

```
# 7A Developing and implementing Decision Tree model on the
# dataset.
```

```
import numpy as nm
import matplotlib.pyplot as mtp
import pandas as pd
```

```
data_set= pd.read_csv('/content/drive/MyDrive/KRAI/User_data.csv')
#Extracting Independent and dependent Variable
x= data_set.iloc[:, [2,3]].values
y= data_set.iloc[:, 4].values
print(x)
print(y)
```

```
[[ 19 19000]
 [ 35 20000]
 [ 26 43000]
 [ 27 57000]
 [ 19 76000]
 [ 27 58000]
 [ 27 84000]
 [ 32 150000]
 [ 25 33000]
 [ 35 65000]
 [ 26 80000]
 [ 26 52000]
 [ 20 86000]
 [ 32 18000]
 [ 18 82000]
 [ 29 80000]
 [ 47 25000]
 [ 45 26000]
 [ 46 28000]
 [ 48 29000]
 [ 45 22000]
 [ 47 49000]
 [ 48 41000]
 [ 45 22000]
 [ 46 23000]
 [ 47 20000]
 [ 49 28000]
 [ 47 30000]
 [ 29 43000]
 [ 31 18000]
 [ 31 74000]
 [ 27 137000]
 [ 21 16000]
 [ 28 44000]
 [ 27 90000]
 [ 35 27000]
 [ 33 28000]
```

```
[ 30 49000]
[ 26 72000]
[ 27 31000]
[ 27 17000]
[ 33 51000]
[ 35 108000]
[ 30 15000]
[ 28 84000]
[ 23 20000]
[ 25 79000]
[ 27 54000]
[ 30 135000]
[ 31 89000]
[ 24 32000]
[ 18 44000]
[ 29 83000]
[ 35 23000]
[ 27 58000]
[ 24 55000]
[ 23 48000]
[ 30 70000]
```

```
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
st_x= StandardScaler()
x_train= st_x.fit_transform(x_train)
x_test= st_x.transform(x_test)
```

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

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

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

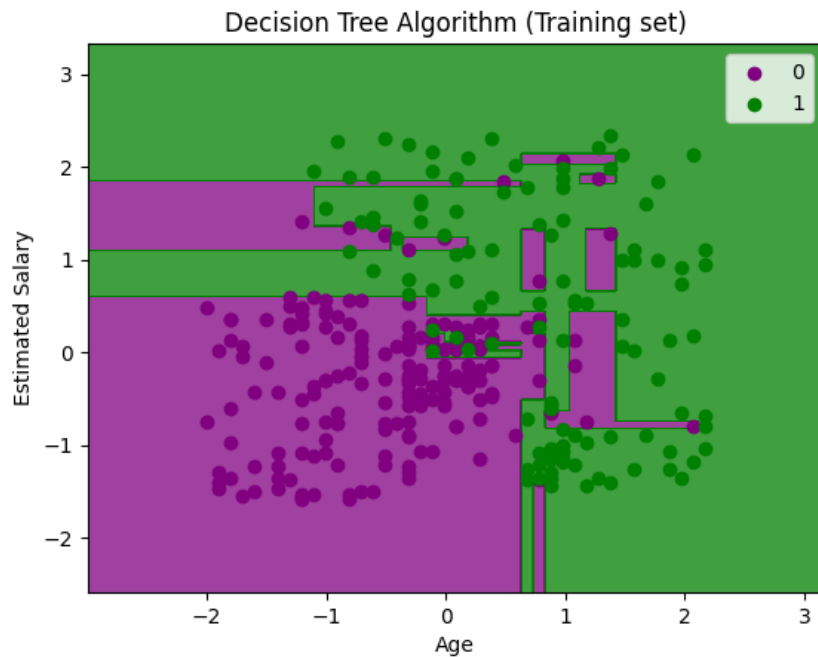
```
array([0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1,
       0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1,
       1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1])
```

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

```
array([[62, 6],
       [ 3, 29]])
```

```
from matplotlib.colors import ListedColormap
x_set, y_set = x_train, y_train
x1, x2 = nm.meshgrid(nm.arange(start = x_set[:, 0].min() - 1, stop = x_set[:, 0].max() + 1, step = 0.01),
nm.arange(start = x_set[:, 1].min() - 1, stop = x_set[:, 1].max() + 1, step = 0.01))
mtp.contourf(x1, x2, classifier.predict(nm.array([x1.ravel(), x2.ravel()]).T).reshape(x1.shape),
alpha = 0.75, cmap = ListedColormap(('purple', 'green' )))
mtp.xlim(x1.min(), x1.max())
mtp.ylim(x2.min(), x2.max())
for i, j in enumerate(nm.unique(y_set)):
    mtp.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1], c = ListedColormap(('purple', 'green'))(i), label = j)
mtp.title('Decision Tree Algorithm (Training set)')
mtp.xlabel('Age')
mtp.ylabel('Estimated Salary')
mtp.legend()
mtp.show()
```

```
<ipython-input-7-7887977755a3>:10: UserWarning: *c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have precedence in
mtp.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1], c = ListedColormap(('purple', 'green'))(i), label = j)
```



```
#Visulaizing the test set result
from matplotlib.colors import ListedColormap
x_set, y_set = x_test, y_test
x1, x2 = nm.meshgrid(nm.arange(start = x_set[:, 0].min() - 1, stop = x_set[:, 0].max() + 1, step = 0.01),
nm.arange(start = x_set[:, 1].min() - 1, stop = x_set[:, 1].max() + 1, step = 0.01))
mtp.contourf(x1, x2, classifier.predict(nm.array([x1.ravel(), x2.ravel()]).T).reshape(x1.shape),
alpha = 0.75, cmap = ListedColormap(('purple', 'green' )))
mtp.xlim(x1.min(), x1.max())
mtp.ylim(x2.min(), x2.max())
for i, j in enumerate(nm.unique(y_set)):
    mtp.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1], c = ListedColormap(('purple', 'green'))(i), label = j)
mtp.title('Decision Tree Algorithm(Test set)')
mtp.xlabel('Age')
mtp.ylabel('Estimated Salary')
mtp.legend()
mtp.show()
```

<ipython-input-8-a6028a9cd450>:11: UserWarning: \*c\* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have precedence in  
mtp.scatter(x\_set[y\_set == j, 0], x\_set[y\_set == j, 1], c = ListedColormap(('purple', 'green'))(i), label = j)

