Python Code for Classification of Wine Data Set using Random Forest and Support Vector Classifier.

The data-set could be downloaded from:
https://archive.ics.uci.edu/ml/datasets/wine+quality
The dataset used here is of the red-wine.
The code is written in python using Jupyter notebooks.

CODE:

#Import required packages for data analysis

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.model_selection import train_test_split, GridSearchCV,
cross_val_score
%matplotlib inline
```

#Loading wine dataset

```
wine = pd.read_csv('../input/winequality-red. csv')
#Add the path in the round bracket where your csv file is being saved
```

#data distribution

```
wine.head()
```

#data columns

```
wine.info()
```

#Here we see that fixed acidity vs quality.

```
fig = plt.figure(figsize = (10,6))
sns.barplot(x = 'quality', y = 'fixed acidity', data = wine)
```

#downing trend in the volatile acidity as we go higher the quality

```
fig = plt.figure(figsize = (10,6))
sns.barplot(x = 'quality', y = 'volatile acidity', data = wine)
```

#Composition of citric acid go higher as we go higher in the quality

```
fig = plt.figure(figsize = (10,6))
sns.barplot(x = 'quality', y = 'citric acid ', data = wine)
```

```
#residual sugar vs quality

fig = plt.figure(figsize = (10,6))
sns.barplot(x = 'quality', y = 'residual sugar', data = wine)

#Composition of chloride goes down as we go higher in the quality of the wine

fig = plt.figure(figsize = (10,6))
sns.barplot(x = 'quality', y = 'chlorides', data = wine)


#free sulfur dioxide vs quality

fig = plt.figure(figsize = (10,6))
sns.barplot(x = 'quality', y = 'free sulfur dioxide', data = wine)

#free sulfur dioxide vs quality

fig = plt.figure(figsize = (10,6))
sns.barplot(x = 'quality', y = 'total sulfur dioxide', data = wine)

#Sulphates level goes high with the quality

fig = plt.figure(figsize = (10,6))
sns.barplot(x = 'quality', y = 'sulphates', data = wine)

#Alcohol level also goes high as the quality increases

fig = plt.figure(figsize = (10,6))
sns.barplot(x = 'quality', y = 'alcohol', data = wine)

#Scatterplot

pd.plotting.scatter_matrix(wine, alpha = 0. 3, figsize = (40,40), diagonal =
'kde');

#Heatmap

correlation = wine.corr()

# display(correlation)

plt.figure(figsize=(14, 12))
heatmap = sns.heatmap(correlation, annot=True, linewidths=0, vmin=-1,
cmap="RdBu_r")

#binary classification for the response variable.
#Dividing wine as good and bad by giving the limit for the quality

bins = (2, 6, 8)
group_names = ['bad', 'good']
wine['quality'] = pd.cut(wine['quality'], bins = bins, labels = group_names)
```

```
#let us assign a label to our quality variable

label_quality = LabelEncoder()

#Bad becomes 0 and good becomes 1

wine['quality'] = label_quality.fit_transform(wine['quality'])
wine['quality'].value_counts()
sns.countplot(wine['quality'])

#seperate the dataset as response variable and feature variabes

X = wine.drop('quality', axis = 1)
y = wine['quality']

#Train and Test splitting of data

X_train, X_test, y_train, y_test = train_te st_split(X, y, test_size = 0.2,
random_state = 42)

#Applying Standard scaling to get optimized output

sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.fit_transform(X_test)
rfc = RandomForestClassifier(n_estimators=200)
rfc.fit(X_train, y_train)
pred_rfc = rfc.predict(X_test)

#Let's see how model has performed

print(classification_report(y_test, pred_rfc))

#Confusion matrix for the random forest classification

print(confusion_matrix(y_test, pred_rfc))
svc = SVC()
svc.fit(X_train, y_train)
pred_svc = svc.predict(X_test)
print(classification_report(y_test, pred_svc))

#Confusion matrix for the support vector classification

print(confusion_matrix(y_test, pred_svc))

#Finding best parameters for our SVC model

param = {
'C': [0.1,0.8,0.9,1,1.1,1.2,1.3,1.4],
'kernel':['linear', 'rbf'],
'gamma' :[0.1,0.8,0.9,1,1.1,1.2,1.3,1.4 ]
}
```

```
grid_svc = GridSearchCV(svc, param_grid=param, scoring='accuracy', cv=10)
grid_svc.fit(X_train, y_train)

#Best parameters for our svc model

grid_svc.best_params_

#Let's run SVC again with the best parameters.

svc2 = SVC(C = 1.2, gamma = 0.9, kernel= ' rbf')
svc2.fit(X_train, y_train)
pred_svc2 = svc2.predict(X_test)
print(classification_report(y_test, pred_svc2))

#let us try to do some evaluation for random forest model using cross validation.

rfc_eval = cross_val_score(estimator = rfc, X = X_train, y = y_train, cv = 10)
rfc_eval.mean()
```