# coding: utf-8

# In[11]:

import math

import numpy as np

import pandas as pd

from sklearn.linear\_model import LogisticRegressionCV

from sklearn.naive\_bayes import GaussianNB

from sklearn.cross\_validation import StratifiedKFold, cross\_val\_score, cross\_val\_predict

from sklearn import preprocessing, metrics

from sklearn.metrics import accuracy\_score, classification\_report

df = pd.read\_csv('/Users/Purva Sawant/breastcancer.csv', header=0)

df.drop(['Id'], 1, inplace=True)

encoder = preprocessing.LabelEncoder()

df['Class'] = encoder.fit\_transform(df['Class'])

X = np.array(df.drop(['Class'], 1))

y = np.array(df['Class'])

scaler = preprocessing.StandardScaler().fit(X)

X = scaler.transform(X)

# 10 fold stratified cross validation

kf = StratifiedKFold(y, n\_folds=10, random\_state=None, shuffle=True)

# Logistic regression with 10 fold stratified cross-validation using model specific cross-validation in scikit-learn

lgclf = LogisticRegressionCV(Cs=list(np.power(10.0, np.arange(-10, 10))),penalty='l2',scoring='roc\_auc',cv=kf)

lgclf.fit(X, y)

y\_pred = lgclf.predict(X)

# Show classification report for the best model (set of parameters) run over the full dataset

print("Classification report:")

print(classification\_report(y, y\_pred))

# Show accuracy and area under ROC curve

print("Accuracy: for Logistic %0.3f" % accuracy\_score(y, y\_pred, normalize=True))

print("Aucroc: %0.3f" % metrics.roc\_auc\_score(y, y\_pred))

# Naive Bayes with 10 fold stratified cross-validation

nbclf = GaussianNB()

scores = cross\_val\_score(nbclf, X, y, cv=kf, scoring='roc\_auc')

# Show accuracy statistics for cross-validation

print("Accuracy for Naive Bayes : %0.3f" % (scores.mean()))

print("Aucroc: %0.3f" % metrics.roc\_auc\_score(y, cross\_val\_predict(nbclf, X, y, cv=kf)))

# In[12]:

# The scoring function that will use the Naive Bayes Classifier to classify new data points

def SuggestDiagnosis(Cl\_thickness, Cell\_size, Cell\_shape, Marg\_adhesion, Epith\_c\_size,

Bare\_nuclei, Bl\_cromatin, Normal\_nucleoli, Mitoses):

X = np.column\_stack([Cl\_thickness, Cell\_size, Cell\_shape, Marg\_adhesion, Epith\_c\_size,

Bare\_nuclei, Bl\_cromatin, Normal\_nucleoli, Mitoses])

X = scaler.transform(X)

return encoder.inverse\_transform(nbclf.predict(X)).tolist()

# In[8]:

import tabpy\_client

connection = tabpy\_client.Client('http://localhost:9004/')

connection.deploy('DiagnosticsDemo2',

SuggestDiagnosis,

'Returns diagnosis suggestion based on ensemble model trained using Wisconsin Breast Cancer dataset')