STUDENT NAME: ODUFUWA ODUTAYO THOMSON

MATRIC NO: 209074209

COURSE CODE: MIT 816

DECISION TREE CLASSIFICATION

TASK 1 - Using decision tree classifier on balanced datasets

Import the necessary libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from imblearn.datasets import make_imbalance
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
```

Loading the Iris dataset and beginning the prediction process

```
data = load_iris() #iris dataset is loaded into a variable named data
```

The next step is to separate the features from the target. This is done below

```
X = data.data #features
y = data.target #target
```

```
X.shape
```

```
(150, 4)
```

```
y[:20]
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

```
X[:20]
```

```
array([[5.1, 3.5, 1.4, 0.2],
       [4.9, 3. , 1.4, 0.2],
       [4.7, 3.2, 1.3, 0.2],
       [4.6, 3.1, 1.5, 0.2],
       [5. , 3.6, 1.4, 0.2],
       [5.4, 3.9, 1.7, 0.4],
       [4.6, 3.4, 1.4, 0.3],
       [5. , 3.4, 1.5, 0.2],
       [4.4, 2.9, 1.4, 0.2],
       [4.9, 3.1, 1.5, 0.1],
       [5.4, 3.7, 1.5, 0.2],
       [4.8, 3.4, 1.6, 0.2],
       [4.8, 3. , 1.4, 0.1],
       [4.3, 3. , 1.1, 0.1],
       [5.8, 4. , 1.2, 0.2],
       [5.7, 4.4, 1.5, 0.4],
       [5.4, 3.9, 1.3, 0.4],
       [5.1, 3.5, 1.4, 0.3],
       [5.7, 3.8, 1.7, 0.3],
       [5.1, 3.8, 1.5, 0.3]])
```

The train_test_split library is applied to split the data into a train and a test part. See code below

```
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state = 42, test_size = 0.3)
```

The variable 'tree' is used to store the DecionTreeClassifier algorithm and used to train the X-train and y_train values. This is further used to predict the value of 'y' given 'X_test'. See code below

```
tree = DecisionTreeClassifier()
```

```
tree = tree.fit(X_train, y_train)
```

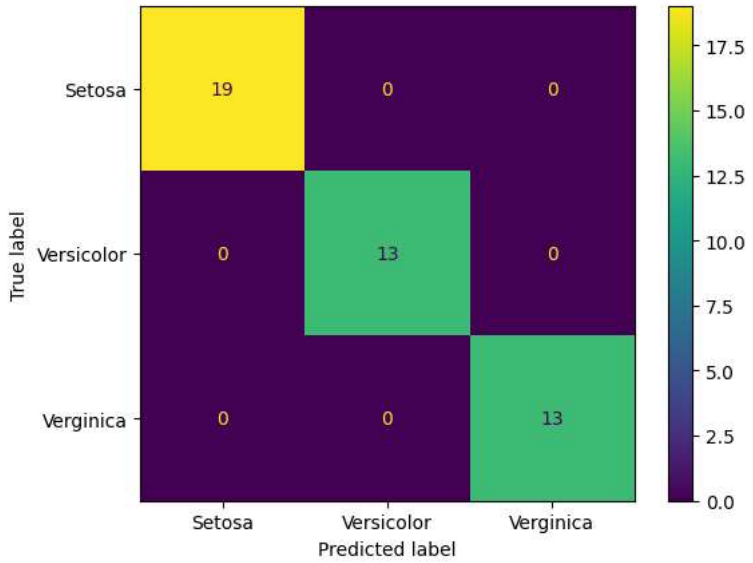
```
y_pred = tree.predict(X_test)
```

Metrics Evaluation

Using Confusion Matrix below, the decision tree classifier would be evaluated to correctness.

```
from sklearn import metrics
cm = metrics.confusion_matrix(y_test, y_pred)
cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = cm, display_labels = ["Setosa", "Versicolor", "Verginica"])
cm_display.plot()
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7cdd5f45fd00>
```




```
f1_score = metrics.f1_score(y_test, y_pred, average="macro")
print(f"F1 score is {f1_score}")
```

```
F1 score is 1.0
```

ASSIGNMENT 1

Question 1 : Interprete the F1 **score** above

A F1 score of 1 indicates perfect precision and recall, which means the classification model has achieved the best possible balance between them.

Perfect Precision (Precision = 1): When precision is 1, it means that every positive prediction made by the model is correct. There are no false positives; every instance predicted as positive is indeed positive.

Perfect Recall (Recall = 1): When recall is 1, it means that the model correctly identifies all actual positive instances. There are no false negatives; the model captures every positive instance.

In practical terms, an F1 score of 1 suggests that the model's positive predictions are all correct, and it doesn't miss any actual positive instances.

Question 2: Use Accuracy metric to evaluate the classifier and interprete the result

```
from sklearn.metrics import accuracy_score


accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy is {accuracy}")
```

```
Accuracy is 1.0
```

The accuracy score will be a value between 0 and 1, where 1 indicates perfect accuracy (all predictions are correct).

While an F1 score of 1 implies perfect precision and recall, the accuracy score tells you the overall proportion of correct predictions. If the accuracy score is also 1, it means your classifier has made zero mistakes on the dataset.