# United Way MLPP: MultinomialNB Optical Digits Example

```
# -*- coding: utf-8 -*-


"""
Title:                  Sample MNBayes1
Project Descriptor:     Machine Learning Pilot Project (United Way)
Project ID:             2016SoE013 (STAT_570_Consulting)
Record:
Author:                 bmarron
Origin Date:            28 May 2016
"""

"""
This script provides an example of the use of the Multinomial Bayes
algorithm as applied to the  Optical Recognition of Handwritten Digits
dataset. The original source of the dataset:
        E. Alpaydin, C. Kaynak
        Department of Computer Engineering
        Bogazici University, 80815 Istanbul Turkey
        alpaydin@boun.edu.tr
        July 1998

The dataset is available at UC Irivine Machine Learning Repository.
"""

"""
Important websites:
    #UCI Machine Learning Repository
http://archive.ics.uci.edu/ml/

    #Python machine learning tools
http://scikit-learn.org/
"""

#%% Import packages


import numpy as np
import cPickle
import urllib
import csv
import operator
import pylab as pl
import pandas as pd
from time import time
from sklearn import cross_validation
from sklearn.utils.fixes import bincount
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import SVC
from sklearn.svm import LinearSVC

#%%

#Download and save the raw, UCI opttical digits data and data info sheet

'''
http://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits
http://archive.ics.uci.edu/ml/machine-learning-databases/optdigits/optdigits.tra
http://archive.ics.uci.edu/ml/machine-learning-databases/optdigits/optdigits.tes
'''

```

```

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.5,
    random_state=random_state)

#%% Confirm data content with info sheet about training dataset

#Data sheet: 0=376, 1=389, 2=380, 9=382


test0 = np.where(raw_optdigits_tr[:, 64] == 0)
test0 = list(test0[0])

test1 = np.where(raw_optdigits_tr[:, 64] == 1)
test1 = list(test1[0])

test2 = np.where(raw_optdigits_tr[:, 64] == 2)
test2 = list(test2[0])

test9 = np.where(raw_optdigits_tr[:, 64] == 9)
test9 = list(test9[0])

print "zeros: %s" %(len(test0))
print "ones: %s" %(len(test1))
print "twos: %s" %(len(test2))
print "nines: %s" %(len(test9))

#%% Split training dataset(tr_d) and test dataset(te_d) into features (X)

#and class (y)


tr_d_X, tr_d_y = raw_optdigits_tr[:,0:64], raw_optdigits_tr[:, 64]
te_d_X, te_d_y = raw_optdigits_te[:,0:64], raw_optdigits_te[:, 64]


#%% Save the split (processed) data


# Save tr_d_X

with open("tr_d_X.pkl", 'wb') as f:
    cPickle.dump(tr_d_X, f, protocol=2)

# Save tr_d_y

with open("tr_d_y.pkl", 'wb') as f:
    cPickle.dump(tr_d_y, f, protocol=2)

# Save te_d_X

with open("te_d_X.pkl", 'wb') as f:
    cPickle.dump(te_d_X, f, protocol=2)

# Save te_d_y

with open("te_d_y.pkl", 'wb') as f:
    cPickle.dump(te_d_y, f, protocol=2)

#%%

```

```
#Reload the processed data into Python, if needed

tr_d_X = cPickle.load(open("/home/bmarron/Desktop/tr_d_X.pkl","rb"))

#Reload the processed data

tr_d_y = cPickle.load(open("/home/bmarron/Desktop/tr_d_y.pkl","rb"))

#Reload the processed data

te_d_X = cPickle.load(open("/home/bmarron/Desktop/te_d_X.pkl","rb"))

#Reload the processed data

te_d_y = cPickle.load(open("/home/bmarron/Desktop/te_d_y.pkl","rb"))


#%% Based on the method  at

#http://scikit-learn.org/stable/auto_examples/text/mlcomp_sparse_document_classification.html#example-text-mlcomp-sparse-document-classification-py

# Benchmark classifiers


def benchmark(clf_class, params, name):
    print("parameters:", params)
    t0 = time()
    clf = clf_class(**params).fit(tr_d_X, tr_d_y)
    print("done in %fs" % (time() - t0))

    if hasattr(clf, 'coef_'):
        print("Percentage of non zeros coef: %f"
              % (np.mean(clf.coef_ != 0) * 100))
        # .coef_ output is a read-only file

        # Python does not define a read-write variable!

        # Export the wgt vector data from .coef_

        wgts = clf.coef_
        with open('wgts_MNBayes.csv', 'w+') as f:
            a= csv.writer(f, delimiter=',')
            a.writerows(wgts)


    print("Predicting the outcomes of the testing set")
    t0 = time()
    pred = clf.predict(te_d_X)
    print("done in %fs" % (time() - t0))

    print("Classification report on test set for classifier:")
    print(clf)
    print()
    print(classification_report(te_d_y, pred))

    cm = confusion_matrix(te_d_y, pred)
    df = pd.DataFrame(cm)
    print("Confusion matrix:")
    print(cm)
    print df.to_latex()


    print ("Accuracy:")
    print accuracy_score(te_d_y, pred)

    # Show confusion matrix

    pl.matshow(cm)
    pl.title('Confusion matrix of the\n %s classifier \n (alpha=0.03)' % name,
        fontsize=12, y=1.10)
    pl.colorbar()
    pl.savefig("MNBayes1.pdf")

#%%

print("Testbenching a MultinomialNB classifier...")
parameters = {'alpha': 0.03}

benchmark(MultinomialNB, parameters, 'MultinomialNB')

pl.show()

#%%

# coef_ is empirical log probability of features given a class, P(x_i|y)

# coef_ in MultinomialNB mirrors the feature_log_prob_ ; useful

#for interpreting MultinomialNB as a linear model


# Import 'wgts_MNBayes.csv' with Spyder (green arrow); import as 'array'

# Take the absolute value of the wgts

# Define a list of feature numbers (0 to 63); actual features run i=1,2,...63


############

#import 'wgts_MNBayes.csv' with Spyder (green arrow)

############

abs_csv = abs(wgts_MNBayescsv)
features =np.arange(64)

# Create tuples (feature #, feature value)

wgts_features=[]
for i in range(10):
    x=zip(features, abs_csv[i])
    wgts_features.append(x)

# Reverse sort on the the second element in each tuple

for i in range(10):
    wgts_features[i]=sorted(wgts_features[i], key=operator.itemgetter(1),
        reverse=True)

df2 = pd.DataFrame(wgts_features)
df2.to_csv('sorted_wgts_features_MNBayes.csv')

#%% Ouput w/o panda dataframe


with open('sorted_wgts_features_MNBayes.csv', 'w+') as f:
            a= csv.writer(f, delimiter=',')
            a.writerows(wgts_features)
```