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:

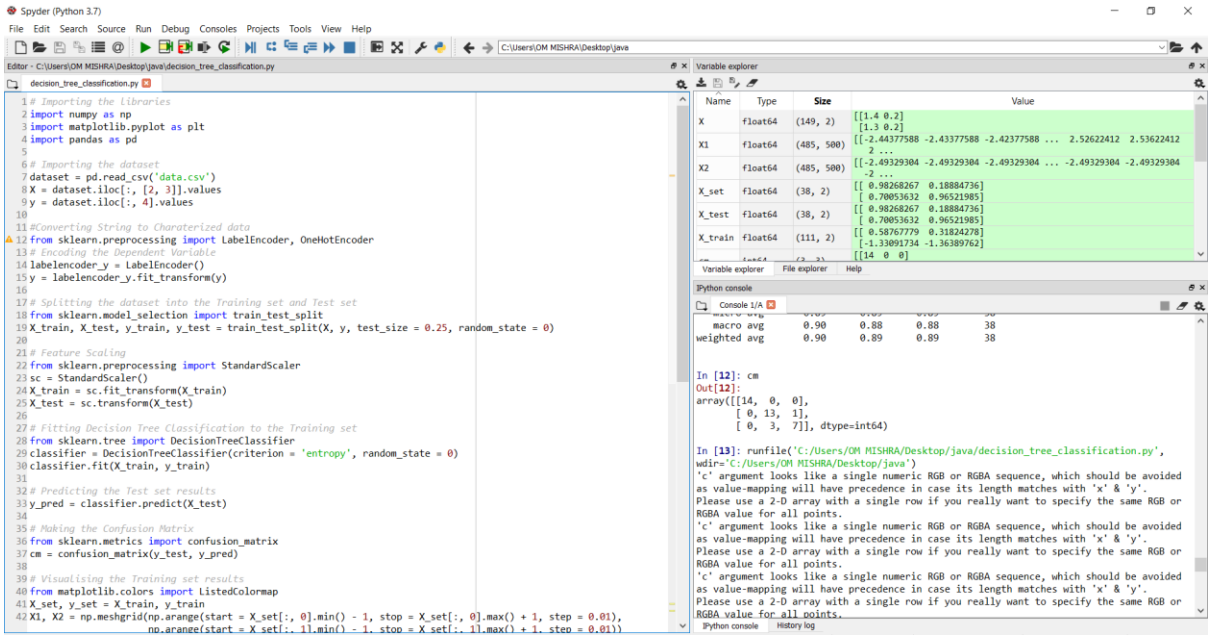
```
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
```

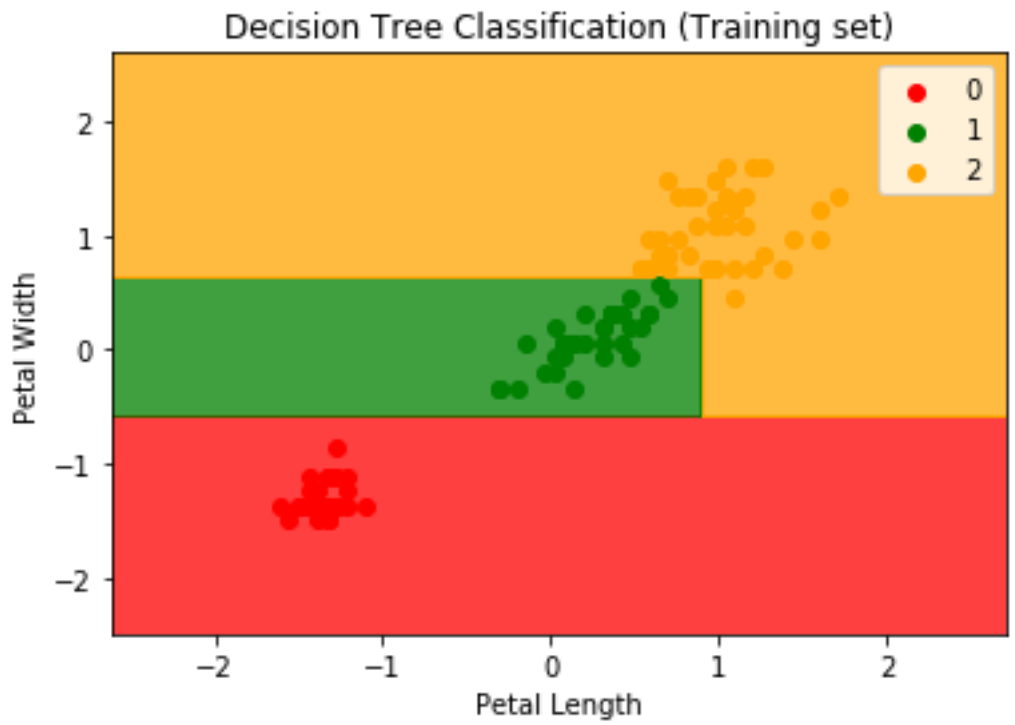
```
import pandas as pd

# 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
```

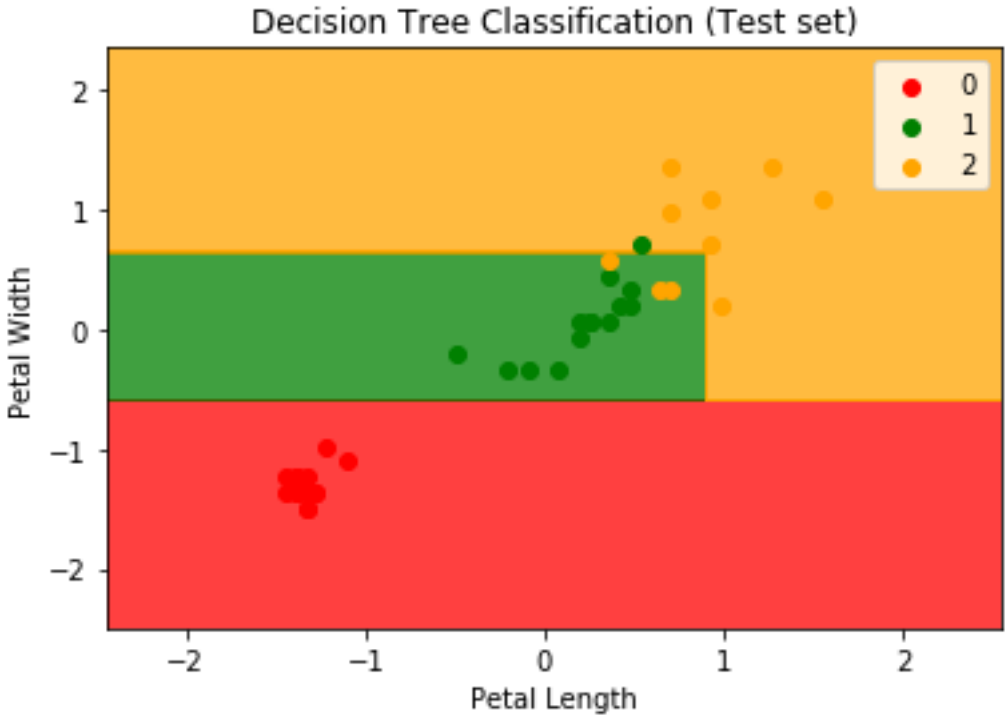
```
from matplotlib.colors import ListedColormap
X_set, y_set = X_train, y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, 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:







	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

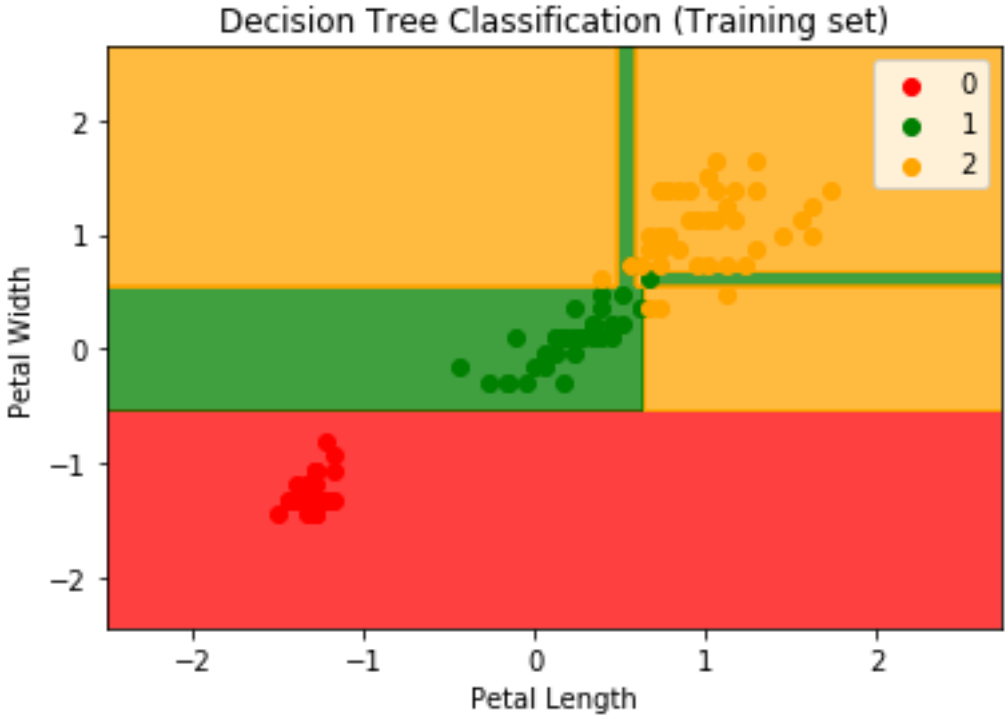
import matplotlib.pyplot as plt

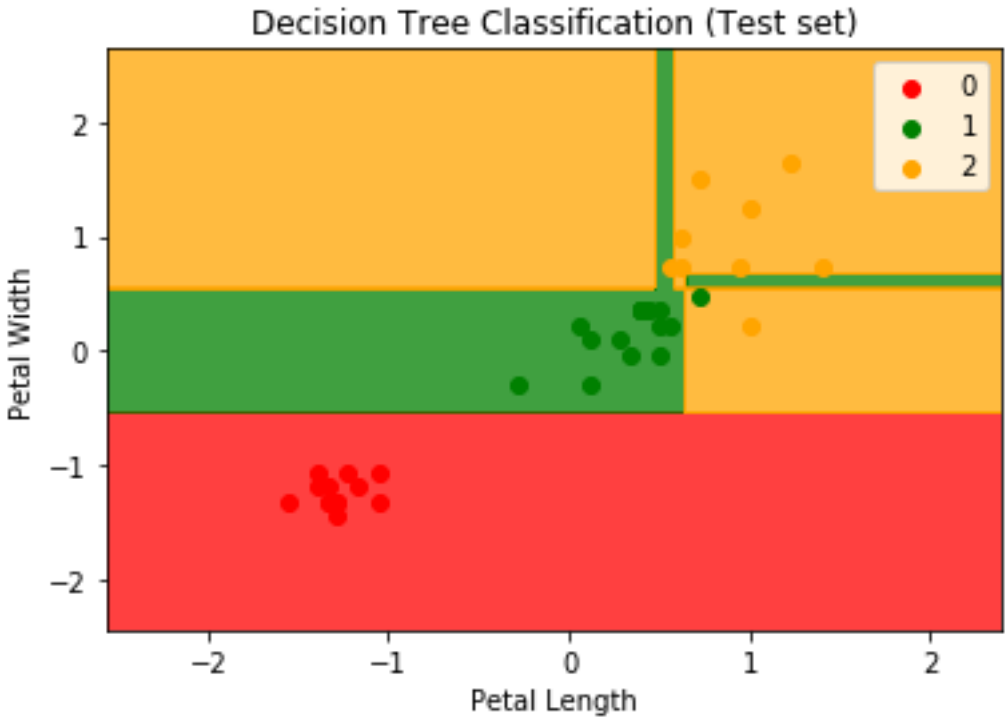
import pandas as pd

```
# 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