Decision Tree (Binary classification)

import required package

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

read data from source and describing

In [2]:
```
df = pd.read_csv('social_network_ads.csv')
df.head()
```

Out[2]:

	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 [3]:
```
print(df.columns)
```

```
Index(['User ID', 'Gender', 'Age', 'EstimatedSalary', 'Purchased'], dtype='object')
```

In [4]:
```
print(df.describe())
```

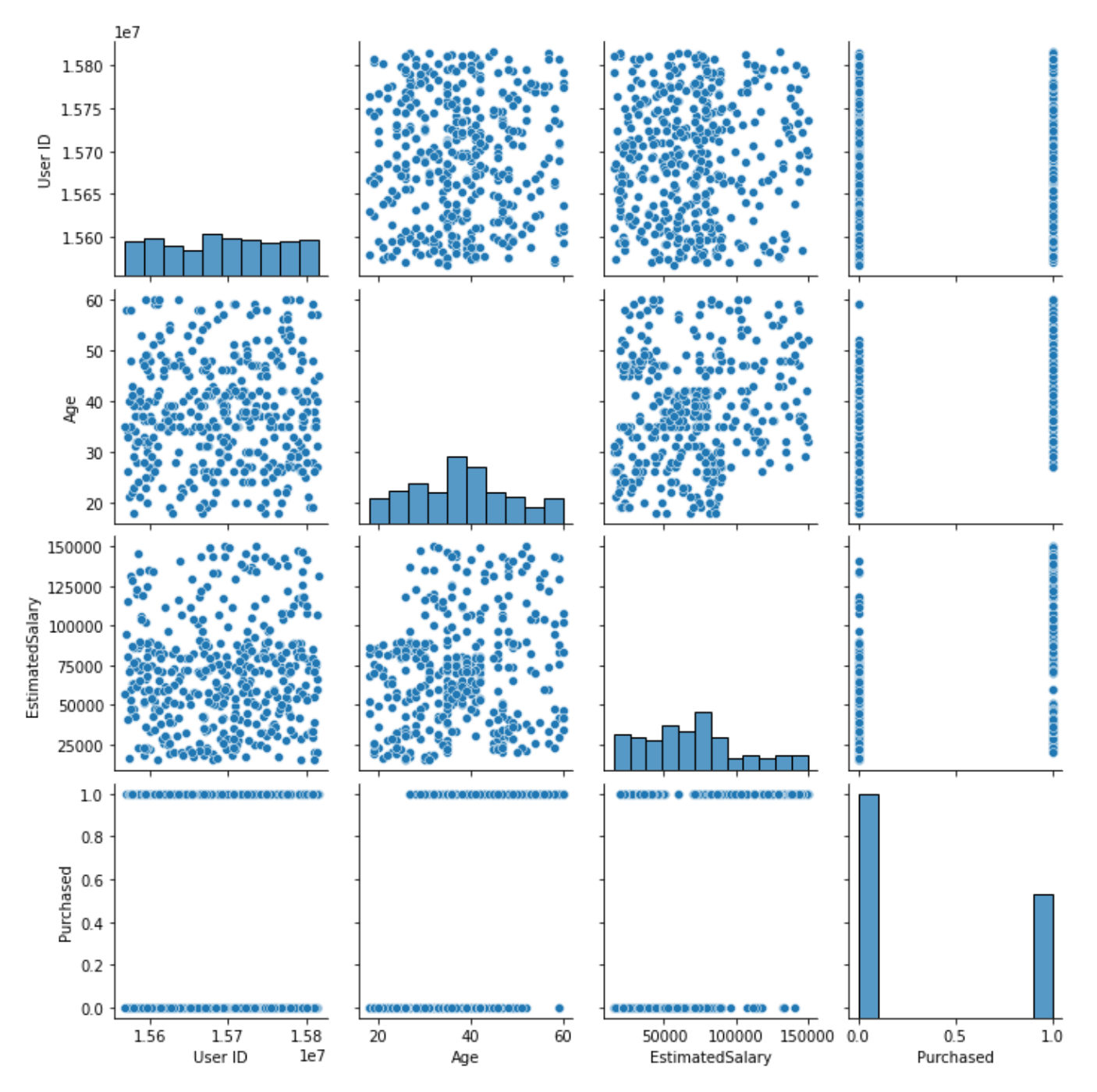
```
            User ID         Age  EstimatedSalary   Purchased
count  4.000000e+02  400.000000       400.000000  400.000000
mean   1.569154e+07   37.655000     69742.500000    0.357500
std    7.165832e+04   10.482877     34096.960282    0.479864
min    1.556669e+07   18.000000     15000.000000    0.000000
25%    1.562676e+07   29.750000     43000.000000    0.000000
50%    1.569434e+07   37.000000     70000.000000    0.000000
75%    1.575036e+07   46.000000     88000.000000    1.000000
max    1.581524e+07   60.000000    150000.000000    1.000000
```

In [5]:
```
## chec for data types
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   User ID          400 non-null    int64 
 1   Gender           400 non-null    object
 2   Age              400 non-null    int64 
 3   EstimatedSalary  400 non-null    int64 
 4   Purchased        400 non-null    int64 
dtypes: int64(4), object(1)
memory usage: 15.8+ KB
None
```

In [6]:
```
sns.pairplot(df)
```

Out[6]: <seaborn.axisgrid.PairGrid at 0x2189672a160>



check the relation between variables

In [7]:
```
corr = df.corr()
corr.style.background_gradient(cmap = 'Greens')
```

Out[7]:

	User ID	Age	EstimatedSalary	Purchased
User ID	1.000000	-0.000721	0.071097	0.007120
Age	-0.000721	1.000000	0.155238	0.622454
EstimatedSalary	0.071097	0.155238	1.000000	0.362083
Purchased	0.007120	0.622454	0.362083	1.000000

select input and op variable

In [9]:
```
x = df.drop(['User ID','Gender','Purchased'], axis=1)
y = df['Purchased']
```

split the data

In [10]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, random_state= 123456, train_
```

creating a model

In [22]:
```
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier(criterion='entropy')
```

In [23]:
```
## fit the data
model.fit(x_train, y_train)
```

Out[23]: DecisionTreeClassifier(criterion='entropy')

parameters to fine tune model

In [13]:
```
print(model.get_params())
```

```
{'ccp_alpha': 0.0, 'class_weight': None, 'criterion': 'gini', 'max_depth': None, 'max_features': None, 'max_leaf_nodes': None, 'min_impurity_decrease': 0.0, 'min_impurity_split': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'min_weight_fraction_leaf': 0.0, 'random_state': None, 'splitter': 'best'}
```

predict the values for unseen data

In [24]:
```
y_prediction = model.predict(x_test)
#print(y_prediction)
```

evaluation of classification model

In [25]:
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_prediction)
print(cm)
```

```
[[51  5]
 [ 6 18]]
```

In [26]:
```
correct = cm[0,0] + cm[1,1]
wrong = cm[1,0] + cm[0,1]
total = correct + wrong
accuracy = correct/total
print(accuracy)
```

```
0.8625
```

In [27]:
```
from sklearn.metrics import accuracy_score
print(accuracy_score(y_test, y_prediction))
```

```
0.8625
```

precision value

In [28]:
```
from sklearn.metrics import precision_score
print(precision_score(y_test, y_prediction))
```

```
0.782608695652174
```

recal value

In [29]:
```
from sklearn.metrics import recall_score
print(recall_score(y_test, y_prediction))
```

```
0.75
```

F1 score

In [30]:
```
from sklearn.metrics import f1_score
print(f1_score(y_test, y_prediction))
```

```
0.7659574468085107
```

classification report

In [31]:
```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_prediction))
```

```
              precision    recall  f1-score   support

           0       0.89      0.91      0.90        56
           1       0.78      0.75      0.77        24

    accuracy                           0.86        80
   macro avg       0.84      0.83      0.83        80
weighted avg       0.86      0.86      0.86        80
```

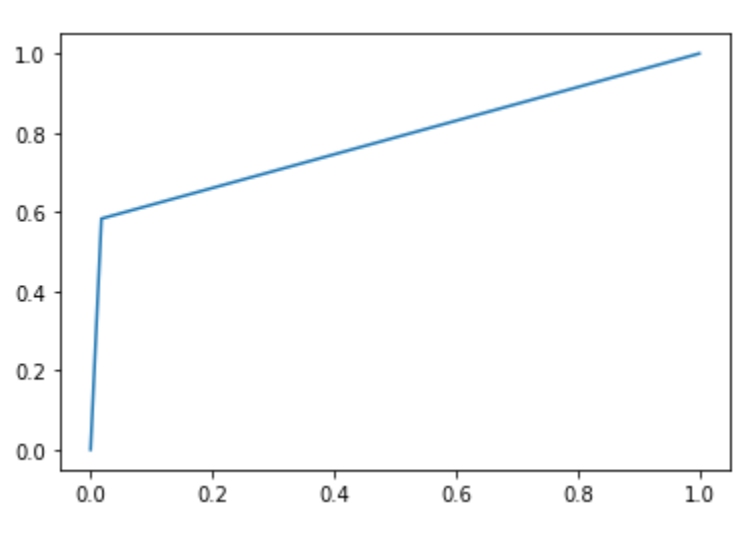
In [32]:
```
from sklearn.metrics import roc_curve, roc_auc_score, plot_roc_curve

print(roc_auc_score(y_test, y_prediction))
```

```
0.8303571428571429
```

In [82]:
```
fpr, tpr, threshold = roc_curve(y_test, y_prediction)
print(fpr)
print(tpr)
print(threshold)
plt.plot(fpr, tpr)
```

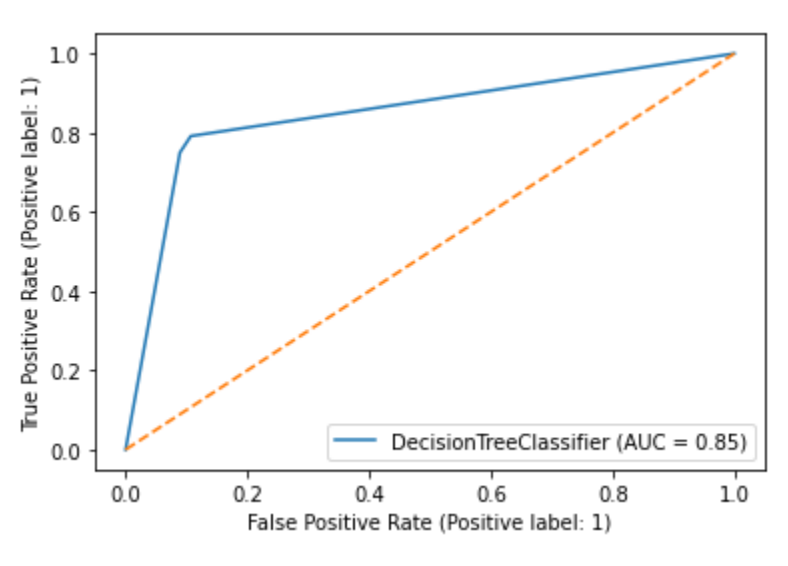
```
[0.         0.01785714 1.        ]
[0.         0.58333333 1.        ]
[2 1 0]
```

Out[82]: [<matplotlib.lines.Line2D at 0x1afab2ab5b0>]



In [33]:
```
plot = plot_roc_curve(model, x_test, y_test)
plt.plot([0,1], [0,1], linestyle = '--')
```

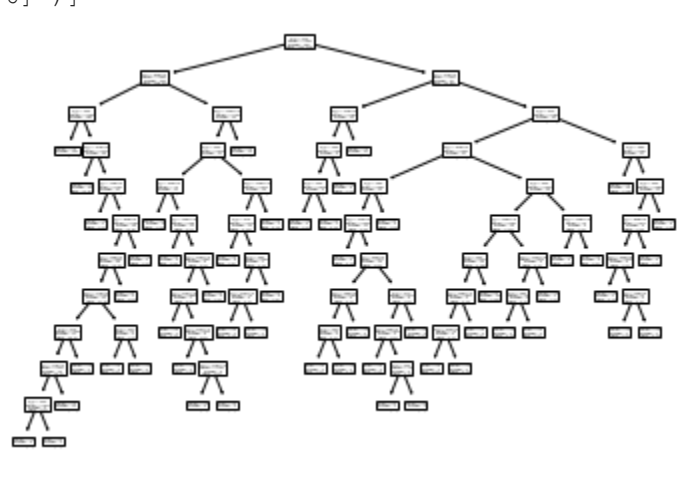
Out[33]: [<matplotlib.lines.Line2D at 0x218c3b86430>]



visualization

In [34]:
```
from sklearn.tree import plot_tree
plot_tree(model)
```

Out[34]:
```
[Text(145.33777173913043, 208.38, 'X[0] <= 44.5\nentropy = 0.952\nsamples = 320\nvalue = [201, 119]'),
 Text(72.78260869565217, 190.26, 'X[1] <= 89500.0\nentropy = 0.648\nsamples = 223\nvalue = [186, 37]'),
 Text(36.391304347826086, 172.14, 'X[0] <= 36.5\nentropy = 0.208\nsamples = 183\nvalue = [177, 6]'),
 Text(29.11304347826087, 154.01999999999998, 'entropy = 0.0\nsamples = 124\nvalue = [124, 0]'),
 Text(43.66956521739131, 154.01999999999998, 'X[1] <= 67500.0\nentropy = 0.474\nsamples = 59\nvalue = [53, 6]'),
 Text(36.391304347826086, 135.89999999999998, 'entropy = 0.0\nsamples = 32\nvalue = [32, 0]'),
 Text(50.947826086956525, 135.89999999999998, 'X[1] <= 70500.0\nentropy = 0.764\nsamples = 27\nvalue = [21, 6]'),
 Text(43.66956521739131, 117.77999999999999, 'entropy = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(58.22608695652174, 117.77999999999999, 'X[1] <= 83500.0\nentropy = 0.706\nsamples = 26\nvalue = [21, 5]'),
 Text(50.947826086956525, 99.66, 'X[1] <= 79500.0\nentropy = 0.634\nsamples = 25\nvalue = [21, 4]'),
 Text(43.66956521739131, 81.53999999999999, 'X[1] <= 77500.0\nentropy = 0.702\nsamples = 21\nvalue = [17, 4]'),
 Text(29.11304347826087, 63.41999999999999, 'X[0] <= 41.5\nentropy = 0.523\nsamples = 17\nvalue = [15, 2]'),
 Text(21.834782608695654, 45.29999999999998, 'X[1] <= 71500.0\nentropy = 0.337\nsamples = 16\nvalue = [15, 1]'),
 Text(14.556521739130435, 27.17999999999998, 'X[0] <= 39.5\nentropy = 0.811\nsamples = 4\nvalue = [3, 1]'),
 Text(7.278260869565218, 9.059999999999974, 'entropy = 0.0\nsamples = 3\nvalue = [3, 0]'),
 Text(21.834782608695654, 9.059999999999974, 'entropy = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(29.11304347826087, 27.17999999999998, 'entropy = 0.0\nsamples = 12\nvalue = [12, 0]'),
 Text(36.391304347826086, 45.29999999999998, 'entropy = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(58.22608695652174, 63.41999999999999, 'X[0] <= 39.0\nentropy = 1.0\nsamples = 4\nvalue = [2, 2]'),
 Text(50.947826086956525, 45.29999999999998, 'entropy = 0.0\nsamples = 2\nvalue = [0, 2]'),
 Text(65.50434782608696, 45.29999999999998, 'entropy = 0.0\nsamples = 2\nvalue = [2, 0]'),
 Text(58.22608695652174, 81.53999999999999, 'entropy = 0.0\nsamples = 4\nvalue = [4, 0]'),
 Text(65.50434782608696, 99.66, 'entropy = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(109.17391304347827, 172.14, 'X[1] <= 133500.0\nentropy = 0.769\nsamples = 40\nvalue = [9, 31]'),
 Text(101.89565217391305, 154.01999999999998, 'X[0] <= 35.5\nentropy = 0.918\nsamples = 27\nvalue = [9, 18]'),
 Text(80.06086956521739, 135.89999999999998, 'X[1] <= 93000.0\nentropy = 1.0\nsamples = 14\nvalue = [7, 7]'),
 Text(72.78260869565217, 117.77999999999999, 'entropy = 0.0\nsamples = 2\nvalue = [2, 0]'),
 Text(87.33913043478262, 117.77999999999999, 'X[1] <= 104000.0\nentropy = 0.98\nsamples = 12\nvalue = [5, 7]'),
 Text(80.06086956521739, 99.66, 'entropy = 0.0\nsamples = 3\nvalue = [0, 3]'),
 Text(94.61739130434783, 99.66, 'X[1] <= 118500.0\nentropy = 0.991\nsamples = 9\nvalue = [5, 4]'),
 Text(87.33913043478262, 81.53999999999999, 'X[1] <= 110000.0\nentropy = 0.863\nsamples = 7\nvalue = [5, 2]'),
 Text(80.06086956521739, 63.41999999999999, 'entropy = 0.0\nsamples = 1\nvalue = [1, 0]'),
 Text(94.61739130434783, 63.41999999999999, 'X[1] <= 112500.0\nentropy = 0.918\nsamples = 6\nvalue = [4, 2]'),
 Text(87.33913043478262, 45.29999999999998, 'entropy = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(101.89565217391305, 45.29999999999998, 'X[1] <= 116500.0\nentropy = 0.722\nsamples = 5\nvalue = [4, 1]'),
 Text(94.61739130434783, 27.17999999999998, 'entropy = 0.0\nsamples = 3\nvalue = [3, 0]'),
 Text(109.17391304347827, 27.17999999999998, 'entropy = 1.0\nsamples = 2\nvalue = [1, 1]'),
 Text(101.89565217391305, 81.53999999999999, 'entropy = 0.0\nsamples = 2\nvalue = [0, 2]'),
 Text(123.7304347826087, 135.89999999999998, 'X[1] <= 129500.0\nentropy = 0.619\nsamples = 13\nvalue = [2, 11]'),
 Text(116.45217391304348, 117.77999999999999, 'X[0] <= 37.5\nentropy = 0.414\nsamples = 12\nvalue = [1, 11]'),
 Text(109.17391304347827, 99.66, 'entropy = 0.0\nsamples = 5\nvalue = [0, 5]'),
 Text(123.7304347826087, 99.66, 'X[0] <= 38.5\nentropy = 0.592\nsamples = 7\nvalue = [1, 6]'),
 Text(116.45217391304348, 81.53999999999999, 'X[1] <= 112500.0\nentropy = 1.0\nsamples = 2\nvalue = [1, 1]'),
 Text(109.17391304347827, 63.41999999999999, 'entropy = 0.0\nsamples = 1\nvalue = [1, 0]'),
 Text(123.7304347826087, 63.41999999999999, 'entropy = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(131.0086956521739, 81.53999999999999, 'entropy = 0.0\nsamples = 5\nvalue = [0, 5]'),
 Text(131.0086956521739, 117.77999999999999, 'entropy = 0.0\nsamples = 1\nvalue = [1, 0]'),
 Text(116.45217391304348, 154.01999999999998, 'entropy = 0.0\nsamples = 13\nvalue = [0, 13]'),
 Text(217.8929347826087, 190.26, 'X[1] <= 41500.0\nentropy = 0.621\nsamples = 97\nvalue = [15, 82]'),
 Text(167.4, 172.14, 'X[1] <= 22500.0\nentropy = 0.196\nsamples = 33\nvalue = [1, 32]'),
 Text(160.12173913043478, 154.01999999999998, 'X[0] <= 46.5\nentropy = 0.65\nsamples = 6\nvalue = [1, 5]'),
 Text(152.84347826086957, 135.89999999999998, 'X[0] <= 45.5\nentropy = 0.918\nsamples = 3\nvalue = [1, 2]'),
 Text(145.56521739130434, 117.77999999999999, 'entropy = 0.0\nsamples = 2\nvalue = [0, 2]'),
 Text(160.12173913043478, 117.77999999999999, 'entropy = 0.0\nsamples = 1\nvalue = [1, 0]'),
 Text(167.4, 135.89999999999998, 'entropy = 0.0\nsamples = 3\nvalue = [0, 3]'),
 Text(174.67826086956524, 154.01999999999998, 'entropy = 0.0\nsamples = 27\nvalue = [0, 27]'),
 Text(268.3858695652174, 172.14, 'X[0] <= 52.5\nentropy = 0.758\nsamples = 64\nvalue = [14, 50]'),
 Text(223.80652173913043, 154.01999999999998, 'X[1] <= 82500.0\nentropy = 0.909\nsamples = 37\nvalue = [12, 25]'),
 Text(181.95652173913044, 135.89999999999998, 'X[0] <= 48.5\nentropy = 0.985\nsamples = 14\nvalue = [8, 6]'),
 Text(174.67826086956524, 117.77999999999999, 'X[1] <= 44000.0\nentropy = 0.994\nsamples = 11\nvalue = [5, 6]'),
 Text(167.4, 99.66, 'entropy = 0.0\nsamples = 1\nvalue = [1, 0]'),
 Text(181.95652173913044, 99.66, 'X[1] <= 55000.0\nentropy = 0.971\nsamples = 10\nvalue = [4, 6]'),
 Text(167.4, 81.53999999999999, 'X[1] <= 48000.0\nentropy = 0.722\nsamples = 5\nvalue = [1, 4]'),
 Text(160.12173913043478, 63.41999999999999, 'X[0] <= 46.0\nentropy = 1.0\nsamples = 2\nvalue = [1, 1]'),
 Text(152.84347826086957, 45.29999999999998, 'entropy = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(167.4, 45.29999999999998, 'entropy = 0.0\nsamples = 1\nvalue = [1, 0]'),
 Text(174.67826086956524, 63.41999999999999, 'entropy = 0.0\nsamples = 3\nvalue = [0, 3]'),
 Text(196.51304347826087, 81.53999999999999, 'X[0] <= 47.0\nentropy = 0.971\nsamples = 5\nvalue = [3, 2]'),
 Text(189.23478260869567, 63.41999999999999, 'X[1] <= 76500.0\nentropy = 0.811\nsamples = 4\nvalue = [3, 1]'),
 Text(181.95652173913044, 45.29999999999998, 'entropy = 0.0\nsamples = 2\nvalue = [2, 0]'),
 Text(196.51304347826087, 45.29999999999998, 'X[0] <= 45.5\nentropy = 1.0\nsamples = 2\nvalue = [1, 1]'),
 Text(189.23478260869567, 27.17999999999998, 'entropy = 0.0\nsamples = 1\nvalue = [1, 0]'),
 Text(203.7913043478261, 27.17999999999998, 'entropy = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(203.7913043478261, 63.41999999999999, 'entropy = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(189.23478260869567, 117.77999999999999, 'entropy = 0.0\nsamples = 3\nvalue = [3, 0]'),
 Text(265.6565217391304, 135.89999999999998, 'X[0] <= 50.5\nentropy = 0.667\nsamples = 23\nvalue = [4, 19]'),
 Text(247.4608695652174, 117.77999999999999, 'X[1] <= 139500.0\nentropy = 0.485\nsamples = 19\nvalue = [2, 17]'),
 Text(232.90434782608696, 99.66, 'X[0] <= 46.5\nentropy = 0.337\nsamples = 16\nvalue = [1, 15]'),
 Text(225.62608695652173, 81.53999999999999, 'X[1] <= 106500.0\nentropy = 0.811\nsamples = 4\nvalue = [1, 3]'),
 Text(218.34782608695653, 63.41999999999999, 'X[1] <= 92000.0\nentropy = 1.0\nsamples = 2\nvalue = [1, 1]'),
 Text(211.0695652173913, 45.29999999999998, 'entropy = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(225.62608695652173, 45.29999999999998, 'entropy = 0.0\nsamples = 1\nvalue = [1, 0]'),
 Text(232.90434782608696, 63.41999999999999, 'entropy = 0.0\nsamples = 2\nvalue = [0, 2]'),
 Text(240.1826086956522, 81.53999999999999, 'entropy = 0.0\nsamples = 12\nvalue = [0, 12]'),
 Text(262.0173913043478, 99.66, 'X[1] <= 142500.0\nentropy = 0.918\nsamples = 3\nvalue = [1, 2]'),
 Text(254.73913043478262, 81.53999999999999, 'X[0] <= 48.5\nentropy = 1.0\nsamples = 2\nvalue = [1, 1]'),
 Text(247.4608695652174, 63.41999999999999, 'entropy = 0.0\nsamples = 1\nvalue = [1, 0]'),
 Text(262.0173913043478, 63.41999999999999, 'entropy = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(269.295652173913, 81.53999999999999, 'entropy = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(283.8521739130435, 117.77999999999999, 'X[1] <= 136000.0\nentropy = 1.0\nsamples = 4\nvalue = [2, 2]'),
 Text(276.5739130434783, 99.66, 'entropy = 0.0\nsamples = 2\nvalue = [2, 0]'),
 Text(291.1304347826087, 99.66, 'entropy = 0.0\nsamples = 2\nvalue = [0, 2]'),
 Text(312.96521739130435, 154.01999999999998, 'X[0] <= 58.5\nentropy = 0.381\nsamples = 27\nvalue = [2, 25]'),
 Text(305.68695652173915, 135.89999999999998, 'entropy = 0.0\nsamples = 16\nvalue = [0, 16]'),
 Text(320.24347826086955, 135.89999999999998, 'X[0] <= 59.5\nentropy = 0.684\nsamples = 11\nvalue = [2, 9]'),
 Text(312.96521739130435, 117.77999999999999, 'X[1] <= 85500.0\nentropy = 0.971\nsamples = 5\nvalue = [2, 3]'),
 Text(305.68695652173915, 99.66, 'X[1] <= 59000.0\nentropy = 0.918\nsamples = 3\nvalue = [2, 1]'),
 Text(298.40869565217395, 81.53999999999999, 'entropy = 0.0\nsamples = 1\nvalue = [1, 0]'),
 Text(312.96521739130435, 81.53999999999999, 'X[1] <= 79500.0\nentropy = 1.0\nsamples = 2\nvalue = [1, 1]'),
 Text(305.68695652173915, 63.41999999999999, 'entropy = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(320.24347826086955, 63.41999999999999, 'entropy = 0.0\nsamples = 1\nvalue = [1, 0]'),
 Text(320.24347826086955, 99.66, 'entropy = 0.0\nsamples = 2\nvalue = [0, 2]'),
 Text(327.5217391304348, 117.77999999999999, 'entropy = 0.0\nsamples = 6\nvalue = [0, 6]')]
```



In [35]:
```
import graphviz
from sklearn import tree
file = open("tree.png", "w")
dot_data = tree.export_graphviz(model, out_file=None,
                      feature_names=x_test.columns,
                      filled=True, rounded=True,
                      special_characters=True)


graph = graphviz.Source(dot_data)
graph.format = 'png'
graph.render('dtree_render',view=True)
# graph
file.close()
```

In [ ]: