SUPERVISED MACHINE LEARNING: CLASSIFICATION

COURSE PROJECT REPORT

Machine learning is connected with the field of education related to algorithms which continuously keeps on learning from various examples and then applying them to real-world problems. Classification is a task of Machine Learning which assigns a label value to a specific class and then can identify a particular type to be of one kind or another. The most basic example can be of the mail spam filtration system where one can classify a mail as either “spam” or “not spam”. You will encounter multiple types of classification challenges and there exist some specific approaches for the type of model that might be used for each challenge.

ABOUT THE DATA

Available on the UCI machine learning repository (https://archive.ics.uci.edu/ml/datasets/wine+quality). The red wine samples were obtained from the north of Portugal to model red wine quality based on physicochemical tests. The dataset contains a total of 12 variables, which were recorded for 1,599 observations. The datasets are also available from http://www3.dsi.uminho.pt/pcortez/wine/

1. Alcohol: the amount of alcohol in wine
2. Volatile acidity: acetic acid content which leading to an unpleasant vinegar taste
3. Sulphates: a wine additive that contributes to SO2 levels and acts as an antimicrobial and antioxidant
4. Citric Acid: acts as a preservative to increase acidity (small quantities add freshness and flavor to wines)
5. Total Sulfur Dioxide: is the amount of SO2
6. Density: sweeter wines have a higher density
7. Chlorides: the amount of salt
8. Fixed acidity: are non-volatile acids that do not evaporate easily
9. pH: the level of acidity
10. Free Sulfur Dioxide: it prevents microbial growth and the oxidation of wine
11. Residual sugar: is the amount of sugar remaining after fermentation stops. (Wines > 45g/ltrs are sweet)

OBJECTIVES FROM THE ANALYSIS

To classify the color of wine as red or white

1. Train test split

2. Simple EDA

- Descriptive statistics and data cleaning
- Numerical features
- Categorical features

3. Model variations

- Apply One-hot encoding

4. Applying classification Algorithms

- Logistic Regression
- KNN Classifier
- Svc Classifier
- SVM kernel Classifier
- Gaussian Nave Bayes
- Decision Tree
- Random Forest
- Compare the metrics

5. Conclusion

METHODS USED

1. Import all the required libraries and data

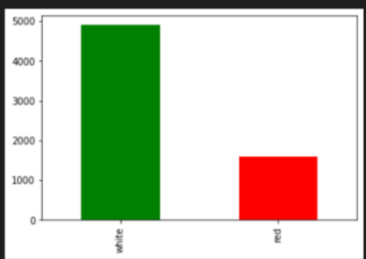
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
%matplotlib inline
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.svm import SVC, LinearSVC
from sklearn.linear_model import SGDClassifier
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.model_selection import train_test_split, GridSearchCV, cross_val_score, StratifiedKFold
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score
```
✓ 15.2s Python

2. Find the value counts and the name of columns.
3. We then apply label encoder on the color column and plot a graph

```
from sklearn import preprocessing
le_color = preprocessing.LabelEncoder()
le_color.fit(['white','red'])
y[:,-1] = le_color.transform(y[:,-1])

df['color'].value_counts().plot.bar(color=['green', 'red'])
```
[23] ✓ 0.6s Python

<AxesSubplot:>



We can see that the data is not highly skewed so we go ahead with train test split

4. Now we apply train test split to the data

```
Y = df.color
X = df.drop('color', axis=1)
```
[24] ✓ 0.1s Python

```
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.2, random_state = 0)
```
[25] ✓ 0.1s Python

5. Now we apply all algorithm models

```python
def models(X_train,Y_train):

    #Using Logistic Regression Algorithm to the Training Set
    from sklearn.linear_model import LogisticRegression
    log = LogisticRegression(random_state = 0)
    log.fit(X_train, Y_train)

    #Using KNeighborsClassifier Method of neighbors class to use Nearest Neighbor algorithm
    from sklearn.neighbors import KNeighborsClassifier
    knn = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p = 2)
    knn.fit(X_train, Y_train)

    #Using SVC method of svm class to use Support Vector Machine Algorithm
    from sklearn.svm import SVC
    svc_lin = SVC(kernel = 'linear', random_state = 0)
    svc_lin.fit(X_train, Y_train)

    #Using SVC method of svm class to use Kernel SVM Algorithm
    from sklearn.svm import SVC
    svc_rbf = SVC(kernel = 'rbf', random_state = 0)
    svc_rbf.fit(X_train, Y_train)

    #Using GaussianNB method of naïve_bayes class to use Naïve Bayes Algorithm
    from sklearn.naive_bayes import GaussianNB
    gauss = GaussianNB()
    gauss.fit(X_train, Y_train)

    #Using DecisionTreeClassifier of tree class to use Decision Tree Algorithm
    from sklearn.tree import DecisionTreeClassifier
    tree = DecisionTreeClassifier(criterion = 'entropy', random_state = 0)
    tree.fit(X_train, Y_train)

    #Using RandomForestClassifier method of ensemble class to use Random Forest Classification algorithm
    from sklearn.ensemble import RandomForestClassifier
    forest = RandomForestClassifier(n_estimators = 10, criterion = 'entropy', random_state = 0)
    forest.fit(X_train, Y_train)

    #print model accuracy on the training data.
    print('[0]Logistic Regression Training Accuracy:', log.score(X_train, Y_train))
    print('[1]K Nearest Neighbor Training Accuracy:', knn.score(X_train, Y_train))
    print('[2]Support Vector Machine (Linear Classifier) Training Accuracy:', svc_lin.score(X_train, Y_train))
    print('[3]Support Vector Machine (RBF Classifier) Training Accuracy:', svc_rbf.score(X_train, Y_train))
    print('[4]Gaussian Naive Bayes Training Accuracy:', gauss.score(X_train, Y_train))
    print('[5]Decision Tree Classifier Training Accuracy:', tree.score(X_train, Y_train))
    print('[6]Random Forest Classifier Training Accuracy:', forest.score(X_train, Y_train))

    return log, knn, svc_lin, svc_rbf, gauss, tree, forest
```

[26] ✓ 0.1s

6. Then we print the training accuracy of all the classifiers

```python
model = models(X_train,y_train)
```

[27] ✓ 6.2s Python

```
c:\Python310\lib\site-packages\sklearn\linear_model\_logistic.py:814: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(

[0]Logistic Regression Training Accuracy: 0.9790263613623245
[1]K Nearest Neighbor Training Accuracy: 0.9570906292091591
[2]Support Vector Machine (Linear Classifier) Training Accuracy: 0.9878776217048297
[3]Support Vector Machine (RBF Classifier) Training Accuracy: 0.9363094092745815
[4]Gaussian Naive Bayes Training Accuracy: 0.9697902636136232
[5]Decision Tree Classifier Training Accuracy: 1.0
[6]Random Forest Classifier Training Accuracy: 0.9998075812969021
```

7. Now we print the confusion matrix and accuracy and precision for all the classifiers

```python
from sklearn.metrics import confusion_matrix
from sklearn.metrics import plot_confusion_matrix
l1=list()
accuracy_scores = list()
precision_scores = list()
for i in range(len(model)):
    cm = confusion_matrix(y_test, model[i].predict(X_test))
    #extracting TN, FP, FN, TP
    TN, FP, FN, TP = confusion_matrix(y_test, model[i].predict(X_test)).ravel()
    print(cm)
    print('Model[{}] Testing Accuracy = "{} !"'.format(i,  (TP + TN) / (TP + TN + FN + FP)))
    acc=(TP + TN) / (TP + TN + FN + FP)
    print('Model[{}] Testing Precision = "{} !"'.format(i,  (TP ) / (TP + FP)))
    prec= (TP) / (TP +  FP)
    print()# Print a new line
    plot_confusion_matrix(model[i], X_test, y_test)
    accuracy_scores.append(acc)
    precision_scores.append(prec)
    l1.append(pd.Series({'model': model[i], 'accuracy':acc,'precision':prec }))
```

```
[[301  10]
 [ 14 975]]
Model[0] Testing Accuracy = "0.9815384615384616 !"
Model[0] Testing Precision = "0.9898477157360406 !"
```

8. Now we extract all this information in table format

```python
models=['LogisticRegression','Knn','SVC_Linear','SVC_RBF','GaussianNB','DecisionTree','RandomForest']
```

```python
performance_df = pd.DataFrame({'Algorithm':models,'Accuracy':accuracy_scores,'Precision':precision_scores}).sort_values('Precision',as
```

```python
performance_df
```

	Algorithm	Accuracy	Precision
6	RandomForest	0.995385	0.998985
5	DecisionTree	0.986923	0.995918
2	SVC_Linear	0.989231	0.992922
4	GaussianNB	0.974615	0.990760
0	LogisticRegression	0.981538	0.989848
1	Knn	0.950000	0.958333
3	SVC_RBF	0.936154	0.937259

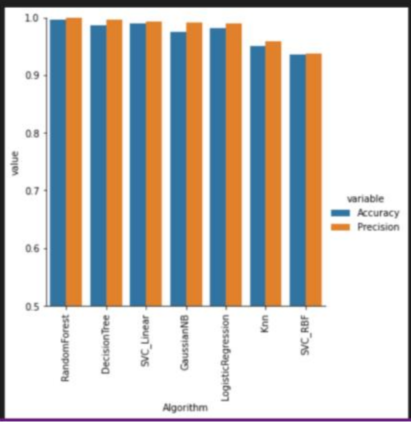
9. For plotting the graph we reformat the table as follows:

```python
performance_df1 = pd.melt(performance_df, id_vars = "Algorithm")
performance_df1
```

	Algorithm	variable	value
0	RandomForest	Accuracy	0.995385
1	DecisionTree	Accuracy	0.986923
2	SVC_Linear	Accuracy	0.989231
3	GaussianNB	Accuracy	0.974615
4	LogisticRegression	Accuracy	0.981538
5	Knn	Accuracy	0.950000
6	SVC_RBF	Accuracy	0.936154
7	RandomForest	Precision	0.998985
8	DecisionTree	Precision	0.995918
9	SVC_Linear	Precision	0.992922
10	GaussianNB	Precision	0.990760
11	LogisticRegression	Precision	0.989848
12	Knn	Precision	0.958333
13	SVC_RBF	Precision	0.937259

10. Now we finally print the comparison graph

```
sns.catplot(x = 'Algorithm', y='value',
                hue = 'variable',data=performance_df1, kind='bar',height=5)
plt.ylim(0.5,1.0)
plt.xticks(rotation='vertical')
plt.show()
```



CONCLUSION

We can now compare the metrics and see that the accuracy of our model is very good and all classifiers perform a great task at it.