

NAME: MOHAMMAD HUSSAM(2033.KHI.DEG.020)

PAIRING WITH : MAVIA ALAM KHAN (2303.KHI.DEG.017)

&

AQSA TAUHEED(2303.KHI.DEG.011)

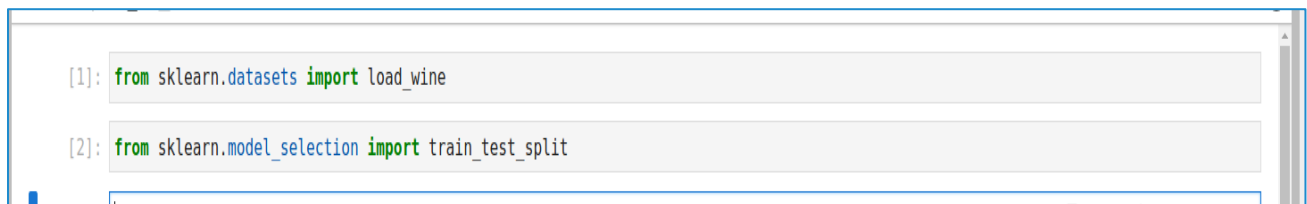
ASSIGNMENT 3.2

Implement a single classification model of your choice and try to achieve at least an 80% F1 score on the wine dataset provided by Sklearn.

SOLUTION:

STEP:1

Firstly we import libraries :



```
[1]: from sklearn.datasets import load_wine

[2]: from sklearn.model_selection import train_test_split
```

from sklearn.datasets import load_wine : sklearn.datasets to load the wine dataset

from sklearn.model_selection import train_test_split :to split the data into training and testing sets

```
[3]: from sklearn.ensemble import RandomForestClassifier  
[4]: from sklearn.metrics import f1_score
```

from sklearn.ensemble import RandomForestClassifier: to train a random forest classifier

from sklearn.metrics import f1_score: to calculate the F1 score

STEP:2

```
[5]: wine_data = load_wine()
```

We loaded the wine dataset using `load_wine()` function and assign it to the variable `wine_data`

STEP:3

```
[6]: X = wine_data.data  
     y = wine_data.target
```

Split the data into X and y variables, where X contains the feature data and y contains the target labels.

STEP:4

```
7]: x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

Split the data into training and testing sets using the `train_test_split` function. This function randomly splits the data into training and testing sets based on the `test_size` parameter (in this case, 20% of the data will be used for testing). The remaining 80% of the data will be used for training.

STEP:5

```
] : model = RandomForestClassifier()  
    model.fit(x_train, y_train)
```

```
] : ▼ RandomForestClassifier  
    RandomForestClassifier()
```

`RandomForestClassifier()` creates an instance of the Random Forest classifier model with default parameters.

`fit(x_train, y_train)` trains the model on the training data `x_train` and their corresponding labels `y_train`. During training, the model learns to map the input features to their corresponding class labels.

STEP:6

```
[9]: y_pred = model.predict(x_test)
```

Make predictions on the testing data using the predict method and assign them to the variable y_pred.

STEP:7

```
]]: f1 = f1_score(y_test, y_pred, average="micro")
```

Calculate the F1 score using the f1_score function. This function takes in the actual target labels (y_test) and the predicted target labels (y_pred) and returns the F1 score. In this case, we're using average="micro", which means we're calculating the F1 score for all classes combined.

OUTPUT:

Then we printed the F1 score as shown in image, which gives the F1 score 0.972222.. or 97.222 %

```
[12]: print("F1 Score =", f1)
```

F1 Score = 0.9722222222222222