

## Unit 3.2 Graded Assignment = Implementation of classification algorithm

### Instructions:

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

### Submitted by:

1. Ali Nasir (2303.KHI.DEG.012)
2. Saif ur Rehman (2303.KHI.DEG.007)

### Solution:

1. First we will import all the required libraries.

```
Import all the required Libraries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
```

0] ✓ 0.0s

2. Import the builtin data from the sklearn library.

```
Dataset

# Load the wine dataset
from sklearn.datasets import load_wine
data = load_wine()

[12] ✓ 0.0s

# Just for Visualization
df = pd.DataFrame(data.data, columns=data.feature_names)
df.head()

[13] ✓ 0.0s
```

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyanins	color_intensity	hue	od280/od315_of_diluted_wines	proline
0	14.23	1.71	2.43	15.6	127.0	2.80	3.06	0.28	2.29	5.64	1.04	3.92	1065.0
1	13.20	1.78	2.14	11.2	100.0	2.65	2.76	0.26	1.28	4.38	1.05	3.40	1050.0
2	13.16	2.36	2.67	18.6	101.0	2.80	3.24	0.30	2.81	5.68	1.03	3.17	1185.0
3	14.37	1.95	2.50	16.8	113.0	3.85	3.49	0.24	2.18	7.80	0.86	3.45	1480.0
4	13.24	2.59	2.87	21.0	118.0	2.80	2.69	0.39	1.82	4.32	1.04	2.93	735.0

3. We have split our dataset into test size 0.2 which means 80 % training data and 20% testing data and a random state of 42 so that any bias ness could be removed.

```
Train Test Split

# Split the dataset into training and testing sets
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(data.data, data.target, test_size=0.2, random_state=42)

4] ✓ 0.0s
```

+ Code + Markdown

4. Here we have implemented the decision tree with max depth 3.

## Decision Tree Classifier Implementation

```
from sklearn.tree import DecisionTreeClassifier, plot_tree

# Create a decision tree classifier with max depth of 3
dt = DecisionTreeClassifier(max_depth=3)

# Train the model using the training data
dt.fit(X_train, y_train)

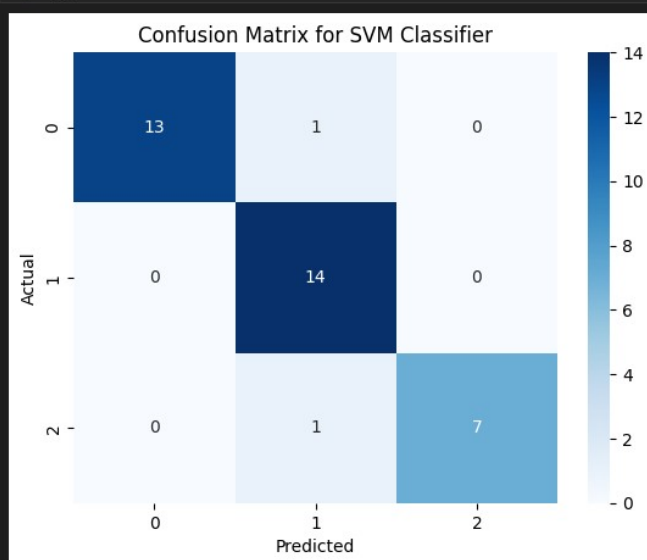
# Use the trained model to predict the labels of the test data
y_pred = dt.predict(X_test)
```

✓ 0.0s

In our classification report, it can be seen that our f1 score is 0.96 with a complete decision tree.

```
matrix = confusion_m (parameter) annot: Any | None
# Plot the confusion
sns.heatmap(matrix, annot=True, cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix for SVM Classifier')
plt.show()
```

✓ 0.2s



```
from sklearn.metrics import classification_report
# Print the classification report
report = classification_report(y_test, y_pred, target_names=data.target_names)
print(report)
```

✓ 0.0s

	precision	recall	f1-score	support
class_0	1.00	0.93	0.96	14
class_1	0.88	1.00	0.93	14
class_2	1.00	0.88	0.93	8
accuracy			0.94	36
macro avg	0.96	0.93	0.94	36
weighted avg	0.95	0.94	0.94	36

```
# Plot the decision tree
plot_tree(dt, filled=True, feature_names=data.feature_names, class_names=data.target_names)
plt.title("Decision Tree")
plt.show()
```

✓ 0.3s

Decision Tree

