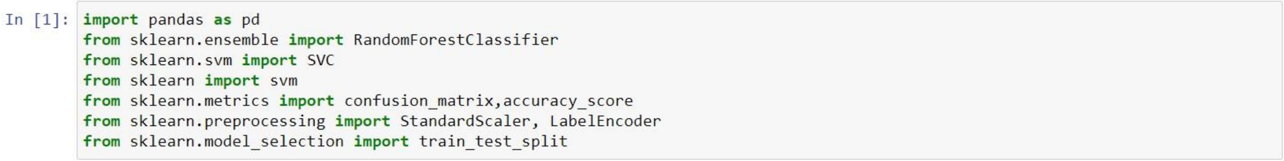
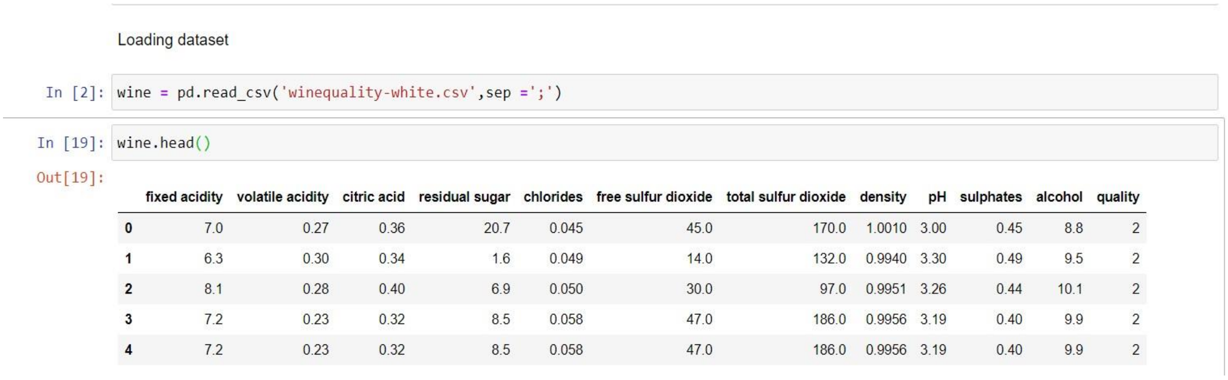
# Task 3.3: ML development from scratch

## 1. Importing libraries:



```
In [1]: import pandas as pd
        from sklearn.ensemble import RandomForestClassifier
        from sklearn.svm import SVC
        from sklearn import svm
        from sklearn.metrics import confusion_matrix,accuracy_score
        from sklearn.preprocessing import StandardScaler, LabelEncoder
        from sklearn.model_selection import train_test_split
```

## 2. Loading dataset:



Loading dataset

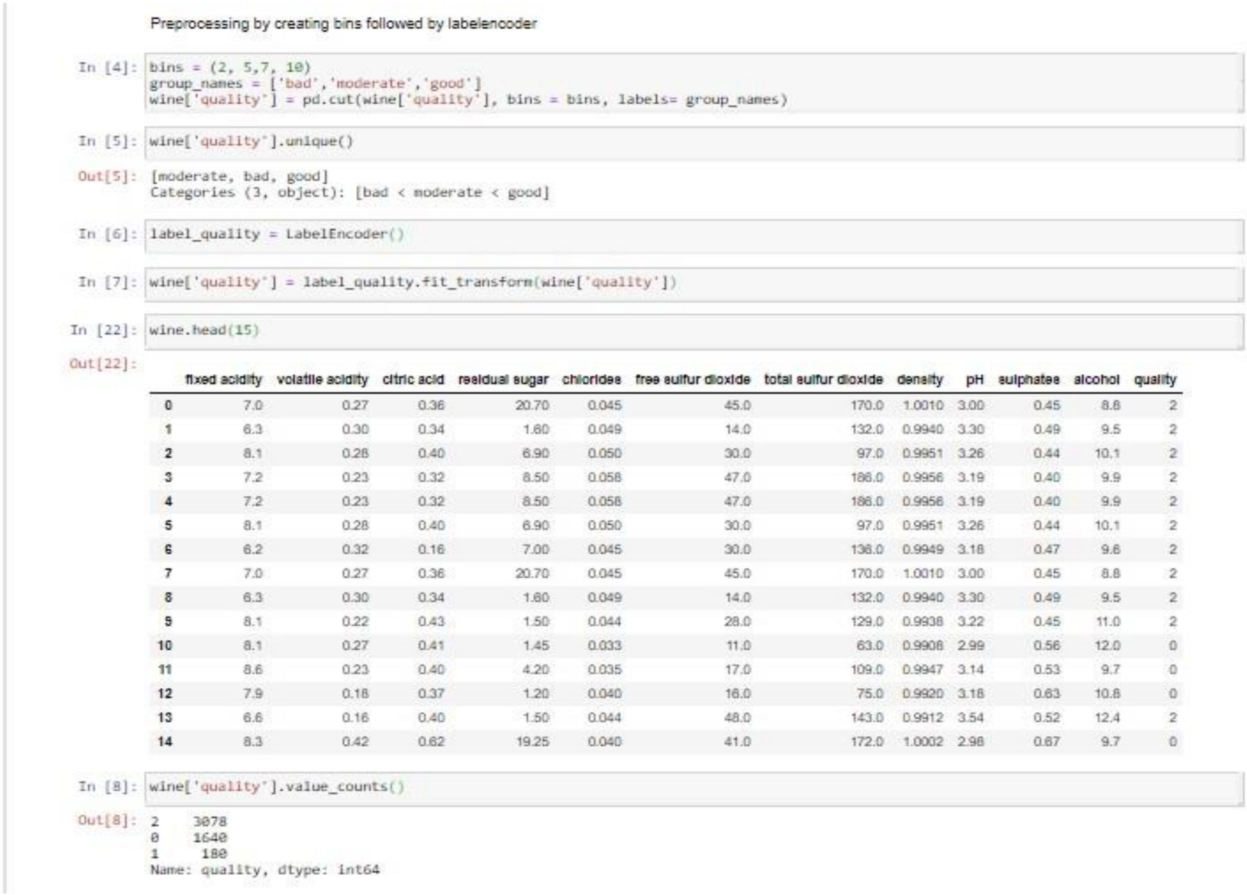
```
In [2]: wine = pd.read_csv('winequality-white.csv',sep =';')
```

```
In [19]: wine.head()
```

Out[19]:

| | fixed acidity | volatile acidity | citric acid | residual sugar | chlorides | free sulfur dioxide | total sulfur dioxide | density | pH | sulphates | alcohol | quality |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 7.0 | 0.27 | 0.36 | 20.7 | 0.045 | 45.0 | 170.0 | 1.0010 | 3.00 | 0.45 | 8.8 | 2 |
| **1** | 6.3 | 0.30 | 0.34 | 1.6 | 0.049 | 14.0 | 132.0 | 0.9940 | 3.30 | 0.49 | 9.5 | 2 |
| **2** | 8.1 | 0.28 | 0.40 | 6.9 | 0.050 | 30.0 | 97.0 | 0.9951 | 3.26 | 0.44 | 10.1 | 2 |
| **3** | 7.2 | 0.23 | 0.32 | 8.5 | 0.058 | 47.0 | 186.0 | 0.9956 | 3.19 | 0.40 | 9.9 | 2 |
| **4** | 7.2 | 0.23 | 0.32 | 8.5 | 0.058 | 47.0 | 186.0 | 0.9956 | 3.19 | 0.40 | 9.9 | 2 |

## 3. Pre-processing:



Preprocessing by creating bins followed by labelencoder

```
In [4]: bins = (2, 5,7, 10)
        group_names = ['bad','moderate','good']
        wine['quality'] = pd.cut(wine['quality'], bins = bins, labels= group_names)
```

```
In [5]: wine['quality'].unique()
```

```
Out[5]: [moderate, bad, good]
        Categories (3, object): [bad < moderate < good]
```

```
In [6]: label_quality = LabelEncoder()
```

```
In [7]: wine['quality'] = label_quality.fit_transform(wine['quality'])
```

```
In [22]: wine.head(15)
```

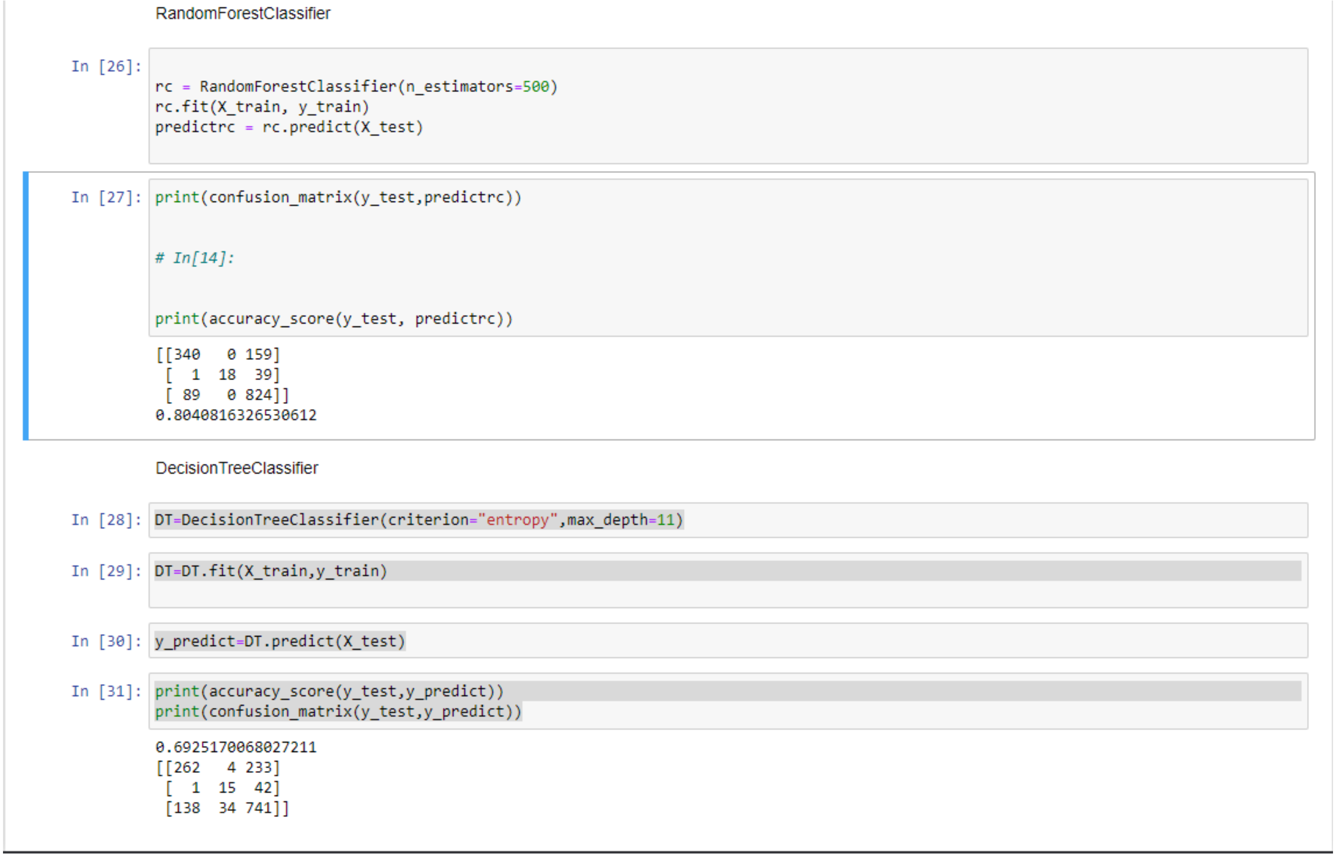
Out[22]:

| | fixed acidity | volatile acidity | citric acid | residual sugar | chlorides | free sulfur dioxide | total sulfur dioxide | density | pH | sulphates | alcohol | quality |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 7.0 | 0.27 | 0.36 | 20.70 | 0.045 | 45.0 | 170.0 | 1.0010 | 3.00 | 0.45 | 8.8 | 2 |
| **1** | 6.3 | 0.30 | 0.34 | 1.60 | 0.049 | 14.0 | 132.0 | 0.9940 | 3.30 | 0.49 | 9.5 | 2 |
| **2** | 8.1 | 0.28 | 0.40 | 6.90 | 0.050 | 30.0 | 97.0 | 0.9951 | 3.26 | 0.44 | 10.1 | 2 |
| **3** | 7.2 | 0.23 | 0.32 | 8.50 | 0.058 | 47.0 | 186.0 | 0.9956 | 3.19 | 0.40 | 9.9 | 2 |
| **4** | 7.2 | 0.23 | 0.32 | 8.50 | 0.058 | 47.0 | 186.0 | 0.9956 | 3.19 | 0.40 | 9.9 | 2 |
| **5** | 8.1 | 0.28 | 0.40 | 6.90 | 0.050 | 30.0 | 97.0 | 0.9951 | 3.26 | 0.44 | 10.1 | 2 |
| **6** | 6.2 | 0.32 | 0.16 | 7.00 | 0.045 | 30.0 | 136.0 | 0.9949 | 3.18 | 0.47 | 9.6 | 2 |
| **7** | 7.0 | 0.27 | 0.36 | 20.70 | 0.045 | 45.0 | 170.0 | 1.0010 | 3.00 | 0.45 | 8.8 | 2 |
| **8** | 6.3 | 0.30 | 0.34 | 1.60 | 0.049 | 14.0 | 132.0 | 0.9940 | 3.30 | 0.49 | 9.5 | 2 |
| **9** | 8.1 | 0.22 | 0.43 | 1.50 | 0.044 | 28.0 | 129.0 | 0.9938 | 3.22 | 0.45 | 11.0 | 2 |
| **10** | 8.1 | 0.27 | 0.41 | 1.45 | 0.033 | 11.0 | 63.0 | 0.9908 | 2.99 | 0.56 | 12.0 | 0 |
| **11** | 8.6 | 0.23 | 0.40 | 4.20 | 0.035 | 17.0 | 109.0 | 0.9947 | 3.14 | 0.53 | 9.7 | 0 |
| **12** | 7.9 | 0.18 | 0.37 | 1.20 | 0.040 | 16.0 | 75.0 | 0.9920 | 3.18 | 0.63 | 10.8 | 0 |
| **13** | 6.6 | 0.16 | 0.40 | 1.50 | 0.044 | 48.0 | 143.0 | 0.9912 | 3.54 | 0.52 | 12.4 | 2 |
| **14** | 8.3 | 0.42 | 0.62 | 19.25 | 0.040 | 41.0 | 172.0 | 1.0002 | 2.98 | 0.67 | 9.7 | 0 |

```
In [8]: wine['quality'].value_counts()
```

```
Out[8]: 2    3078
        0    1640
        1     180
        Name: quality, dtype: int64
```

## 4. Building model 1 and model 2:



RandomForestClassifier

```
In [26]:
rc = RandomForestClassifier(n_estimators=500)
rc.fit(X_train, y_train)
predictrc = rc.predict(X_test)
```

```
In [27]: print(confusion_matrix(y_test,predictrc))


# In[14]:


print(accuracy_score(y_test, predictrc))
```

```
[[340   0 159]
 [  1  18  39]
 [ 89   0 824]]
0.8040816326530612
```

DecisionTreeClassifier

```
In [28]: DT=DecisionTreeClassifier(criterion="entropy",max_depth=11)
```

```
In [29]: DT=DT.fit(X_train,y_train)
```

```
In [30]: y_predict=DT.predict(X_test)
```

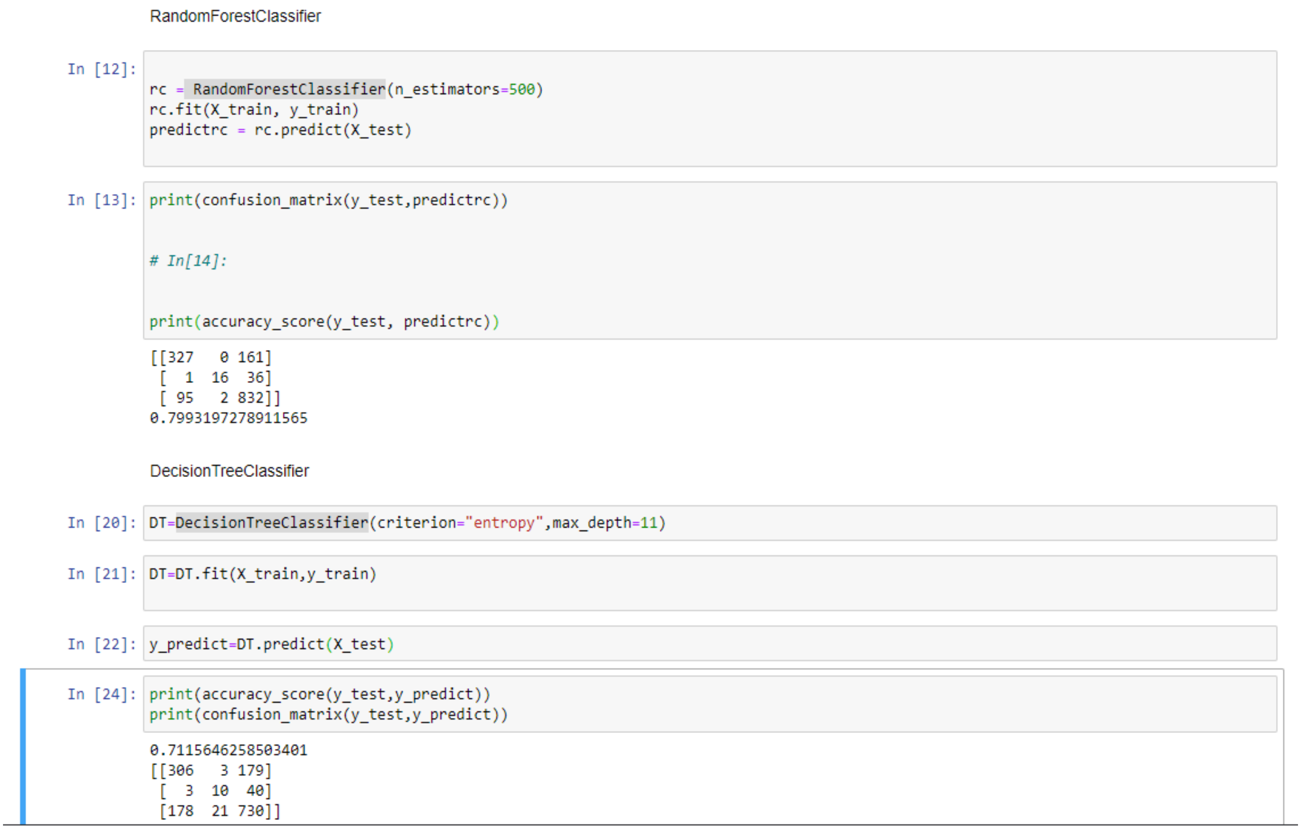
```
In [31]: print(accuracy_score(y_test,y_predict))
         print(confusion_matrix(y_test,y_predict))
```

```
0.6925170068027211
[[262   4 233]
 [  1  15  42]
 [138  34 741]]
```

## 5. Code:

```
import pandas as pd
from sklearn.ensemble import RandomForestClassifier from
sklearn.svm import SVC from sklearn import svm from
sklearn.metrics import confusion_matrix,accuracy_score from
sklearn.preprocessing import StandardScaler, LabelEncoder from
sklearn.model_selection import train_test_split bins = (2,
5,7, 10) group_names = ['bad','moderate','good']
wine['quality'] = pd.cut(wine['quality'], bins = bins, labels= group_names)
wine['quality'].unique() label_quality
= LabelEncoder()
wine['quality'] = label_quality.fit_transform(wine['quality']) wine.head(15)
wine['quality'].value_counts() X
= wine.drop('quality', axis =1) y
= wine['quality']
X_train , X_test, y_train, y_test = train_test_split(X,y,test_size=0.3) sc
= StandardScaler()
X_train = sc.fit_transform(X_train) X_test
= sc.transform(X_test)
rc = RandomForestClassifier(n_estimators=500)
rc.fit(X_train, y_train) predictrc =
rc.predict(X_test)
print(confusion_matrix(y_test,predictrc))
print(accuracy_score(y_test, predictrc))
DT=DecisionTreeClassifier(criterion="entropy”
,max_depth=11)

DT=DT.fit(X_train,y_train)

y_predict=DT.predict(X_test)

print(accuracy_score(y_test,y_predict))

print(confusion_matrix(y_test,y_predict))
```

## 6. Output:



RandomForestClassifier

```
In [12]: 
rc = RandomForestClassifier(n_estimators=500)
rc.fit(X_train, y_train)
predictrc = rc.predict(X_test)
```

```
In [13]: print(confusion_matrix(y_test,predictrc))


# In[14]:


print(accuracy_score(y_test, predictrc))
```

```
[[327   0 161]
 [  1  16  36]
 [ 95   2 832]]
0.7993197278911565
```

DecisionTreeClassifier

```
In [20]: DT=DecisionTreeClassifier(criterion="entropy",max_depth=11)
```

```
In [21]: DT=DT.fit(X_train,y_train)
```

```
In [22]: y_predict=DT.predict(X_test)
```

```
In [24]: print(accuracy_score(y_test,y_predict))
print(confusion_matrix(y_test,y_predict))
```

```
0.7115646258503401
[[306   3 179]
 [  3  10  40]
 [178  21 730]]
```

## 7. Comparing results of two matrix:

- Random Tree classifier:

```
 accuracy_score = 0.8040816326530612
confusion_matrix = [[340   0 159]
                    [  1  18  39]
                    [ 89   0 824]]
```

- Decision Tree Classifier:

```
accuracy_score: 0.6925170068027211
confusion_matrix: [[262   4 233]
                   [  1  15  42]
                   [138  34 741]]
```