First Name: Twinkle Deepakkumar

Last Name: Jariwala

NJIT UCID: tj257

Preferred email address: tj257@njit.edu

Project option number: Option 1

Project option name: Supervised data mining(Classification)

Algorithms to be used in the project: Category 1 (Support Vector Machines): LIBSVM radial basis function (RBF) kernel AND Category 5 (Naïve Bayes): NaiveBayes

Software tools and platforms: Category 10 (Python)

Tools: Jupyter notebook

Data to be used in the project: https://archive.ics.uci.edu/ml/datasets/Wine

CSV file of Data can be downloaded from: https://www.kaggle.com/brynja/wineuci

Operating systems to be used in the project: Windows 10

Programming languages to be used in the project: Python 3

Hardware to be used in the project: intel core i7(10th gen) 16 GB ram

Algorithms to be used in project:

Naive Bayes:

It is a probabilistic machine learning classifier based on the Bayes Theorem with an assumption of independence among predictors, in other words, this algorithm considers that a presence of a feature in a class is independent of any other features.

All naive Bayes classifiers assume that the value of a particular feature is independent of the value of any other feature, given the class variable. For example, a fruit may be considered to be an apple if it is red, round, and about 10 cm in diameter. A naive Bayes classifier considers each of these features to contribute independently to the probability that this fruit is an apple, regardless of any possible correlations between the color, roundness, and diameter features.

Support Vector Machine (Radial Basis Function):

It is a supervised learning model which can achieve good results in text categorization. Basically, this classifier locates the best possible boundaries to separate between positive and negative training samples. SVM is based on the idea of finding a hyperplane that best separates the features into different domains. Gaussian RBF(Radial Basis Function) is another popular Kernel method used in SVM models for more. RBF kernel is a function whose value depends on the distance from the origin or from some point.

Information about the data to be used in the project:

URL of the data: https://archive.ics.uci.edu/ml/datasets/Wine

CSV file of the data: https://www.kaggle.com/brynja/wineuci

- Abstract:

Using chemical analysis determine the origin of wines

- Content:

This data set is the result of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines.

- Labels:

Number of instances of each wine class:

Name	Number denoting a specific wine class
Class 1	59
Class 2	71
Class 3	48

- Features:

1. Alcohol
2. Malic acid
3. Ash
4. Alcalinity of ash
5. Magnesium
6. Total phenols
7. Flavanoids
8. Nonflavanoid phenols
9. Proanthocyanins

10. Color intensity
11. Hue
12. OD280/OD315 of diluted wines
13. Proline

Source Code URL:

https://www.kaggle.com/abhikaggle8/wine-classification

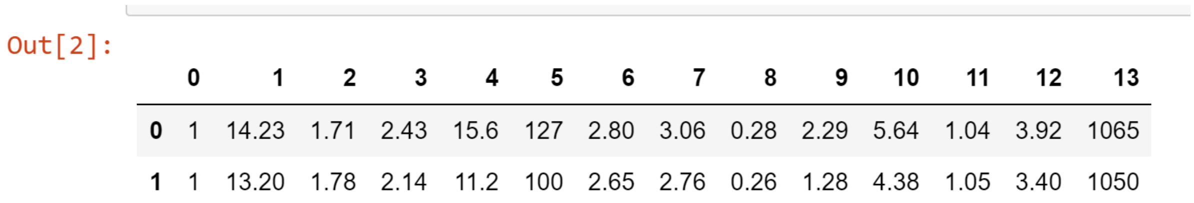
Source Code with output:

Loading the useful packages:

```
# In[1]:



import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt
```

Loading Data and displaying first two record of data:

```
# In[2]:


df=pd.read_csv(r"C:\Users\Twinkle\Downloads\Wine.csv",header=None)
df.head(2)
```

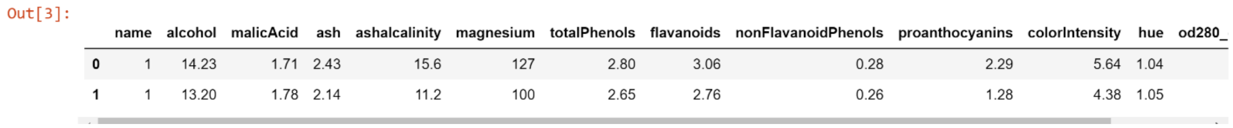


Out[2]:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	1	14.23	1.71	2.43	15.6	127	2.80	3.06	0.28	2.29	5.64	1.04	3.92	1065
1	1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26	1.28	4.38	1.05	3.40	1050

Adding Headers to the columns and displaying first two records:

```
# In[3]:


df.columns = [   'name'
                     ,'alcohol'
                 ,'malicAcid'
                 ,'ash'
                ,'ashalcalinity'
                 ,'magnesium'
                ,'totalPhenols'
                 ,'flavanoids'
                 ,'nonFlavanoidPhenols'
                 ,'proanthocyanins'
                ,'colorIntensity'
                 ,'hue'
                 ,'od280_od315'
                 ,'proline'
                    ]

df.head(2)
```

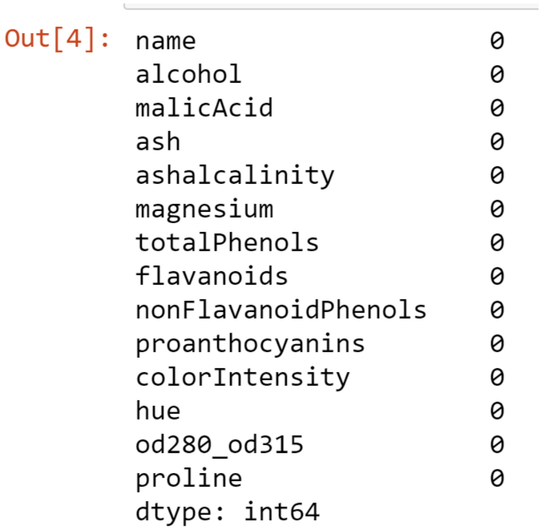


Out[3]:

	name	alcohol	malicAcid	ash	ashalcalinity	magnesium	totalPhenols	flavanoids	nonFlavanoidPhenols	proanthocyanins	colorIntensity	hue	od280_
0	1	14.23	1.71	2.43	15.6	127	2.80	3.06	0.28	2.29	5.64	1.04	
1	1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26	1.28	4.38	1.05	

Check for any Missing Values in dataset:

```
# In[4]:


df.isnull().sum()
```



```
Out[4]: name                   0
        alcohol                0
        malicAcid              0
        ash                    0
        ashalcalinity          0
        magnesium              0
        totalPhenols           0
        flavanoids             0
        nonFlavanoidPhenols    0
        proanthocyanins        0
        colorIntensity         0
        hue                    0
        od280_od315            0
        proline                0
        dtype: int64
```

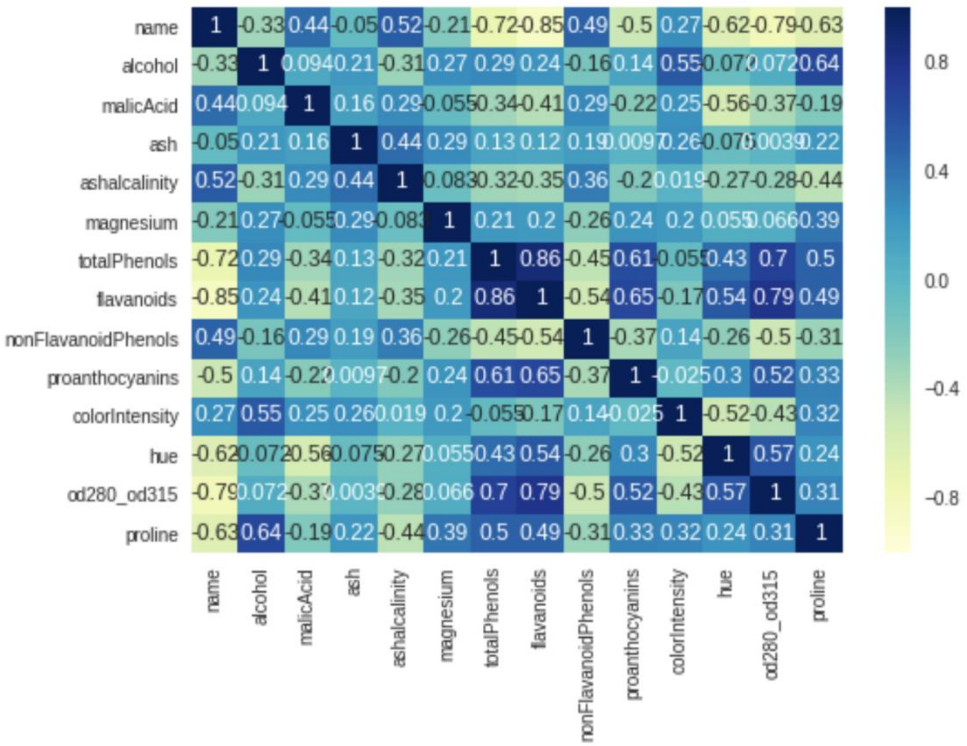
From the output, there are no null records.

Heatmap for Correlation of Parameters:

A heatmap contains values representing various shades of the same color for each value to be plotted. Usually the darker shades of the chart represent higher values than the lighter shade. For a very different value a completely different color can also be used.

Here, displaying the correlation between different features of dataset with the heatmap.

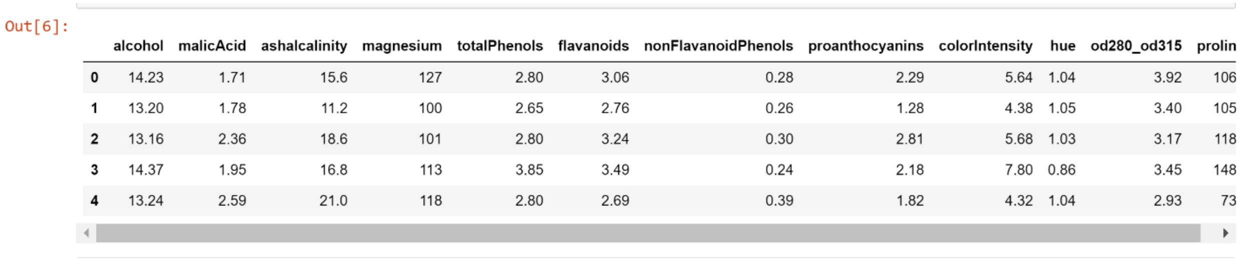
```
# In[5]:


import seaborn as sns
corr = df[df.columns].corr()
sns.heatmap(corr, cmap="YlGnBu", annot = True)
```



From the heatmap, we can see that ‘ash' is the least correlated with other features. So, dropping ‘ash’ is better before classifying the data.

After dropping ‘ash’ , Updated Features and Displaying the Data:

```
# In[6]:


X= df.drop(['name','ash'], axis=1)

X.head()
```

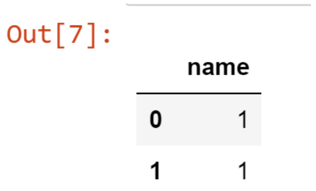


Out[6]:

	alcohol	malicAcid	ashalcalinity	magnesium	totalPhenols	flavanoids	nonFlavanoidPhenols	proanthocyanins	colorIntensity	hue	od280_od315	prolin
0	14.23	1.71	15.6	127	2.80	3.06	0.28	2.29	5.64	1.04	3.92	106
1	13.20	1.78	11.2	100	2.65	2.76	0.26	1.28	4.38	1.05	3.40	105
2	13.16	2.36	18.6	101	2.80	3.24	0.30	2.81	5.68	1.03	3.17	118
3	14.37	1.95	16.8	113	3.85	3.49	0.24	2.18	7.80	0.86	3.45	148
4	13.24	2.59	21.0	118	2.80	2.69	0.39	1.82	4.32	1.04	2.93	73

Displaying the Labels of first two records:

```
# In[7]:


Y=df.iloc[:,:1]
Y.head(2)
```



Out[7]:

	name
0	1
1	1

Doing the Train-Test Split:

```
# In[8]:


from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3,
random_state=0)

print(X_train.shape)
print(X_test.shape)
```

```
(124, 12)
(54, 12)
```

There are 124 rows and 12 columns in train data and 54 rows and 12 columns in test data.

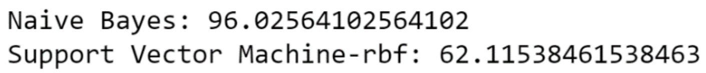
Importing Classifier Models:

```
# In[9]:


from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC

models = []


models.append(("Naive Bayes:",GaussianNB()))
models.append(("Support Vector Machine-rbf:",SVC(kernel="rbf")))



print('Models appended...')
```

```
Models appended...
```

Displaying the accuracy of Model by doing 10-fold cross validation:

```
# In[10]:


results = []
names = []
for name,model in models:
    kfold = KFold(n_splits=10, random_state=0)
    cv_result = cross_val_score(model,X_train,Y_train.values.ravel(), cv =
kfold,scoring = "accuracy")
    names.append(name)
    results.append(cv_result)
for i in range(len(names)):
    print(names[i],results[i].mean()*100)
```



```
Naive Bayes: 96.02564102564102
Support Vector Machine-rbf: 62.11538461538463
```

From a theoretical point of view, it is a little bit hard to compare these two methods. One is probabilistic in nature, while the second one is geometric.

Here, the accuracy of models is displayed. From this, we can say that Naïve Bayes is more accurate on Wine data. But accuracy is not always a good choice for comparing classification models specially when the data is imbalanced. It should not be used as it will not give a true picture. For example, the accuracy of the model might be 96% and one might think that model is performing extremely well but in reality, the model might be predicting only majority class and if the main purpose of the model is to predict minority class then model is of no use. Hence recall, precision and f1-score should be used for measuring the performance of the model.

Source code URL: https://stackoverflow.com/questions/42562146/classification-report-with-nested-cross-validation-in-sklearn-average-individua

Displaying and comparing classification report of two models:

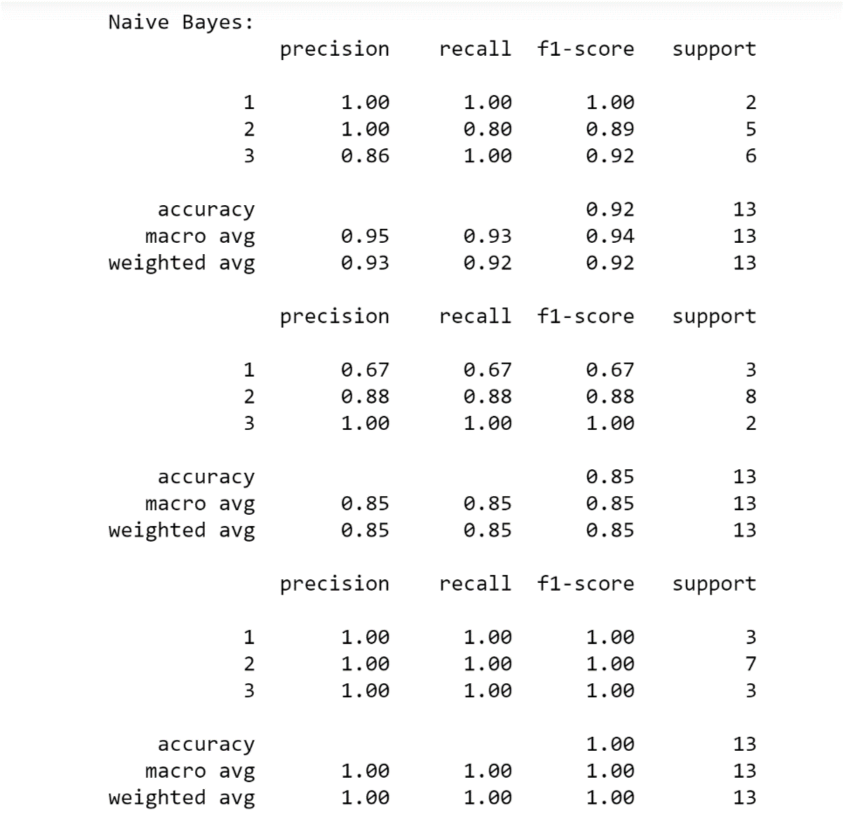
```
# In[69]:


from sklearn.metrics import classification_report, accuracy_score,
make_scorer

def classification_report_with_accuracy_score(y_true, y_pred):

    print(classification_report(y_true, y_pred))# print classification
report
    return accuracy_score(y_true, y_pred) # return accuracy score
```

```
# In[70]:


from sklearn.metrics import confusion_matrix
# Nested CV with parameter optimization
results_report= []
names_report = []
# Variables for average classification report


# Non_nested parameter search and scoring
for name,model in models:
    # Nested CV with parameter optimization
    print(name)
    nested_score = cross_val_score(model, X_train, Y_train.values.ravel(),
cv=kfold,scoring=make_scorer(classification_report_with_accuracy_score))
    names_report.append(name)
    results_report.append(nested_score)


for i in range(len(names)):
    print(names[i])
    print(results_report[i].mean())
```

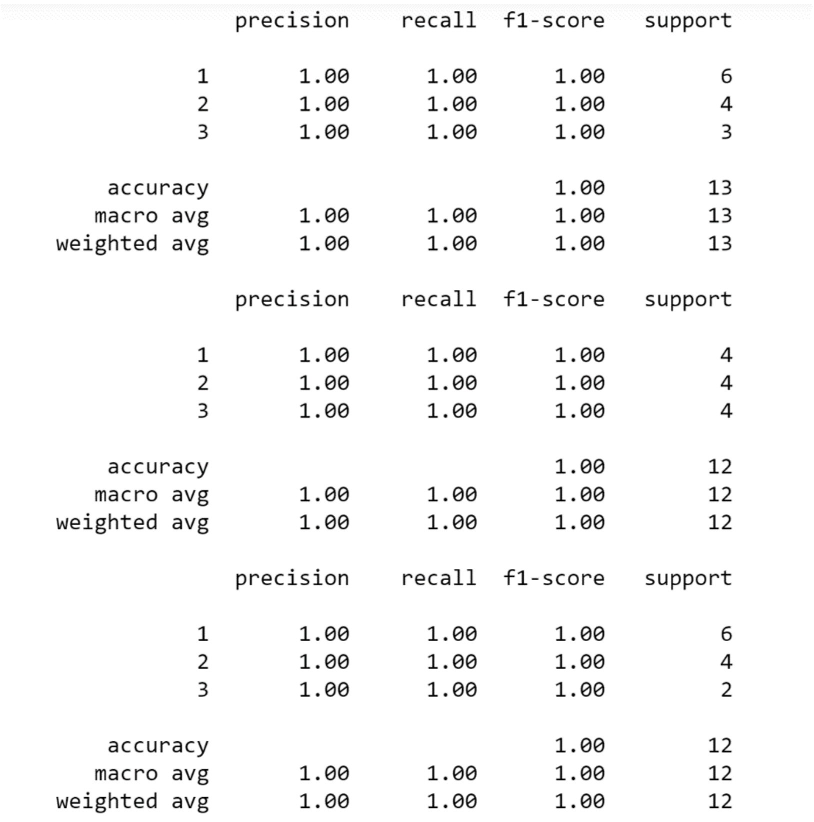
This code displays the classification report for 10 different folds for Naïve Bayes and SVM-rbf.

In the end, it produces the average accuracy score of the model.

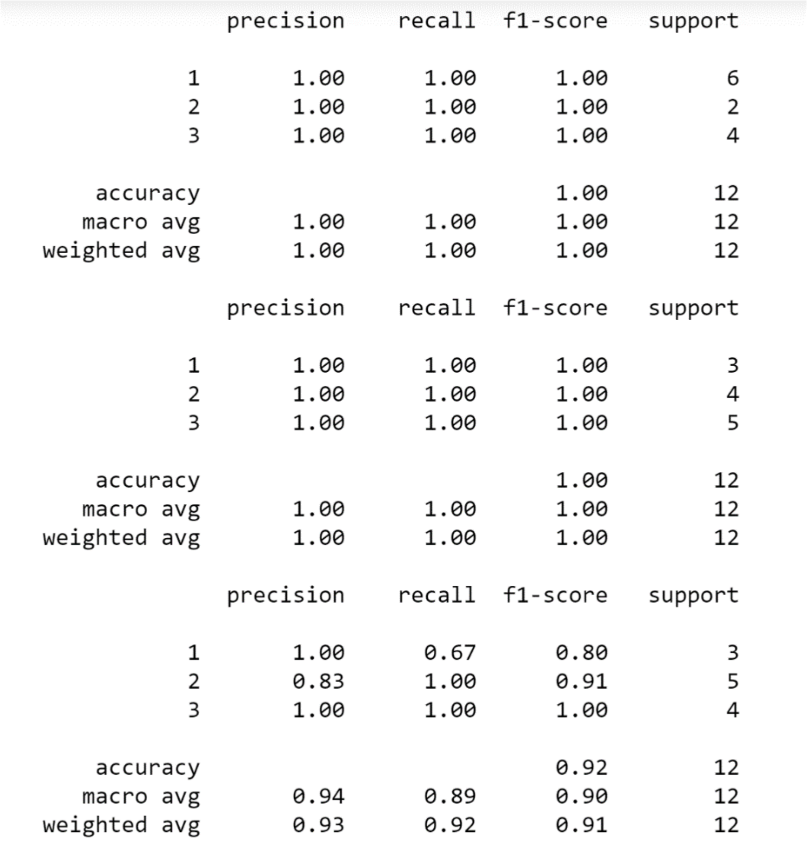
Output: Classificationn reports for Niave Bayes



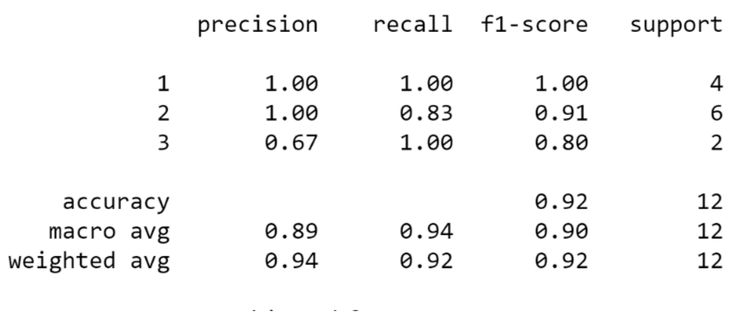
```
Naive Bayes:
              precision    recall  f1-score   support

           1       1.00      1.00      1.00         2
           2       1.00      0.80      0.89         5
           3       0.86      1.00      0.92         6

    accuracy                           0.92        13
   macro avg       0.95      0.93      0.94        13
weighted avg       0.93      0.92      0.92        13

              precision    recall  f1-score   support

           1       0.67      0.67      0.67         3
           2       0.88      0.88      0.88         8
           3       1.00      1.00      1.00         2

    accuracy                           0.85        13
   macro avg       0.85      0.85      0.85        13
weighted avg       0.85      0.85      0.85        13

              precision    recall  f1-score   support

           1       1.00      1.00      1.00         3
           2       1.00      1.00      1.00         7
           3       1.00      1.00      1.00         3

    accuracy                           1.00        13
   macro avg       1.00      1.00      1.00        13
weighted avg       1.00      1.00      1.00        13
```



```
              precision    recall  f1-score   support

           1       1.00      1.00      1.00         6
           2       1.00      1.00      1.00         4
           3       1.00      1.00      1.00         3

    accuracy                           1.00        13
   macro avg       1.00      1.00      1.00        13
weighted avg       1.00      1.00      1.00        13

              precision    recall  f1-score   support

           1       1.00      1.00      1.00         4
           2       1.00      1.00      1.00         4
           3       1.00      1.00      1.00         4

    accuracy                           1.00        12
   macro avg       1.00      1.00      1.00        12
weighted avg       1.00      1.00      1.00        12

              precision    recall  f1-score   support

           1       1.00      1.00      1.00         6
           2       1.00      1.00      1.00         4
           3       1.00      1.00      1.00         2

    accuracy                           1.00        12
   macro avg       1.00      1.00      1.00        12
weighted avg       1.00      1.00      1.00        12
```

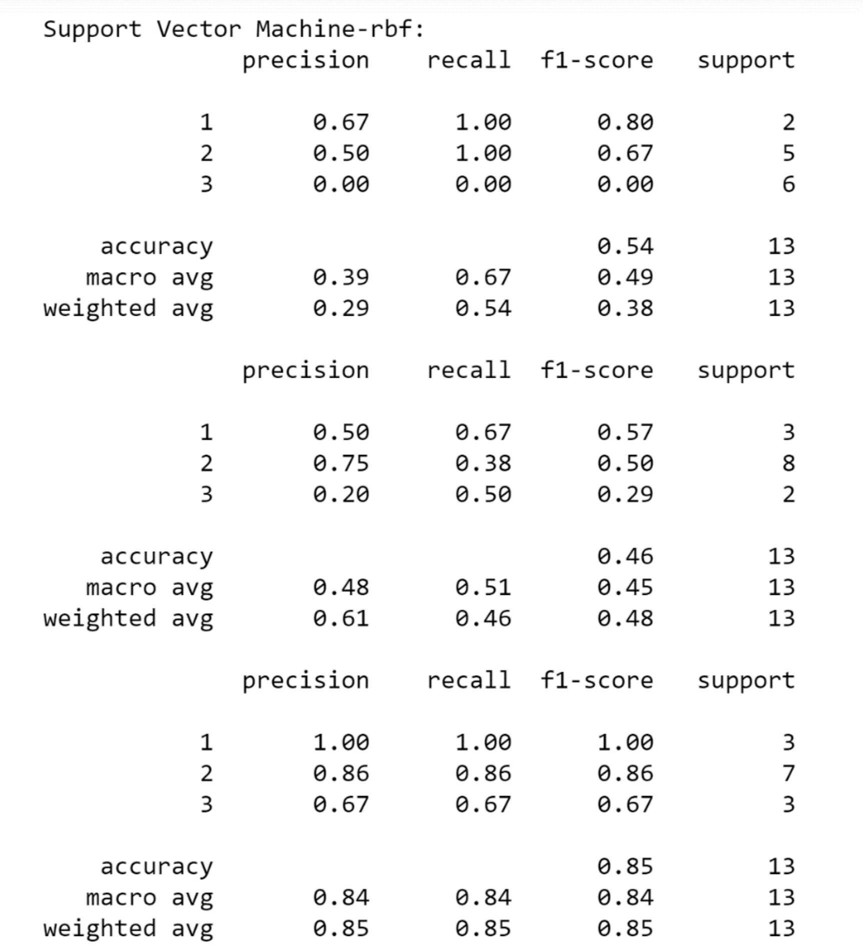


```
              precision    recall  f1-score   support

           1       1.00      1.00      1.00         6
           2       1.00      1.00      1.00         2
           3       1.00      1.00      1.00         4

    accuracy                           1.00        12
   macro avg       1.00      1.00      1.00        12
weighted avg       1.00      1.00      1.00        12

              precision    recall  f1-score   support

           1       1.00      1.00      1.00         3
           2       1.00      1.00      1.00         4
           3       1.00      1.00      1.00         5

    accuracy                           1.00        12
   macro avg       1.00      1.00      1.00        12
weighted avg       1.00      1.00      1.00        12

              precision    recall  f1-score   support

           1       1.00      0.67      0.80         3
           2       0.83      1.00      0.91         5
           3       1.00      1.00      1.00         4

    accuracy                           0.92        12
   macro avg       0.94      0.89      0.90        12
weighted avg       0.93      0.92      0.91        12
```

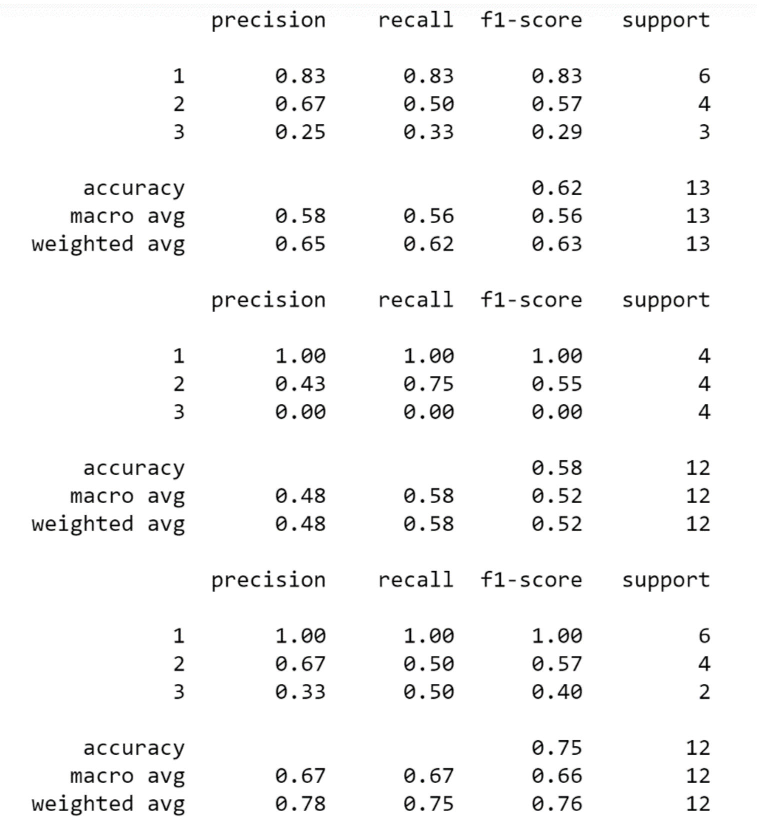


```
              precision    recall  f1-score   support

           1       1.00      1.00      1.00         4
           2       1.00      0.83      0.91         6
           3       0.67      1.00      0.80         2

    accuracy                           0.92        12
   macro avg       0.89      0.94      0.90        12
weighted avg       0.94      0.92      0.92        12
```

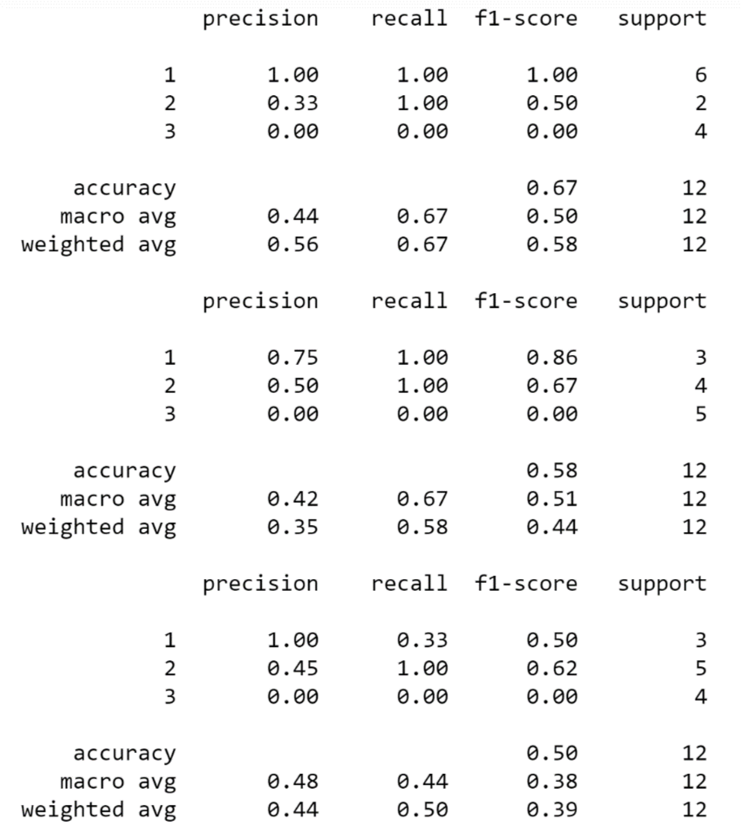
Output: Classification reports for SVM-rbf



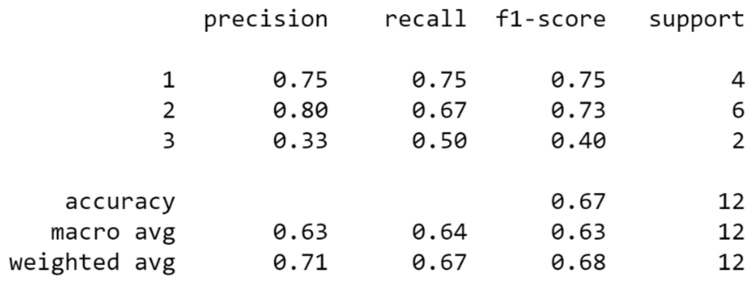
```
Support Vector Machine-rbf:
              precision    recall  f1-score   support

           1       0.67      1.00      0.80         2
           2       0.50      1.00      0.67         5
           3       0.00      0.00      0.00         6

    accuracy                           0.54        13
   macro avg       0.39      0.67      0.49        13
weighted avg       0.29      0.54      0.38        13

              precision    recall  f1-score   support

           1       0.50      0.67      0.57         3
           2       0.75      0.38      0.50         8
           3       0.20      0.50      0.29         2

    accuracy                           0.46        13
   macro avg       0.48      0.51      0.45        13
weighted avg       0.61      0.46      0.48        13

              precision    recall  f1-score   support

           1       1.00      1.00      1.00         3
           2       0.86      0.86      0.86         7
           3       0.67      0.67      0.67         3

    accuracy                           0.85        13
   macro avg       0.84      0.84      0.84        13
weighted avg       0.85      0.85      0.85        13
```



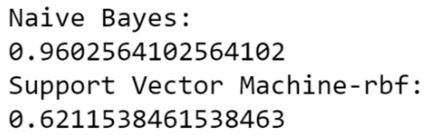
```
              precision    recall  f1-score   support

           1       0.83      0.83      0.83         6
           2       0.67      0.50      0.57         4
           3       0.25      0.33      0.29         3

    accuracy                           0.62        13
   macro avg       0.58      0.56      0.56        13
weighted avg       0.65      0.62      0.63        13
              precision    recall  f1-score   support

           1       1.00      1.00      1.00         4
           2       0.43      0.75      0.55         4
           3       0.00      0.00      0.00         4

    accuracy                           0.58        12
   macro avg       0.48      0.58      0.52        12
weighted avg       0.48      0.58      0.52        12
              precision    recall  f1-score   support

           1       1.00      1.00      1.00         6
           2       0.67      0.50      0.57         4
           3       0.33      0.50      0.40         2

    accuracy                           0.75        12
   macro avg       0.67      0.67      0.66        12
weighted avg       0.78      0.75      0.76        12
```



```
              precision    recall  f1-score   support

           1       1.00      1.00      1.00         6
           2       0.33      1.00      0.50         2
           3       0.00      0.00      0.00         4

    accuracy                           0.67        12
   macro avg       0.44      0.67      0.50        12
weighted avg       0.56      0.67      0.58        12
              precision    recall  f1-score   support

           1       0.75      1.00      0.86         3
           2       0.50      1.00      0.67         4
           3       0.00      0.00      0.00         5

    accuracy                           0.58        12
   macro avg       0.42      0.67      0.51        12
weighted avg       0.35      0.58      0.44        12
              precision    recall  f1-score   support

           1       1.00      0.33      0.50         3
           2       0.45      1.00      0.62         5
           3       0.00      0.00      0.00         4

    accuracy                           0.50        12
   macro avg       0.48      0.44      0.38        12
weighted avg       0.44      0.50      0.39        12
```



```
              precision    recall  f1-score   support

           1       0.75      0.75      0.75         4
           2       0.80      0.67      0.73         6
           3       0.33      0.50      0.40         2

    accuracy                           0.67        12
   macro avg       0.63      0.64      0.63        12
weighted avg       0.71      0.67      0.68        12
```



```
Naive Bayes:
0.9602564102564102
Support Vector Machine-rbf:
0.6211538461538463
```

The values for classification report for 10 different folds in Naïve Bayes and SVM-rbf both are displayed here. We can now compare these two classification models with these values.

Interpreting classification report:

- Precision: The precision is the ratio tp / (tp + fp) where tp is the number of true positives and fp the number of false positives. The precision is intuitively the ability of the classifier not to label as positive a sample that is negative.
    - Comparison:
        - For some folds in Naïve Bayes, precision is 100% for all classes and also in other cases it is very high.
        - For some folds in SVM-rbf, precision is high for one class and very low for other classes. If we take average of these, precision is high for only one class compare to other classes.
        - By evaluating classification reports of these two models, we can say that the Average precision of Naïve Bayes is much higher than the average precision in SVM-rbf for all classes.
- Recall: The recall is the ratio tp / (tp + fn) where tp is the number of true positives and fn the number of false negatives. The recall is intuitively the ability of the classifier to find all the positive samples.
    - Comaprison:
        - For Naïve-Bayes, Recall is mostly 100% for all the classes.
        - For SVM-rbf, Recall is high for class1 but low for class 2 and 3.

        - Same as Precision, average recall value is much higher for Naïve Bayes than SVM-rbf.

- F1-score:

  F1 score is harmonic mean between precision and recall.

  F1 = 2 * (precision * recall) / (precision + recall)

  The scores corresponding to every class will tell you the accuracy of the classifier in classifying the data points in that class compared to all other classes. It tells you how precise your classifier is (how many instances it classifies correctly), as well as how robust it is (it does not miss a significant number of instances). The greater F1 score the better performance.

    - Comparison:
        - F1-score is the mean of Precision and Recall. Here, F1 is also higher for Naïve bayes as its precision and recall value is higher.
        - As F1-score determines how precise and robust is the classifier, with the higher value , we can say that Naive Bayes is more precise than SVM-rbf for this data.

- Support: The support is the number of samples of the true response that lie in that class in training data.
- Accuracy: As we have seen earlier, Naïve Bayes has higher accuracy in classifying Wine data than SVM-rbf with a difference of 34%.

Conclusion:

From the classification report, we can say that Naïve Bayes performs better on Wine data than SVM-rbf. Also, it can be said that the Wine data is linear because Naïve Bayes works better with linear data and SVM-rbf works better with non-linear data.