Objective :

"Predict behavior to retain customers. You can analyze all relevant customer data and develop focused customer retention programs."

Step 1 : Import Library and Dataset

```
import pandas as pd
import numpy as np
```

```
# Read the data in
import pandas as pd
from google.colab import drive        # Importing data from google drive.
drive.mount("/gdrive")                # mount is used when you have added external device as SSD, Hard

%cd /gdrive/My Drive/Imarticus/Machine_Learning_with_Python/Decision_Tree
employee = pd.read_csv(r"churn.csv")
```

```
Mounted at /gdrive
/gdrive/My Drive/Imarticus/Machine_Learning_with_Python/Decision_Tree
```

```
employee.head(5)
```

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	Mult
0	7590-VHVEG	Female	No	Yes	No	1	No	
1	5575-GNVDE	Male	No	No	No	34	Yes	
2	3668-QPYBK	Male	No	No	No	2	Yes	
3	7795-CFOCW	Male	No	No	No	45	No	
4	9237-HQITU	Female	No	No	No	2	Yes	

5 rows × 21 columns

Step 2 : Data Pre-Processing

Univariate Analysis

```
employee.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 21 columns):
```

```
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   customerID        7043 non-null   object
 1   gender            7043 non-null   object
 2   SeniorCitizen     7043 non-null   object
 3   Partner           7043 non-null   object
 4   Dependents        7043 non-null   object
 5   tenure            7043 non-null   int64
 6   PhoneService      7043 non-null   object
 7   MultipleLines     7043 non-null   object
 8   InternetService   7043 non-null   object
 9   OnlineSecurity    7043 non-null   object
 10  OnlineBackup      7043 non-null   object
 11  DeviceProtection  7043 non-null   object
 12  TechSupport       7043 non-null   object
 13  StreamingTV       7043 non-null   object
 14  StreamingMovies   7043 non-null   object
 15  Contract          7043 non-null   object
 16  PaperlessBilling  7043 non-null   object
 17  PaymentMethod     7043 non-null   object
 18  MonthlyCharges    7043 non-null   float64
 19  TotalCharges      7043 non-null   object
 20  Churn             7043 non-null   object
dtypes: float64(1), int64(1), object(19)
memory usage: 1.1+ MB
```

```
employee.describe()
```

	tenure	MonthlyCharges
count	7043.000000	7043.000000
mean	32.371149	64.761692
std	24.559481	30.090047
min	0.000000	18.250000
25%	9.000000	35.500000
50%	29.000000	70.350000
75%	55.000000	89.850000
max	72.000000	118.750000

Removing Irrelavent Variable

```
employee = employee.drop(['customerID'],axis=1)
employee.columns
```

```
Index(['gender', 'SeniorCitizen', 'Partner', 'Dependents', 'tenure',
       'PhoneService', 'MultipleLines', 'InternetService', 'OnlineSecurity',
       'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV',
       'StreamingMovies', 'Contract', 'PaperlessBilling', 'PaymentMethod',
       'MonthlyCharges', 'TotalCharges', 'Churn'],
      dtype='object')
```

```
#Replacing spaces with null values in total charges column
employee['TotalCharges'] =employee["TotalCharges"].replace(" ",np.nan).astype(float)
# string cannot be convert float direclty
```

```
employee.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 20 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   gender            7043 non-null   object 
 1   SeniorCitizen     7043 non-null   object 
 2   Partner           7043 non-null   object 
 3   Dependents        7043 non-null   object 
 4   tenure            7043 non-null   int64  
 5   PhoneService      7043 non-null   object 
 6   MultipleLines     7043 non-null   object 
 7   InternetService   7043 non-null   object 
 8   OnlineSecurity    7043 non-null   object 
 9   OnlineBackup      7043 non-null   object 
 10  DeviceProtection  7043 non-null   object 
 11  TechSupport       7043 non-null   object 
 12  StreamingTV       7043 non-null   object 
 13  StreamingMovies   7043 non-null   object 
 14  Contract          7043 non-null   object 
 15  PaperlessBilling  7043 non-null   object 
 16  PaymentMethod     7043 non-null   object 
 17  MonthlyCharges    7043 non-null   float64
 18  TotalCharges      7032 non-null   float64
 19  Churn             7043 non-null   object 
dtypes: float64(2), int64(1), object(17)
memory usage: 1.1+ MB
```

Checking Missing Value

```
# Do we have NA's in data
employee.isna().sum()  ## is = check & as = convert
```

```
gender               0
SeniorCitizen        0
Partner              0
Dependents           0
tenure               0
PhoneService         0
MultipleLines        0
InternetService      0
OnlineSecurity       0
OnlineBackup         0
DeviceProtection     0
TechSupport          0
StreamingTV          0
StreamingMovies      0
Contract             0
PaperlessBilling     0
PaymentMethod        0
MonthlyCharges       0
TotalCharges        11
Churn                0
dtype: int64
```

```
employee.TotalCharges.fillna(employee.TotalCharges.mean(),inplace=True) # one column at a time bb
```

```
# Do we have NA's in data
employee.isna().sum()
```

```
gender              0
SeniorCitizen       0
Partner             0
Dependents          0
tenure              0
PhoneService        0
MultipleLines       0
InternetService     0
OnlineSecurity      0
OnlineBackup        0
DeviceProtection    0
TechSupport         0
StreamingTV         0
StreamingMovies     0
Contract            0
PaperlessBilling    0
PaymentMethod       0
MonthlyCharges      0
TotalCharges        0
Churn               0
dtype: int64
```

```
employee.describe() # describe works for number by default
```

	tenure	MonthlyCharges	TotalCharges
count	7043.000000	7043.000000	7043.000000
mean	32.371149	64.761692	2283.300441
std	24.559481	30.090047	2265.000258
min	0.000000	18.250000	18.800000
25%	9.000000	35.500000	402.225000
50%	29.000000	70.350000	1400.550000
75%	55.000000	89.850000	3786.600000
max	72.000000	118.750000	8684.800000

```
employee.head()
```

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	I
0	Female	No	Yes	No	1	No	No phone service	
1	Male	No	No	No	34	Yes	No	
2	Male	No	No	No	2	Yes	No	
3	Male	No	No	No	45	No	No phone service	
4	Female	No	No	No	2	Yes	No	

```
employee.OnlineSecurity.value_counts(ascending=False)
```

```
No                     3498
Yes                    2019
```

```
No internet service    1526
Name: OnlineSecurity, dtype: int64
```

```
3498+1526
```

```
5024
```

```
employee.OnlineSecurity=employee.OnlineSecurity.replace({'No internet service' : 'No'})
```

```
employee.OnlineSecurity.value_counts()
```

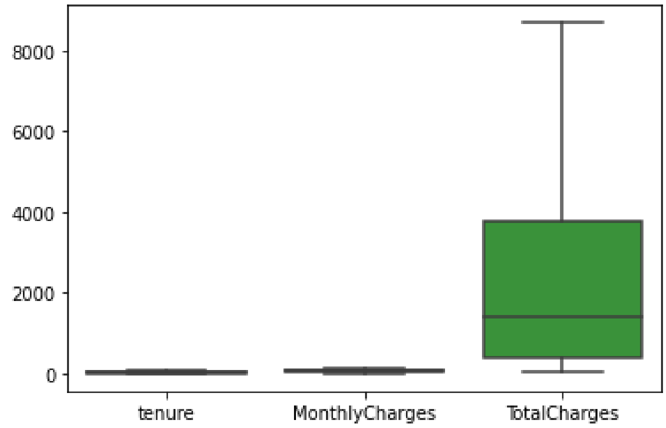
```
No     5024
Yes    2019
Name: OnlineSecurity, dtype: int64
```

```
employee.OnlineBackup=employee.OnlineBackup.replace({'No internet service' : 'No'})
employee.DeviceProtection=employee.DeviceProtection.replace({'No internet service' : 'No'})
employee.TechSupport=employee.TechSupport.replace({'No internet service' : 'No'})
employee.StreamingTV=employee.StreamingTV.replace({'No internet service' : 'No'})
employee.StreamingMovies=employee.StreamingMovies.replace({'No internet service' : 'No'})
employee.MultipleLines=employee.MultipleLines.replace({'No phone service' : 'No'})
```

Outlier

```
import seaborn as sns
sns.boxplot(data=employee)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f15af57a7c0>
```



Churn Rate Analysis

```
employee.Churn.value_counts()
```

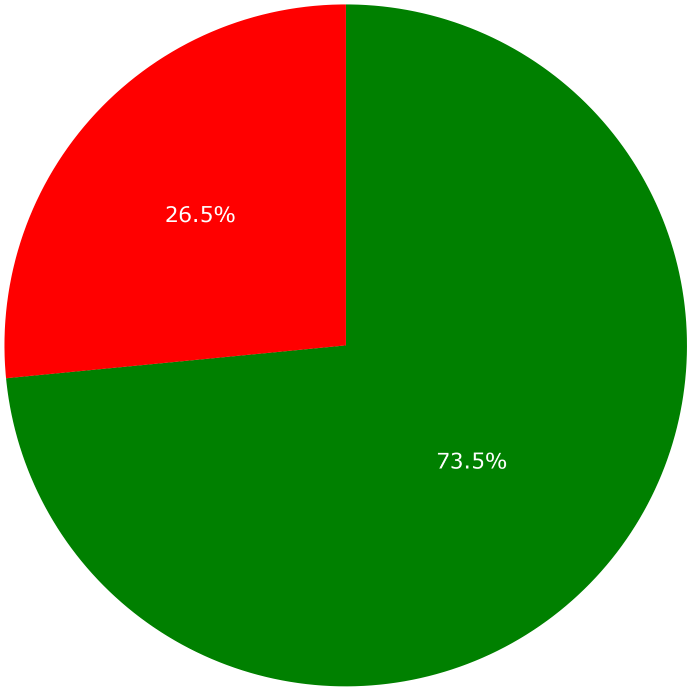
```
No     5174
Yes    1869
Name: Churn, dtype: int64
```

```
(1869/7043)*100
```

```
26.536987079369588
```

```
import plotly.express as px

fig = px.pie(employee,names='Churn',color='Churn',
             color_discrete_map={'Yes':'red',
                                 'No':'green'})
fig.show()
```



Trend Analysis

```
employee.Churn.value_counts()
```

```
No     5174
Yes    1869
Name: Churn, dtype: int64
```

```
(1869/7043)*100
```

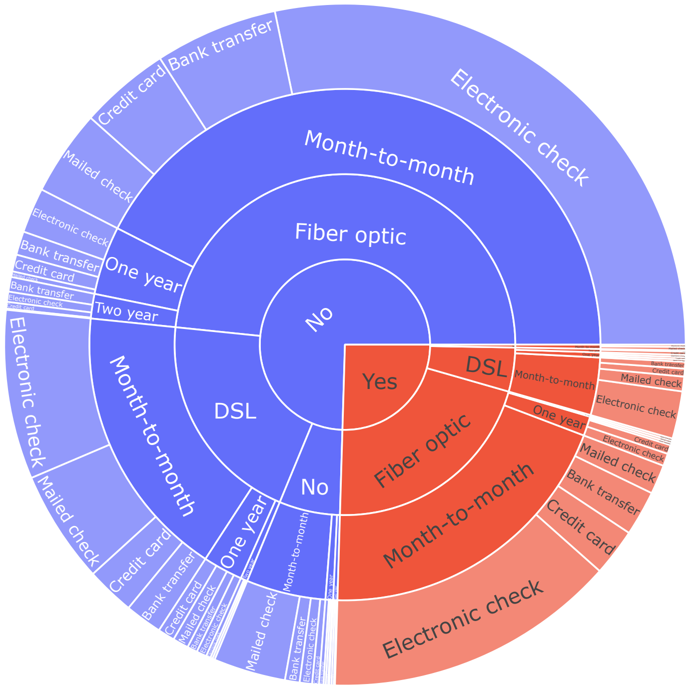
```
26.536987079369588
```

```
Churn_Customer= employee[employee["Churn"] == "Yes"]
Churn_Customer
```

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines
2	Male	No	No	No	2	Yes	No
4	Female	No	No	No	2	Yes	No
5	Female	No	No	No	8	Yes	Yes
8	Female	No	Yes	No	28	Yes	Yes
13	Male	No	No	No	49	Yes	Yes
...	...	...	...	...	...	...	..
7021	Male	No	No	No	12	Yes	No
7026	Female	No	No	No	9	Yes	No
7032	Male	Yes	No	No	1	Yes	Yes
7034	Female	No	No	No	67	Yes	Yes
7041	Male	Yes	Yes	No	4	Yes	Yes

1869 rows × 20 columns

```
fig = px.sunburst(Churn_Customer, path=["SeniorCitizen",'InternetService',
                                         "Contract", "PaymentMethod"])
fig.show()
```



Conclusion :- Customer Trend Analysis

- Customer who leave the service are
- Citizen = Youth , Internet = Fiber Optic & Month-to-Month & Payment = Electronic Check

Taking subset data of Number

```
employee.select_dtypes(include=[np.number]).columns.tolist()
```

```
['tenure', 'MonthlyCharges', 'TotalCharges']
```

```
# #Employee Numeric columns
employee_num = employee[employee.select_dtypes(include=[np.number]).columns.tolist()]
employee_num.head(3)
```

	tenure	MonthlyCharges	TotalCharges
0	1	29.85	29.85
1	34	56.95	1889.50
2	2	53.85	108.15

Taking subset data of Category

```
employee_dummies = employee[employee.select_dtypes(include=['object']).columns.tolist()]
employee_dummies.head(3)
```

	gender	SeniorCitizen	Partner	Dependents	PhoneService	MultipleLines	InternetS
0	Female	No	Yes	No	No	No	
1	Male	No	No	No	Yes	No	
2	Male	No	No	No	Yes	No	

Converting Quality Variable to Number

```
from sklearn.preprocessing import LabelEncoder
employee_dummies=employee_dummies.apply(LabelEncoder().fit_transform)
employee_dummies.head(3)
# label in ascending order
```

	gender	SeniorCitizen	Partner	Dependents	PhoneService	MultipleLines	InternetS
0	0	0	1	0	0	0	
1	1	0	0	0	1	0	
2	1	0	0	0	1	0	

Combine to Dataset

```
employee_combined = pd.concat([employee_num, employee_dummies],axis=1)
```

```
employee_combined.head()
```

	tenure	MonthlyCharges	TotalCharges	gender	SeniorCitizen	Partner	Dependents
0	1	29.85	29.85	0	0	1	0
1	34	56.95	1889.50	1	0	0	0
2	2	53.85	108.15	1	0	0	0
3	45	42.30	1840.75	1	0	0	0
4	2	70.70	151.65	0	0	0	0

Step 3: Data Partition

```
#Dividing data into train and test dataset
from sklearn.model_selection import train_test_split
#from random import seed

#seed(20)
x = employee_combined.drop(['Churn'],axis=1)
y = employee_combined[['Churn']]

# Train test split

X_train, X_test, y_train, y_test =train_test_split(x,y,test_size=0.3,random_state=231)
```

Step 4: Model Building

```
#Import Tree Classifier model
from sklearn import tree

dt = tree.DecisionTreeClassifier()  # by default it use Gini index for split
#Train the model using the training sets
dt.fit(X_train,y_train)  # Model = dt
```

```
DecisionTreeClassifier()
```

Step 5: Plotting the Tree

Ploting Tree

import graphviz from six import StringIO

from sklearn.externals.six import StringIO

from IPython.display import Image

from sklearn.tree import

export_graphviz

import pydotplus

import pydot

```
train=pd.concat([y_train,X_train],axis=1)
train.head()
```

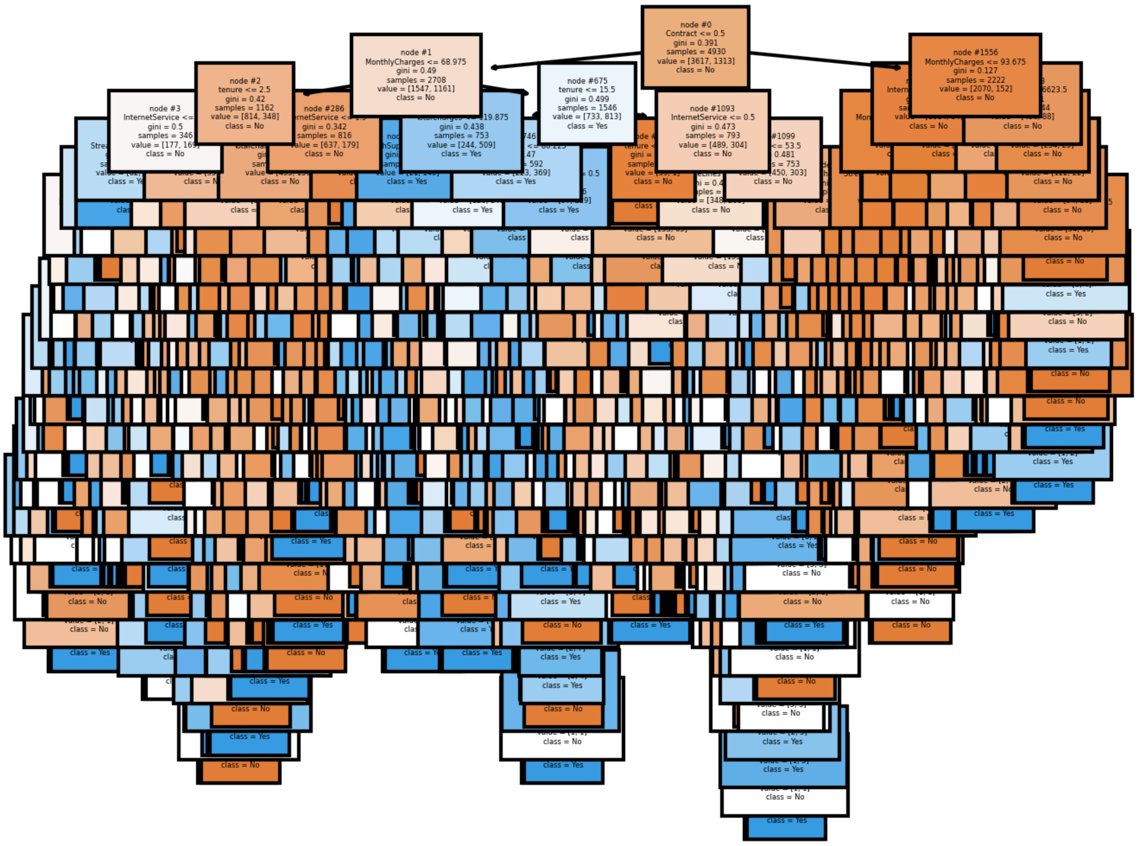
	Churn	tenure	MonthlyCharges	TotalCharges	gender	SeniorCitizen	Partner	De
1583	0	6	48.95	273.25	0	0	1	
6791	1	19	39.65	733.35	1	0	0	
4812	1	9	66.25	620.55	0	0	0	
6282	0	4	19.55	68.80	1	0	1	
2479	0	56	75.85	4261.20	1	0	1	

```
independent_variable = list(train.columns[1:])
independent_variable
```

```
['tenure',
 'MonthlyCharges',
 'TotalCharges',
 'gender',
 'SeniorCitizen',
 'Partner',
 'Dependents',
 'PhoneService',
 'MultipleLines',
 'InternetService',
 'OnlineSecurity',
 'OnlineBackup',
 'DeviceProtection',
 'TechSupport',
 'StreamingTV',
 'StreamingMovies',
 'Contract',
 'PaperlessBilling',
 'PaymentMethod']
```

```
from sklearn import tree
import matplotlib.pyplot as plt

churn=['No', 'Yes']  # array
fig, axes = plt.subplots(nrows = 1,ncols = 1,figsize = (5,4), dpi=300)
tree.plot_tree(dt,  # Model
               feature_names = independent_variable,  # column name
               class_names=churn, # Yes , No
               filled = True, # colour
              node_ids=True, # node number
              fontsize=2); #
#fig.savefig('imagename.png')
```



Step 6 : Predictions on Train Dataset

```
train.head()
```

	Churn	tenure	MonthlyCharges	TotalCharges	gender	SeniorCitizen	Partner	De
1583	0	6	48.95	273.25	0	0	1	
6791	1	19	39.65	733.35	1	0	0	
4812	1	9	66.25	620.55	0	0	0	
6282	0	4	19.55	68.80	1	0	1	
2479	0	56	75.85	4261.20	1	0	1	

```
train['Predicted']=dt.predict(X_train)  # MODEL = dt
train.head()
```

	Churn	tenure	MonthlyCharges	TotalCharges	gender	SeniorCitizen	Partner	De
1583	0	6	48.95	273.25	0	0	1	
6791	1	19	39.65	733.35	1	0	0	
4812	1	9	66.25	620.55	0	0	0	
6282	0	4	19.55	68.80	1	0	1	
2479	0	56	75.85	4261.20	1	0	1	

5 rows × 21 columns

Step 7 : Model Performance Metrics

```
from sklearn.metrics import confusion_matrix
matrix = confusion_matrix(train['Predicted'],train['Churn'])
print(matrix)
```

```
[[3616    7]
 [   1 1306]]
```

Final accuracy of Model Before Pruning

```
Accuracy_Train=((3616+1306)/(4930)*100)
print(Accuracy_Train)   # overfit or High accuracy
```

```
99.83772819472617
```

```
from sklearn.metrics import classification_report
print(classification_report(train['Churn'], train['Predicted']))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      3617
           1       1.00      0.99      1.00      1313

    accuracy                           1.00      4930
   macro avg       1.00      1.00      1.00      4930
weighted avg       1.00      1.00      1.00      4930
```

Model Improvement by Pruning Method (Cut Tree)

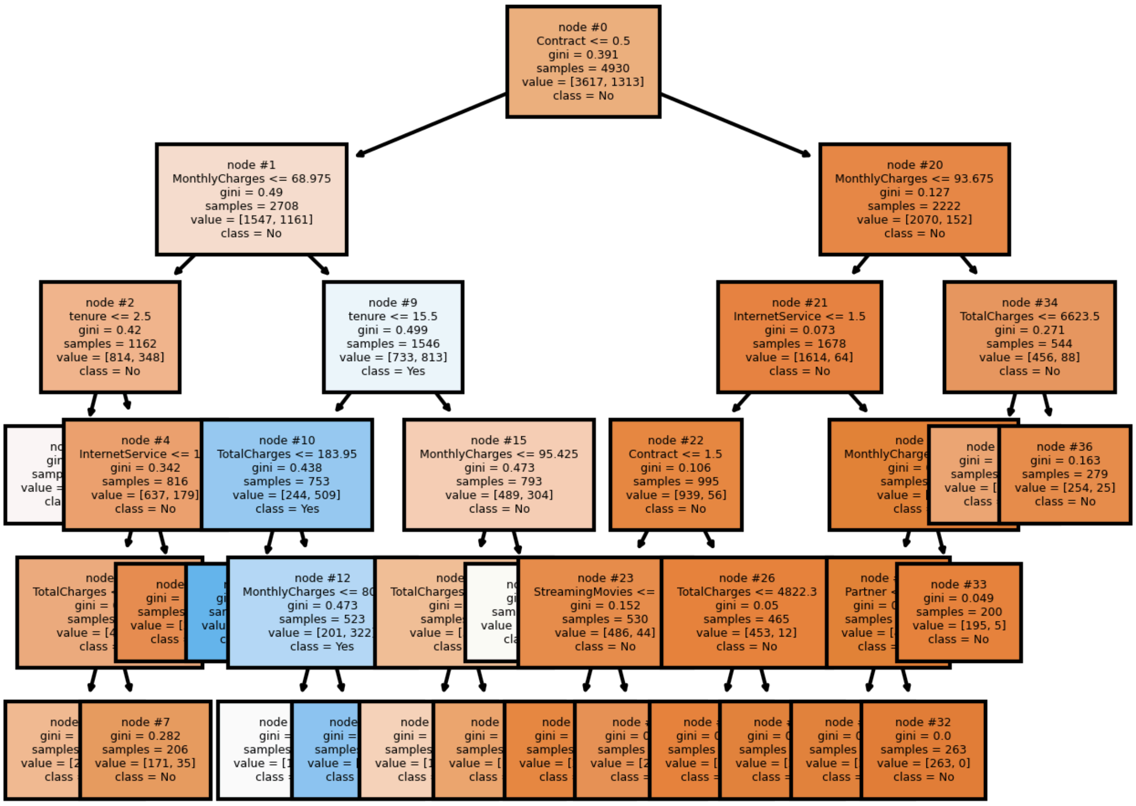
```
#Import Tree Classifier model
from sklearn import tree

dt = tree.DecisionTreeClassifier(criterion='gini',  #splitter
                                 min_samples_leaf=200, ## child
                                 min_samples_split=50, #parent
                                 max_depth=6)  #branches
#Train the model using the training sets
dt.fit(X_train,y_train)
```

```
DecisionTreeClassifier(max_depth=6, min_samples_leaf=200, min_samples_split=50)
```

```
from sklearn import tree
import matplotlib.pyplot as plt

churn=['No', 'Yes']  # array
fig, axes = plt.subplots(nrows = 1,ncols = 1,figsize = (5,4), dpi=300)
tree.plot_tree(dt,  # Model
               feature_names = independent_variable,  # column name
               class_names=churn, # Yes , No
               filled = True, # colour
              node_ids=True, # node number
              fontsize=3); #
#fig.savefig('imagename.png')
```



Strategy & Prediction

- Contract = Month-to-Month & Monthly Charges > 68 & Tenure <= 15.5

```
train['Predicted']=dt.predict(X_train)  # MODEL = dt
train.head()
```

	Churn	tenure	MonthlyCharges	TotalCharges	gender	SeniorCitizen	Partner	De
1583	0	6	48.95	273.25	0	0	1	
6791	1	19	39.65	733.35	1	0	0	
4812	1	9	66.25	620.55	0	0	0	
6282	0	4	19.55	68.80	1	0	1	
2479	0	56	75.85	4261.20	1	0	1	

5 rows × 21 columns

Final accuracy of Model after Pruning

```
from sklearn.metrics import confusion_matrix
matrix = confusion_matrix(train['Predicted'],train['Churn'])
print(matrix)
```

```
[[3373  804]
 [ 244  509]]
```

```
Accuracy_Train=((3373+509)/(4930)*100)
print(Accuracy_Train)
```

```
78.74239350912778
```

```
from sklearn.metrics import classification_report
print(classification_report(train['Churn'], train['Predicted']))
```

```
              precision    recall  f1-score   support

           0       0.81      0.93      0.87      3617
           1       0.68      0.39      0.49      1313

    accuracy                           0.79      4930
   macro avg       0.74      0.66      0.68      4930
weighted avg       0.77      0.79      0.77      4930
```

Step 8 : Predictions on Test Dataset

```
test=pd.concat([X_test,y_test],axis=1)
test.head()
```

```
test['Predicted']=dt.predict(X_test)
test.head()
```

	tenure	MonthlyCharges	TotalCharges	gender	SeniorCitizen	Partner	Dependent
1358	10	70.15	735.50	1	0	0	
5471	29	74.20	1993.25	0	0	0	
2693	72	19.30	1414.80	1	0	0	
1077	41	114.50	4527.45	0	0	0	
6663	1	54.65	54.65	0	0	0	

5 rows × 21 columns

Step 9 : Model Performance Metrics on Test data

```
from sklearn.metrics import confusion_matrix
confusion_matrix = confusion_matrix(test['Predicted'],test['Churn'])
print(confusion_matrix)
```

```
[[1449  342]
 [ 108  214]]
```

```
Accuracy_test=((1449+214)/(2113)*100)
Accuracy_test
```

```
78.70326549929011
```

Sensitivity & Specificity

Train

```
from sklearn.metrics import classification_report
print(classification_report(train['Churn'], train['Predicted']))
```

```
              precision    recall  f1-score   support

           0       0.81      0.93      0.87      3617
           1       0.68      0.39      0.49      1313

    accuracy                           0.79      4930
```