

```
In [ ]: Pooja Dhumal
LGM TASK
TASK 1
NAME: Prediction using decision algorithm
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
```

```
In [20]: iris=pd.read_csv(r"C:\Users\Pooja\Downloads\Iris csv.csv")
```

```
In [21]: iris
```

```
Out[21]:
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species	
	0	1	5.1	3.5	1.4	0.2	Iris-setosa
	1	2	4.9	3.0	1.4	0.2	Iris-setosa
	2	3	4.7	3.2	1.3	0.2	Iris-setosa
	3	4	4.6	3.1	1.5	0.2	Iris-setosa
	4	5	5.0	3.6	1.4	0.2	Iris-setosa

	145	146	6.7	3.0	5.2	2.3	Iris-virginica
	146	147	6.3	2.5	5.0	1.9	Iris-virginica
	147	148	6.5	3.0	5.2	2.0	Iris-virginica
	148	149	6.2	3.4	5.4	2.3	Iris-virginica
	149	150	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 6 columns

```
In [22]: iris.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column          Non-Null Count  Dtype
---  -
 0   Id              150 non-null   int64
 1   SepalLengthCm   150 non-null   float64
 2   SepalWidthCm    150 non-null   float64
 3   PetalLengthCm   150 non-null   float64
 4   PetalWidthCm    150 non-null   float64
 5   Species         150 non-null   object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

```
In [32]: iris.head()
```

```
Out[32]:
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species	Species_class
0	1	5.1	3.5	1.4	0.2	Iris-setosa	3
1	2	4.9	3.0	1.4	0.2	Iris-setosa	3
2	3	4.7	3.2	1.3	0.2	Iris-setosa	3
3	4	4.6	3.1	1.5	0.2	Iris-setosa	3
4	5	5.0	3.6	1.4	0.2	Iris-setosa	3

```
In [33]: iris.tail()
```

```
Out[33]:
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species	Species_class
145	146	6.7	3.0	5.2	2.3	Iris-virginica	1
146	147	6.3	2.5	5.0	1.9	Iris-virginica	1
147	148	6.5	3.0	5.2	2.0	Iris-virginica	1
148	149	6.2	3.4	5.4	2.3	Iris-virginica	1
149	150	5.9	3.0	5.1	1.8	Iris-virginica	1

```
In [37]: iris.columns
```

```
Out[37]: Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm',  
              'Species', 'Species_class'],  
            dtype='object')
```

```
In [25]: iris.Species.value_counts()
```

```
Out[25]: Iris-setosa      50  
Iris-versicolor    50  
Iris-virginica      50  
Name: Species, dtype: int64
```

```
In [26]: iris['Species_class']=np.where(iris.Species=='Iris-virginica',1,np.where(iris.Species=='Iris-versicolor',2,3
```

```
In [28]: iris.Species_class.value_counts()
```

```
Out[28]: 3      50  
2      50  
1      50  
Name: Species_class, dtype: int64
```

```
In [45]: cols=['SepalLengthCm','SepalWidthCm','PetalLengthCm','PetalWidthCm']
```

```
In [46]: from sklearn.model_selection import train_test_split  
  
train_X, test_X, train_y, test_y = train_test_split( iris[cols],  
                                                    iris['Species_class'],  
                                                    test_size = 0.2,  
                                                    random_state = 123 )
```

```
In [50]: #Model Building  
param_grid = {'max_depth': np.arange(2,8),  
              'max_features': np.arange(2,5)}
```

```
In [51]: from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeClassifier, export_graphviz
tree = GridSearchCV(DecisionTreeClassifier(), param_grid, cv = 10, verbose=1, n_jobs=-1)
tree.fit( train_X, train_y )
```

```
Out[51]: Fitting 10 folds for each of 18 candidates, totalling 180 fits
GridSearchCV(cv=10, estimator=DecisionTreeClassifier(), n_jobs=-1,
              param_grid={'max_depth': array([2, 3, 4, 5, 6, 7]),
                           'max_features': array([2, 3, 4])},
              verbose=1)
```

```
In [52]: tree.best_score_
```

```
Out[52]: 0.9583333333333334
```

```
In [54]: tree.best_estimator_
```

```
Out[54]: DecisionTreeClassifier(max_depth=5, max_features=3)
```

```
In [55]: tree.best_params_
```

```
Out[55]: {'max_depth': 5, 'max_features': 3}
```

```
In [56]: train_pred = tree.predict(train_X)
```

```
In [57]: test_pred = tree.predict(test_X)
```

```
In [59]: import sklearn.metrics as metrics
print(metrics.classification_report(test_y, test_pred))
```

	precision	recall	f1-score	support
1	1.00	0.82	0.90	11
2	0.75	1.00	0.86	6
3	1.00	1.00	1.00	13

accuracy			0.93	30
macro avg	0.92	0.94	0.92	30
weighted avg	0.95	0.93	0.93	30

```
In [62]: #Building Final Decision Tree  
clf_tree =DecisionTreeClassifier( max_depth =4, max_features=2)  
clf_tree.fit( train_X, train_y)
```

```
Out[62]: DecisionTreeClassifier(max_depth=4, max_features=2)
```

```
In [63]: tree_test_pred = pd.DataFrame({'actual': test_y, 'pedicted':clf_tree.predict(test_X)})
```

```
In [64]: tree_test_pred.sample(n=10)
```

```
Out[64]:
```

	actual	pedicted
133	1	2
4	3	3
90	2	2
24	3	3
37	3	3
8	3	3
88	2	2
13	3	3
104	1	1
138	1	1

```
In [70]: metrics.accuracy_score( tree_test_pred.actual, tree_test_pred.pedicted )
```

```
Out[70]: 0.9666666666666667
```

```
In [76]: tree_cm = metrics.confusion_matrix( tree_test_pred.predicted,
                                             tree_test_pred.actual,
                                             [1,0] )

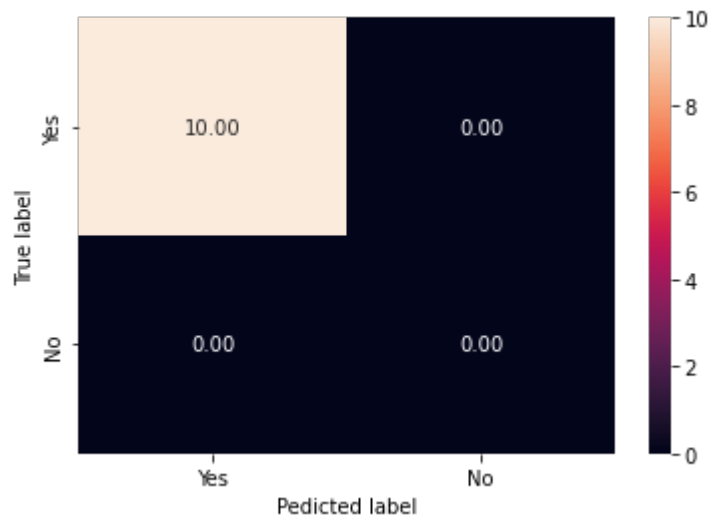
sns.heatmap(tree_cm, annot=True,
            fmt='.2f',
            xticklabels = ["Yes", "No"] , yticklabels = ["Yes", "No"] )

plt.ylabel('True label')
plt.xlabel('Predicted label')
```

C:\Users\urmil\anaconda3\lib\site-packages\sklearn\utils\validation.py:70: FutureWarning: Pass labels=[1, 0] as keyword args. From version 1.0 (renaming of 0.25) passing these as positional arguments will result in an error

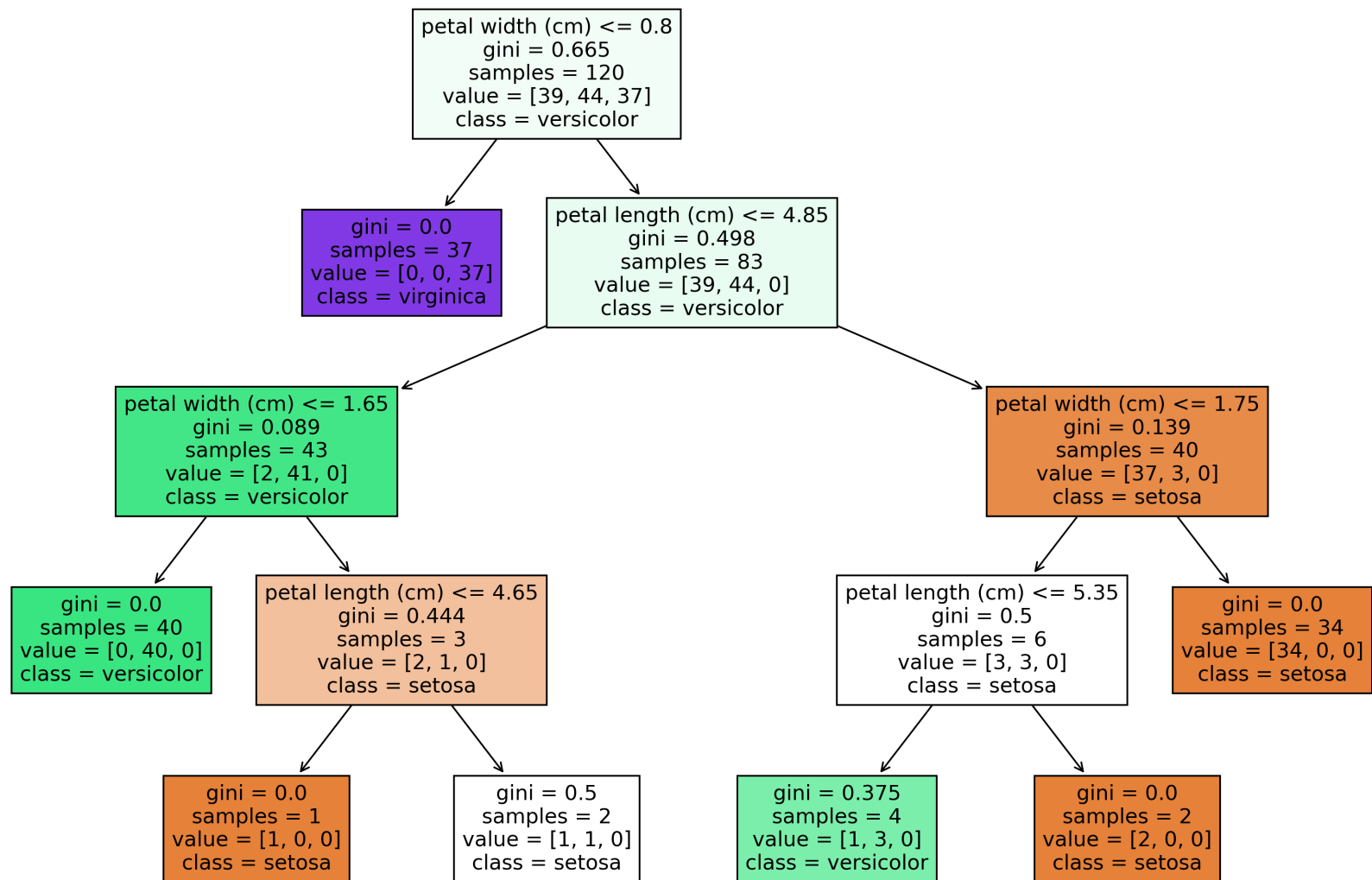
```
warnings.warn(f"Pass {args_msg} as keyword args. From version "
              Text(0.5, 15.0, 'Predicted label'))
```

Out[76]:



In [77]:

```
#Graphical Representation of Decision Tree
from sklearn import tree
fn=['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
cn=['setosa', 'versicolor', 'virginica']
fig, axes = plt.subplots(nrows = 1,ncols = 1,figsize = (15,10), dpi=300)
tree.plot_tree(clf_tree,
                feature_names = fn,
                class_names=cn,
                filled = True);
fig.savefig('imagename.png')
```



In []: