

# Machine Learning Lab Assignment

## Decision Tree

Name: Om Ashish Mishra

Reg. No.: 16BCE0789

Slot: F2

The Dataset:

	A	B	C	D	E	F	G
1	5.1	3.5	1.4	0.2	Iris-setosa		
2	4.9	3	1.4	0.2	Iris-setosa		
3	4.7	3.2	1.3	0.2	Iris-setosa		
4	4.6	3.1	1.5	0.2	Iris-setosa		
5	5	3.6	1.4	0.2	Iris-setosa		
6	5.4	3.9	1.7	0.4	Iris-setosa		
7	4.6	3.4	1.4	0.3	Iris-setosa		
8	5	3.4	1.5	0.2	Iris-setosa		
9	4.4	2.9	1.4	0.2	Iris-setosa		
10	4.9	3.1	1.5	0.1	Iris-setosa		
11	5.4	3.7	1.5	0.2	Iris-setosa		
12	4.8	3.4	1.6	0.2	Iris-setosa		
13	4.8	3	1.4	0.1	Iris-setosa		
14	4.3	3	1.1	0.1	Iris-setosa		
15	5.8	4	1.2	0.2	Iris-setosa		
16	5.7	4.4	1.5	0.4	Iris-setosa		
17	5.4	3.9	1.3	0.4	Iris-setosa		

### ID3 Algorithm

The Code:

```
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
```

```
import pandas as pd

# Importing the dataset
dataset = pd.read_csv('data.csv')
X = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:, 4].values

#Converting String to Charaterized data
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
# Encoding the Dependent Variable
labelencoder_y = LabelEncoder()
y = labelencoder_y.fit_transform(y)

# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)

# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

# Fitting Decision Tree Classification to the Training set
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(criterion = 'entropy', random_state = 0)
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)

# Visualising the Training set results
```

```

from matplotlib.colors import ListedColormap

X_set, y_set = X_train, y_train

X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, step = 0.01))

plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('red', 'green','orange')))

plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())

for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green','orange'))(i), label = j)

plt.title('Decision Tree Classification (Training set)')
plt.xlabel('Petal Length')
plt.ylabel('Petal Width')
plt.legend()
plt.show()

```

# Visualising the Test set results

```

from matplotlib.colors import ListedColormap

X_set, y_set = X_test, y_test

X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, step = 0.01))

plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('red', 'green','orange')))

plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())

for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green','orange'))(i), label = j)

plt.title('Decision Tree Classification (Test set)')
plt.xlabel('Petal Length')
plt.ylabel('Petal Width')
plt.legend()
plt.show()

```

```

from sklearn.metrics import classification_report

```

```
print(classification_report(y_test,y_pred))
```

The Output:

Spyder (Python 3.7)

File Edit Search Source Run Debug Consoles Projects Tools View Help

Editor - C:\Users\OM MISHRA\Desktop\java\decision\_tree\_classification.py

```
1 # Importing the libraries
2 import numpy as np
3 import matplotlib.pyplot as plt
4 import pandas as pd
5
6 # Importing the dataset
7 dataset = pd.read_csv('data.csv')
8 X = dataset.iloc[:, [2, 3]].values
9 y = dataset.iloc[:, 4].values
10
11 #Converting String to Characterized data
12 from sklearn.preprocessing import LabelEncoder, OneHotEncoder
13 # Encoding the Dependent Variable
14 labelencoder_y = LabelEncoder()
15 y = labelencoder_y.fit_transform(y)
16
17 # Splitting the dataset into the Training set and Test set
18 from sklearn.model_selection import train_test_split
19 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)
20
21 # Feature Scaling
22 from sklearn.preprocessing import StandardScaler
23 sc = StandardScaler()
24 X_train = sc.fit_transform(X_train)
25 X_test = sc.transform(X_test)
26
27 # Fitting Decision Tree Classification to the Training set
28 from sklearn.tree import DecisionTreeClassifier
29 classifier = DecisionTreeClassifier(criterion = 'entropy', random_state = 0)
30 classifier.fit(X_train, y_train)
31
32 # Predicting the Test set results
33 y_pred = classifier.predict(X_test)
34
35 # Making the Confusion Matrix
36 from sklearn.metrics import confusion_matrix
37 cm = confusion_matrix(y_test, y_pred)
38
39 # Visualising the Training set results
40 from matplotlib.colors import ListedColormap
41 X_set, y_set = X_train, y_train
42 X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, step = 0.01),
43                      np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, step = 0.01))
```

Variable explorer

Name	Type	Size	Value
X	float64	(149, 2)	[[1.4 0.2] [1.3 0.2] 2 ...
X1	float64	(485, 500)	[[-2.44377588 -2.43377588 -2.42377588 ... 2.52622412 2.53622412 2 ...
X2	float64	(485, 500)	[[-2.49329304 -2.49329304 -2.49329304 ... -2.49329304 -2.49329304 -2 ...
X_set	float64	(38, 2)	[[ 0.98268267 0.18884736] [ 0.70053632 0.96521985]
X_test	float64	(38, 2)	[[ 0.98268267 0.18884736] [ 0.70053632 0.96521985]
X_train	float64	(111, 2)	[[ 0.58767779 0.31824278] [-1.33891734 -1.36389762] [14 0 0]

Python console

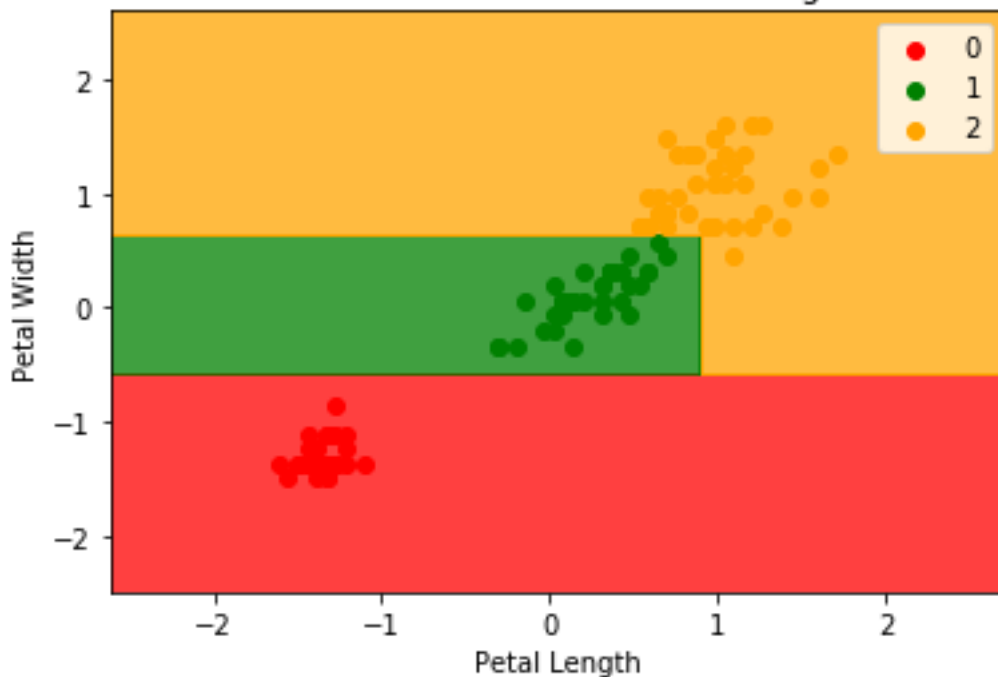
```
In [12]: cm
Out[12]:
array([[14,  0,  0],
       [ 0, 13,  1],
       [ 0,  3,  7]], dtype=int64)

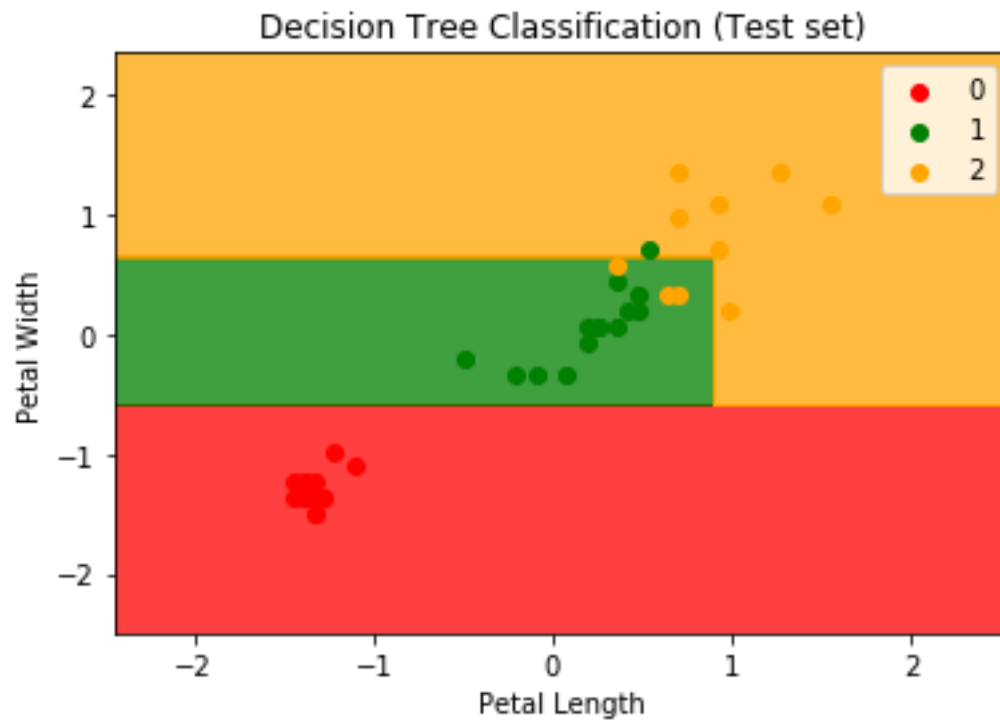
In [13]: runfile('C:/Users/OM MISHRA/Desktop/java/decision_tree_classification.py',
                wdir='C:/Users/OM MISHRA/Desktop/java')
```

Console I/O

```
macro avg 0.90 0.88 0.88 38
weighted avg 0.90 0.89 0.89 38
```

Decision Tree Classification (Training set)





	precision	recall	f1-score	support
0	1.00	1.00	1.00	14
1	0.81	0.93	0.87	14
2	0.88	0.70	0.78	10
micro avg	0.89	0.89	0.89	38
macro avg	0.90	0.88	0.88	38
weighted avg	0.90	0.89	0.89	38

In [14]: cm

Out[14]:

```
array([[14,  0,  0],
       [ 0, 13,  1],
       [ 0,  3,  7]], dtype=int64)
```

## CART Algorithm

The Code:

```
# Importing the libraries
```

```
import numpy as np
```

```
import matplotlib.pyplot as plt
```

```
import pandas as pd
```

```
# Importing the dataset
dataset = pd.read_csv('iris.csv')
X = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:, 4].values

#Converting String to Charaterized data
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
# Encoding the Dependent Variable
labelencoder_y = LabelEncoder()
y = labelencoder_y.fit_transform(y)

# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)

# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

# Fitting Decision Tree Classification to the Training set
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(criterion = 'entropy', random_state = 0)
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)

# Visualising the Training set results
from matplotlib.colors import ListedColormap
```

```

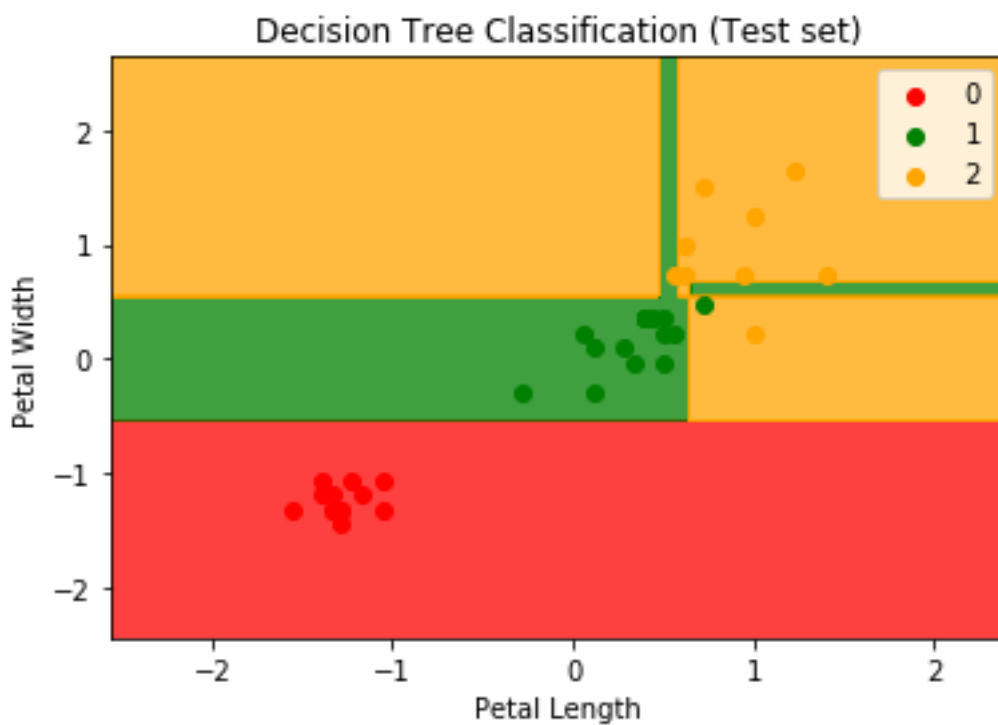
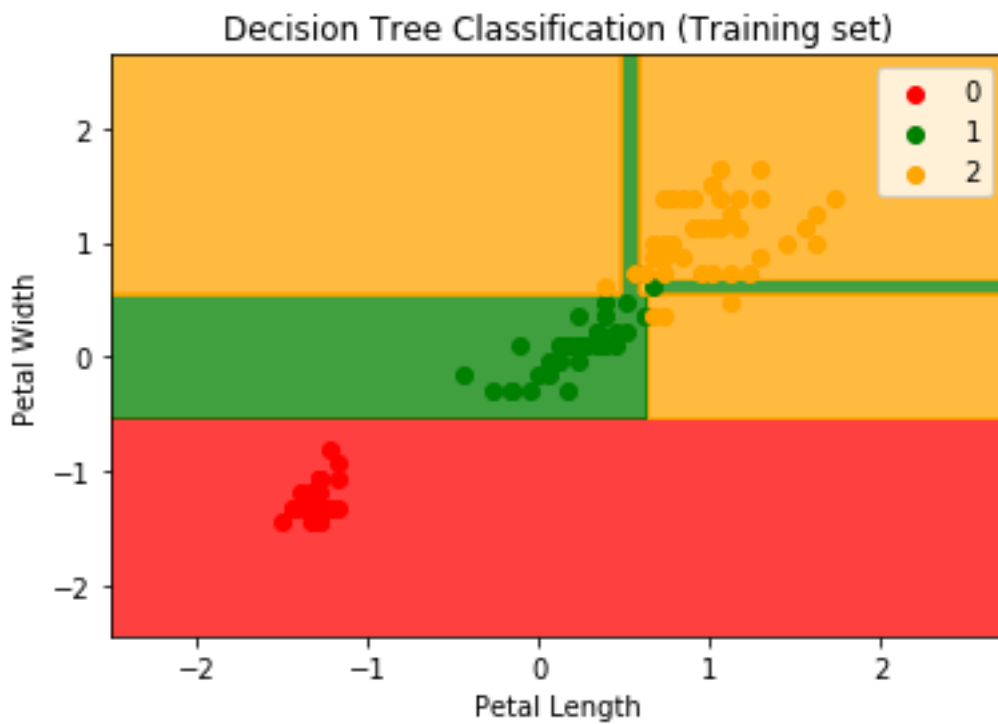
X_set, y_set = X_train, y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('red', 'green','orange')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green','orange'))(i), label = j)
plt.title('Decision Tree Classification (Training set)')
plt.xlabel('Petal Length')
plt.ylabel('Petal Width')
plt.legend()
plt.show()

# Visualising the Test set results
from matplotlib.colors import ListedColormap
X_set, y_set = X_test, y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('red', 'green','orange')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green','orange'))(i), label = j)
plt.title('Decision Tree Classification (Test set)')
plt.xlabel('Petal Length')
plt.ylabel('Petal Width')
plt.legend()
plt.show()

from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred))

```

The Output:



```
Out[10]:  
array([[13,  0,  0],  
       [ 0, 15,  1],  
       [ 0,  1,  8]], dtype=int64)
```



	precision	recall	f1-score	support
0	1.00	1.00	1.00	13
1	0.94	0.94	0.94	16
2	0.89	0.89	0.89	9
micro avg	0.95	0.95	0.95	38
macro avg	0.94	0.94	0.94	38
weighted avg	0.95	0.95	0.95	38