code for graphical confusion matrix

In [69]:

```
def plot_confusion_matrix(cm,
                          target_names,
                          title='Confusion matrix',
                          cmap=None,
                          normalize=True):
    """
    given a sklearn confusion matrix (cm), make a nice plot

    Arguments
    ---------
    cm:           confusion matrix from sklearn.metrics.confusion_matrix

    target_names: given classification classes such as [0, 1, 2]
                  the class names, for example: ['high', 'medium', 'low']

    title:        the text to display at the top of the matrix

    cmap:         the gradient of the values displayed from matplotlib.pyplot.cm
                  see http://matplotlib.org/examples/color/colormaps_reference.html
                  plt.get_cmap('jet') or plt.cm.Blues

    normalize:    If False, plot the raw numbers
                  If True, plot the proportions

    Usage
    -----
    plot_confusion_matrix(cm           = cm,                  # confusion matrix created by
                                                              # sklearn.metrics.confusion_m
                          normalize    = True,                # show proportions
                          target_names = y_labels_vals,       # list of names of the classe
                          title        = best_estimator_name) # title of graph

    Citiation
    ---------
    http://scikit-learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html

    """


    accuracy = np.trace(cm) / float(np.sum(cm))
    misclass = 1 - accuracy

    if cmap is None:
        cmap = plt.get_cmap('Blues')

    plt.figure(figsize=(8, 6))
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()

    if target_names is not None:
        tick_marks = np.arange(len(target_names))
        plt.xticks(tick_marks, target_names, rotation=45)
        plt.yticks(tick_marks, target_names)

    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
```

```
    thresh = cm.max() / 1.5 if normalize else cm.max() / 2
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        if normalize:
            plt.text(j, i, "{:0.4f}".format(cm[i, j]),
                     horizontalalignment="center",
                     color="white" if cm[i, j] > thresh else "black")
        else:
            plt.text(j, i, "{:,}".format(cm[i, j]),
                     horizontalalignment="center",
                     color="white" if cm[i, j] > thresh else "black")


    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label\naccuracy={:0.4f}; misclass={:0.4f}'.format(accuracy, miscl
    plt.show()
```

importing packages

In [70]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import scipy.fftpack
from sklearn import preprocessing
from sklearn.metrics import confusion_matrix, f1_score
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.svm import SVC
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
import keras
from keras.models import Sequential
from keras.layers import Dense, Dropout
import itertools

# pandas_entropy is a .py file tht calculates entropy
#import pandas_entropy as pe

#import plot_conf_mat as cnf
```

Loading Data

In [71]:

```
df1 = pd.read_csv('./sensor_data/sensorData_23-03-2018_15217898569-12_55.txt')
df2 = pd.read_csv('./sensor_data/sensorData_23-03-2018_15217899985-1_02.txt')
df3 = pd.read_csv('./sensor_data/sensorData_23-03-2018_15217903881-1_06.txt')
df4 = pd.read_csv('./sensor_data/sensorData_23-03-2018_15217906095-1_09.txt')
df5 = pd.read_csv('./sensor_data/sensorData_23-03-2018_15217913809-1_24.txt')
df6 = pd.read_csv('./sensor_data/sensorData_23-03-2018_15217883948-12_31.txt')
df7 = pd.read_csv('./sensor_data/sensorData_23-03-2018_15217908058-1_13.txt')
df8 = pd.read_csv('./sensor_data_old/sensorData_19-03-2018_15214527501_15-18.txt')
df9 = pd.read_csv('./sensor_data_old/sensorData_19-03-2018_15214537070_15-34.txt')

# skip below file : NaN problem with data
#df10 = pd.read_csv('./sensor_data_old/sensorData_19-03-2018_15214540891_15-41.txt')

df11 = pd.read_csv('./sensor_data_old/sensorData_19-03-2018_15214551353_15-58.txt')
df12 = pd.read_csv('./sensor_data_old/sensorData_19-03-2018_15214560034_16-13.txt')
df13 = pd.read_csv('./sensor_data_new/sensorData_04-04-2018_15228184860-10_53.txt')
df14 = pd.read_csv('./sensor_data_new/sensorData_04-04-2018_15228194973-10_58.txt')
df15 = pd.read_csv('./sensor_data_new/sensorData_04-04-2018_15228197814-11_02.txt')
df16 = pd.read_csv('./sensor_data_new/sensorData_04-04-2018_15228200158-11_06.txt')
df17 = pd.read_csv('./sensor_data_new/sensorData_04-04-2018_15228202727-11_14.txt')
df18 = pd.read_csv('./sensor_data_new/sensorData_04-04-2018_15228207467-11_36.txt')
df19 = pd.read_csv('./sensor_data_new/sensorData_04-04-2018_15228271593-1_24.txt')
df20 = pd.read_csv('./sensor_data_new/sensorData_04-04-2018_15228285553-1_47.txt')


# dataframes for pothole annotated files

pdf1 = pd.read_csv('./potholes/sensorData_23-03-2018_15217898607-12_55.txt')
pdf2 = pd.read_csv('./potholes/sensorData_23-03-2018_15217899997-1_02.txt')
pdf3 = pd.read_csv('./potholes/sensorData_23-03-2018_15217903905-1_06.txt')
pdf4 = pd.read_csv('./potholes/sensorData_23-03-2018_15217906130-1_09.txt')
pdf5 = pd.read_csv('./potholes/sensorData_23-03-2018_15217914003-1_24.txt')
pdf6 = pd.read_csv('./potholes/sensorData_23-03-2018_15217883957-12_31.txt')
pdf7 = pd.read_csv('./potholes/sensorData_23-03-2018_15217908081-1_13.txt')
pdf8 = pd.read_csv('./potholes_old/sensorData_19-03-2018_15214527532_15-18.txt')
pdf9 = pd.read_csv('./potholes_old/sensorData_19-03-2018_15214537329_15-34.txt')

# below file contains Nan values
#pdf10 = pd.read_csv('./potholes_old/sensorData_19-03-2018_15214540946_15-41.txt')

pdf11 = pd.read_csv('./potholes_old/sensorData_19-03-2018_15214551395_15-58.txt')
pdf12 = pd.read_csv('./potholes_old/sensorData_19-03-2018_15214560096_16-13.txt')
pdf13 = pd.read_csv('./potholes_new/sensorData_04-04-2018_15228184880-10_53.txt')
pdf14 = pd.read_csv('./potholes_new/sensorData_04-04-2018_15228194989-10_58.txt')
pdf15 = pd.read_csv('./potholes_new/sensorData_04-04-2018_15228197815-11_02.txt')
pdf16 = pd.read_csv('./potholes_new/sensorData_04-04-2018_15228200151-11_06.txt')
pdf17 = pd.read_csv('./potholes_new/sensorData_04-04-2018_15228202691-11_14.txt')
pdf18 = pd.read_csv('./potholes_new/sensorData_04-04-2018_15228207479-11_36.txt')
pdf19 = pd.read_csv('./potholes_new/sensorData_04-04-2018_15228271641-1_24.txt')

pdf20 = pd.read_csv('./potholes_new/sensorData_04-04-2018_15228285574-1_47.txt')

df1.head()
```

Out[71]:

	timestamp	accx	accy	accz	gyrx	gyry	gyrz	longitude	
0	152178985702	0.152901	-0.657606	9.569601	-0.002742	0.008070	-0.005377	77.643354	1

	timestamp	accx	accy	accz	gyrx	gyry	gyrz	longitude	
1	152178985720	-0.444470	-0.501953	9.585513	0.000966	-0.007851	-0.005371	77.643354	1
2	152178985742	-0.195615	-0.610109	10.012871	0.000354	-0.009804	-0.004395	77.643354	1
3	152178985764	-0.310112	-0.708696	9.713529	0.000969	-0.009312	-0.004761	77.643354	1
4	152178985784	-0.400679	-0.345822	10.250119	-0.000861	-0.003700	-0.004639	77.643354	1

In [72]:

```
#pe.ID3_entropies(df1)
```

In [73]:

```
df_main = pd.read_csv('./features.txt')
df_main
```

Out[73]:

file_id	ts_start	ts_end	mean_ax	mean_ay	mean_az	mean_gx	mean_gy	mean_gz	sd_ax

0 rows × 82 columns

feature extraction

In [74]:

```
# this is to calcualte features by aggregating 10 data pts
df_index =0

for j in range(0,19):

    if(df_index == 0):
        df = df1
        pdf = pdf1
    elif(df_index == 1):
        df = df2
        pdf = pdf2
    elif(df_index == 2):
        df = df3
        pdf = pdf3
    elif(df_index == 3):
        df = df4
        pdf = pdf4
    elif(df_index == 4):
        df = df5
        pdf = pdf5
    elif(df_index == 5):
        df = df6
        pdf = pdf6
    elif(df_index == 6):
        df = df7
        pdf = pdf7
    elif(df_index == 7):
        df = df8
        pdf = pdf8
    elif(df_index == 8):
        df = df9
        pdf = pdf9

    elif(df_index == 9):
        df = df11
        pdf = pdf11
    elif(df_index == 10):
        df = df12
        pdf = pdf12
    elif(df_index == 11):
        df = df13
        pdf = pdf13
    elif(df_index == 12):
        df = df14
        pdf = pdf14
    elif(df_index == 13):
        df = df15
        pdf = pdf15
    elif(df_index == 14):
        df = df16
        pdf = pdf16
    elif(df_index == 15):
        df = df17
        pdf = pdf17
    elif(df_index == 16):
        df = df18
        pdf = pdf18
    elif(df_index == 17):
        df = df19
```

```
        pdf = pdf19
    elif(df_index == 18):
        df = df20
        pdf = pdf20
    else:       # skipping 10 no file bcz of NaN error for now
        df = df10
        pdf = pdf10

    df_index += 1
    count = 0
    k = 0

    for i in range(1,len(df),10):    # step size is 10 means aggregrating 10 data pts means
        if(i+9 >= len(df)):
            break
        #print(i)
        dt = df[i-1:i+10]      # chunking the given dataframe into smaller dataframe contai
        start = dt.timestamp[i-1]
        end = dt.timestamp[i+9]


#time domain features : mean , max , min , var , std dev, median , interquartile range,
#                       mean of abs deviation , skewness < left : root mean sq error , entr
        # mean
        a = dt.mean()      # will give an array of mean of columns of dt
        mean_ax = a[1]
        mean_ay = a[2]
        mean_az = a[3]

        mean_gx = a[4]
        mean_gy = a[5]
        mean_gz = a[6]

        # min
        a = dt.min()
        min_ax = a[1]
        min_ay = a[2]
        min_az = a[3]

        min_gx = a[4]
        min_gy = a[5]
        min_gz = a[6]

        # max
        a = dt.max()
        max_ax = a[1]
        max_ay = a[2]
        max_az = a[3]

        max_gx = a[4]
        max_gy = a[5]
        max_gz = a[6]

        # std dev
        a = dt.std()
        sd_ax = a[1]
        sd_ay = a[2]
        sd_az = a[3]

        sd_gx = a[4]
        sd_gy = a[5]
```

```
        sd_gz = a[6]

        # variance
        a = dt.var()
        var_ax = a[1]
        var_ay = a[2]
        var_az = a[3]

        var_gx = a[4]
        var_gy = a[5]
        var_gz = a[6]

        #adding max-min
        mm_x = max_ax - min_ax
        mm_y = max_ay - min_ay
        mm_z = max_az - min_az


        # median coln wise of acc data
        a = dt.median()
        med_ax = a[1]
        med_ay = a[2]
        med_az = a[3]

        med_gx = a[4]
        med_gy = a[5]
        med_gz = a[6]

        # entropy coln wise of acc data

        # interquantile ranges
        a = dt.quantile(.25)
        quant1_ax = a[1]
        quant1_ay = a[2]
        quant1_az = a[3]

        quant1_gx = a[4]
        quant1_gy = a[5]
        quant1_gz = a[6]

        a = dt.quantile(.5)
        quant2_ax = a[1]
        quant2_ay = a[2]
        quant2_az = a[3]

        quant2_gx = a[4]
        quant2_gy = a[5]
        quant2_gz = a[6]

        a = dt.quantile(.75)
        quant3_ax = a[1]
        quant3_ay = a[2]
        quant3_az = a[3]

        quant3_gx = a[4]
        quant3_gy = a[5]
        quant3_gz = a[6]


        # mean absolute deviation
        a = dt.mad()
```

```
        mad_ax = a[1]
        mad_ay = a[2]
        mad_az = a[3]

        mad_gx = a[4]
        mad_gy = a[5]
        mad_gz = a[6]

        # skewness
        a = dt.skew()
        skew_ax = a[1]
        skew_ay = a[2]
        skew_az = a[3]

        skew_gx = a[4]
        skew_gy = a[5]
        skew_gz = a[6]


# gradient based features : gradient with respect to timestamp

        #taking gradients
        arx = dt['accx']
        ary = dt['accy']
        arz = dt['accz']

        grx = dt['gyrx']
        gry = dt['gyry']
        grz = dt['gyrz']

        tm = dt['timestamp']
        adx = np.gradient(arx, tm).max()
        ady = np.gradient(ary, tm).max()
        adz = np.gradient(arz, tm).max()
        gdx = np.gradient(grx, tm).max()
        gdy = np.gradient(gry, tm).max()
        gdz = np.gradient(grz, tm).max()


# frequency domain features : fft , spectral energy ,

        #taking fourier transforms
        ft = scipy.fftpack.fft(dt)

        fft_ax = ft[1].max().imag
        fft_ay = ft[2].max().imag
        fft_az = ft[3].max().imag

        #getting spectral energy
        sp_ax = np.mean(np.square(ft[1].real) + np.square(ft[1].imag))
        sp_ay = np.mean(np.square(ft[2].real) + np.square(ft[2].imag))
        sp_az = np.mean(np.square(ft[3].real) + np.square(ft[3].imag))

        file_id = j + 1

        '''
        #adding label
        if(k >= len(pdf)):
            break

        if(pdf['timestamp'][k] > start and pdf['timestamp'][k] <= end ):
```

```
            label = 1
            k = k + 1
            #print("haha")

            if(k >= len(pdf)):
                break
            while(pdf['timestamp'][k] > start and pdf['timestamp'][k] <= end):
                k = k + 1
                if(k >= len(pdf)):
                        break
        else:
            label = 0
        '''

        if(k >= len(pdf)):
            break

        if(pdf['timestamp'][k] > start and pdf['timestamp'][k] <= end ):
            if(pdf['type'][k] == "pothole"):
                label = 1   # 1 means pothole
            else:
                label = 2   # 2 means others

            k = k + 1
            #print("haha")

            if(k >= len(pdf)):
                break
            while(pdf['timestamp'][k] > start and pdf['timestamp'][k] <= end):
                k = k + 1
                if(k >= len(pdf)):
                        break
        else:
            label = 0


        df_temp = pd.DataFrame([[file_id,start,end,mean_ax,mean_ay,mean_az,mean_gx,mean_gy,
                                 sd_ay,sd_az,sd_gx,sd_gy,sd_gz,min_ax,min_ay,min_az,min_gx,
                                 max_ax,max_ay,max_az,max_gx,max_gy,max_gz,var_ax,var_ay,va
                                 var_gz,med_ax,med_ay,med_az,med_gx,med_gy,med_gz,quant1_ax
                                 ,quant1_gx,quant1_gy,quant1_gz,quant2_ax,quant2_ay,quant2_
                                 quant2_gy,quant2_gz,quant3_ax,quant3_ay,quant3_az,quant3_g
                                 quant3_gz,mad_ax,mad_ay,mad_az,mad_gx,mad_gy,mad_gz,skew_a
                                 skew_az,skew_gx,skew_gy,skew_gz,adx,ady,adz,gdx,gdy,gdz,ff
                                 sp_ax,sp_ay,sp_az,label]],

                                columns = ('file_id','ts_start','ts_end','mean_ax','mean_ay',
                                           'mean_gz','sd_ax','sd_ay','sd_az','sd_gx','sd_gy',
                                           ,'min_az',
                                           'min_gx','min_gy','min_gz','max_ax','max_ay','max_
                                           'var_ax','var_ay','var_az','var_gx','var_gy','var_
                                           ,'med_az','med_gx',
                                           'med_gy','med_gz','quant1_ax','quant1_ay','quant1_
                                           'quant1_gy',
                                           'quant1_gz','quant2_ax','quant2_ay','quant2_az','q
                                           ,
                                           'quant2_gz','quant3_ax','quant3_ay','quant3_az','q
                                           'quant3_gz',
                                           'mad_ax','mad_ay','mad_az','mad_gx','mad_gy','mad_
                                           'skew_ay','skew_az',
```

```
                                        'skew_gx','skew_gy','skew_gz','adx','ady','adz','g
                                        ,'fft_ax','fft_ay','fft_az',
                                        'sp_ax','sp_ay','sp_az','label'))

        df_main = df_main.append(df_temp)
        #count = count + 1
        #i = i+20
```

Imbalanced Data

In [75]:

```
df_main['label'].value_counts()
```

Out[75]:

```
0    3155
1     182
2     143
Name: label, dtype: int64
```

In [76]:

```
print(df_main.shape)
df_main.head()
#df_main['file_id'].unique()
```

```
(3480, 82)
```

Out[76]:

	file_id	ts_start	ts_end	mean_ax	mean_ay	mean_az	mean_gx	mean_gy	m
0	1	152178985702	152178985912	-0.333256	-0.529265	9.790426	0.001520	-0.000742	-0
0	1	152178985912	152178986124	-0.422345	-0.421185	9.919289	0.005632	-0.011685	-0
0	1	152178986124	152178986337	-0.312917	-0.081383	9.855358	0.006088	-0.000170	-0
0	1	152178986337	152178986549	-0.374978	0.672824	9.570090	0.008233	-0.016506	-0
0	1	152178986549	152178986762	-0.313581	0.429496	9.818923	0.000324	-0.010486	-0

5 rows × 82 columns

In [77]:

```
# putting time stamps at the end
cols = list(df_main.columns.values) #Make a list of all of the columns in the df
cols.pop(cols.index('ts_start')) #Remove b from list
cols.pop(cols.index('ts_end')) #Remove x from list
cols.pop(cols.index('label')) # remove label
cols.pop(cols.index('file_id'))  # remove file_id
df_main = df_main[cols+['ts_start','ts_end', 'label' , 'file_id']]
df_main.head()
```

Out[77]:

	mean_ax	mean_ay	mean_az	mean_gx	mean_gy	mean_gz	sd_ax	sd_ay	sd_az
0	-0.333256	-0.529265	9.790426	0.001520	-0.000742	-0.004106	0.242658	0.162682	0.286165
0	-0.422345	-0.421185	9.919289	0.005632	-0.011685	-0.034437	0.254251	0.158947	0.201366
0	-0.312917	-0.081383	9.855358	0.006088	-0.000170	-0.020531	0.279257	0.514140	0.223400
0	-0.374978	0.672824	9.570090	0.008233	-0.016506	-0.037798	0.235341	0.213451	0.271166
0	-0.313581	0.429496	9.818923	0.000324	-0.010486	-0.039492	0.153429	0.272872	0.344815

5 rows × 82 columns

In [78]:

```
df_main_copy = df_main.copy()
df_main = df_main_copy

testdf_16 = df_main[df_main.file_id == 18]    # taking out 16th file for testing purpose se
print("no of pts in test file " , testdf_16.shape)

df_main = df_main[df_main.file_id != 18]
print("remaining pts for training and testing " , df_main.shape )

testdf_16['label'].value_counts()
```

```
no of pts in test file  (606, 82)
remaining pts for training and testing  (2874, 82)
```

Out[78]:

```
0    558
1     29
2     19
Name: label, dtype: int64
```

In [79]:

```
df_main['fft_ax'] = preprocessing.scale(df_main['fft_ax'])
df_main['fft_ay'] = preprocessing.scale(df_main['fft_ay'])
df_main['fft_az'] = preprocessing.scale(df_main['fft_az'])

df_main['sp_ax'] = preprocessing.scale(df_main['sp_ax'])
df_main['sp_ay'] = preprocessing.scale(df_main['sp_ay'])
df_main['sp_az'] = preprocessing.scale(df_main['sp_az'])
df_main.head()
```

```
/home/tolani/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:1:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/s
table/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pand
as-docs/stable/indexing.html#indexing-view-versus-copy)
  """Entry point for launching an IPython kernel.
/home/tolani/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:2:
 SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/s
table/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pand
as-docs/stable/indexing.html#indexing-view-versus-copy)

/home/tolani/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:3:
 SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/s
table/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pand
as-docs/stable/indexing.html#indexing-view-versus-copy)
  This is separate from the ipykernel package so we can avoid doing imports
 until
/home/tolani/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/dat
a.py:164: UserWarning: Numerical issues were encountered when centering the
 data and might not be solved. Dataset may contain too large values. You may
 need to prescale your features.
  warnings.warn("Numerical issues were encountered "
/home/tolani/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:5:
 SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/s
table/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pand
as-docs/stable/indexing.html#indexing-view-versus-copy)
  """
/home/tolani/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:6:
 SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/s
table/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pand
as-docs/stable/indexing.html#indexing-view-versus-copy)
```

```
/home/tolani/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:7:
 SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy)
  import sys
```

Out[79]:

	mean_ax	mean_ay	mean_az	mean_gx	mean_gy	mean_gz	sd_ax	sd_ay	sd_az
0	-0.333256	-0.529265	9.790426	0.001520	-0.000742	-0.004106	0.242658	0.162682	0.286165
0	-0.422345	-0.421185	9.919289	0.005632	-0.011685	-0.034437	0.254251	0.158947	0.201366
0	-0.312917	-0.081383	9.855358	0.006088	-0.000170	-0.020531	0.279257	0.514140	0.223400
0	-0.374978	0.672824	9.570090	0.008233	-0.016506	-0.037798	0.235341	0.213451	0.271166
0	-0.313581	0.429496	9.818923	0.000324	-0.010486	-0.039492	0.153429	0.272872	0.344815

5 rows × 82 columns

dividing data into feature matrix and a target vector

In [80]:

```
data = np.array(df_main)

x = data[:,0:-4]
y = data[:,-2:-1]
y = y.astype(int)
```

In [81]:

```
#y_train
print(x.shape)
y.shape
```

```
(2874, 78)
```

Out[81]:

```
(2874, 1)
```

In [ ]:

```

```

In [82]:

```
#df_main.isnull().any()
```

applying pca on data to visualize it , coln std the feature

matrix

In [83]:

```
# Data-preprocessing: Standardizing the data matrix 'x'

from sklearn.preprocessing import StandardScaler
standardized_data = StandardScaler().fit_transform(x)
print(standardized_data.shape)

# coln std our feature matrix

x = standardized_data
```

```
(2874, 78)

/home/tolani/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.
py:475: DataConversionWarning: Data with input dtype object was converted to
float64 by StandardScaler.
  warnings.warn(msg, DataConversionWarning)
```

In [84]:

```
# initializing the pca

from sklearn import decomposition
pca = decomposition.PCA()
```

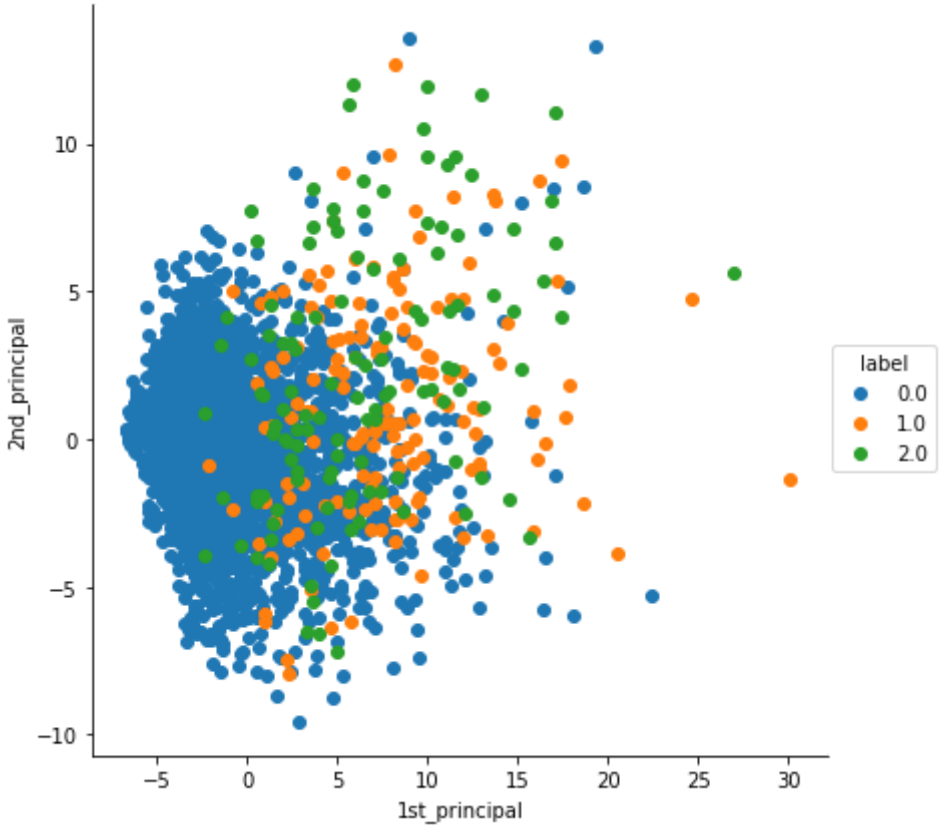
In [85]:

```
# configuring the parameteres
# the number of components = 2
pca.n_components = 2     # so as we wnt top two eigen vectors we pass 2 here
pca_data = pca.fit_transform(x)  # note tht sampled_data has been standardized already

# pca_reduced will contain the 2-d projects of simple data
print("shape of pca_reduced.shape = ", pca_data.shape)
```

```
shape of pca_reduced.shape =  (2874, 2)
```

In [86]:

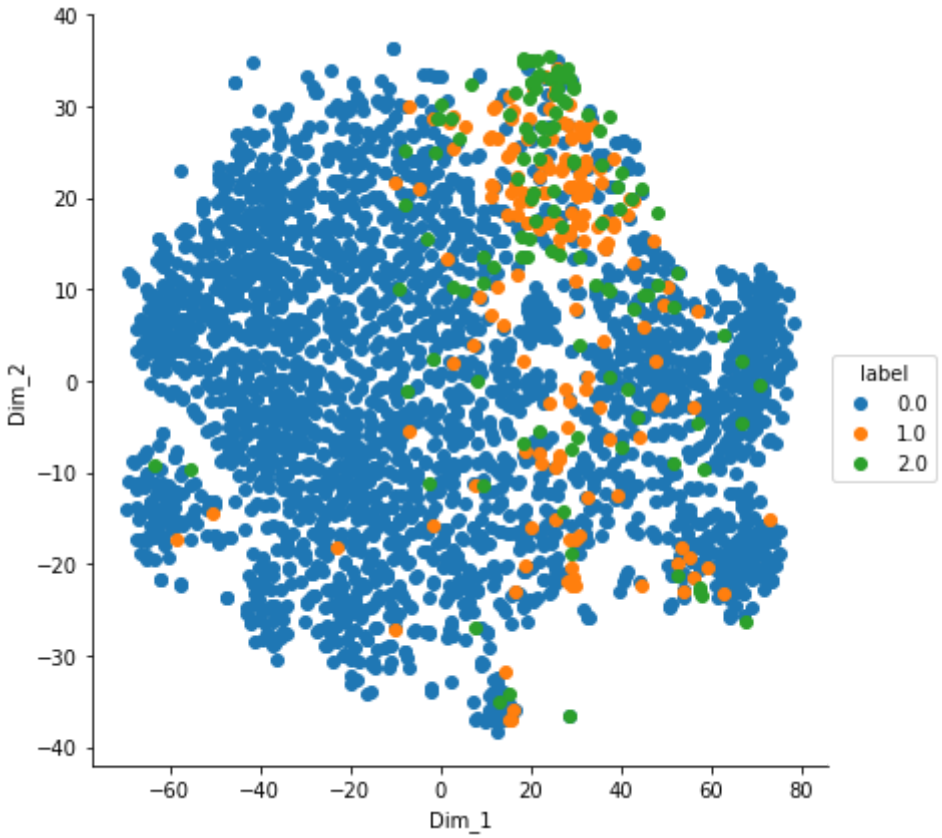
```
# attaching the label for each 2-d data point
import seaborn as sn

#print(pca_data.shape , " " , y.shape)

pca_data = np.hstack((pca_data, y))

# creating a new data fram which help us in ploting the result data
pca_df = pd.DataFrame(data=pca_data, columns=("1st_principal", "2nd_principal", "label"))
sn.FacetGrid(pca_df, hue="label", size=6).map(plt.scatter, '1st_principal', '2nd_principal'
plt.show()

# https://scipython.com/book/chapter-6-numpy/examples/vstack-and-hstack/
```



applying T SNE on dataset for better visualization of data

In [22]:

```
# TSNE

from sklearn.manifold import TSNE

model = TSNE(n_components=2, random_state=0)

tsne_data = model.fit_transform(x)

# creating a new data frame which help us in ploting the result data
tsne_data = np.hstack((tsne_data, y))
tsne_df = pd.DataFrame(data=tsne_data, columns=("Dim_1", "Dim_2", "label"))

# Ploting the result of tsne
sn.FacetGrid(tsne_df, hue="label", size=6).map(plt.scatter, 'Dim_1', 'Dim_2').add_legend()
plt.show()
```

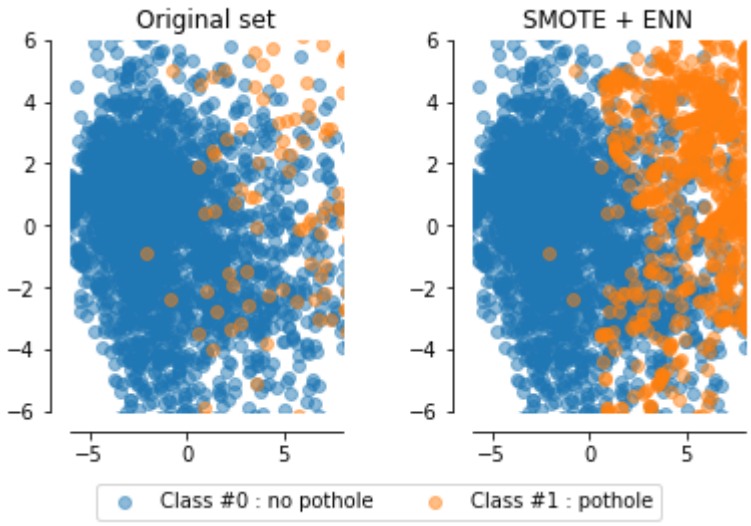


balancing the data by upsampling & downsampling method : SMOTE + ENN

In [87]:

```
import matplotlib.pyplot as plt
from sklearn.datasets import make_classification
from sklearn.decomposition import PCA

from imblearn.combine import SMOTEENN
from imblearn.over_sampling import SMOTE
from imblearn.ensemble import BalanceCascade

print(__doc__)

# Generate the dataset
#X, y = make_classification(n_classes=2, class_sep=2, weights=[0.1, 0.9],
#                           n_informative=3, n_redundant=1, flip_y=0,
#                           n_features=20, n_clusters_per_class=1,
#                           n_samples=100, random_state=10)

# Instanciate a PCA object for the sake of easy visualisation
pca = PCA(n_components=2)
# Fit and transform x to visualise inside a 2D feature space
X_vis = pca.fit_transform(x)

# Apply SMOTE + ENN
sm = SMOTE(k = 5 , kind='svm')
X_resampled, y_resampled = sm.fit_sample(x,y)
X_res_vis = pca.transform(X_resampled)

# Two subplots, unpack the axes array immediately
f, (ax1, ax2) = plt.subplots(1, 2)

#print(X_vis.shape, " " , X_res_vis.shape , " " , y.shape , " " , y_resampled.shape)

y = y.reshape(y.shape[0],)
#print(y_resampled)

c0 = ax1.scatter(X_vis[y == 0, 0], X_vis[y == 0, 1], label="Class #0 : no pothole",alpha=0.
c1 = ax1.scatter(X_vis[y == 1, 0], X_vis[y == 1, 1], label="Class #1 : pothole",alpha=0.5)
ax1.set_title('Original set')

ax2.scatter(X_res_vis[y_resampled == 0, 0], X_res_vis[y_resampled == 0, 1],label="Class #0
ax2.scatter(X_res_vis[y_resampled == 1, 0], X_res_vis[y_resampled == 1, 1],label="Class #1
ax2.set_title('SMOTE + ENN')

# make nice plotting
for ax in (ax1, ax2):
    ax.spines['top'].set_visible(False)
    ax.spines['right'].set_visible(False)
    ax.get_xaxis().tick_bottom()
    ax.get_yaxis().tick_left()
    ax.spines['left'].set_position(('outward', 10))
    ax.spines['bottom'].set_position(('outward', 10))
    ax.set_xlim([-6, 8])
    ax.set_ylim([-6, 6])

f.legend((c0, c1), ('Class #0 : no pothole', 'Class #1 : pothole'), loc='lower center',
         ncol=2, labelspacing=0.)
plt.tight_layout(pad=3)
plt.show()
```

```
Automatically created module for IPython interactive environment

/home/tolani/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.
py:578: DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples, ), for example usi
ng ravel().
  y = column_or_1d(y, warn=True)
/home/tolani/anaconda3/lib/python3.6/site-packages/imblearn/utils/deprecatio
n.py:50: DeprecationWarning: 'k' is deprecated from 0.2 and will be removed
in 0.4. Use 'k_neighbors' instead.
  category=DeprecationWarning)
```



In [88]:

```
X_resampled.shape
y_resampled.shape
#y_resampled['label'].value_counts()
a,b = np.unique(y_resampled,return_counts=True)
print(a,b)
```

```
[0 1 2] [2597 2596 2596]
```

applying Logistic Regression one vs rest for multiclassficaiton

In [89]:

```
#acc_sum = 0
#for i in range(100):
x_train,x_test,y_train,y_test = train_test_split(X_resampled,y_resampled,test_size = 0.2)

model = LogisticRegression(multi_class='ovr')
model.fit(x_train,y_train)
y_pred = model.predict(x_test)

accuracy = accuracy_score(y_pred,y_test) * 100
#acc_sum = acc_sum + accuracy
print(accuracy)
#acc_sum/100
print(x_train.shape, " ", x_test.shape)
```

```
89.9229781771502
(6231, 78)   (1558, 78)
```

In [90]:

```
confusion_matrix(y_test, y_pred)
```

Out[90]:

```
array([[441,  18,  18],
       [ 16, 488,  33],
       [ 37,  35, 472]])
```

looking at the most important features in classification task : top 20 features out of 81 features

In [91]:

```
abs_weights = np.fabs(model.coef_)

#print(model.coef_.shape)
#print(model.coef_)

#abs_weights.reshape(78,)

arr0 = abs_weights[0,:]
arr0.reshape(78,)
#print(arr0)
#print(arr0.shape)

sorted_asc = np.argsort(arr0)
#print(sorted_asc)

sorted_desc = np.flip(sorted_asc,axis =0)
#print("sorted arr shape",sorted_desc.shape)

top_15_features = sorted_desc[:20]
top = top_15_features.ravel()
print(top)

features_names = df_main.columns
#print(features_names)
for i in range(0,79):
    if(i in top):
        print(features_names[i])
    else:
        pass
```

```
[27 15 54 11 23 57 41  6 48 28 14 17 29 26 16 36 21 24 38  7]
sd_ax
sd_ay
sd_gz
min_az
min_gx
min_gy
min_gz
max_gx
max_gz
var_ax
var_az
var_gx
var_gy
var_gz
quant1_ax
quant1_az
quant1_gz
quant3_ax
mad_ax
mad_gx
```

applying svm with rbf kernel for multiclassfn

In [92]:

```
model = SVC(kernel='rbf', C = 10)
model.fit(x_train,y_train)
y_pred = model.predict(x_test)

accuracy_score(y_pred,y_test)
```

Out[92]:

```
0.9878048780487805
```

In [93]:

```
confusion_matrix(y_test, y_pred)
```

Out[93]:

```
array([[462,  11,   4],
       [  2, 535,   0],
       [  1,   1, 542]])
```

applying svm with default kernel

In [133]:

```
pca = PCA(0.99)
```

In [134]:

```
pca.fit(x_train)
```

Out[134]:

```
PCA(copy=True, iterated_power='auto', n_components=0.99, random_state=None,
  svd_solver='auto', tol=0.0, whiten=False)
```

In [135]:

```
x_trans_train = pca.transform(x_train)
x_trans_test = pca.transform(x_test)
```

In [136]:

```
x_trans_train.shape
```

Out[136]:

```
(7460, 33)
```

In [137]:

```
model = SVC()
model.fit(x_trans_train,y_train)
y_pred = model.predict(x_trans_test)

accuracy_score(y_pred,y_test)
```

Out[137]:

```
0.986058981233244
```

In [85]:

```
confusion_matrix(y_test, y_pred)
```

Out[85]:

```
array([[606,   3,   4],
       [  4, 593,   0],
       [  9,   0, 595]])
```

Applying nerural network for multi classfn

one hot encoding of classes : https://machinelearningmastery.com/multi-class-classification-tutorial-keras-deep-learning-library/ (https://machinelearningmastery.com/multi-class-classification-tutorial-keras-deep-learning-library/)

In [58]:

```
from sklearn.preprocessing import LabelEncoder
from keras.utils import np_utils

# encode class values as integers
encoder = LabelEncoder()
encoder.fit(y_train)
encoded_y_train = encoder.transform(y_train)
# convert integers to dummy variables (i.e. one hot encoded)
dummy_y_train = np_utils.to_categorical(encoded_y_train)


# encode class values as integers
encoder = LabelEncoder()
encoder.fit(y_test)
encoded_y_test = encoder.transform(y_test)
# convert integers to dummy variables (i.e. one hot encoded)
dummy_y_test = np_utils.to_categorical(encoded_y_test)



#print(dummy_y[0:50])
```

In [59]:

```
# uncomment following to run neural net

#acc = []
#for i in range(50):
# x_train,x_test,y_train,y_test = train_test_split(X_resampled,y_resampled,test_size = 0.33
# model
model_nn = Sequential()
model_nn.add(Dense(units=100,activation="relu",input_dim =x_train.shape[1]))
model_nn.add(Dropout(0.3))
model_nn.add(Dense(units=50,activation="relu"))
model_nn.add(Dropout(0.3))
model_nn.add(Dense(units=10,activation="relu"))
model_nn.add(Dropout(0.3))
model_nn.add(Dense(units=3,activation="sigmoid"))       # 3 units for 3 classes
#model.add(Dense(units=10,activation="softmax"))

#compile
model_nn.compile(optimizer='adam',loss="categorical_crossentropy",metrics=["accuracy"])

#train
model_nn.fit(x_train,dummy_y_train, validation_data= (x_test, dummy_y_test), batch_size= 50
```

```
4870/4870 [==============================] - 0s 50us/step - loss: 0.1839
- acc: 0.9458 - val_loss: 0.2170 - val_acc: 0.9409
Epoch 12/100
4870/4870 [==============================] - 0s 53us/step - loss: 0.1755
- acc: 0.9511 - val_loss: 0.2051 - val_acc: 0.9466
Epoch 13/100
4870/4870 [==============================] - 0s 53us/step - loss: 0.1544
- acc: 0.9530 - val_loss: 0.2055 - val_acc: 0.9475
Epoch 14/100
4870/4870 [==============================] - 0s 53us/step - loss: 0.1584
- acc: 0.9530 - val_loss: 0.2057 - val_acc: 0.9417
Epoch 15/100
4870/4870 [==============================] - 0s 54us/step - loss: 0.1531
- acc: 0.9567 - val_loss: 0.2053 - val_acc: 0.9475
Epoch 16/100
4870/4870 [==============================] - 0s 52us/step - loss: 0.1456
- acc: 0.9565 - val_loss: 0.2116 - val_acc: 0.9458
Epoch 17/100
4870/4870 [==============================] - 0s 51us/step - loss: 0.1347
- acc: 0.9575 - val_loss: 0.2014 - val_acc: 0.9499
```

In [60]:

```
y_pred = model_nn.predict_classes(x_test)
accuracy_score(y_test, y_pred)
```

Out[60]:

```
0.9704433497536946
```

In [61]:

```
cnf_mat = confusion_matrix(y_test, y_pred)
```

In [62]:

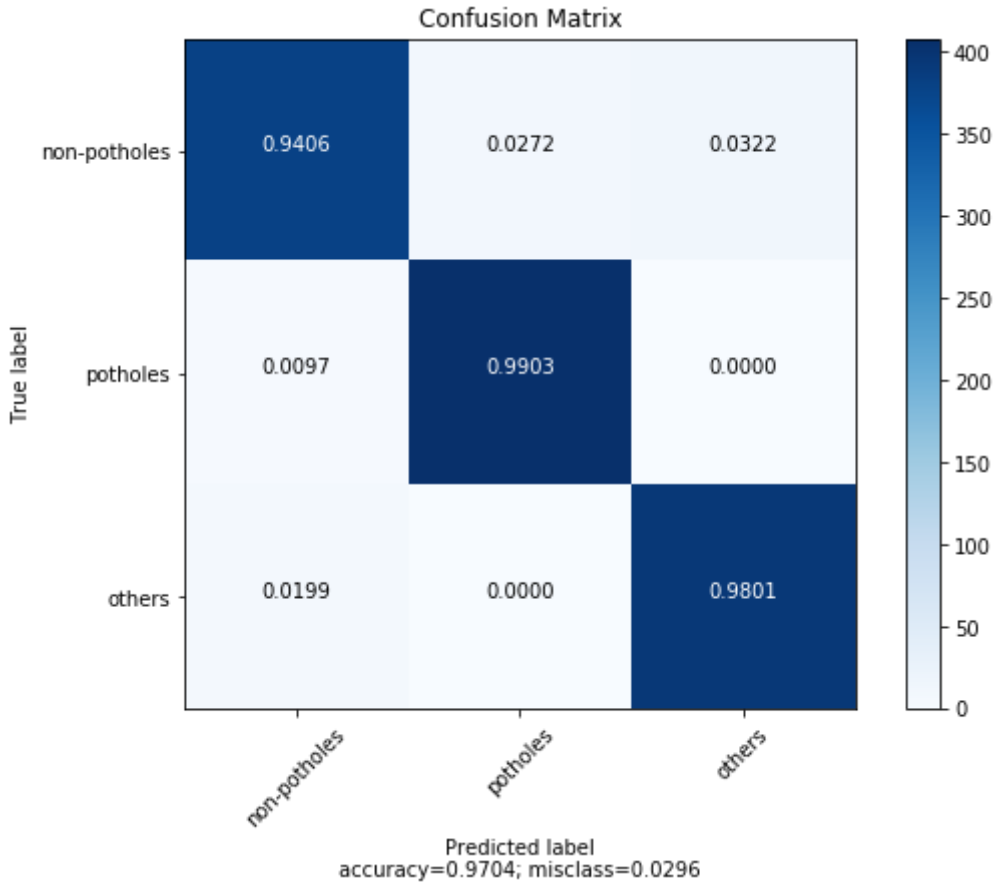
```
f1_score(y_test, y_pred, average='weighted')
```

Out[62]:

```
0.9703257102162579
```

In [63]:

```
plot_confusion_matrix(cm           =cnf_mat,
                      normalize    = True,
                      target_names = ['non-potholes', 'potholes', 'others'],
                      title        = "Confusion Matrix")
```



saving model

In [94]:

```
import pickle
pickle.dump(model,open('model_svm.pkl', 'wb'))
```

In [95]:

```
loaded_model = pickle.load(open('model_svm.pkl', 'rb'))
```

In [96]:

```
data_test = np.array(testdf_16)

x = data_test[:,0:-4]
y = data_test[:,-2:-1]
y = y.astype(int)

x.shape
y.shape


from sklearn.preprocessing import StandardScaler
standardized_data = StandardScaler().fit_transform(x)
print(standardized_data.shape)

# coln std our feature matrix

x = standardized_data
```

```
(606, 78)

/home/tolani/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.
py:475: DataConversionWarning: Data with input dtype object was converted to
float64 by StandardScaler.
  warnings.warn(msg, DataConversionWarning)
```

In [97]:

```
confusion_matrix(y,loaded_model.predict(x))
```

Out[97]:

```
array([[539,  11,   8],
       [ 14,  10,   5],
       [ 10,   2,   7]])
```

In [98]:

```
from keras.models import load_model

model_nn.save('model_nn.h5')

model_nn2 = load_model('model_nn.h5')

confusion_matrix(y,model_nn2.predict_classes(x))
```

Out[98]:

```
array([[537,  13,   8],
       [ 18,   5,   6],
       [  4,   2,  13]])
```

converting pandas df to numpy array

In [ ]:

```
np_array_for_server = df_main.as_matrix()
```

In [ ]:

```
np_array_for_serveray_for_server.shape
```

In [ ]:

```
np_array_for_server
```

In [ ]:

```

```