er.py:62: QueueRunner.__init__ (from tensorflow.python.training.queue_runner_
impl) is deprecated and will be removed in a future version.
Instructions for updating:
To construct input pipelines, use the `tf.data` module.
WARNING:tensorflow:From C:\Users\sekaranh\AppData\Local\Continuum\anaconda3\l
ib\site-packages\tensorflow\python\estimator\inputs\queues\feeding_functions.
py:500: add_queue_runner (from tensorflow.python.training.queue_runner_impl)
is deprecated and will be removed in a future version.
Instructions for updating:
To construct input pipelines, use the `tf.data` module.
INFO:tensorflow:Calling model_fn.
INFO:tensorflow:Done calling model_fn.
INFO:tensorflow:Create CheckpointSaverHook.
INFO:tensorflow:Graph was finalized.
INFO:tensorflow:Restoring parameters from ongoing/train15\model.ckpt-2000
INFO:tensorflow:Running local_init_op.
INFO:tensorflow:Done running local_init_op.
WARNING:tensorflow:From C:\Users\sekaranh\AppData\Local\Continuum\anaconda3\l
ib\site-packages\tensorflow\python\training\monitored_session.py:804: start_q
ueue_runners (from tensorflow.python.training.queue_runner_impl) is deprecate
d and will be removed in a future version.
Instructions for updating:
To construct input pipelines, use the `tf.data` module.
INFO:tensorflow:Saving checkpoints for 2000 into ongoing/train15\model.ckpt.
INFO:tensorflow:loss = 87.6553, step = 2001
INFO:tensorflow:global_step/sec: 79.6178
INFO:tensorflow:loss = 53.99681, step = 2101 (1.264 sec)
INFO:tensorflow:global_step/sec: 100.806
INFO:tensorflow:loss = 56.268463, step = 2201 (0.992 sec)
INFO:tensorflow:global_step/sec: 98.8142
INFO:tensorflow:loss = 70.87938, step = 2301 (1.020 sec)
INFO:tensorflow:global_step/sec: 90.5772
INFO:tensorflow:loss = 78.64027, step = 2401 (1.092 sec)
INFO:tensorflow:global_step/sec: 102.88
INFO:tensorflow:loss = 41.530903, step = 2501 (0.972 sec)
INFO:tensorflow:global_step/sec: 88.3422
INFO:tensorflow:loss = 61.45598, step = 2601 (1.136 sec)
INFO:tensorflow:global_step/sec: 83.0565
INFO:tensorflow:loss = 65.99566, step = 2701 (1.204 sec)
INFO:tensorflow:global_step/sec: 87.4124
INFO:tensorflow:loss = 70.94962, step = 2801 (1.140 sec)
INFO:tensorflow:global_step/sec: 87.4125
INFO:tensorflow:loss = 70.54324, step = 2901 (1.144 sec)
INFO:tensorflow:global_step/sec: 96.5253
INFO:tensorflow:loss = 53.988316, step = 3001 (1.036 sec)
INFO:tensorflow:global_step/sec: 102.456
INFO:tensorflow:loss = 59.1893, step = 3101 (0.976 sec)
INFO:tensorflow:global_step/sec: 103.737
INFO:tensorflow:loss = 70.96391, step = 3201 (0.964 sec)
INFO:tensorflow:global_step/sec: 98.4251
INFO:tensorflow:loss = 72.90112, step = 3301 (1.020 sec)
INFO:tensorflow:global_step/sec: 100.402
INFO:tensorflow:loss = 82.619774, step = 3401 (0.992 sec)
INFO:tensorflow:global_step/sec: 102.041
INFO:tensorflow:loss = 52.372383, step = 3501 (0.980 sec)
```

```
INFO:tensorflow:global_step/sec: 97.6535
INFO:tensorflow:loss = 60.038322, step = 3601 (1.024 sec)
INFO:tensorflow:global_step/sec: 102.881
INFO:tensorflow:loss = 57.729553, step = 3701 (0.972 sec)
INFO:tensorflow:global_step/sec: 103.309
INFO:tensorflow:loss = 75.908615, step = 3801 (0.968 sec)
INFO:tensorflow:global_step/sec: 100.806
INFO:tensorflow:loss = 66.50431, step = 3901 (0.996 sec)
INFO:tensorflow:Saving checkpoints for 4000 into ongoing/train15\model.ckpt.
INFO:tensorflow:Loss for final step: 53.982426.
```

Out[115]:

```
<tensorflow.python.estimator.canned.dnn_linear_combined.DNNLinearCombinedClas
sifier at 0x1f003ca7780>
```

In [116]:

```
model.evaluate(input_fn=get_input_fn(df_test,
                                      num_epochs=1,
                                      n_batch = 128,
                                      shuffle=False),
                                      steps=2000)
```

```
INFO:tensorflow:Calling model_fn.
WARNING:tensorflow:Trapezoidal rule is known to produce incorrect PR-AUCs; pl
ease switch to "careful_interpolation" instead.
WARNING:tensorflow:Trapezoidal rule is known to produce incorrect PR-AUCs; pl
ease switch to "careful_interpolation" instead.
INFO:tensorflow:Done calling model_fn.
INFO:tensorflow:Starting evaluation at 2019-06-16-15:30:08
INFO:tensorflow:Graph was finalized.
INFO:tensorflow:Restoring parameters from ongoing/train15\model.ckpt-4000
INFO:tensorflow:Running local_init_op.
INFO:tensorflow:Done running local_init_op.
INFO:tensorflow:Finished evaluation at 2019-06-16-15:30:10
INFO:tensorflow:Saving dict for global step 4000: accuracy = 0.7605634, accur
acy_baseline = 0.7505762, auc = 0.63622624, auc_precision_recall = 0.3926016
7, average_loss = 0.53554624, global_step = 4000, label/mean = 0.24942382, lo
ss = 67.461555, precision = 0.76, prediction/mean = 0.21062137, recall = 0.05
852156
INFO:tensorflow:Saving 'checkpoint_path' summary for global step 4000: ongoin
g/train15\model.ckpt-4000
```

Out[116]:

```
{'accuracy': 0.7605634,
 'accuracy_baseline': 0.7505762,
 'auc': 0.63622624,
 'auc_precision_recall': 0.39260167,
 'average_loss': 0.53554624,
 'label/mean': 0.24942382,
 'loss': 67.461555,
 'precision': 0.76,
 'prediction/mean': 0.21062137,
 'recall': 0.05852156,
 'global_step': 4000}
```

In [117]:
```
COLUMNS_FEATURE_REDUCED = ['harshAcceleration', 'harshAcceleration1', 'harshBr
aking',
       'harshBraking1', 'harsh_orientationChange', 'harsh_speed_limit',
       'harshTurn', 'harshTurn1', 'harshBump', 'harshBump1', 'harshPitch',
       'harshPitch1', 'harshRoll', 'harshRoll1']
continuous_features_reduce = [tf.feature_column.numeric_column(k) for k in COL
UMNS_FEATURE]
```

In [118]:
```
model = tf.estimator.DNNLinearCombinedClassifier(
    n_classes=2,
    model_dir="ongoing/train15",
    linear_optimizer=tf.train.FtrlOptimizer(learning_rate=0.001,l1_regularizat
ion_strength=0.7,l2_regularization_strength=5),
    dnn_feature_columns=continuous_features_reduce,
    dnn_hidden_units=[256,256, 50],
    dnn_optimizer=tf.train.AdagradOptimizer(learning_rate=0.01))
```

```
INFO:tensorflow:Using default config.
INFO:tensorflow:Using config: {'_model_dir': 'ongoing/train15', '_tf_random_s
eed': None, '_save_summary_steps': 100, '_save_checkpoints_steps': None, '_sa
ve_checkpoints_secs': 600, '_session_config': allow_soft_placement: true
graph_options {
  rewrite_options {
    meta_optimizer_iterations: ONE
  }
}
, '_keep_checkpoint_max': 5, '_keep_checkpoint_every_n_hours': 10000, '_log_s
tep_count_steps': 100, '_train_distribute': None, '_device_fn': None, '_proto
col': None, '_eval_distribute': None, '_experimental_distribute': None, '_ser
vice': None, '_cluster_spec': <tensorflow.python.training.server_lib.ClusterS
pec object at 0x000001F027394588>, '_task_type': 'worker', '_task_id': 0, '_g
lobal_id_in_cluster': 0, '_master': '', '_evaluation_master': '', '_is_chie
f': True, '_num_ps_replicas': 0, '_num_worker_replicas': 1}
```

In [119]:
```
LABEL= 'label'
def get_input_fn(data_set, num_epochs=None, n_batch = 128, shuffle=True):
    return tf.estimator.inputs.pandas_input_fn(
       x=pd.DataFrame({k: data_set[k].values for k in COLUMNS_FEATURE}),
       y = pd.Series(data_set[LABEL].values),
       batch_size=n_batch,
       num_epochs=num_epochs,
       shuffle=shuffle)
```

In [120]:
```
model.train(input_fn=get_input_fn(df_train,
                                      num_epochs=None,
                                      n_batch = 128,
                                      shuffle=False),
                                      steps=2000)
```

```
INFO:tensorflow:Calling model_fn.
INFO:tensorflow:Done calling model_fn.
INFO:tensorflow:Create CheckpointSaverHook.
INFO:tensorflow:Graph was finalized.
INFO:tensorflow:Restoring parameters from ongoing/train15\model.ckpt-4000
INFO:tensorflow:Running local_init_op.
INFO:tensorflow:Done running local_init_op.
INFO:tensorflow:Saving checkpoints for 4000 into ongoing/train15\model.ckpt.
INFO:tensorflow:loss = 86.371216, step = 4001
INFO:tensorflow:global_step/sec: 80.3839
INFO:tensorflow:loss = 53.204716, step = 4101 (1.244 sec)
INFO:tensorflow:global_step/sec: 99.2064
INFO:tensorflow:loss = 56.04718, step = 4201 (1.012 sec)
INFO:tensorflow:global_step/sec: 96.9018
INFO:tensorflow:loss = 69.7899, step = 4301 (1.032 sec)
INFO:tensorflow:global_step/sec: 95.4169
INFO:tensorflow:loss = 77.1274, step = 4401 (1.044 sec)
INFO:tensorflow:global_step/sec: 96.525
INFO:tensorflow:loss = 41.418243, step = 4501 (1.036 sec)
INFO:tensorflow:global_step/sec: 95.0574
INFO:tensorflow:loss = 60.528717, step = 4601 (1.052 sec)
INFO:tensorflow:global_step/sec: 99.6016
INFO:tensorflow:loss = 64.737305, step = 4701 (1.004 sec)
INFO:tensorflow:global_step/sec: 102.041
INFO:tensorflow:loss = 70.24129, step = 4801 (0.980 sec)
INFO:tensorflow:global_step/sec: 91.5774
INFO:tensorflow:loss = 69.197464, step = 4901 (1.100 sec)
INFO:tensorflow:global_step/sec: 71.427
INFO:tensorflow:loss = 53.21539, step = 5001 (1.396 sec)
INFO:tensorflow:global_step/sec: 88.0304
INFO:tensorflow:loss = 58.148575, step = 5101 (1.136 sec)
INFO:tensorflow:global_step/sec: 97.2763
INFO:tensorflow:loss = 70.24313, step = 5201 (1.024 sec)
INFO:tensorflow:global_step/sec: 96.1539
INFO:tensorflow:loss = 72.057724, step = 5301 (1.040 sec)
INFO:tensorflow:global_step/sec: 104.167
INFO:tensorflow:loss = 81.337166, step = 5401 (0.960 sec)
INFO:tensorflow:global_step/sec: 100.401
INFO:tensorflow:loss = 51.693245, step = 5501 (1.000 sec)
INFO:tensorflow:global_step/sec: 75.7558
INFO:tensorflow:loss = 58.897972, step = 5601 (1.316 sec)
INFO:tensorflow:global_step/sec: 92.9394
INFO:tensorflow:loss = 57.17653, step = 5701 (1.080 sec)
INFO:tensorflow:global_step/sec: 98.0365
INFO:tensorflow:loss = 74.88202, step = 5801 (1.016 sec)
INFO:tensorflow:global_step/sec: 97.2789
INFO:tensorflow:loss = 65.49578, step = 5901 (1.028 sec)
INFO:tensorflow:Saving checkpoints for 6000 into ongoing/train15\model.ckpt.
INFO:tensorflow:Loss for final step: 53.405113.
```

Out[120]:

```
<tensorflow.python.estimator.canned.dnn_linear_combined.DNNLinearCombinedClas
sifier at 0x1f027394668>
```

In [121]:
```
model.evaluate(input_fn=get_input_fn(df_test,
                                      num_epochs=1,
                                      n_batch = 128,
                                      shuffle=False),
                                      steps=2000)
```

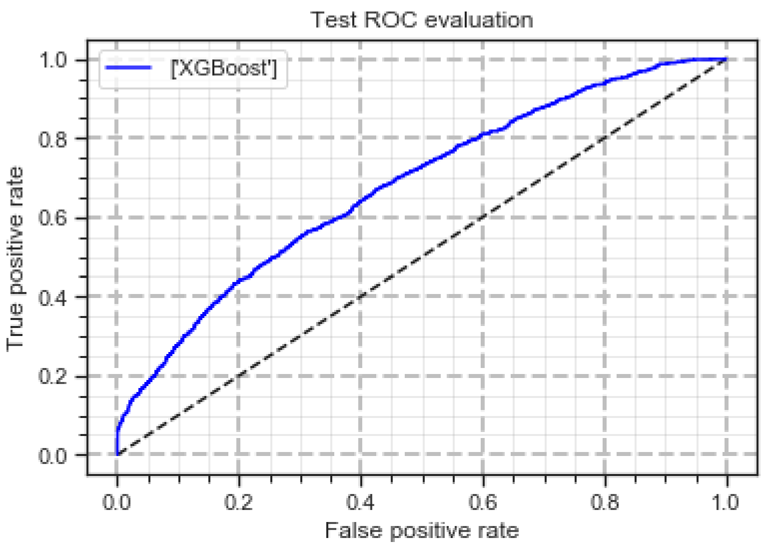
```
INFO:tensorflow:Calling model_fn.
WARNING:tensorflow:Trapezoidal rule is known to produce incorrect PR-AUCs; pl
ease switch to "careful_interpolation" instead.
WARNING:tensorflow:Trapezoidal rule is known to produce incorrect PR-AUCs; pl
ease switch to "careful_interpolation" instead.
INFO:tensorflow:Done calling model_fn.
INFO:tensorflow:Starting evaluation at 2019-06-16-15:32:45
INFO:tensorflow:Graph was finalized.
INFO:tensorflow:Restoring parameters from ongoing/train15\model.ckpt-6000
INFO:tensorflow:Running local_init_op.
INFO:tensorflow:Done running local_init_op.
INFO:tensorflow:Finished evaluation at 2019-06-16-15:32:47
INFO:tensorflow:Saving dict for global step 6000: accuracy = 0.7590269, accur
acy_baseline = 0.7505762, auc = 0.63638526, auc_precision_recall = 0.3918356,
average_loss = 0.53741264, global_step = 6000, label/mean = 0.24942382, loss
= 67.69666, precision = 0.6813187, prediction/mean = 0.21266793, recall = 0.0
6365503
INFO:tensorflow:Saving 'checkpoint_path' summary for global step 6000: ongoin
g/train15\model.ckpt-6000
```

Out[121]:
```
{'accuracy': 0.7590269,
 'accuracy_baseline': 0.7505762,
 'auc': 0.63638526,
 'auc_precision_recall': 0.3918356,
 'average_loss': 0.53741264,
 'label/mean': 0.24942382,
 'loss': 67.69666,
 'precision': 0.6813187,
 'prediction/mean': 0.21266793,
 'recall': 0.06365503,
 'global_step': 6000}
```

To improve the accuracy we need more Data, last model is using only COLUMNS_FEATURE_REDUCED 'harshAcceleration', 'harshBraking','harsh_orientationChange', 'harsh_speed_limit','harshTurn',, 'harshBump', 'harshPitch', 'harshRoll.

In [182]:
```
best_model = grid_search.best_estimator_
prob = best_model.predict_proba(x_test)[:,1]

print('The validation AUC is :', roc_auc_score(y_test,prob))
fpr, tpr, __ = roc_curve(y_test,prob)
plt.figure()
plt.plot([0, 1], [0, 1], 'k--')

plt.plot(fpr,tpr ,label=['XGBoost'],color='blue')
plt.grid(True, lw = 2, ls = '--', c = '.75')
plt.minorticks_on()
plt.grid(b=True, which='minor', color='#999999', linestyle='-', alpha=0.2)
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('Test ROC evaluation')
plt.legend(loc='best')
plt.show()
```

```
The validation AUC is : 0.6790687685200391
```



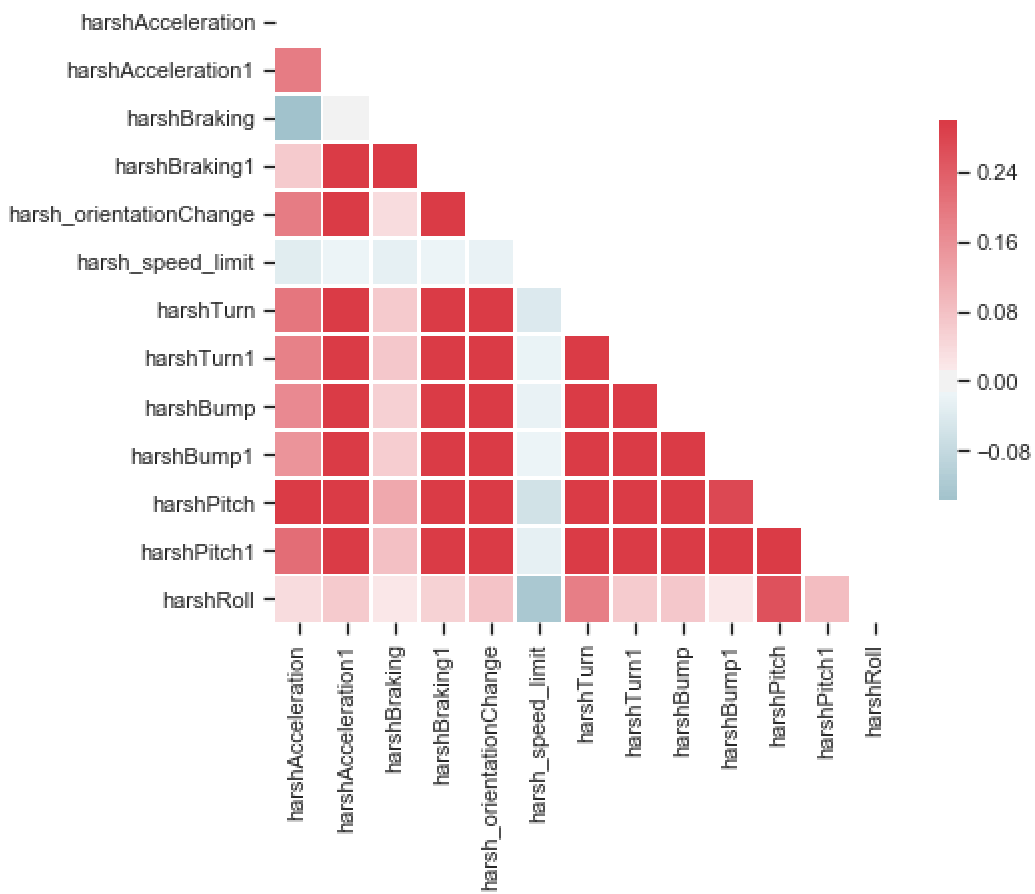
In [170]:
```
from sklearn.metrics import accuracy_score, confusion_matrix, roc_auc_score, r
oc_curve
from matplotlib import pyplot
def plot_acc_curve_1(model):
    labels_predict = model.predict(x_test)
    print('The validation AUC is :', roc_auc_score(y_test,labels_predict))
    cm = confusion_matrix(labels_predict, y_test, labels=normalizedFeatures.la
bel.unique())
    cm_normalized = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
    fpr, tpr, __ = roc_curve(y_test,labels_predict)
    fig= plt.figure(figsize=(15,4))
    ax1=fig.add_subplot(1,3,1)
    sns.heatmap(cm_normalized, annot=True, fmt=".4f",
            cmap='Blues', square=True,
            xticklabels=normalizedFeatures.label.unique(),
            yticklabels=normalizedFeatures.label.unique())
    ax1.set_xlabel('Predicted Activity')
    ax1.set_ylabel('True Activity', )
    ax2=fig.add_subplot(1,3,2)
    ax2.plot([0, 1], [0, 1], 'k--')
    ax2.plot(fpr,tpr ,label=['ROC'],color='blue')
    ax2.grid(True, lw = 2, ls = '--', c = '.75')
    ax2.minorticks_on()
    ax2.grid(b=True, which='minor', color='#999999', linestyle='-', alpha=0.2)
    ax2.set_xlabel('False positive rate')
    ax2.set_ylabel('True positive rate')
    #ax2.title('Test ROC evaluation')
    ax2.legend(loc='best')
    plt.show()
```

In [219]:
```
from sklearn.metrics import accuracy_score, confusion_matrix, roc_auc_score, r
oc_curve
from matplotlib import pyplot
def plot_acc_curve_2(model):
    best_model = grid_search.best_estimator_
    prob = best_model.predict_proba(x_test)[:,1]
    print('The validation AUC is :', roc_auc_score(y_test,prob))
    y_pred = [1 if x > 0.60 else 0 for x in prob]
    cm = confusion_matrix(y_pred, y_test, labels=normalizedFeatures.label.uniq
ue())
    cm_normalized = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
    fpr, tpr, __ = roc_curve(y_test,prob)
    fig= plt.figure(figsize=(15,4))
    ax1=fig.add_subplot(1,3,1)
    sns.heatmap(cm_normalized, annot=True, fmt=".4f",
            cmap='Blues', square=True,
            xticklabels=normalizedFeatures.label.unique(),
            yticklabels=normalizedFeatures.label.unique())
    ax1.set_xlabel('Predicted Activity')
    ax1.set_ylabel('True Activity', )
    ax2=fig.add_subplot(1,3,2)
    ax2.plot([0, 1], [0, 1], 'k--')
    ax2.plot(fpr,tpr ,label=['ROC'],color='blue')
    ax2.grid(True, lw = 2, ls = '--', c = '.75')
    ax2.minorticks_on()
    ax2.grid(b=True, which='minor', color='#999999', linestyle='-', alpha=0.2)
    ax2.set_xlabel('False positive rate')
    ax2.set_ylabel('True positive rate')
    #ax2.title('Test ROC evaluation')
    ax2.legend(loc='best')
    plt.savefig('final_modle')
    plt.show()
```

In [212]:
```
RELEVANT_EVENTS = ['harshAcceleration','harshAcceleration1', 'harshBraking',
'harshBraking1','harsh_orientationChange','harsh_speed_limit','harshTurn',
        'harshTurn1', 'harshBump', 'harshBump1', 'harshPitch','harshPitch1','h
arshRoll',  ]
drop_these = list(set(list(normalizedFeatures)) - set(RELEVANT_EVENTS))
df_final =  normalizedFeatures.drop(drop_these, axis = 1)
```

In [221]:
```
create_heat_map_plt(df_final)
```

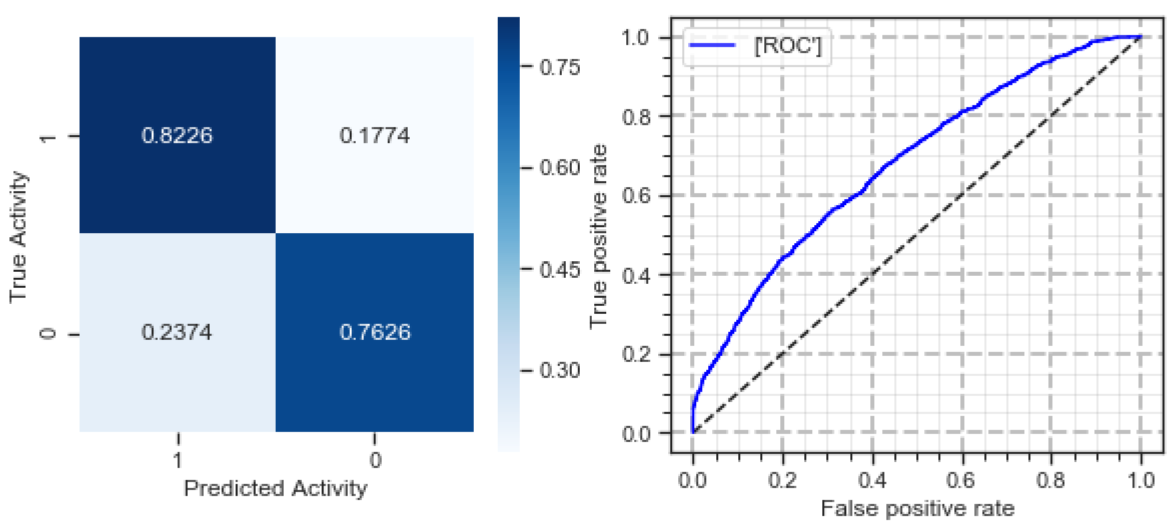
```
<Figure size 432x288 with 0 Axes>
```



gradient boosted decision trees (using XGBOOST)

In [220]:
```
plot_acc_curve_2(best_model)
```

```
The validation AUC is : 0.6790687685200391
```



In [209]:
```
normalizedFeatures.columns
```

Out[209]:
```
Index(['bookingID', 'harshAcceleration', 'harshAcceleration1', 'harshBrakin
g',
       'harshBraking1', 'harsh_orientationChange', 'harsh_speed_limit',
       'harshTurn', 'harshTurn1', 'harshBump', 'harshBump1', 'harshPitch',
       'harshPitch1', 'harshRoll', 'harshRoll1', 'f_pitch_mean', 'f_pitch_ma
x',
       'f_pitch_std', 'f_roll_mean', 'f_roll_max', 'f_roll_std', 'f_gyro_mea
n',
       'f_gyro_max', 'f_gyro_std', 'f_accel_mean', 'f_accel_max',
       'f_accel_std', 'f_bearing_mean', 'f_bearing_max', 'f_bearing_std',
       'f_speed_mean', 'f_speed_max', 'f_speed_std', 'f_diffPitchMean_mean',
       'f_diffPitchMean_max', 'f_diffPitchMean_std', 'f_diffRollMean_mean',
       'f_diffRollMean_max', 'f_diffRollMean_std', 'label'],
      dtype='object')
```

Conclusion

Best model is gradient boosted decision tree, need more data to increase accuracy, try Deep Neural Network like LinearClassification, LinearDNNClassifier, Custom Estimators. Still we cannot increase the accuracy.

Even consider bucketizing the features and added second order polynomials features, still cannot increase the accuracy. I need more data to increase accuracy and prove my hypothesis.

In [ ]:
```
```