Applying Decision Tree on User Dataset

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

In [2]:
```
#importing datasets
data_set=pd.read_csv('user_data.csv')
```

In [3]:
```
data_set
```

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
...	...	...	...	...	...
395	15691863	Female	46	41000	1
396	15706071	Male	51	23000	1
397	15654296	Female	50	20000	1
398	15755018	Male	36	33000	0
399	15594041	Female	49	36000	1

400 rows × 5 columns

In [4]:
```
#extracting independent and dependent variables
x=data_set.iloc[:,[2,3]]
y=data_set.iloc[:,4]
```

In [5]:
```
#splitting the dataset into training 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)
```

In [6]:
```
x_train
```

Out[6]:

	Age	EstimatedSalary
250	44	39000
63	32	120000
312	38	50000
159	32	135000
283	52	21000
...	...	...
323	48	30000
192	29	43000
117	36	52000
47	27	54000
172	26	118000

300 rows × 2 columns

In [7]:
```
x_test
```

Out[7]:

	Age	EstimatedSalary
132	30	87000
309	38	50000
341	35	75000
196	30	79000
246	35	50000
...	...	...
146	27	96000
135	23	63000
390	48	33000
264	48	90000
364	42	104000

100 rows × 2 columns

In [8]:
```
y_train
```

Out[8]:
```
250    0
63     1
312    0
159    1
283    1
      ..
323    1
192    0
117    0
47     0
172    0
Name: Purchased, Length: 300, dtype: int64
```

In [9]:
```
y_test
```

Out[9]:
```
132    0
309    0
341    0
196    0
246    0
      ..
146    1
135    0
390    1
264    1
364    1
Name: Purchased, Length: 100, dtype: int64
```

In [10]:
```
#feature scaling
from sklearn.preprocessing import StandardScaler
st_x=StandardScaler()
x_train=st_x.fit_transform(x_train)
x_test=st_x.fit_transform(x_test)
```

In [11]:
```
x_train
```

Out[11]:
```
array([[ 0.58164944, -0.88670699],
       [-0.60673761,  1.46173768],
       [-0.01254409, -0.5677824 ],
       [-0.60673761,  1.89663484],
       [ 1.37390747, -1.40858358],
       [ 1.47293972,  0.99784738],
       [ 0.08648817, -0.79972756],
       [-0.01254409, -0.24885782],
       [-0.21060859, -0.5677824 ],
       [-0.21060859, -0.19087153],
       [-0.30964085, -1.29261101],
       [-0.30964085, -0.5677824 ],
       [ 0.38358493,  0.09905991],
       [ 0.8787462 , -0.59677555],
       [ 2.06713324, -1.17663843],
       [ 1.07681071, -0.13288524],
       [ 0.68068169,  1.78066227],
       [-0.70576986,  0.56295021],
       [ 0.77971394,  0.35999821],
```

In [12]:
```
x_test
```

Out[12]:
```
array([[-0.54748976,  0.5130727 ],
       [ 0.15442019, -0.61825566],
       [-0.10879604,  0.14615539],
       [-0.54748976,  0.26846116],
       [-0.10879604, -0.61825566],
       [-0.81070599, -1.53554892],
       [-0.45975102, -1.68843113],
       [-0.0210573 ,  2.25592989],
       [-1.60035469, -0.0678797 ],
       [ 0.94406888, -0.83229075],
       [-0.54748976, -0.6488321 ],
       [-0.72296725, -0.46537345],
       [ 0.06668145, -0.46537345],
       [ 0.24215893,  0.20730828],
       [-1.4248772 ,  0.48249625],
       [-0.37201227,  1.43036596],
       [ 0.06668145,  0.20730828],
       [-1.51261594,  0.45191981],
       [ 1.64597884,  1.8278597 ],
       [-0.10879604, -1.47439603],
       [-0.10879604, -0.70998498],
       [ 0.94406888,  2.25592989],
       [ 0.41763642, -0.58767922],
       [ 0.94406888,  1.06344865],
       [-1.16166097, -1.29093738],
       [ 1.11954637,  2.16420057],
       [-0.72296725,  0.5130727 ],
       [-0.63522851,  0.2990376 ],
       [ 0.06668145, -0.25133835],
       [-0.37201227,  0.48249625],
       [-1.33713846,  0.54364914],
       [ 0.06668145,  0.26846116],
       [ 1.82145632, -0.31249124],
       [ 0.06668145, -0.52652633],
       [-1.07392223, -0.37364412],
       [-1.60035469, -0.55710277],
       [-1.24939971,  0.32961404],
       [-0.19653479, -0.83229075],
       [-0.45975102, -1.10747873],
       [ 1.11954637, -1.04632585],
       [-0.81070599,  0.54364914],
       [ 0.41763642, -0.55710277],
       [-0.81070599,  0.42134337],
       [-0.10879604, -1.53554892],
       [ 0.59311391,  1.27748375],
       [-0.81070599, -0.37364412],
       [ 0.06668145,  0.2990376 ],
       [ 1.3827626 ,  0.60480202],
       [-0.89844474, -1.2297845 ],
       [ 1.11954637,  0.48249625],
       [ 1.82145632,  1.58324817],
       [-0.19653479, -1.38266671],
       [-0.10879604, -0.40422056],
       [-0.19653479,  1.36921307],
```

```
       [ 1.99693381,  0.54364914],
       [ 0.7685914 , -1.16863161],
       [-0.63522851,  0.39076693],
       [-0.89844474,  0.2990376 ],
       [ 1.11954637, -1.29093738],
       [-1.16166097, -1.53554892],
       [-0.37201227, -1.5967018 ],
       [ 2.08467255, -0.86286719],
       [-1.51261594,  0.17673183],
       [-0.0210573 ,  0.87999   ],
       [-1.51261594, -1.35209027],
       [ 2.08467255,  0.39076693],
       [-1.07392223,  0.57422558],
       [-0.81070599, -0.37364412],
       [ 0.32989768, -0.70998498],
       [ 0.50537516, -0.00672682],
       [-0.37201227,  2.43938854],
       [-0.10879604,  0.20730828],
       [-1.24939971, -0.22076191],
       [ 0.7685914 , -1.47439603],
       [-0.81070599,  0.57422558],
       [-1.60035469,  0.36019049],
       [ 0.50537516,  0.26846116],
       [ 0.32989768, -0.31249124],
       [ 1.47050135, -1.10747873],
       [ 0.94406888,  1.12460154],
       [ 1.90919507,  2.25592989],
       [ 1.99693381,  0.39076693],
       [-1.07392223, -0.46537345],
       [-0.89844474, -1.07690229],
       [ 1.90919507, -0.98517296],
       [ 0.50537516,  0.2990376 ],
       [ 0.32989768,  0.14615539],
       [ 1.99693381,  1.8278597 ],
       [ 0.85633014, -0.89344364],
       [ 0.41763642, -0.31249124],
       [ 0.50537516, -0.19018547],
       [ 0.06668145,  2.31708278],
       [-1.16166097, -0.67940854],
       [-0.98618348, -1.13805517],
       [-1.07392223,  0.42134337],
       [-0.81070599,  0.78826068],
       [-1.16166097, -0.22076191],
       [ 1.03180763, -1.13805517],
       [ 1.03180763,  0.60480202],
       [ 0.50537516,  1.03287221]])
```

In [13]:

```python
#Fitting a Decision Tree algorithm to the training set
from sklearn.tree import DecisionTreeClassifier
classifier=DecisionTreeClassifier(criterion='entropy',random_state=0)
classifier.fit(x_train,y_train)
```

Out[13]:

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

In [14]:
```
#now we will check whether the Decision is better by considering confusion matrix
#while considering confusion matrix we need to check predicted values with actual
#if correct predictions are more compared to incorrect prediction then we can cor
#predicting the test set result
y_pred=classifier.predict(x_test)
```

In [15]:
```
y_pred
```

Out[15]:
```
array([0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1,
       0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 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, 1, 0, 0, 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], dtype=int64)
```

In [16]:
```
#Creating confusion Matrix
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,y_pred)
```

In [17]:
```
cm
```

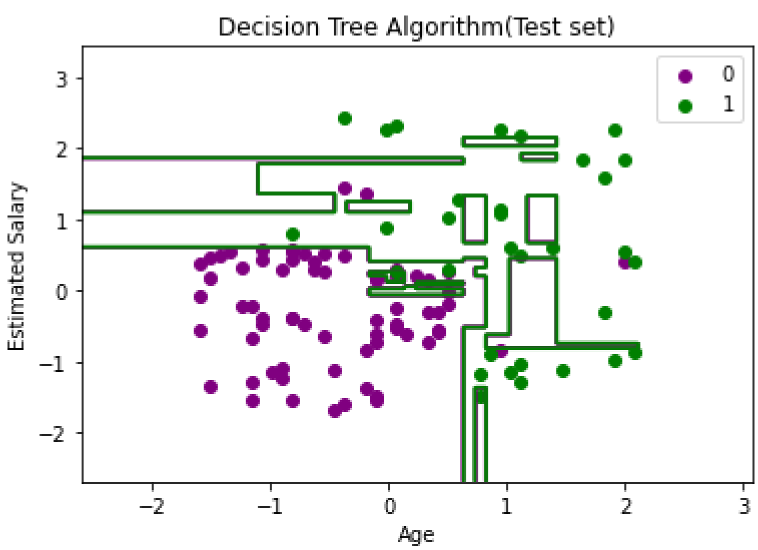
Out[17]:
```
array([[61,  7],
       [ 3, 29]], dtype=int64)
```

In [18]:
```
#as there are 7+3=10 incorrect predictions and 61+29=90 correct predictions we co
#visualizing the training dataset
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
np.arange(start=x_set[:,1].min()-1,stop=x_set[:,1].min()+1,step=0.01))

plt.contourf(x1,x2,classifier.predict(np.array([x1.ravel(),x2.ravel()]).T).reshap
             alpha=0.75,cmap=ListedColormap(('purple','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(('purple','green'))(i),label=j)
plt.title('Decision Tree Algorithm(Training 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
avoided as value-mapping will have precedence in case its length matches with *
x* & *y*.  Please use the *color* keyword-argument or provide a 2D array with a
single row if you intend to specify the same RGB or RGBA value for all points.
*c* argument looks like a single numeric RGB or RGBA sequence, which should be
avoided as value-mapping will have precedence in case its length matches with *
x* & *y*.  Please use the *color* keyword-argument or provide a 2D array with a
single row if you intend to specify the same RGB or RGBA value for all points.
```

In [19]:
```
#Visualizing the Test set result
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
np.arange(start=x_set[:,1].min()-1,stop=x_set[:,1].max()+1,step=0.01))

plt.contour(x1,x2,classifier.predict(np.array([x1.ravel(),x2.ravel()]).T).reshape
           alpha=0.75,cmap=ListedColormap(('purple','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(('purple','green'))(i),label=j)

plt.title('Decision Tree Algorithm(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
avoided as value-mapping will have precedence in case its length matches with *
x* & *y*.  Please use the *color* keyword-argument or provide a 2D array with a
single row if you intend to specify the same RGB or RGBA value for all points.
*c* argument looks like a single numeric RGB or RGBA sequence, which should be
avoided as value-mapping will have precedence in case its length matches with *
x* & *y*.  Please use the *color* keyword-argument or provide a 2D array with a
single row if you intend to specify the same RGB or RGBA value for all points.
```



In [ ]:

```

```