```
In [1]: import pandas as pd
```

```
In [2]: df=pd.read_csv('Admission_Predict.csv',sep=',')
```

```
In [5]: df
```

Out[5]:

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
0	1	337	118	4	4.5	4.5	9.65	1	0.92
1	2	324	107	4	4.0	4.5	8.87	1	0.76
2	3	316	104	3	3.0	3.5	8.00	1	0.72
3	4	322	110	3	3.5	2.5	8.67	1	0.80
4	5	314	103	2	2.0	3.0	8.21	0	0.65
...	...	...	...	...	...	...	...	...	...
395	396	324	110	3	3.5	3.5	9.04	1	0.82
396	397	325	107	3	3.0	3.5	9.11	1	0.84
397	398	330	116	4	5.0	4.5	9.45	1	0.91
398	399	312	103	3	3.5	4.0	8.78	0	0.67
399	400	333	117	4	5.0	4.0	9.66	1	0.95

400 rows × 9 columns

```
In [7]: df.columns
```

```
Out[7]: Index(['Serial No.', 'GRE Score', 'TOEFL Score', 'University Rating', 'SOP',
               'LOR ', 'CGPA', 'Research', 'Chance of Admit '],
         dtype='object')
```

```
In [9]: df.head()
```

Out[9]:

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
0	1	337	118	4	4.5	4.5	9.65	1	0.92
1	2	324	107	4	4.0	4.5	8.87	1	0.76

2	3	316	104	3	3.0	3.5	8.00	1	0.72
3	4	322	110	3	3.5	2.5	8.67	1	0.80
4	5	314	103	2	2.0	3.0	8.21	0	0.65

```
In [11]: df.shape
```

```
Out[11]: (400, 9)
```

```
In [13]: df.columns=df.columns.str.rstrip()
```

```
In [15]: df.columns
```

```
Out[15]: Index(['Serial No.', 'GRE Score', 'TOEFL Score', 'University Rating', 'SOP',
                'LOR', 'CGPA', 'Research', 'Chance of Admit'],
          dtype='object')
```

```
In [17]: df.isnull().sum()
```

```
Out[17]:  Serial No.           0
          GRE Score            0
          TOEFL Score          0
          University Rating    0
          SOP                  0
          LOR                  0
          CGPA                 0
          Research              0
          Chance of Admit       0
          dtype: int64
```

```
In [19]: # replace values in in Chance of Admit column by 0 or 1 based on threshold value
          df.loc[df['Chance of Admit'] >=0.80,'Chance of Admit']=1 df.loc[df['Chance of
          Admit'] < 0.80,'Chance of Admit']=0
```

```
In [21]: df['Chance of Admit']
```

```
Out[21]: 0      1.0 1
          0.0
          2   0.0
          3   1.0
          4   0.0       ... 395    1.0
          396    1.0
          397    1.0
          398    0.0
          399    1.0
          Name: Chance of Admit, Length: 400, dtype: float64
```

```
In [23]: df=df.drop('Serial No.',axis=1)
```

```
In [25]: df
```

Out[25]:

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
0	337	118	4	4.5	4.5	9.65	1	1.0

1	324	107	4	4.0	4.5	8.87	1	0.0
2	316	104	3	3.0	3.5	8.00	1	0.0
3	322	110	3	3.5	2.5	8.67	1	1.0
4	314	103	2	2.0	3.0	8.21	0	0.0
...	...	...	...	...	...	...	...	...
395	324	110	3	3.5	3.5	9.04	1	1.0
396	325	107	3	3.0	3.5	9.11	1	1.0
397	330	116	4	5.0	4.5	9.45	1	1.0
398	312	103	3	3.5	4.0	8.78	0	0.0
399	333	117	4	5.0	4.0	9.66	1	1.0

400 rows × 8 columns

```
In [27]: X = df.iloc[:,0:7].values y=df.iloc[:,7].values
```

```
In [29]: X
```

```
Out[29]: array([[337.  , 118.  ,   4.  , ...,   4.5 ,   9.65,   1.  ],
                 [324.  , 107.  ,   4.  , ...,   4.5 ,   8.87,   1.  ],
                [316.  , 104.  ,   3.  , ...,   3.5 ,   8.  ,   1.  ],
         ...,
                [330.  , 116.  ,   4.  , ...,   4.5 ,   9.45,   1.  ],
                [312.  , 103.  ,   3.  , ...,   4.  ,   8.78,   0.  ],
                [333.  , 117.  ,   4.  , ...,   4.  ,   9.66,   1.  ]])
```

```
In [31]: y
```

```
Out[31]: array([1., 0., 0., 1., 0., 1., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0.,
                  0., 0., 0., 0., 0., 1., 1., 1., 1., 0., 0., 0., 0., 0., 0., 1., 1.,
         1., 1., 0., 0., 0., 0., 0., 0., 0., 1., 1., 1., 1., 1., 1., 0., 0.,        0.,
         0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,        0., 0.,
         1., 1., 1., 1., 0., 0., 0., 0., 0., 0., 0., 1., 1., 1., 1.,        0., 0., 0.,
         0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 1., 0., 0., 0.,        0., 0., 0., 0.,
         1., 1., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,        0., 1., 1., 0., 0.,
         0., 0., 1., 0., 1., 1., 1., 0., 0., 0., 1., 1.,        0., 0., 1., 0., 1., 1.,
         1., 1., 1., 1., 0., 1., 1., 0., 1., 1., 1.,        0., 1., 0., 0., 0., 0., 0.,
         0., 0., 0., 0., 1., 0., 0., 0., 0., 0.,        0., 1., 1., 1., 1., 1., 1., 1.,
         0., 0., 0., 0., 0., 0., 0., 1., 1.,        1., 1., 1., 1., 1., 1., 1., 0., 0.,
         0., 0., 0., 0., 0., 0., 1., 1.,        0., 0., 0., 0., 0., 0., 0., 1., 1., 1.,
         1., 1., 1., 1., 1., 0., 0.,        0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0.,
         0., 0., 1., 1., 1., 1.,        0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 1., 0.,
         0., 0., 0., 1., 1.,        0., 0., 0., 0., 1., 1., 0., 0., 0., 0., 0., 0., 0.,
         1., 0., 0., 0.,        0., 0., 0., 0., 1., 0., 0., 0., 0., 1., 1., 1., 1., 1.,
         1., 1., 1.,        0., 0., 0., 0., 0., 0., 0., 0., 1., 1., 0., 0., 0., 0., 0.,
         0., 0.,        0., 1., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0.,
         0.,        0., 0., 1., 0., 0., 1., 0., 1., 0., 0., 0., 0., 1., 0., 1., 1., 1.,
         0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,        0.,
                0., 1., 1., 1., 1., 0., 0., 1., 0., 0., 0., 0., 0., 1., 1., 0.,
                 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 1., 1., 0., 0., 0., 0., 0.,
                 0., 1., 0., 1., 1., 1., 1., 0., 1.])
```

```
In [33]: from sklearn.model_selection import train_test_split,StratifiedKFold,cross_val_s
          X_train,X_test,y_train,y_test= train_test_split(X,y,test_size=0.25,random_state=
```

```
In [34]: print(X_train.shape,end='  ') print(X_test.shape)
```

```
(300, 7)  (100, 7)
```

```
In [35]: from sklearn.tree import DecisionTreeClassifier


          import matplotlib.pyplot as plt
```

```
In [36]: model = DecisionTreeClassifier(criterion='entropy')
          model.fit(X_train,y_train) y_pred=model.predict(X_test)
```

```
In [39]: from sklearn.metrics import confusion_matrix
```

```
In [40]: matrix=confusion_matrix(y_test,y_pred,labels=[0.0,1.0])
```

```
In [41]: matrix
Out[41]:  array([[64,  7],
                 [ 7, 22]], dtype=int64)
In [42]: from sklearn.metrics import accuracy_score
```

```
In [49]: acc = accuracy_score(y_test,y_pred)
          print('Accuracy of Decision Tree model =  ',acc)
```

```
          from sklearn.metrics import classification_report cr
          =classification_report(y_test,y_pred)
          print('Classification Report ', cr )
```

```
         Accuracy of Decision Tree model =   0.86
In [51]:



         Classification Report                 precision    recall  f1-score   support


                  0.0       0.90      0.90      0.90       71
         1.0       0.76      0.76      0.76        29


             accuracy                           0.86       100
         macro avg       0.83      0.83      0.83       100 weighted
         avg       0.86      0.86      0.86       100
```

In [53]:

```
feature_names=df.columns[0:7]
print(feature_names,end=' ') class_names=[str(x)
for x in model.classes_] class_names
```

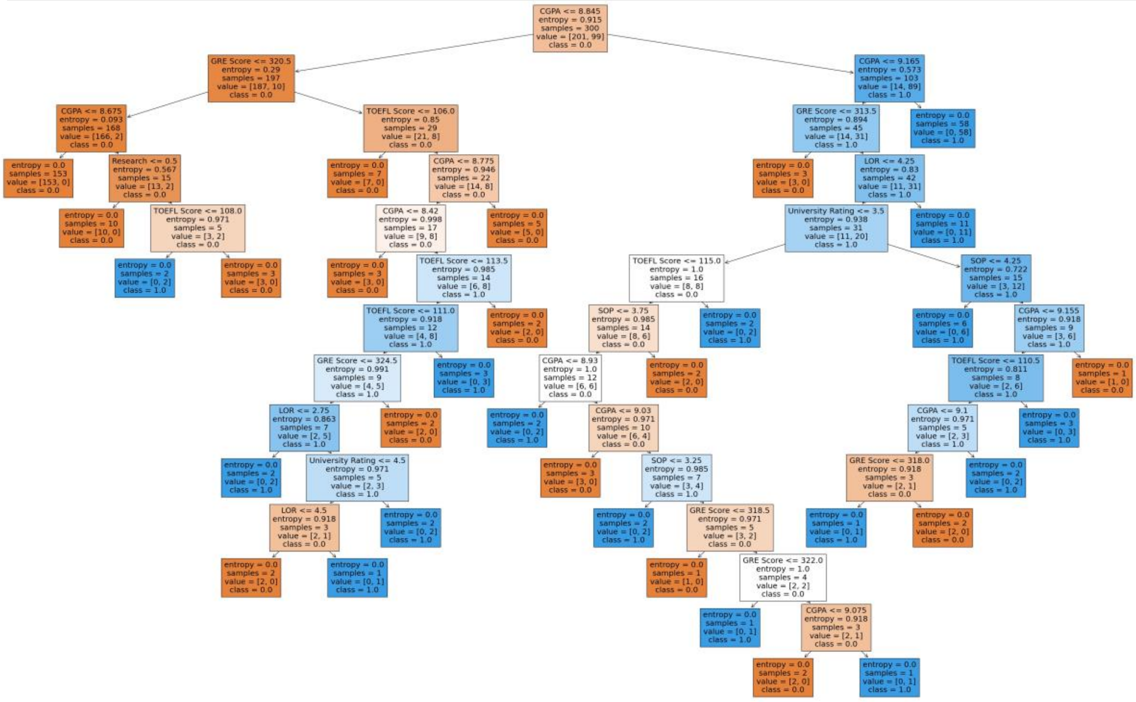
```
Index(['GRE Score', 'TOEFL Score', 'University Rating', 'SOP', 'LOR', 'CGPA',
       'Research'],
dtype='object')
```

Out[53]:

```
['0.0', '1.0']
```

In [55]:

```
from sklearn.tree import plot_tree fig=plt.figure(figsize=(50,30))
plot_tree(model,feature_names=feature_names,class_names=class_names,filled=True
) plt.savefig('tree_visualization.png')
```


CGPA <= 8.845 entropy = 0.915 samples = 300 value = [201, 99] class = 0.0
GRE Score <= 320.5 entropy = 0.29 samples = 197 value = [187, 10] class = 0.0
CGPA <= 9.165 entropy = 0.573 samples = 103 value = [14, 89] class = 1.0
CGPA <= 8.675 entropy = 0.093 samples = 168 value = [166, 2] class = 0.0
TOEFL Score <= 106.0 entropy = 0.85 samples = 29 value = [21, 8] class = 0.0
GRE Score <= 313.5 entropy = 0.894 samples = 45 value = [14, 31] class = 1.0
entropy = 0.0 samples = 58 value = [0, 58] class = 1.0
entropy = 0.0 samples = 153 value = [153, 0] class = 0.0
Research <= 0.5 entropy = 0.567 samples = 15 value = [13, 2] class = 0.0
entropy = 0.0 samples = 7 value = [7, 0] class = 0.0
CGPA <= 8.775 entropy = 0.946 samples = 22 value = [14, 8] class = 0.0
entropy = 0.0 samples = 3 value = [3, 0] class = 0.0
LOR <= 4.25 entropy = 0.83 samples = 42 value = [11, 31] class = 1.0
entropy = 0.0 samples = 10 value = [10, 0] class = 0.0
TOEFL Score <= 108.0 entropy = 0.971 samples = 5 value = [3, 2] class = 0.0
CGPA <= 8.42 entropy = 0.998 samples = 17 value = [9, 8] class = 0.0
entropy = 0.0 samples = 5 value = [5, 0] class = 0.0
University Rating <= 3.5 entropy = 0.938 samples = 31 value = [11, 20] class = 1.0
entropy = 0.0 samples = 11 value = [0, 11] class = 1.0
entropy = 0.0 samples = 2 value = [0, 2] class = 1.0
entropy = 0.0 samples = 3 value = [3, 0] class = 0.0
entropy = 0.0 samples = 3 value = [3, 0] class = 0.0
TOEFL Score <= 113.5 entropy = 0.985 samples = 14 value = [6, 8] class = 1.0
TOEFL Score <= 115.0 entropy = 1.0 samples = 16 value = [8, 8] class = 0.0
SOP <= 4.25 entropy = 0.722 samples = 15 value = [3, 12] class = 1.0
TOEFL Score <= 111.0 entropy = 0.918 samples = 12 value = [4, 8] class = 1.0
entropy = 0.0 samples = 2 value = [2, 0] class = 0.0
SOP <= 3.75 entropy = 0.985 samples = 14 value = [8, 6] class = 0.0
entropy = 0.0 samples = 2 value = [0, 2] class = 1.0
entropy = 0.0 samples = 6 value = [0, 6] class = 1.0
CGPA <= 9.155 entropy = 0.918 samples = 9 value = [3, 6] class = 1.0
GRE Score <= 324.5 entropy = 0.991 samples = 9 value = [4, 5] class = 1.0
entropy = 0.0 samples = 3 value = [0, 3] class = 1.0
CGPA <= 8.93 entropy = 1.0 samples = 12 value = [6, 6] class = 0.0
entropy = 0.0 samples = 2 value = [2, 0] class = 0.0
TOEFL Score <= 110.5 entropy = 0.811 samples = 8 value = [2, 6] class = 1.0
entropy = 0.0 samples = 1 value = [1, 0] class = 0.0
LOR <= 2.75 entropy = 0.863 samples = 7 value = [2, 5] class = 1.0
entropy = 0.0 samples = 2 value = [2, 0] class = 0.0
entropy = 0.0 samples = 2 value = [0, 2] class = 1.0
CGPA <= 9.03 entropy = 0.971 samples = 10 value = [6, 4] class = 0.0
CGPA <= 9.1 entropy = 0.971 samples = 5 value = [2, 3] class = 1.0
entropy = 0.0 samples = 3 value = [0, 3] class = 1.0
entropy = 0.0 samples = 2 value = [0, 2] class = 1.0
University Rating <= 4.5 entropy = 0.971 samples = 5 value = [2, 3] class = 1.0
entropy = 0.0 samples = 3 value = [3, 0] class = 0.0
SOP <= 3.25 entropy = 0.985 samples = 7 value = [3, 4] class = 1.0
GRE Score <= 318.0 entropy = 0.918 samples = 3 value = [2, 1] class = 0.0
entropy = 0.0 samples = 2 value = [0, 2] class = 1.0
LOR <= 4.5 entropy = 0.918 samples = 3 value = [2, 1] class = 0.0
entropy = 0.0 samples = 2 value = [0, 2] class = 1.0
entropy = 0.0 samples = 2 value = [0, 2] class = 1.0
GRE Score <= 318.5 entropy = 0.971 samples = 5 value = [3, 2] class = 0.0
entropy = 0.0 samples = 1 value = [0, 1] class = 1.0
entropy = 0.0 samples = 2 value = [2, 0] class = 0.0
entropy = 0.0 samples = 2 value = [2, 0] class = 0.0
entropy = 0.0 samples = 1 value = [0, 1] class = 1.0
entropy = 0.0 samples = 1 value = [1, 0] class = 0.0
GRE Score <= 322.0 entropy = 1.0 samples = 4 value = [2, 2] class = 0.0
entropy = 0.0 samples = 1 value = [0, 1] class = 1.0
CGPA <= 9.075 entropy = 0.918 samples = 3 value = [2, 1] class = 0.0
entropy = 0.0 samples = 2 value = [2, 0] class = 0.0
entropy = 0.0 samples = 1 value = [0, 1] class = 1.0

In [ ]:

In [57]:

```
pip install graphviz
```

```
Collecting graphviz
  Downloading graphviz-0.20.3-py3-none-any.whl.metadata (12 kB)
Downloading graphviz-0.20.3-py3-none-any.whl (47 kB)
   ---------------------------------------- 0.0/47.1 kB ? eta -:--:--
   -------- ------------------------------- 10.2/47.1 kB ? eta -:--:-   -----------
----------------------------- 47.1/47.1 kB 594.9 kB/s eta 0:00:00
Installing collected packages: graphviz
Successfully installed graphviz-0.20.3 Note: you may need to
restart the kernel to use updated packages.
```

In [58]:

```
import graphviz from sklearn import tree dot_data =
tree.export_graphviz(model,out_file=None, feature_names=feature_name
s graph=graphviz.Source(dot_data,format="png")
```

```
In [59]: sf = StratifiedKFold(n_splits=5,shuffle=True,random_state=0)
```

```
In [60]: depth=[1,2,3,4,5,6,7,8,9,10]

         for d in depth :
             score = cross_val_score(tree.DecisionTreeClassifier(criterion='entropy',max
        _print("Average score for depth {} is {}  :".format(d,score.mean()))
```

```
Average score for depth 1 is 0.9199999999999999  :
Average score for depth 2 is 0.9199999999999999  :
Average score for depth 3 is 0.9233333333333332  :
Average score for depth 4 is 0.9033333333333333  :
Average score for depth 5 is 0.8833333333333334  :
Average score for depth 6 is 0.9  :
Average score for depth 7 is 0.89  :
Average score for depth 8 is 0.8866666666666667  :
Average score for depth 9 is 0.9  :
Average score for depth 10 is 0.9033333333333333  :
```

```
In [61]: score.mean()

Out[61]: 0.9033333333333333
```

```
In [62]: maxdepth=[]
         gini_acc=[]
         entropy_acc=[]

         for i in range(1,11):
             dtree=DecisionTreeClassifier(criteria='gini',max_depth=i)
         dtree.fit(X_train,y_train)     pred=dtree.predict(y_test,pred)
         gini_acc.append(accuracy_score(y_test,y_pred))
         maxdepth.append(i)
```

```
---------------------------------------------------------------------------TypeError
                                          Traceback (most recent call last)
Cell In[62], line 6
      3 entropy_acc=[]
      5 for i in range(1,11):
----> 6     dtree=DecisionTreeClassifier(criteria='gini',max_depth=i)
7       dtree.fit(X_train,y_train)
8       pred=dtree.predict(y_test,pred)

TypeError: DecisionTreeClassifier.__init__() got an unexpected keyword argument
'criteria'
```

```
In [ ]:
```