In [1]:

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

In [2]:

```
dataset = pd.read_csv('suv_data.csv')
```

In [3]:

```
dataset.head()
```

Out[3]:

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0

In [4]:

```
dataset = dataset.drop(['User ID', 'Gender'], axis = 1)
```

In [5]:

```
x= dataset.iloc[:,:-1].values
y=dataset.iloc[:,-1].values
```

In [6]:

```
dataset.head()
```

Out[6]:

	Age	EstimatedSalary	Purchased
0	19	19000	0
1	35	20000	0
2	26	43000	0
3	27	57000	0
4	19	76000	0

In [7]:

```
dataset.shape
```

Out[7]:

```
(400, 3)
```

In [8]:

```
dataset.info
```

Out[8]:

```
<bound method DataFrame.info of      Age  EstimatedSalary  Purchased
0     19            19000          0
1     35            20000          0
2     26            43000          0
3     27            57000          0
4     19            76000          0
..   ...              ...        ...
395   46            41000          1
396   51            23000          1
397   50            20000          1
398   36            33000          0
399   49            36000          1

[400 rows x 3 columns]>
```

In [9]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=.20, random_state=1)
```

In [10]:

```
x_train.shape
```

Out[10]:

```
(320, 2)
```

In [11]:

```
x_test[:3]
```

Out[11]:

```
array([[    36,  33000],
       [    39,  61000],
       [    36, 118000]], dtype=int64)
```

In [12]:

```
print(y_train)
```

```
[0 1 1 0 0 0 1 0 1 0 1 1 1 0 0 1 1 1 0 0 0 1 0 1 0 1 0 1 0 0 0 0 0 0 1 0 0
 0 0 1 0 0 1 1 0 0 0 0 0 1 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0
 1 1 0 0 0 1 0 1 1 0 0 0 1 1 0 0 1 1 0 1 0 0 0 0 0 0 0 1 0 0 0 1 0 0 1 0 1
 0 1 0 0 0 0 1 0 0 1 0 0 1 1 0 1 0 0 0 1 1 1 0 0 0 0 0 1 1 0 0 0 1 0 0 0 1
 0 1 1 0 1 1 0 0 0 0 1 0 0 1 0 0 0 0 0 1 1 1 0 0 0 0 0 1 0 0 1 1 0 0 0 0 1
 0 1 0 1 0 0 1 0 1 0 0 0 0 1 0 1 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 1 0 0 0 0 1
 0 1 0 0 0 0 0 1 0 0 1 1 1 0 0 0 1 0 0 1 1 0 1 0 1 0 0 0 0 0 1 0 1 0 0 1 1
 0 0 1 1 1 0 0 0 1 0 1 1 1 0 0 1 0 0 0 1 0 0 0 1 0 1 0 1 0 1 1 0 0 1 0 0 1
 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1 0]
```

In [13]:

```
print(y_test)
```

```
[0 0 1 1 0 0 0 1 0 0 0 0 0 1 1 1 1 0 0 1 0 1 1 0 1 0 1 0 1 0 0 0 0 1 0 0 0
 0 1 0 1 1 0 0 1 1 1 1 0 1 0 0 1 1 1 0 1 0 1 1 0 0 0 0 1 1 0 0 0 0 0 0 1 0
 0 1 0 0 0 0]
```

In [14]:

```
# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train=sc.fit_transform(x_train)
x_test = sc.fit_transform(x_test)
```

In [15]:

```
print(x_train[0:10])
```

```
[[-0.80330081 -1.19121795]
 [ 0.75697997 -1.36859801]
 [ 0.85449752  1.43991958]
 [-0.51074816 -1.48685138]
 [-1.48592365  0.37563923]
 [-1.19337101  0.55301929]
 [ 1.04953262 -1.04340124]
 [-0.21819552 -0.30431766]
 [ 0.95201507 -1.33903467]
 [-1.09585346 -1.07296458]]
```

In [16]:

```
print(x_test[0:5])
```

```
[[-0.29863069 -1.23842019]
 [-0.02918947 -0.42323911]
 [-0.29863069  1.23623667]
 [-0.02918947  1.35269111]
 [-1.19676812  1.23623667]]
```

In [17]:

```
#build a model
from sklearn.ensemble import RandomForestClassifier
from sklearn import tree
classifier = RandomForestClassifier(n_estimators=5, criterion='entropy', random_state = 1)
classifier.fit(x_train,y_train)
```

Out[17]:

```
RandomForestClassifier(criterion='entropy', n_estimators=5, random_state=1)
```

In [18]:

```
#predicting a new result
print(classifier.predict(sc.fit_transform([[30,87000]])))
```

```
[0]
```

In [19]:

```
#predicting the test set results
y_pred = classifier.predict(x_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1))
```

```
[[0 0]
 [0 0]
 [1 1]
 [1 1]
 [1 0]
 [0 0]
 [0 0]
 [0 1]
 [0 0]
 [1 0]
 [0 0]
 [0 0]
 [0 0]
 [1 1]
 [1 1]
 [1 1]
 [1 1]
 [0 0]
 [0 0]
 [1 1]
 [0 0]
 [1 1]
 [1 1]
 [1 0]
 [0 1]
 [0 0]
 [1 1]
 [0 0]
 [1 1]
 [1 0]
 [0 0]
 [0 0]
 [0 0]
 [1 1]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [1 1]
 [0 0]
 [1 1]
 [1 1]
 [1 0]
 [0 0]
 [1 1]
 [0 1]
 [1 1]
 [1 1]
 [0 0]
 [1 1]
 [0 0]
 [0 0]
 [0 1]
 [0 1]
 [0 1]
 [0 0]
 [1 1]
 [0 0]
 [1 1]
 [1 1]
 [0 0]
 [0 0]
 [1 0]
 [0 0]
 [1 1]
 [1 1]
 [0 0]
 [0 0]
 [1 0]
 [0 0]
 [1 0]
 [0 0]
 [1 1]
 [0 0]
 [0 0]
 [1 1]
 [0 0]
 [0 0]
 [0 0]
 [0 0]]
```

In [20]:

```
#confusion matrix
from sklearn.metrics import confusion_matrix, accuracy_score
cm=confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)
```

```
[[40  8]
 [ 6 26]]
```

Out[20]:

```
0.825
```

In [23]:

```
plt.figure(figsize=(15,10))
tree.plot_tree(classifier.estimators_[4], filled = True)
```

Out[23]:

```
[Text(395.25, 513.4, 'X[0] <= 0.513\nentropy = 0.931\nsamples = 204\nvalue
= [209, 111]'),
 Text(201.5, 453.0, 'X[1] <= 0.656\nentropy = 0.666\nsamples = 152\nvalue
= [200, 42]'),
 Text(93.0, 392.6, 'X[0] <= -0.072\nentropy = 0.237\nsamples = 129\nvalue
= [198, 8]'),
 Text(62.0, 332.2, 'entropy = 0.0\nsamples = 85\nvalue = [142, 0]'),
 Text(124.0, 332.2, 'X[0] <= 0.026\nentropy = 0.544\nsamples = 44\nvalue =
[56, 8]'),
 Text(62.0, 271.8, 'X[1] <= -0.142\nentropy = 0.9\nsamples = 9\nvalue = [1
3, 6]'),
 Text(31.0, 211.39999999999998, 'entropy = 0.0\nsamples = 4\nvalue = [7,
0]'),
 Text(93.0, 211.39999999999998, 'X[1] <= 0.065\nentropy = 1.0\nsamples = 5
\nvalue = [6, 6]'),
 Text(62.0, 151.0, 'entropy = 0.0\nsamples = 1\nvalue = [0, 4]'),
 Text(124.0, 151.0, 'X[1] <= 0.287\nentropy = 0.811\nsamples = 4\nvalue =
```

In [ ]: