```
import pandas as pd
```

```
df=pd.read_csv("Admission_Predict.csv")
```

```
df
```

	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

Next steps: Generate code with df View recommended plots New interactive sheet

```
df.isnull().sum()
```

	0
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

```
df.columns
```

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

```
df.dtypes
```

```
                         0
Serial No.           int64
GRE Score            int64
TOEFL Score          int64
University Rating    int64
SOP                float64
LOR                float64
CGPA               float64
Research             int64
Chance of Admit    float64

dtype: object
```

```
df['Chance of Admit '] = df['Chance of Admit '].apply(lambda x:1 if x>=0.75 else 0)
```

```
df
```

```
     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                1
1             2        324          107                  4  4.0  4.5  8.87         1                1
2             3        316          104                  3  3.0  3.5  8.00         1                0
3             4        322          110                  3  3.5  2.5  8.67         1                1
4             5        314          103                  2  2.0  3.0  8.21         0                0
...         ...        ...          ...                ...  ...  ...   ...       ...              ...
395         396        324          110                  3  3.5  3.5  9.04         1                1
396         397        325          107                  3  3.0  3.5  9.11         1                1
397         398        330          116                  4  5.0  4.5  9.45         1                1
398         399        312          103                  3  3.5  4.0  8.78         0                0
399         400        333          117                  4  5.0  4.0  9.66         1                1

400 rows × 9 columns
```

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
features = ['GRE Score','TOEFL Score','University Rating','SOP','LOR ','CGPA','Research']
```

```
x=df[features]
y=df['Chance of Admit ']
```

```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report,confusion_matrix
```

```
import matplotlib.pyplot as plt
import seaborn as sns
```

```
df=df.drop(columns=['Serial No.'])
```

```
df
```

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

400 rows × 8 columns

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=42)
```

```
clf = DecisionTreeClassifier()
```

```
clf.fit(X_train, y_train)
```

```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

```
y_pred=clf.predict(X_test)
```

```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.94      0.78      0.85        74
           1       0.72      0.91      0.81        46

    accuracy                           0.83       120
   macro avg       0.83      0.85      0.83       120
weighted avg       0.85      0.83      0.84       120
```

```
cm = confusion_matrix(y_test, y_pred)
```

```
import matplotlib.pyplot as plt
import seaborn as sns
```

```
plt.figure(figsize=(20, 20))
```

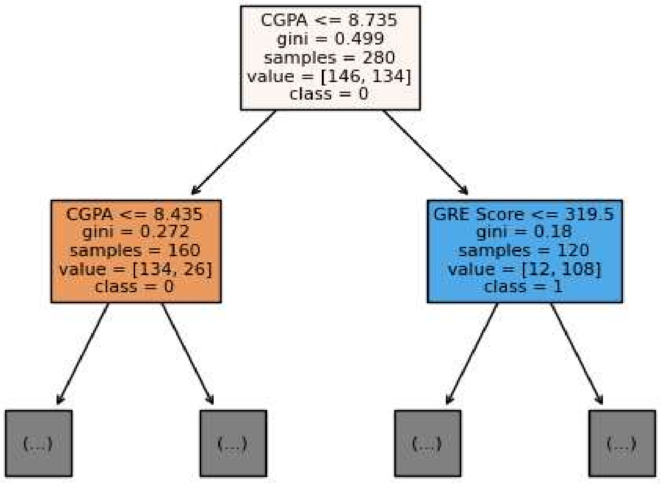
```
<Figure size 2000x2000 with 0 Axes>
<Figure size 2000x2000 with 0 Axes>
```

```
from sklearn.tree import plot_tree
plt.figure(figsize=(20,20))
```

```
<Figure size 2000x2000 with 0 Axes>
<Figure size 2000x2000 with 0 Axes>
```

```
import numpy as np
plot_tree(clf, filled=True, feature_names=x.columns,class_names=np.unique(y).astype(str),max_depth=1)
```

```
[Text(0.5, 0.8333333333333334, 'CGPA <= 8.735\ngini = 0.499\nsamples = 280\nvalue = [146, 134]\nclass = 0'),
 Text(0.25, 0.5, 'CGPA <= 8.435\ngini = 0.272\nsamples = 160\nvalue = [134, 26]\nclass = 0'),
 Text(0.125, 0.16666666666666666, '\n  (...)  \n'),
 Text(0.375, 0.16666666666666666, '\n  (...)  \n'),
 Text(0.75, 0.5, 'GRE Score <= 319.5\ngini = 0.18\nsamples = 120\nvalue = [12, 108]\nclass = 1'),
 Text(0.625, 0.16666666666666666, '\n  (...)  \n'),
 Text(0.875, 0.16666666666666666, '\n  (...)  \n')]
```



```
clf = DecisionTreeClassifier(criterion='entropy')
```

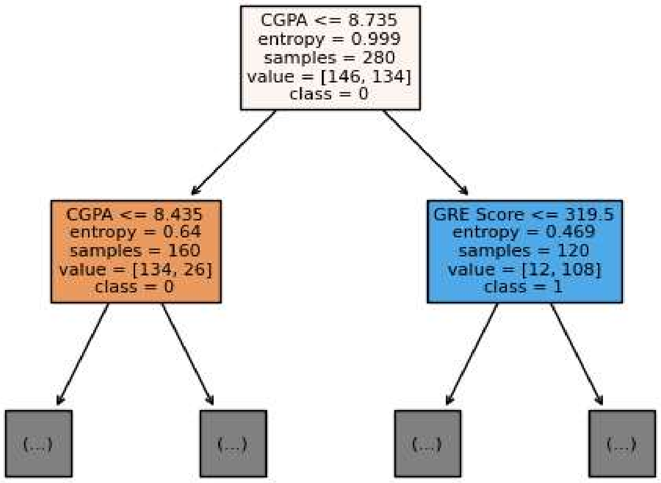
```
clf.fit(X_train, y_train)
```

```
▾ DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy')
```

```
y_pred=clf.predict(X_test)
```

```
import numpy as np
plot_tree(clf, filled=True, feature_names=x.columns,class_names=np.unique(y).astype(str),max_depth=1)
```

```
[Text(0.5, 0.8333333333333334, 'CGPA <= 8.735\nentropy = 0.999\nsamples = 280\nvalue = [146, 134]\nclass = 0'),
 Text(0.25, 0.5, 'CGPA <= 8.435\nentropy = 0.64\nsamples = 160\nvalue = [134, 26]\nclass = 0'),
 Text(0.125, 0.16666666666666666, '\n  (...)  \n'),
 Text(0.375, 0.16666666666666666, '\n  (...)  \n'),
 Text(0.75, 0.5, 'GRE Score <= 319.5\nentropy = 0.469\nsamples = 120\nvalue = [12, 108]\nclass = 1'),
 Text(0.625, 0.16666666666666666, '\n  (...)  \n'),
 Text(0.875, 0.16666666666666666, '\n  (...)  \n')]
```



```
import matplotlib.patches as patches
```

Start coding or generate with AI.