```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

```
dataset = pd.read_csv("/content/drive/MyDrive/DataSet/Social_Network_Ads.csv")
```

```
X = dataset.iloc[:, [2, 3]].values
```

```
y = dataset.iloc[:, 4].values
```

```
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 =
```

```
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)
```

```
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(criterion='entropy', random_state=0)
classifier.fit(X_train, y_train)
```

```
DecisionTreeClassifier(criterion='entropy', random_state=0)
```

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

```
for i in range(len(y_pred)):
    print(y_test[i], y_pred[i])
```

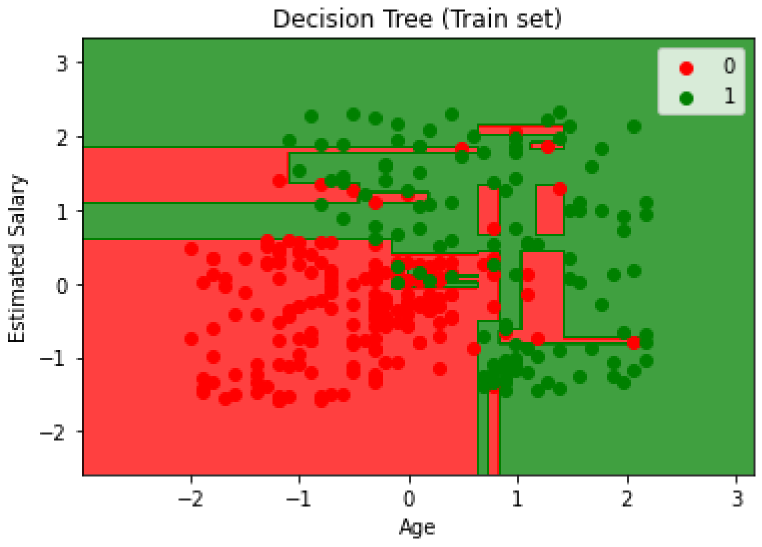
```
0 0
0 0
1 1
0 0
0 0
1 1
0 0
1 1
1 1
0 0
0 0
0 1
1 1
1 1
0 0
0 0
1 1
0 0
0 0
1 1
0 0
1 1
```

```
    0 0
    1 1
    0 0
    0 0
    0 0
    0 1
    1 1
    0 0
    0 0
    1 1
    0 0
    0 0
    0 0
    0 0
    1 1
    1 1
    1 1
    0 1
    0 0
    0 0
    1 1
    1 0
    0 0
    1 1
    1 1
    0 0
    0 0
    1 1
    0 0
    0 0
    0 0
    1 1
    0 0
    1 1
    1 1
    1 1
```

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
```

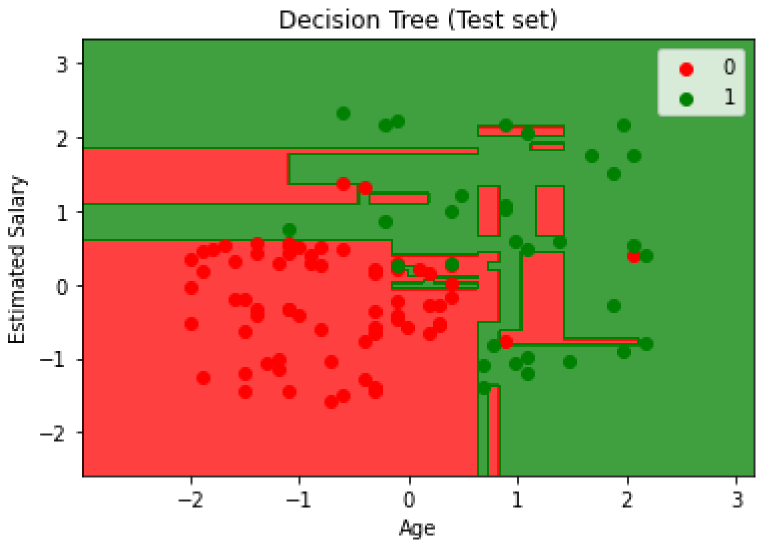
```
# Visualising the Training set results
from matplotlib.colors import ListedColormap
X_set, y_set = X_train, y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.s
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
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'))(i), label = j)
plt.title('Decision Tree (Train set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

```
    *c* argument looks like a single numeric RGB or RGBA sequence, which should be avoide
    *c* argument looks like a single numeric RGB or RGBA sequence, which should be avoide
```



```
# 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
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.s
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
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'))(i), label = j)
plt.title('Decision Tree (Test set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

```
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoide
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoide
```



✓ 0s completed at 7:19 PM