Project Report studentID: 15205032

Analysis of Wisconsin Breast Cancer Dataset

Objective: Predictive analysis of the **Wisconsin Breast Cancer** data (Extracted from Kaggle Datasets database) in view of predicting whether the tumor would be classified as malignant or benign based on the given features. And compare the accuracies of Predictions by various classification methods.

Data description provided by the source [1]:

The given features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image. n the 3-dimensional space is that described in: [K. P. Bennett and O. L. Mangasarian: "Robust Linear Programming Discrimination of Two Linearly Inseparable Sets", Optimization Methods and Software 1, 1992, 23-34]. The data set is also available through the WDBC website, UCI Machine Learning Repository.

The data set has 32 attributes and 569 observations. All feature values are recoded with four significant digits. There are no missing attribute values. Reported 357 tumors benign and 212 malignant. **Attributes details:** 1. ID number 2. Diagnosis (M = malignant, B = benign) and 10 real-valued features are computed for each cell nucleus: i) radius (mean of distances from center to points on the perimeter), ii) texture (standard deviation of gray-scale values), iii) perimeter, iv) area, v) smoothness (local variation in radius lengths), vi) compactness, vii) concavity (severity of concave portions of the contour), vii) concave points (number of concave portions of the contour), iX) symmetry, x) fractal dimension. The mean (variables with _mean in the attribute name), standard error (with '_se' in the attribute name) and "worst" or largest (with '_worst' in the attribute name) of these features were computed for each image, resulting in 30 features. For example, field 4 is Mean Texture, field 14 is Texture Standard error, field 24 is Worst Texture.

Data Exploration:

The extra empty column and ID variable have been removed and the 'Diagnoses' variable which would be the response variable in the predictive analysis have been converted to binary variable from the string format. To facilitate view of the correlations and to look at the histograms of the variables 3 subsets have been created such that the _mean variables constitute one subset, the _se variables constitute one subset and _worst variables another subset. The histograms plotted for all the 3 subsets showed the non normal distributions. The data transformation could reduce these distribution problems. The data have been log transformed for the 3 subsets. The box plot for original subset and the log transformed subset for each subset has been presented in Fig.1. Due to the lengthy variable names, the labels are affected. The box plot produced clearly showed the vast differences in the distribution of values for different variables.

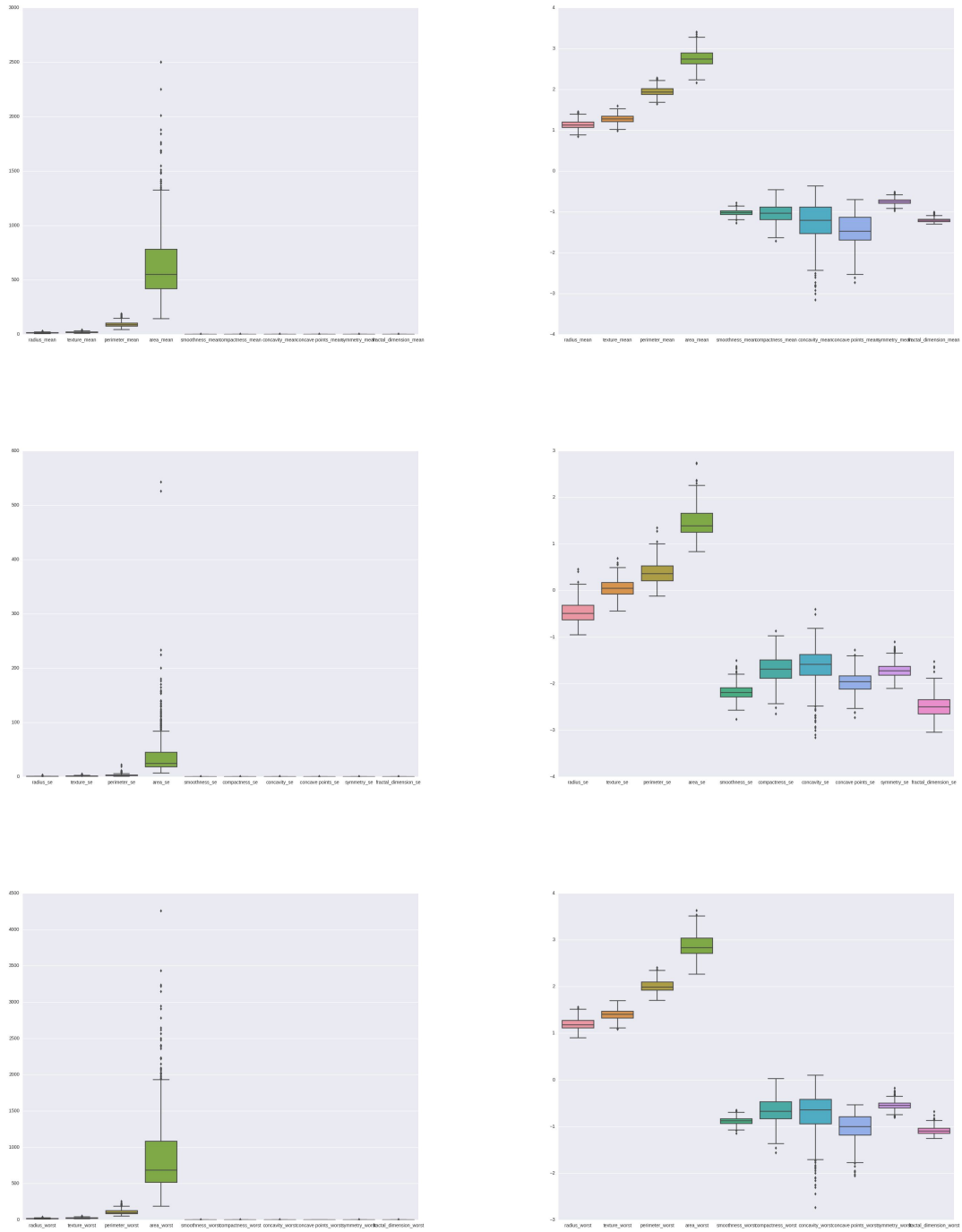

Fig. 1 First row _mean variables original (left) and log transformed (right)
Second row _se variables original (left) and log transformed (right)
Third row _worst variables original (left) and log transformed (right)

Log transformation of the data brought the overall distribution up. Expected multicollinearity among the variables radius, perimeter, area. Similarly between concavity, concavity points, symmetry and fractal dimension. Overlapping have been seen among the variables smoothness, compactness, concavity, concavity-points, symmetry and fractal dimension. The model analysis have been carried out on the normalized data and the log transformed data using various classifiers and the outcomes have been reported in the following section.

Predictive Analysis:

Grid search CV algorithm [2] used to optimise the exhaustive list of chosen parameters and give the best fit for the classifiers Logistic Regression, Random Forest, Extra trees, Gradient boosting and Adaboosting with 15 cross validations, and accuracy as the scoring scheme. The n-jobs = -1 parameter make use of the processors on the system to run the algorithm parallely.

The given data split into train test sets carried out using scikit learn, train_tes_split method at a given random state, chosen 3:1 split. The classification performed with log transformed data produced infinity, NaN in the process. To eliminate them the variables with concavity and concavity points (total 6 variables) have been dropped from the log transformed dataset.

Noted in the Gridsearch process different optimal parameters have been selected for the normalized and log transformed datasets. The logistic and Adaboost classification data is presented in open table format below, to look at the optimal parameters for the 2 types of datasets and the statistical outcomes precision, support etc for each class in the response variable separately.

Classification by Logistic Regression:

	Normalized Data				Log-transformed Data			
Classification report: (optimal C = 10)					(Optimal C = 2000)			
	Precision	recall	f1-score	support	Precision	recall	f1-score	support
0	0.99	0.99	0.99	88	0.97	0.97	0.97	115
1	0.98	0.98	0.98	55	0.93	0.95	0.94	56
total	0.99	0.99	0.99	143	0.96	0.96	0.96	171

Classification by Adaboost classifier:

	Normalized Data				Log-transformed Data			
Classification report: (n-estimators = 200)					(Optimal n-estimators = 400)			
	precision	recall	f1-score	support	precision	recall	f1-score	support
0	0.99	0.95	0.97	88	0.99	0.99	0.99	115
1	0.93	0.98	0.96	55	0.98	0.98	0.98	56
total	0.97	0.97	0.97	143	0.99	0.99	0.99	171

The precision varied ranging from 95% to 99% depending on the random state of selection and the train, test set ratio. Noted that the recall, f1-score and the precision values are equal for a specific classifier. The algorithms working at equal levels with precision and recall and hence F1-score which depend on these to working at their level. The support value (number of true occurrences) for the Benign class is higher than for Malignant class, with every classification approach. Part of the required parameters in classifiers Adaboost and Gradient boosting are chosen the default values. By tuning those appropriately, the support for Malignant class might improve.

Table. 1 Percentage precision of each classifier employed on the data

Classifier	Normalized Data	Log Transformed Data
Logistic Regression	99% C =10	96% C=1000
Random Forest	99% N-estimators 150	97% N-estimators 150
Extra Trees	98% N-estimators 100	97% N-estimators 100
Gradient Boosting	97% N-estimators 150	95% N-estimators 150
Adaboosting	97% N-estimators 200	99% N-estimators 400

Conclusions:

Log transformed data classification scores are 1 to 2% less than the normalized scores. Due to the random state dependence of the train test split, the observation is sceptical. The normalized data with Logistic regression produced 99% accuracy. Investigated the classification of the normalized data set using 2 other classifiers K nearest neighbours (KNN) and support vector machines (SVM). k=3 nearest neighbours have been used for KNN. SVC with probability, KNN (k=3) classifiers prediction accuracy turned out 99.99%. The ROC curve is shown in Fig. 2.

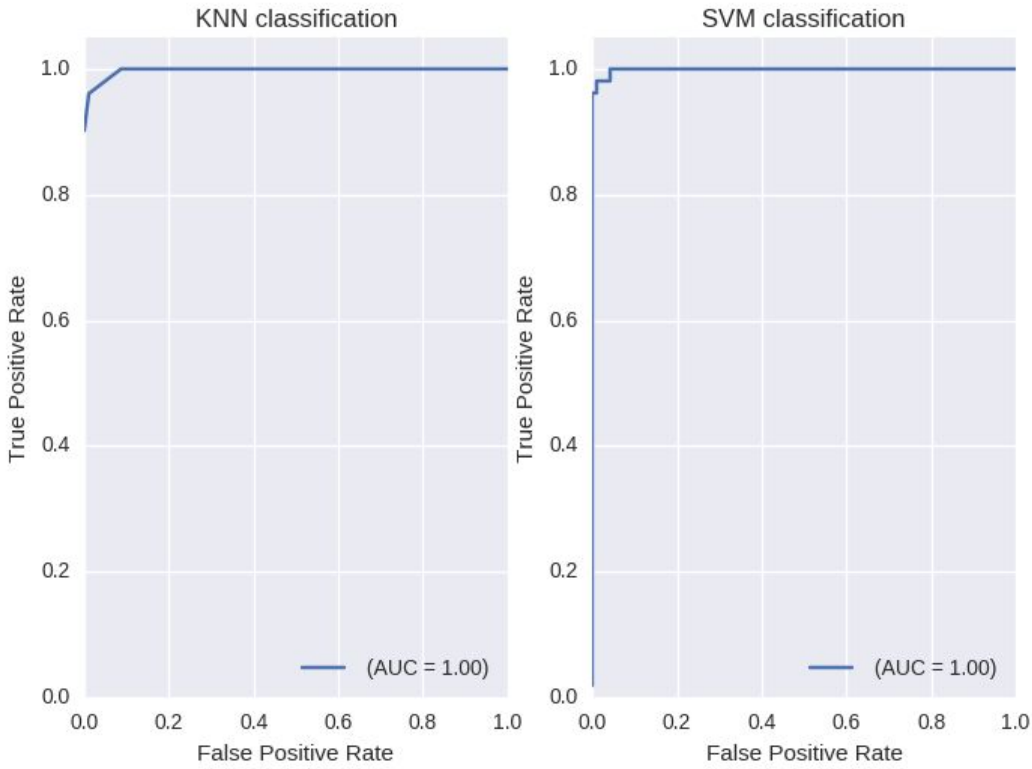

Fig. 2 ROC curve of KNN and SVM classifiers

The expected collinearities present in the data have not affected the prediction accuracies. But the best model, with only the important features could not be decided, for example the analysis done by dropping perimeter affected the accuracies depicting it as an important feature in the prediction process along with area variable. Overall predicted scores are good using SVM, KNN, Logistic Regression. Because of the critical nature of the diagnosis of the tumors, it is advised that the models have to be suggested with a caution that there are no discrepancies and on the basis of analysing heavily populated data sets with all the related features. Again how large is the length requirement and how many features are important depends on the problem of interest and speculative.

References:

1. https://www.kaggle.com/zcbmxvnyico/d/uciml/breast-cancer-wisconsin-data
2. http://scikit-learn.org/stable/modules/grid_search.html
3. http://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_recall_fscore_support.html

Appendix:
Python Code:

```
# Sree Latha Vallabhaneni Student_Id: 15205032
#project under Strand 1: Statistical modelling/machine Learning
# Analysis of data, downloaded from kaggle website
#https://www.kaggle.com/zcbmxvnyico/d/uciml/breast-cancer-wisconsin-data

#import libraries and required modules
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
import seaborn as sbn
from sklearn.metrics import classification_report,roc_curve, auc
from sklearn.model_selection import train_test_split

#classifiers
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import ExtraTreesClassifier,GradientBoostingClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC,LinearSVC
from sklearn.gaussian_process.kernels import RBF

#Load data.
data = pd.read_csv('Documents/dataProgPython/input/breastCancerW.csv')
data.head(10)
type(data)
#data cleaning, arrange or transform data variables
data = data.drop("id",1)
data = data.drop("Unnamed: 32",1)
data.head(3)
data[[0]].head(10)

#checking for null values
data.apply(lambda x: sum(x.isnull()),axis=0) #no null values

data.describe()# distribution of mean values is different for the different
characterstics
data.info()# size 569 * 30 float 1 catogorical -'diagnosis'
```

```
dir(data)
data.diagnosis.unique() # view categories

#map 'Diagnosis' string names(B/M) to binary(0/1)
cl = {'M': 1, 'B': 0}
data['diagnosis'] = data['diagnosis'].map(cl).astype(int)
data['diagnosis'].head()

#--------------------exploratory analysis--------------------#

print(data.columns)
#10 subtitled _mean, 10 with _se and another 10 with _worst excluding diagnosis

# cut into chunks 1: _mean, 2: _se and 3: _worst and explore histograms etc
dcut1=data.ix[:,1:11].copy()
dcut2=data.ix[:,11:21].copy()
dcut3=data.ix[:,21:].copy()

#check for each chunk variables
print(dcut1.columns) # variables with _mean in column name
print(dcut2.columns) # variables with _se in column name
print(dcut3.columns) #variables with _worst in column name

#histograms
dcut1 = (dcut1-dcut1.mean())/dcut1.std()
dcut2 = (dcut2-dcut2.mean())/dcut2.std()
dcut3 = (dcut3-dcut3.mean())/dcut3.std()

dcut1.hist(bins=10)
dcut2.hist(bins=10)
dcut3.hist(bins=10)

#correlation in each chunk
dcut1.corr(method='pearson', min_periods=1)
dcut2.corr(method='pearson', min_periods=1)
dcut3.corr(method='pearson', min_periods=1)
#
#plot predictor variables with _mean
#dcut1.boxplot()
plt.rcParams['figure.figsize']=(15,12)
sbn.boxplot(dcut1)
plt.show()
```

```
plt.savefig('Documents/dataProgPython/output/boxCut1.png')
plt.clf()
#log transform might bring down the difference in distribution
#log transform cut1 (_mean data) and see
cut1_columns = ['radius_mean', 'texture_mean', 'perimeter_mean',
'area_mean','smoothness_mean', 'compactness_mean', 'concavity_mean',
'concave points_mean', 'symmetry_mean','fractal_dimension_mean']
log_dcut1 = dcut1.loc[:,cut1_columns]
log_dcut1[cut1_columns] = log_dcut1[cut1_columns].apply(np.log10)

#plot log transformed predictor variables with _mean
plt.rcParams['figure.figsize']=(15,12)
sbn.boxplot(log_dcut1)
plt.show()
plt.savefig('Documents/dataProgPython/output/boxLogCut1.png')
plt.clf()
#
#plot predictor variables with _se
plt.rcParams['figure.figsize']=(15,12)
sbn.boxplot(dcut2)
plt.show()
plt.savefig('Documents/dataProgPython/output/boxCut2.png')
plt.clf()
# log transform dcut2 variables
cut2_columns = ['radius_se', 'texture_se', 'perimeter_se',
'area_se','smoothness_se', 'compactness_se', 'concavity_se',
'concave points_se', 'symmetry_se','fractal_dimension_se']
log_dcut2=dcut2.loc[:,cut2_columns]
log_dcut2[cut2_columns] = log_dcut2[cut2_columns].apply(np.log10)

#plot log transformed predictor variables with _se (dcut2)
plt.rcParams['figure.figsize']=(15,12)
sbn.boxplot(log_dcut2)
plt.show()
plt.savefig('Documents/dataProgPython/output/boxLogCut2.png')
plt.clf()
#
#plot predictor variables with _worst (dcut3)
plt.rcParams['figure.figsize']=(15,12)
sbn.boxplot(dcut3)
plt.show()
plt.savefig('Documents/dataProgPython/output/boxCut3.png')
```

```
# log transform dcut3 variables
cut3_columns = [ 'radius_worst','texture_worst', 'perimeter_worst',
'area_worst','smoothness_worst','compactness_worst', 'concavity_worst',
'concave points_worst' , 'symmetry_worst','fractal_dimension_worst']
log_dcut3 =dcut3.loc[:,cut3_columns]
log_dcut3[cut3_columns] = log_dcut3[cut3_columns].apply(np.log10)

#plot log transformed predictor variables (with _worst) (dcut3)
plt.clf()
plt.rcParams['figure.figsize']=(15,12)
sbn.boxplot(log_dcut3)
plt.show()
plt.savefig('Documents/dataProgPython/output/boxLogCut3.png')


####Using grid search to tune for optimal parameters for the classifiers

def fit_model(model_type,model_name,tuning_parameters):
clf = GridSearchCV(model_type, tuning_parameters, cv=15,
scoring = 'accuracy')
clf.fit(X_train, y_train)
print(" Best parameters with train data for the classifier, %s" %model_name )
print( clf.best_params_)
print("Grid scores for train data: ")
means = clf.cv_results_['mean_test_score']
stds = clf.cv_results_['std_test_score']
for mean, std, params in zip(means, stds, clf.cv_results_['params']):
print(" %0.5f (+/-%0.03f) for %r" % (mean, std * 2, params))
y_pred = clf.predict(X_test)
print(" Classification report:")
print(classification_report(y_test, y_pred))
Return

#prepare data for model analysis
data.head()
y = data['diagnosis']
y.head()

#Gridsearch with normal data (without dropping any columns)
X = data.ix[:,1:]
```

```
#Normalize the data
X = (X-X.mean())/X.std()
X.head()
#Split dataset into Train and Test
X_train, X_test, y_train, y_test= train_test_split(X,y, test_size = 0.25)

#------------------fit models--------------

#1) Logistic Regression:
fit_model(LogisticRegression(),'Logit',[{'C': [10, 100, 1000,2000, 3000]}])

#2) Random Forest Classifier:
fit_model(RandomForestClassifier(n_jobs=-1),'RF',[{'n_estimators':[10,50,100,150]}])

#3) Extra Trees Classifier:
fit_model(ExtraTreesClassifier(n_jobs=-1),'ET',[{'n_estimators':[10,50,100]}])

#4) Gradient Boosting Classifier: (Optional paramaters are
learning_rate=1.0,max_depth=1,random_state=0
fit_model(GradientBoostingClassifier(),'GradientBoost',[{'n_estimators':[10,50,100,15
0]}])

#5) Adaboosting Classifier: Optional paramaters are
learning_rate=1.0,random_state=0
fit_model(AdaBoostClassifier(),'AdaBoost',[{'n_estimators':[200,300,400,450]}])

#Gridsearch with log transformed data (dropped columns with Concavity)

log_columns = ['radius_mean', 'texture_mean', 'perimeter_mean',
'area_mean','smoothness_mean', 'compactness_mean',
'symmetry_mean','fractal_dimension_mean',
'radius_se', 'texture_se', 'perimeter_se',
'area_se','smoothness_se', 'compactness_se',
'symmetry_se','fractal_dimension_se',
'radius_worst','texture_worst', 'perimeter_worst',
'area_worst','smoothness_worst','compactness_worst',
'symmetry_worst','fractal_dimension_worst']
X = data.ix[:,log_columns].apply(np.log10) #log tansforming data
X.head()

#Split dataset into Train and Test
X_train, X_test, y_train, y_test= train_test_split(X,y, test_size = 0.25)
```

```
#------------------fit models---------------

#1) Logistic Regression:
fit_model(LogisticRegression(),'Logit',[{'C': [10, 100, 1000,2000, 3000]}])

#2) Random Forest Classifier:
fit_model(RandomForestClassifier(n_jobs=-1),'RF',[{'n_estimators':[10,50,100,150]}])

#3) Extra Trees Classifier:
fit_model(ExtraTreesClassifier(n_jobs=-1),'ET',[{'n_estimators':[10,50,100]}])

#4) Gradient Boosting Classifier: (Optional paramaters are
learning_rate=1.0,max_depth=1,random_state=0
fit_model(GradientBoostingClassifier(),'GradientBoost',[{'n_estimators':[10,50,100,15
0]}])

#5) Adaboosting Classifier: Optional paramaters are
learning_rate=1.0,random_state=0
fit_model(AdaBoostClassifier(),'AdaBoost',[{'n_estimators':[200,300,400,450]}])



#------------------KNN and SVM for ROC plots-----------------#
X = data.ix[:,1:]
#Normalize the data
X = (X-X.mean())/X.std()
X.head()
#Split dataset into Train and Test
X_train, X_test, y_train, y_test= train_test_split(X,y, test_size = 0.25)

# KNN classification using normalized data
knn3 = KNeighborsClassifier(n_neighbors=3)
#fit
knn3fit = knn3.fit(X_train, y_train)
#predict
knn3testPred = knn3fit.predict(X_test)
#get probabilities
knn3testProb = knn3fit.predict_proba(X_test)
#auc
roc_knn = roc_curve(y_test, knn3testProb[:,1]) # Returns fpr, tpr, cutoffs
knn_auc3 = auc(roc_knn[0],roc_knn[1])
```

```
#                   svm fit and prediction
svm_clf = SVC(probability=True)
svm_clf_fit = svm_clf.fit(X_train, y_train)
svm_test_prob = svm_clf_fit.predict_proba(X_test)
roc_svm = roc_curve(y_test, svm_test_prob[:,1])
svm_auc = auc(roc_svm[0],roc_svm[1])
print [knn_auc3, svm_auc]
###############
plt.clf()
plt.figure()
plt.title('ROC curves for KNN, SVM classifications of Wisconcin Breast Cancer data')

ax1= plt.subplot(121)
ax1.set_xlim([0.0, 1.0])
ax1.set_ylim([0.0, 1.05])
ax1.set_xlabel('False Positive Rate')
ax1.set_ylabel('True Positive Rate')
ax1.set_title('KNN classification')
ax1.plot(roc_knn[0],roc_knn[1],label=' (AUC = {0:0.2f})'.format(knn_auc3))
ax1.legend(loc='lower right')

ax2= plt.subplot(122)
ax2.set_xlim([0.0, 1.0])
ax2.set_ylim([0.0, 1.05])
ax2.set_xlabel('False Positive Rate')
ax2.set_ylabel('True Positive Rate')
ax2.set_title('SVM classification')

ax2.plot(roc_svm[0],roc_svm[1],label=' (AUC = {0:0.2f})'.format(svm_auc))
ax2.legend(loc='lower right')
plt.show()
plt.savefig('Documents/dataProgPython/output/rocKS.png')
```

#===#