```
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

```
from sklearn.cluster import KMeans, AgglomerativeClustering, DBSCANS

from mlxtend.frequent_patterns import apriori, association_rules

from google.colab import files
uploaded = files.upload()
```

Choose Files No file chosen   Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
df = pd.read_csv("placement_data.csv")
df.head()
```

	student_id	gender	degree	stream	internship_score	cgpa	placed
0	1	Male	B.Tech	IT	80	8.5	Yes
1	2	Female	B.Tech	CS	70	8.0	Yes
2	3	Male	B.E	EXTC	50	6.5	No
3	4	Female	B.E	IT	90	9.1	Yes
4	5	Male	B.Tech	MECH	40	6.0	No

```
# Check missing values
print(df.isnull().sum())
```

```
student_id         0
gender             0
degree             0
stream             0
internship_score   0
cgpa               0
placed             0
dtype: int64
```

```
df.fillna(df.mean(numeric_only=True), inplace=True)
```

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
for col in df.select_dtypes(include=['object']).columns:
    df[col] = le.fit_transform(df[col])
```

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaled_data = scaler.fit_transform(df)
df_scaled = pd.DataFrame(scaled_data, columns=df.columns)
df_scaled.head()
```

	student_id	gender	degree	stream	internship_score	cgpa	placed
0	-1.566699	1.0	0.816497	0.538816	0.904534	0.831670	0.816497
1	-1.218544	-1.0	0.816497	-1.257237	0.301511	0.359130	0.816497
2	-0.870388	1.0	-1.224745	-0.359211	-0.904534	-1.058490	-1.224745
3	-0.522233	-1.0	-1.224745	0.538816	1.507557	1.398718	0.816497
4	-0.174078	1.0	0.816497	1.436842	-1.507557	-1.531030	-1.224745

```
print(df.describe())   # summary (mean, min, max, etc.)
```

```
       student_id     gender    degree     stream  internship_score  \
count   10.00000  10.000000  10.000000  10.000000         10.000000
mean     5.50000   0.500000   0.600000   1.400000         65.000000
std      3.02765   0.527046   0.516398   1.173788         17.480147
min      1.00000   0.000000   0.000000   0.000000         40.000000
25%      3.25000   0.000000   0.000000   0.250000         51.250000
50%      5.50000   0.500000   1.000000   1.500000         65.000000
75%      7.75000   1.000000   1.000000   2.000000         78.750000
max     10.00000   1.000000   1.000000   3.000000         90.000000

           cgpa     placed
count  10.000000  10.000000
mean    7.620000   0.600000
std     1.115347   0.516398
min     6.000000   0.000000
25%     6.675000   0.000000
50%     7.750000   1.000000
75%     8.475000   1.000000
max     9.100000   1.000000
```
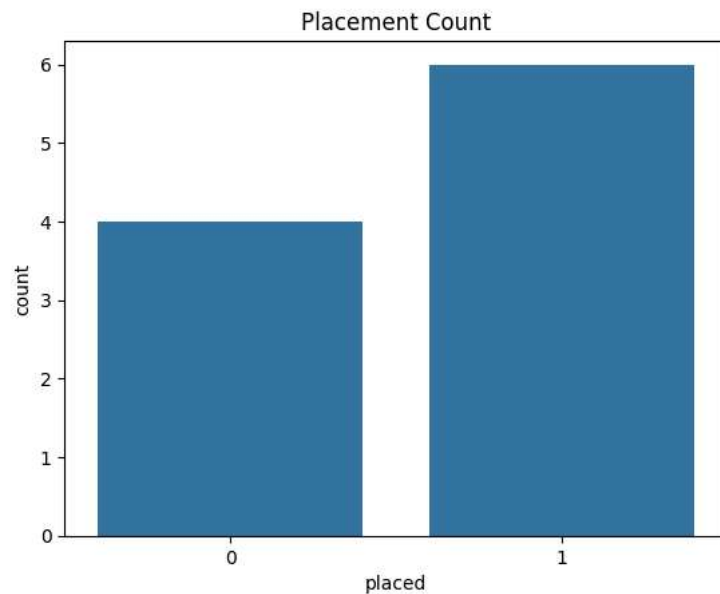
```
plt.figure(figsize=(10,6))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.title("Correlation Heatmap")
plt.show()
```


Correlation Heatmap
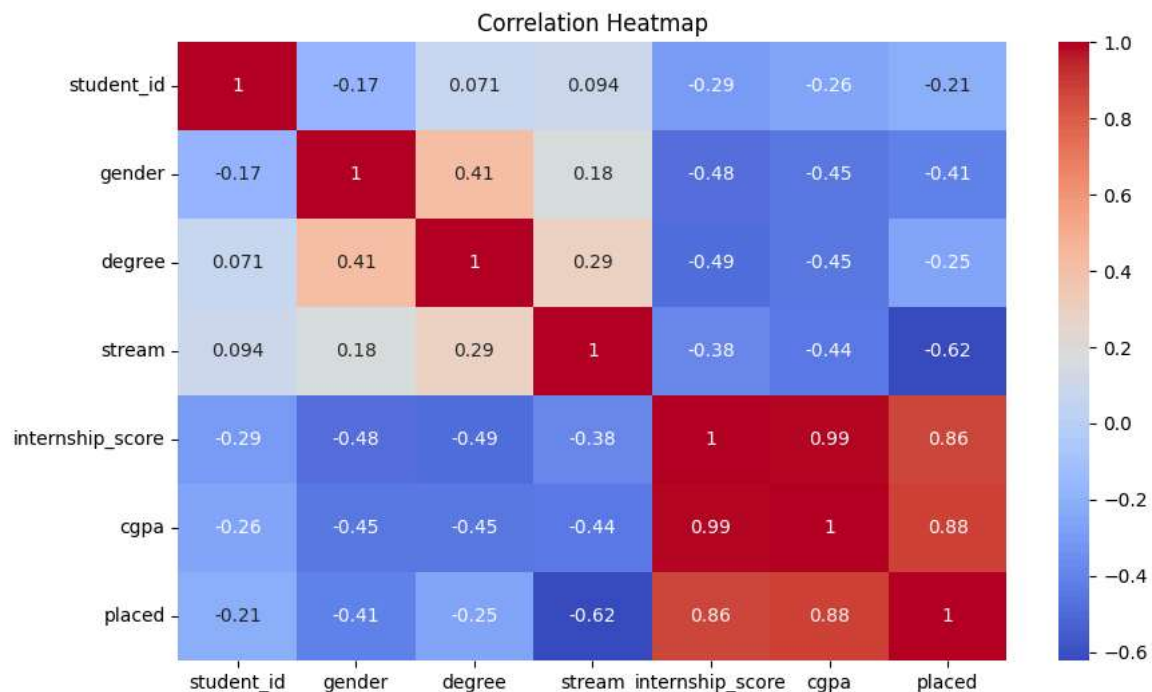
```
sns.countplot(x='placed', data=df)
plt.title("Placement Count")
plt.show()
```

Placement Count


```
X = df.drop('placed', axis=1)
y = df['placed']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
dt = DecisionTreeClassifier()
dt.fit(X_train, y_train)
y_pred_dt = dt.predict(X_test)
```

```
cm = confusion_matrix(y_test, y_pred_dt)
plt.figure(figsize=(6,4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix Heatmap')
plt.show()
```
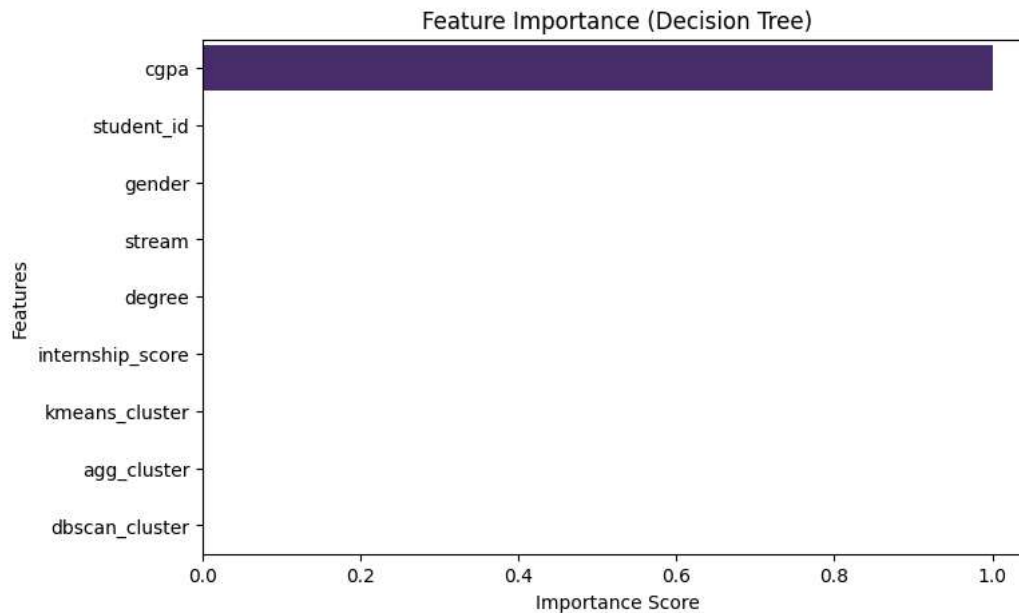
Confusion Matrix Heatmap
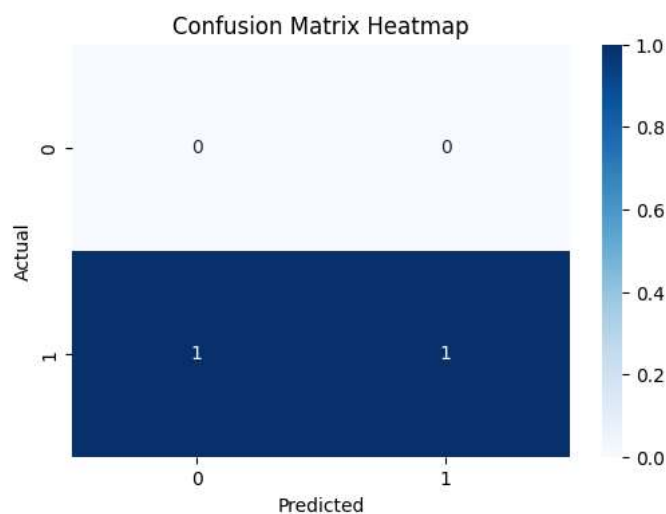

```
importances = dt.feature_importances_
feature_names = X.columns
feat_imp = pd.Series(importances, index=feature_names).sort_values(ascending=False)

plt.figure(figsize=(8,5))
sns.barplot(x=feat_imp.values, y=feat_imp.index, palette='viridis')
plt.title('Feature Importance (Decision Tree)')
plt.xlabel('Importance Score')
plt.ylabel('Features')
plt.show()
```

```
pv.show()
```

```
/tmp/ipython-input-916379644.py:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `le
  sns.barplot(x=feat_imp.values, y=feat_imp.index, palette='viridis')
```

Feature Importance (Decision Tree)


```
nb = GaussianNB()
nb.fit(X_train, y_train)
y_pred_nb = nb.predict(X_test)

print("Naive Bayes Accuracy:", accuracy_score(y_test, y_pred_nb))
print(confusion_matrix(y_test, y_pred_nb))
print(classification_report(y_test, y_pred_nb))
```

```
Naive Bayes Accuracy: 1.0
[[2]]
              precision    recall  f1-score   support

           1       1.00      1.00      1.00         2

    accuracy                           1.00         2
   macro avg       1.00      1.00      1.00         2
weighted avg       1.00      1.00      1.00         2

/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_classification.py:407: UserWarning: A single label was found in 'y_true' and
  warnings.warn(
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.cluster import KMeans, AgglomerativeClustering, DBSCAN
```

```
from google.colab import files
uploaded = files.upload()
```

Choose Files No file chosen   Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to
enable.

```
df = pd.read_csv("placement_data.csv")
print("Dataset loaded successfully!")
df.head()
```

```
Dataset loaded successfully!
```

	student_id	gender	degree	stream	internship_score	cgpa	placed
0	1	Male	B.Tech	IT	80	8.5	Yes
1	2	Female	B.Tech	CS	70	8.0	Yes
2	3	Male	B.E	EXTC	50	6.5	No
3	4	Female	B.E	IT	90	9.1	Yes
4	5	Male	B.Tech	MECH	40	6.0	No

```
display(df.head())
```

	student_id	gender	degree	stream	internship_score	cgpa	placed
0	1	Male	B.Tech	IT	80	8.5	Yes
1	2	Female	B.Tech	CS	70	8.0	Yes
2	3	Male	B.E	EXTC	50	6.5	No
3	4	Female	B.E	IT	90	9.1	Yes
4	5	Male	B.Tech	MECH	40	6.0	No

```
df.fillna(df.mean(numeric_only=True), inplace=True)
```

```
display(df.head())
```

```
le = LabelEncoder()
for col in df.select_dtypes(include=['object']).columns:
    df[col] = le.fit_transform(df[col])
```

```
scaler = StandardScaler()
scaled_data = scaler.fit_transform(df)
df_scaled = pd.DataFrame(scaled_data, columns=df.columns)
```

```
print(df.columns)
```

```
Index(['student_id', 'gender', 'degree', 'stream', 'internship_score', 'cgpa',
       'placed', 'kmeans_cluster', 'agg_cluster', 'dbscan_cluster'],
      dtype='object')
```
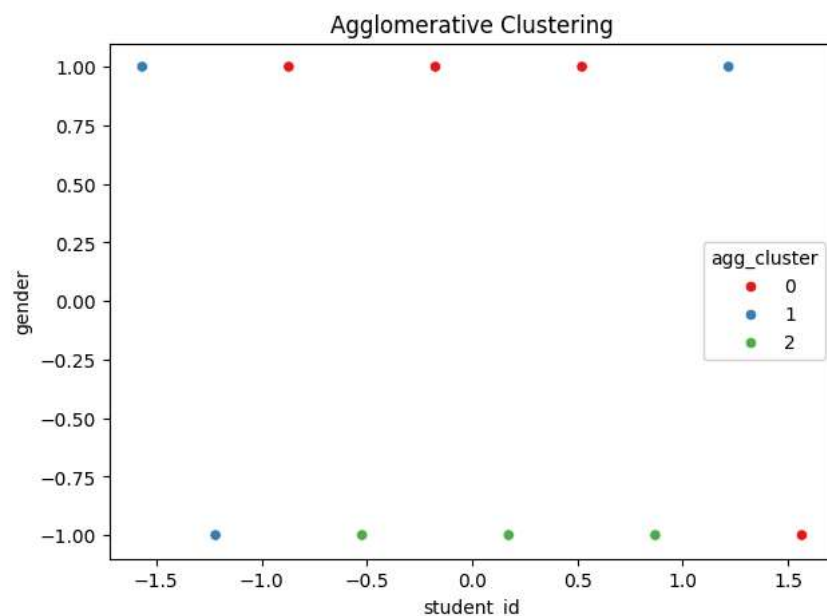
```
kmeans = KMeans(n_clusters=3, random_state=42)
df['kmeans_cluster'] = kmeans.fit_predict(df_scaled)
```

```
plt.figure(figsize=(7,5))
sns.scatterplot(x=df_scaled.iloc[:,0], y=df_scaled.iloc[:,1], hue=df['kmeans_cluster'], palette='Set2')
plt.title("K-Means Clustering")
plt.show()
```

K-Means Clustering
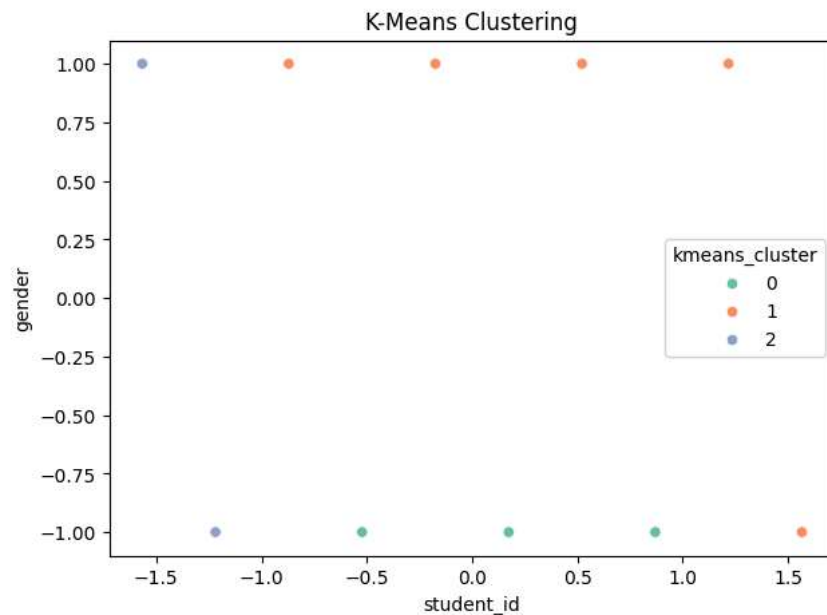

```
agg = AgglomerativeClustering(n_clusters=3)
df['agg_cluster'] = agg.fit_predict(df_scaled)
```

```
plt.figure(figsize=(7,5))
sns.scatterplot(x=df_scaled.iloc[:,0], y=df_scaled.iloc[:,1], hue=df['agg_cluster'], palette='Set1')
plt.title("Agglomerative Clustering")
plt.show()
```

Agglomerative Clustering


```
dbscan = DBSCAN(eps=1.5, min_samples=3)
df['dbscan_cluster'] = dbscan.fit_predict(df_scaled)
```

```
plt.figure(figsize=(7,5))
sns.scatterplot(x=df_scaled.iloc[:,0], y=df_scaled.iloc[:,1], hue=df['dbscan_cluster'], palette='Set3')
plt.title("DBSCAN Clustering")
plt.show()
```

DBSCAN Clustering


```
print(df[['kmeans_cluster', 'agg_cluster', 'dbscan_cluster']].head())
```

```
   kmeans_cluster  agg_cluster  dbscan_cluster  student_id
0               2            1              -1
1               2            1              -1
2               1            0              -1
3               0            2              -1
4               1            0              -1
```

```
# ✅ Use existing column names
df_apriori = df[['gender', 'degree', 'placed']]   # replace degree_t with actual column name
df_apriori = pd.get_dummies(df_apriori)

# Find frequent itemsets
frequent_itemsets = apriori(df_apriori, min_support=0.2, use_colnames=True)

# Generate rules
rules = association_rules(frequent_itemsets, metric='lift', min_threshold=1)
print(rules[['antecedents', 'consequents', 'support', 'confidence', 'lift']])
```

```
         antecedents        consequents  support  confidence      lift
0           (gender)           (degree)      0.4    0.800000  1.333333
1           (degree)           (gender)      0.4    0.666667  1.333333
2   (gender, placed)           (degree)      0.2    1.000000  1.666667
3   (degree, placed)           (gender)      0.2    0.666667  1.333333
4           (gender)   (degree, placed)      0.2    0.400000  1.333333
5           (degree)   (gender, placed)      0.2    0.333333  1.666667
```

```
df_apriori = df[['gender','degree','placed']]
df_apriori = pd.get_dummies(df_apriori)
```

```
frequent_itemsets = apriori(df_apriori, min_support=0.2, use_colnames=True)
rules = association_rules(frequent_itemsets, metric='lift', min_threshold=1)
print(rules[['antecedents', 'consequents', 'support', 'confidence', 'lift']])
```