# Untitled

February 10, 2026

[1]:
```
import pandas as pd
import numpy as np

from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.tree import plot_tree
from sklearn.metrics import␣
 ↪confusion_matrix,classification_report,accuracy_score
```

[2]:
```
df = pd.read_csv(r"C:/Users/USER/Downloads/college_student_placement_dataset.
 ↪csv")
df.head()
```

[2]:
```
  College_ID   IQ  Prev_Sem_Result  CGPA  Academic_Performance  \
0    CLG0030  107             6.61  6.28                     8
1    CLG0061   97             5.52  5.37                     8
2    CLG0036  109             5.36  5.83                     9
3    CLG0055  122             5.47  5.75                     6
4    CLG0004   96             7.91  7.69                     7

  Internship_Experience  Extra_Curricular_Score  Communication_Skills  \
0                    No                       8                     8
1                    No                       7                     8
2                    No                       3                     1
3                   Yes                       1                     6
4                    No                       8                    10

   Projects_Completed Placement
0                   4        No
1                   0        No
2                   1        No
3                   1        No
4                   2        No
```

[3]:
```
df.shape
```

[3]:
```
(10000, 10)
```

[4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 10 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   College_ID              10000 non-null  object
 1   IQ                      10000 non-null  int64
 2   Prev_Sem_Result         10000 non-null  float64
 3   CGPA                    10000 non-null  float64
 4   Academic_Performance    10000 non-null  int64
 5   Internship_Experience   10000 non-null  object
 6   Extra_Curricular_Score  10000 non-null  int64
 7   Communication_Skills    10000 non-null  int64
 8   Projects_Completed      10000 non-null  int64
 9   Placement               10000 non-null  object
dtypes: float64(2), int64(5), object(3)
memory usage: 781.4+ KB
```

[5]:
```
df.describe()
```

[5]:
```
                 IQ  Prev_Sem_Result          CGPA  Academic_Performance  \
count  10000.000000     10000.000000  10000.000000          10000.000000
mean      99.471800         7.535673      7.532379              5.546400
std       15.053101         1.447519      1.470141              2.873477
min       41.000000         5.000000      4.540000              1.000000
25%       89.000000         6.290000      6.290000              3.000000
50%       99.000000         7.560000      7.550000              6.000000
75%      110.000000         8.790000      8.770000              8.000000
max      158.000000        10.000000     10.460000             10.000000

       Extra_Curricular_Score  Communication_Skills  Projects_Completed
count            10000.000000          10000.000000        10000.000000
mean                 4.970900              5.561800            2.513400
std                  3.160103              2.900866            1.715959
min                  0.000000              1.000000            0.000000
25%                  2.000000              3.000000            1.000000
50%                  5.000000              6.000000            3.000000
75%                  8.000000              8.000000            4.000000
max                 10.000000             10.000000            5.000000
```

[6]:
```
X = df.drop(columns=["College_ID","Placement"])
y = df["Placement"]
```

[7]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.
↪2,random_state=42)
```

[8]:
```
categorical_cols = ['Internship_Experience']
encoders = {}

for col in categorical_cols:
    enc = LabelEncoder()
    X_train[col] = enc.fit_transform(X_train[col])
    X_test[col] = enc.transform(X_test[col])
    encoders[col] = enc
```

[9]:
```
y_encoder = LabelEncoder()

y_train = y_encoder.fit_transform(y_train)
y_test = y_encoder.transform(y_test)
```

[26]:
```
model = DecisionTreeClassifier()
```

[27]:
```
model.fit(X_train,y_train)
```

[27]:
```
DecisionTreeClassifier()
```

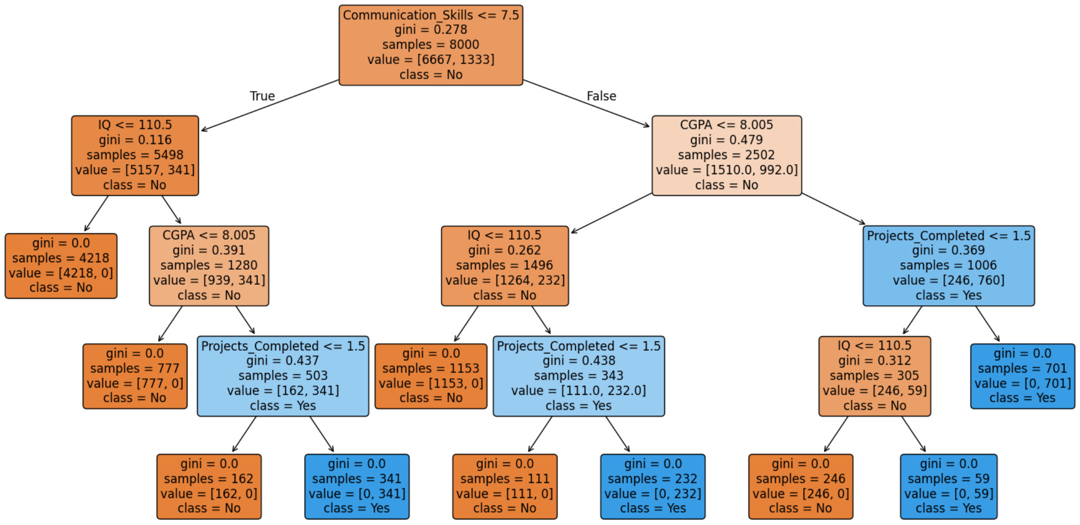
[28]:
```
y_pred = model.predict(X_test)
```

[29]:
```
print("Accuracy:", accuracy_score(y_test, y_pred))
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test, y_pred))
```

```
Accuracy: 1.0
[[1674    0]
 [   0  326]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1674
           1       1.00      1.00      1.00       326

    accuracy                           1.00      2000
   macro avg       1.00      1.00      1.00      2000
weighted avg       1.00      1.00      1.00      2000
```

[31]:
```
import matplotlib.pyplot as plt

plt.figure(figsize=(20,10))
plot_tree(model,
          feature_names=X_train.columns,
          class_names=y_encoder.classes_,
          filled=True,
          rounded=True,
          fontsize=12)
```

```
plt.show()
```



```
[10]: model = RandomForestClassifier(criterion='gini',
                                      max_depth=5,
                                      min_samples_leaf=10,
                                      random_state=42
      )
```

```
<IPython.core.display.Javascript object>
```

```
[11]: model.fit(X_train,y_train)
```

```
[11]: RandomForestClassifier(max_depth=5, min_samples_leaf=10, random_state=42)
```

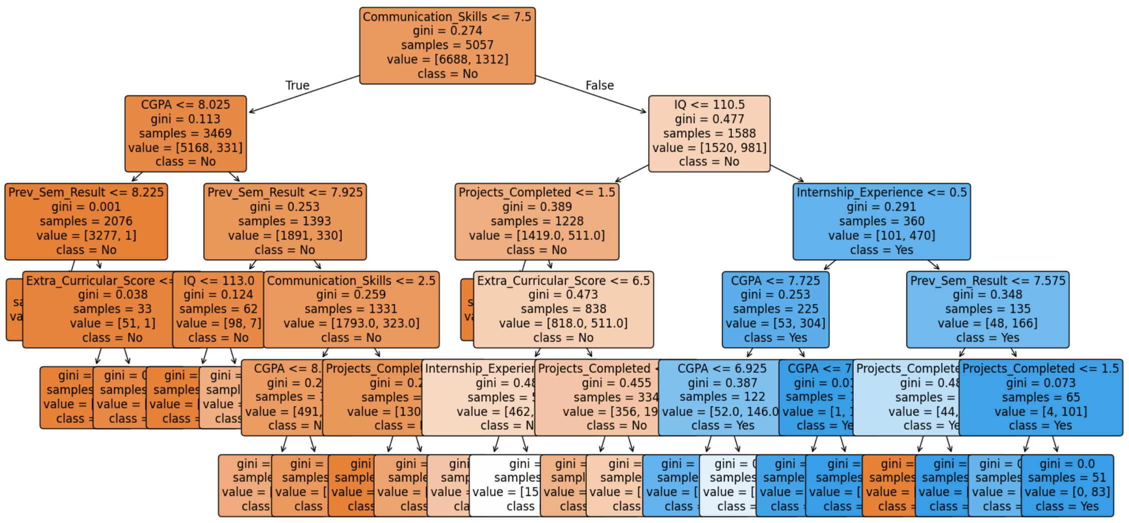
```
[12]: y_pred = model.predict(X_test)
```

```
[13]: print("Accuracy:", accuracy_score(y_test, y_pred))
      print(confusion_matrix(y_test,y_pred))
      print(classification_report(y_test, y_pred))
```

```
Accuracy: 0.999
[[1674    0]
 [   2  324]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1674
           1       1.00      0.99      1.00       326

    accuracy                           1.00      2000
```

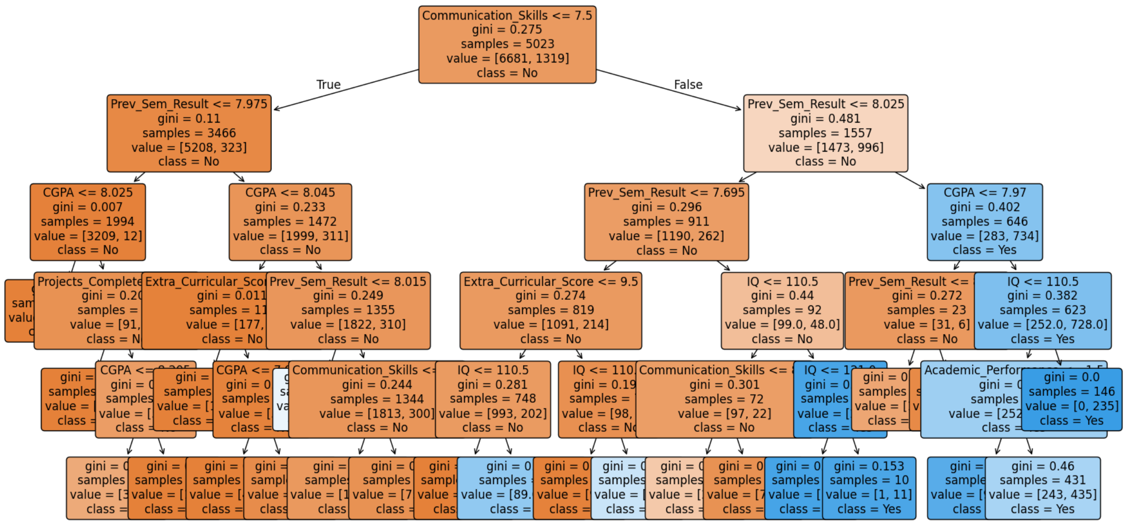
```
   macro avg       1.00      1.00      1.00      2000
weighted avg       1.00      1.00      1.00      2000
```

[15]:
```python
import matplotlib.pyplot as plt
tree = model.estimators_[0]

plt.figure(figsize=(20, 10))
plot_tree(
    tree,
    feature_names=X_train.columns,
    class_names=y_encoder.classes_,
    filled=True,
    rounded=True,
    fontsize=12
)
plt.show()
```



[16]:
```python
tree = model.estimators_[4]

plt.figure(figsize=(20, 10))
plot_tree(
    tree,
    feature_names=X_train.columns,
    class_names=y_encoder.classes_,
    filled=True,
    rounded=True,
    fontsize=12
)
```

```
plt.show()
```


Communication_Skills <= 7.5
gini = 0.275
samples = 5023
value = [6681, 1319]
class = No
True
False
Prev_Sem_Result <= 7.975
gini = 0.11
samples = 3466
value = [5208, 323]
class = No
Prev_Sem_Result <= 8.025
gini = 0.481
samples = 1557
value = [1473, 996]
class = No
CGPA <= 8.025
gini = 0.007
samples = 1994
value = [3209, 12]
class = No
CGPA <= 8.045
gini = 0.233
samples = 1472
value = [1999, 311]
class = No
Prev_Sem_Result <= 7.695
gini = 0.296
samples = 911
value = [1190, 262]
class = No
CGPA <= 7.97
gini = 0.402
samples = 646
value = [283, 734]
class = Yes
Prev_Sem_Result <= 8.015
gini = 0.249
samples = 1355
value = [1822, 310]
class = No
Extra_Curricular_Score <= 9.5
gini = 0.274
samples = 819
value = [1091, 214]
class = No
IQ <= 110.5
gini = 0.44
samples = 92
value = [99.0, 48.0]
class = No
Prev_Sem_Result <=
gini = 0.272
samples = 23
value = [31, 6]
class = No
IQ <= 110.5
gini = 0.382
samples = 623
value = [252.0, 728.0]
class = Yes
Communication_Skills <=
gini = 0.244
samples = 1344
value = [1813, 300]
class = No
IQ <= 110.5
gini = 0.281
samples = 748
value = [993, 202]
class = No
Communication_Skills <=
gini = 0.301
samples = 72
value = [97, 22]
class = No
gini = 0.0
samples = 146
value = [0, 235]
class = Yes
gini = 0.153
samples = 10
value = [1, 11]
class = Yes
gini = 0.46
samples = 431
value = [243, 435]
class = Yes

```
[ ]:
```