39. Pre and Post Pruning in a Decision Tree

- Pruning is performed to avoid your model from over-fitting
- **Pre-Pruning:** You perfrom pruning before making model
- **Post-Pruning:** You perfrom pruning after making model

In [2]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [3]:
```
dataset = pd.read_csv(r'Data/Social_Network_Ads_2.csv')
dataset.head(3)
```

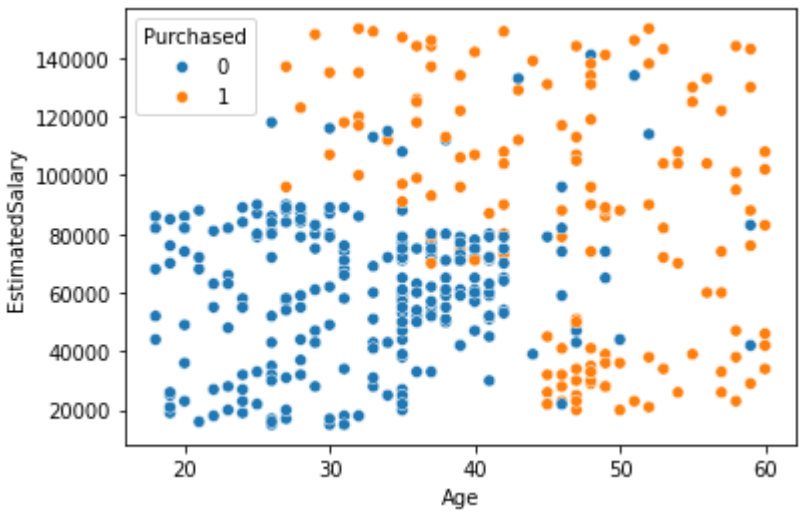
Out[3]:

	Age	EstimatedSalary	Purchased
0	19	19000	0
1	35	20000	0
2	26	43000	0

To see how splitting is taking place through graph

- Decision tree is non-linear algorithm

In [4]:
```
sns.scatterplot(x="Age", y="EstimatedSalary", data=dataset, hue="Purchased")
plt.show()
```



- So this is non-linear graph

Step 1: Check for missing data

In [5]:
```
dataset.isnull().sum()
```

Out[5]:
```
Age                0
EstimatedSalary    0
Purchased          0
dtype: int64
```

Step 2: Split the data into dependent and independent variables

In [6]:
```
x = dataset.iloc[:,:-1]
x
```

Out[6]:

	Age	EstimatedSalary
0	19	19000
1	35	20000
2	26	43000
3	27	57000
4	19	76000
...	...	...
395	46	41000
396	51	23000
397	50	20000
398	36	33000
399	49	36000

400 rows × 2 columns

In [7]:
```
y = dataset['Purchased']
y
```

Out[7]:
```
0      0
1      0
2      0
3      0
4      0
      ..
395    1
396    1
397    1
398    0
399    1
Name: Purchased, Length: 400, dtype: int64
```

Step 3: Do scaling of data

In [8]:
```
dataset.head(3)
```

Out[8]:

	Age	EstimatedSalary	Purchased
0	19	19000	0
1	35	20000	0
2	26	43000	0

Scaling is needed b/c there is huge difference between values of Age and EstimatedSalary.
So there is need to do scaling of data before model building

In [9]:
```
from sklearn.preprocessing import StandardScaler
```

In [10]:
```
sc = StandardScaler()
sc.fit(x)
# Next step will transform (sc.transform(x)) the data and will convert into datafra
x = pd.DataFrame(sc.transform(x), columns=x.columns)
```

In [11]:
```
x
```

Out[11]:

	Age	EstimatedSalary
0	-1.781797	-1.490046
1	-0.253587	-1.460681
2	-1.113206	-0.785290
3	-1.017692	-0.374182
4	-1.781797	0.183751
...	...	...
395	0.797057	-0.844019
396	1.274623	-1.372587
397	1.179110	-1.460681
398	-0.158074	-1.078938
399	1.083596	-0.990844

400 rows × 2 columns

Now our has been scalled

Step 3: Split the data into train and test dataset

In [12]:
```
from sklearn.model_selection import train_test_split
```

In [13]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.20, random_st
```

Step 4: Build Model through Decision Tree

- Decision tree can work for both classification through **DecisionTreeClassifier** or for regression through **DecisionTreeRegressor**
- As our output (dataset['Purchased']) consists of 0 and 1 form, so DecisionTreeClassifier will be used

In [14]:
```
from sklearn.tree import DecisionTreeClassifier
```

In [15]:
```
# default: DecisionTreeClassifier(criterion='gini')
dt = DecisionTreeClassifier()
dt.fit(x_train, y_train)
```

Out[15]:

▾ DecisionTreeClassifier

DecisionTreeClassifier()

Step 5: Check Accuracy of Built Model

In [16]:
```
dt.score(x_test, y_test)*100
```

Out[16]: 83.75

Step 6: Perform Predictions on Built Model

In [17]:
```
dataset.head(3)
```

Out[17]:

	Age	EstimatedSalary	Purchased
0	19	19000	0
1	35	20000	0
2	26	43000	0

In [18]:
```
dt.predict([[19,19000]])
```

```
C:\Users\rashi\AppData\Local\Programs\Python\Python39\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature names, but DecisionTreeClassifier was fitted with feature names
  warnings.warn(
```

Out[18]: array([1], dtype=int64)

It gave wrong prediction

dt.predict([[35,20000]])

again wrong prediction

39.1 Perform Pruning

- First check whether your model is over-fit, the model will be over-fit, if accuracy of the training model is high and testing model accuracy is significantly low.

In [20]:
```
dt.score(x_test, y_test)*100
```

Out[20]: 83.75

In [28]:
```
dt.score(x_train, y_train)*100
```

Out[28]: 99.6875

- See hug difference b/w accuracies of training and testing data, so the model is over-fit

39.2 Perform Pre-Pruning

In [31]:
```
# default: DecisionTreeClassifier(criterion='gini')
dtpre = DecisionTreeClassifier(max_depth=5)
dtpre.fit(x_train, y_train)
```

Out[31]:
```
DecisionTreeClassifier
DecisionTreeClassifier(max_depth=5)
```

In [32]:
```
dtpre.score(x_test, y_test)*100
```

Out[32]: 90.0

In [34]:
```
dtpre.score(x_train, y_train)*100
```

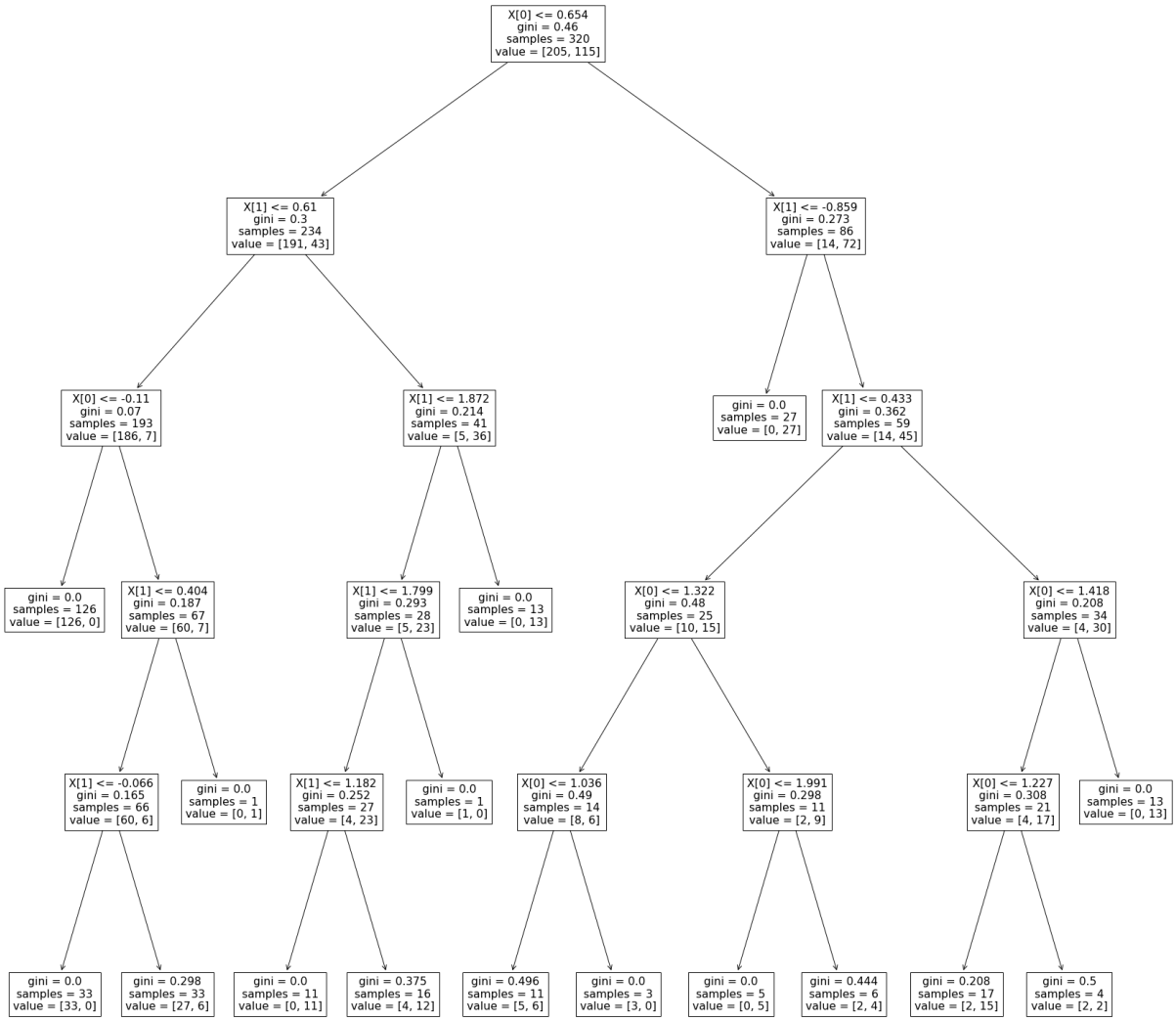
Out[34]: 93.4375

- See the difference between training data and testing data accuracy has been reduced, and hence over-fitting is reduced

39.2.1 Analysis of Model through Graph

In [37]:
```
from sklearn.tree import plot_tree
```

In [39]:
```
# plot_tree(decision_tree)
plt.figure(figsize=(30,30))
plot_tree(dtpre)
plt.savefig(r'Generated_images/decision-tree-demo-pre-prunning.jpg')
plt.show()
```

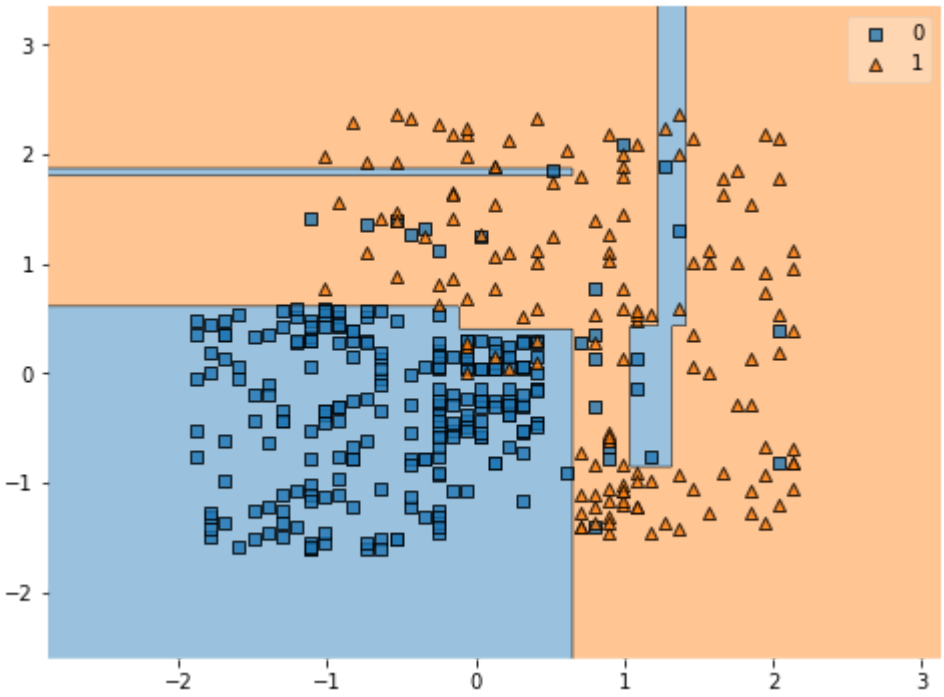


39.2.2 To see Non-linear line splitting

In [25]:
```
from mlxtend.plotting import plot_decision_regions
```

In [35]:
```
plt.figure(figsize=(8,6))
plot_decision_regions(x.to_numpy(),y.to_numpy(),clf=dtpre)
plt.show()
```

```
C:\Users\rashi\AppData\Local\Programs\Python\Python39\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature names, but DecisionTreeClassifier was fitted with feature names
  warnings.warn(
```



In [ ]:

39.3 Perform Post-Pruning

In [42]:
```
for i in range(1, 19):
    dtpost = DecisionTreeClassifier(max_depth=i)
    dtpost.fit(x_train, y_train)
    print(dtpost.score(x_train, y_train)*100, dtpost.score(x_test, y_test)*100, i)
```

```
82.1875 90.0 1
91.875 91.25 2
91.875 91.25 3
93.125 91.25 4
93.4375 90.0 5
95.0 86.25 6
96.875 85.0 7
97.5 85.0 8
98.125 85.0 9
98.4375 85.0 10
99.0625 83.75 11
99.375 83.75 12
99.375 83.75 13
99.6875 83.75 14
99.6875 83.75 15
99.6875 83.75 16
99.6875 83.75 17
99.6875 83.75 18
```

- the difference in training and testing accuracy is negligible for model 2 and 3, **so it means that max_depth value should be 2 or 3**

- We can take max_depth till 5, as there is no major difference b/w accuracies of training and testing models

In [48]:
```
dtpost1 = DecisionTreeClassifier(max_depth=3)
dtpost1.fit(x_train, y_train)
```

Out[48]:
```
DecisionTreeClassifier
DecisionTreeClassifier(max_depth=3)
```

In [50]:
```
dtpost1.score(x_test, y_test)*100
```

Out[50]: 91.25

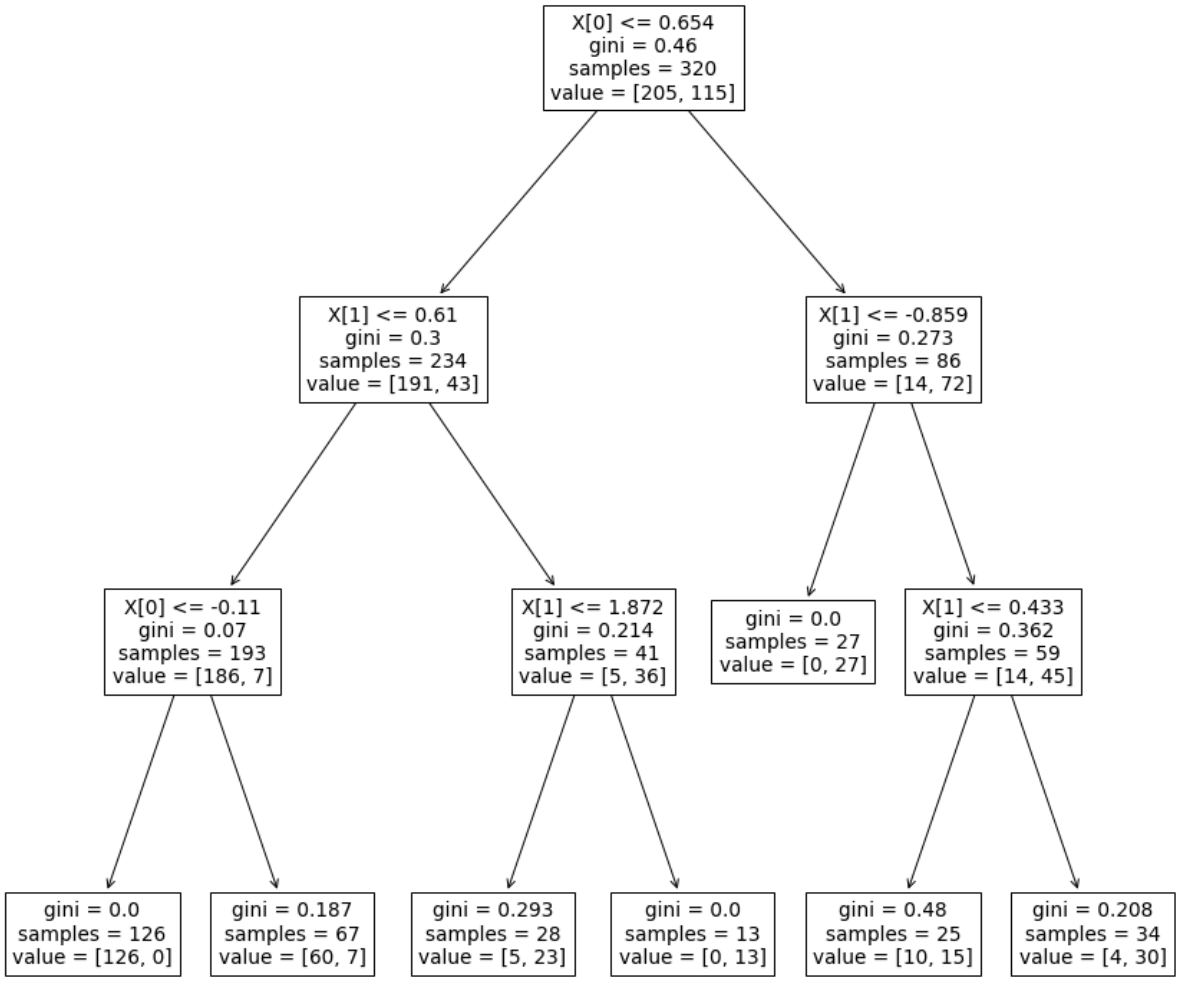
In [51]:
```
dtpost1.score(x_train, y_train)*100
```

Out[51]: 91.875

So Our model is no more over-fit

39.3.1 Analysis of Model through Graph

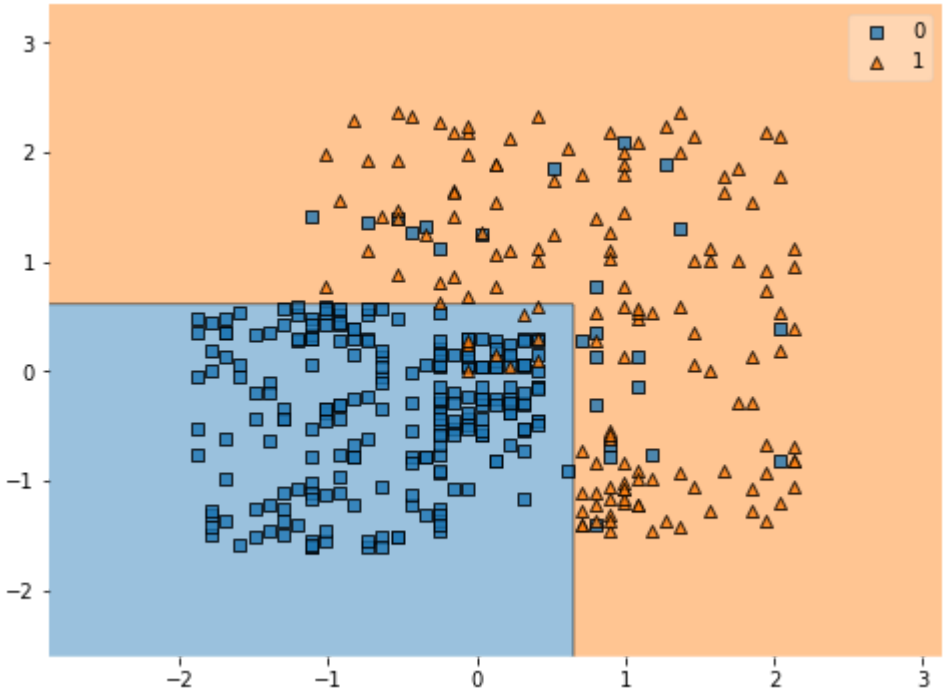
In [54]:
```
# plot_tree(decision_tree)
plt.figure(figsize=(15,15))
plot_tree(dtpost1)
plt.savefig(r'Generated_images/decision-tree-demo-post-prunning.jpg')
plt.show()
```



39.3.2 To see Non-linear line splitting

In [52]:
```
plt.figure(figsize=(8,6))
plot_decision_regions(x.to_numpy(),y.to_numpy(),clf=dtpost1)
plt.show()
```

```
C:\Users\rashi\AppData\Local\Programs\Python\Python39\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature names, but DecisionTreeClassifier was fitted with feature names
  warnings.warn(
```



In [ ]: