

Machine Learning Assignment 1

March 13, 2026

Course: Practical Machine Learning

Dr. Hager Saleh

Assignment 1

Name: Amro Ismail Mohamed

0.0.1 import the Libraries

```
[93]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
import warnings
warnings.filterwarnings("ignore")
```

0.0.2 Load the Students CSV File by Pandas Library

```
[94]: df = pd.read_csv('student_placement_synthetic.csv')
df.head()
```

```
[94]:
```

	branch	college_tier	cgpa	backlogs	coding_skills	dsa_score	\
0	ECE	Tier-3	6.70	0	7.6	4.4	
1	Chemical	Tier-2	5.70	0	5.4	7.9	
2	EE	Tier-2	7.19	0	5.6	6.8	
3	CE	Tier-2	6.48	0	5.2	3.1	
4	CSE	Tier-2	6.71	1	5.9	4.7	

	aptitude_score	communication_skills	ml_knowledge	system_design	\
0	49.5		3.7	6.4	0.3
1	72.0		8.3	6.3	1.9
2	79.1		7.4	4.4	5.2
3	48.4		5.0	1.1	6.7
4	61.2		4.3	2.7	2.8

	internships	projects_count	certifications	hackathons	\
0	1	4	4	3	

1	0	4	0	0
2	1	3	2	1
3	1	4	3	0
4	1	2	0	3

	open_source_contributions	extracurriculars	placement_status	\
0	2	1	1	
1	0	0	0	
2	2	0	1	
3	0	0	0	
4	0	1	1	

	salary_package_lpa
0	14.75
1	NaN
2	19.06
3	NaN
4	13.42

```
[95]: df.dtypes
```

```
[95]: branch                object
      college_tier          object
      cgpa                  float64
      backlogs              int64
      coding_skills         float64
      dsa_score             float64
      aptitude_score        float64
      communication_skills  float64
      ml_knowledge          float64
      system_design         float64
      internships           int64
      projects_count        int64
      certifications        int64
      hackathons            int64
      open_source_contributions int64
      extracurriculars      int64
      placement_status      int64
      salary_package_lpa    float64
      dtype: object
```

```
[96]: df.columns
```

```
[96]: Index(['branch', 'college_tier', 'cgpa', 'backlogs', 'coding_skills',
            'dsa_score', 'aptitude_score', 'communication_skills', 'ml_knowledge',
            'system_design', 'internships', 'projects_count', 'certifications',
            'hackathons', 'open_source_contributions', 'extracurriculars',
```

```

        'placement_status', 'salary_package_lpa'],
        dtype='object')

```

0.0.3 Check the Missing Values & drop the Columns with 30% Null & Fill the missing values

```
[97]: df.isnull().sum()
```

```

[97]: branch                0
      college_tier          0
      cgpa                  0
      backlogs              0
      coding_skills         0
      dsa_score              0
      aptitude_score        0
      communication_skills  0
      ml_knowledge           0
      system_design         0
      internships           0
      projects_count        0
      certifications        0
      hackathons            0
      open_source_contributions 0
      extracurriculars      0
      placement_status      0
      salary_package_lpa    31525
      dtype: int64

```

```

[98]: missing_percentage = df.isnull().mean() * 100
      threshold = 30
      # Filter columns with more than the threshold percentage of missing values
      columns_to_drop = missing_percentage[missing_percentage > threshold].index
      # Drop the columns
      df = df.drop(columns=columns_to_drop)
      # Display the resulting dataframe after removing columns with too many missing
      ↪ values
      print(f"Columns dropped: {list(columns_to_drop)}")
      print(f"Remaining columns: {df.columns}")

```

```

Columns dropped: ['salary_package_lpa']
Remaining columns: Index(['branch', 'college_tier', 'cgpa', 'backlogs',
'coding_skills',
      'dsa_score', 'aptitude_score', 'communication_skills', 'ml_knowledge',
      'system_design', 'internships', 'projects_count', 'certifications',
      'hackathons', 'open_source_contributions', 'extracurriculars',
      'placement_status'],
      dtype='object')

```

```
[99]: #Pre-processing on dataset
label_encoder = LabelEncoder()
for column in df.select_dtypes(include=['object']).columns:
    df[column] = label_encoder.fit_transform(df[column])
df.head()
```

```
[99]:
```

	branch	college_tier	cgpa	backlogs	coding_skills	dsa_score \
0	3	2	6.70	0	7.6	4.4
1	2	1	5.70	0	5.4	7.9
2	4	1	7.19	0	5.6	6.8
3	0	1	6.48	0	5.2	3.1
4	1	1	6.71	1	5.9	4.7

	aptitude_score	communication_skills	ml_knowledge	system_design \
0	49.5	3.7	6.4	0.3
1	72.0	8.3	6.3	1.9
2	79.1	7.4	4.4	5.2
3	48.4	5.0	1.1	6.7
4	61.2	4.3	2.7	2.8

	internships	projects_count	certifications	hackathons \
0	1	4	4	3
1	0	4	0	0
2	1	3	2	1
3	1	4	3	0
4	1	2	0	3

	open_source_contributions	extracurriculars	placement_status
0	2	1	1
1	0	0	0
2	2	0	1
3	0	0	0
4	0	1	1

```
[100]: # Fill missing values in columns with mode
for column in df.select_dtypes(include=['float64']).columns:
    mean_value = df[column].mean() # Get the mode (most frequent value)
    df[column] = df[column].fillna(mean_value)
# Fill missing values in categorical columns with mode
for column in df.select_dtypes(include=['object']).columns:
    mode_value = df[column].mode()[0] # Get the mode (most frequent value)
    df[column] = df[column].fillna(mode_value)
df.isnull().sum()
```

```
[100]: branch          0
college_tier         0
cgpa                 0
```

```

backlogs                0
coding_skills            0
dsa_score                0
aptitude_score          0
communication_skills    0
ml_knowledge             0
system_design            0
internships              0
projects_count           0
certifications           0
hackathons               0
open_source_contributions 0
extracurriculars         0
placement_status         0
dtype: int64

```

0.0.4 Identify the Outliers

```

[101]: #Using IQR (Interquartile Range):
data=df.copy()
def detect_outliers_iqr(column):
    Q1 = column.quantile(0.25)
    Q3 = column.quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    # Return a boolean series indicating if the value is an outlier
    return (column < lower_bound) | (column > upper_bound)

# Apply the IQR method to each feature in the DataFrame
for col in data.columns:
    data[f'{col}_Outlier'] = detect_outliers_iqr(data[col])

# Filter the DataFrame to display rows where any outlier flag is True
outliers_only = data[data.filter(like='_Outlier').any(axis=1)]
# Display the filtered DataFrame with outliers
outliers_only

```

```

[101]:
   branch  college_tier  cgpa  backlogs  coding_skills  dsa_score  \
0         3            2  6.70         0           7.6         4.4
4         1            1  6.71         1           5.9         4.7
8         0            1  7.03         0           5.1         2.5
9         4            1  7.34         3           4.7         6.2
13        1            1  6.14         0           6.4         9.3
...      ...          ...  ...      ...      ...      ...
99973    2            0  5.44         3           5.2         8.1
99974    6            1  6.33         3           5.9         6.2

```

99978	3	2	7.66	0	5.2	4.8
99987	6	1	6.84	3	5.6	5.8
99990	5	2	8.03	0	3.4	8.6

	aptitude_score	communication_skills	ml_knowledge	system_design	...	\
0	49.5		3.7	6.4	0.3	...
4	61.2		4.3	2.7	2.8	...
8	54.7		6.4	6.2	3.2	...
9	78.8		5.9	6.7	5.2	...
13	56.6		5.9	3.4	3.1	...
...	...	...	...	...	...	...
99973	66.7		5.9	5.9	5.7	...
99974	89.0		6.7	7.4	0.0	...
99978	72.4		4.4	6.9	5.7	...
99987	68.5		6.2	0.3	6.3	...
99990	63.7		4.4	3.1	1.0	...

	communication_skills_Outlier	ml_knowledge_Outlier	\
0	False	False	
4	False	False	
8	False	False	
9	False	False	
13	False	False	
...	...	...	
99973	False	False	
99974	False	False	
99978	False	False	
99987	False	False	
99990	False	False	

	system_design_Outlier	internships_Outlier	projects_count_Outlier	\
0	False	False	False	
4	False	False	False	
8	False	False	False	
9	False	False	False	
13	False	False	False	
...	...	...	...	
99973	False	False	False	
99974	False	False	False	
99978	False	False	False	
99987	False	False	False	
99990	False	False	False	

	certifications_Outlier	hackathons_Outlier	\
0	True	True	
4	False	True	
8	False	True	

9	False	False
13	False	True
...	...	...
99973	False	False
99974	False	False
99978	False	True
99987	False	False
99990	False	True

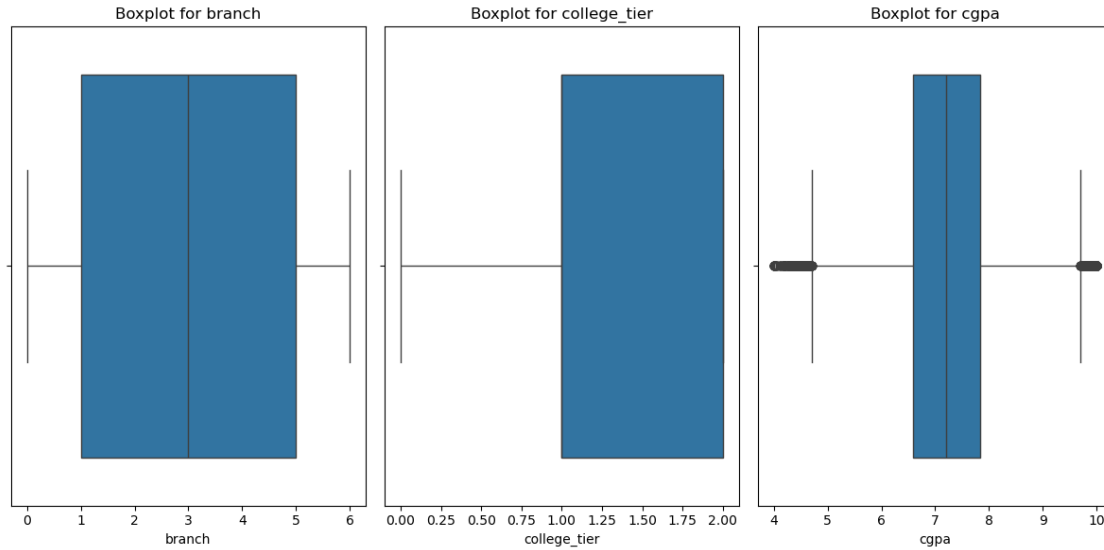
	open_source_contributions_Outlier	extracurriculars_Outlier	\
0	False	False	
4	False	False	
8	False	False	
9	False	False	
13	False	False	
...	...	...	
99973	False	False	
99974	False	False	
99978	False	False	
99987	False	False	
99990	False	False	

	placement_status_Outlier
0	False
4	False
8	False
9	False
13	False
...	...
99973	False
99974	False
99978	False
99987	False
99990	False

[16525 rows x 34 columns]

[102]: *#Using Visualization:*

```
plt.figure(figsize=(12, 6))
for i, col in enumerate(df.columns[:3]): # Only plotting the first 3 features
    plt.subplot(1, 3, i+1) # Create subplots for each feature
    sns.boxplot(x=df[col])
    plt.title(f"Boxplot for {col}")
plt.tight_layout()
plt.show()
```



```
[103]: #Using Z-Score:
# Function to detect outliers in a column using Z-Score
data=df.copy()
def detect_outliers_zscore(column, threshold=3):
    mean = np.mean(column)
    std_dev = np.std(column)
    z_scores = (column - mean) / std_dev
    # Return a boolean series where True indicates an outlier
    return np.abs(z_scores) > threshold

# Apply the Z-Score method to each feature in the DataFrame
for col in data.columns:
    data[f'{col}_Outlier'] = detect_outliers_zscore(data[col])

# Filter the DataFrame to display rows where any outlier flag is True
outliers_only = data[data.filter(like='_Outlier').any(axis=1)]
outliers_only
```

```
[103]:
```

	branch	college_tier	cgpa	backlogs	coding_skills	dsa_score	\
60	3	2	7.93	3	1.4	8.7	
216	3	2	4.41	2	7.1	5.2	
255	0	1	7.57	0	8.5	6.1	
278	4	2	7.99	1	5.8	6.0	
326	0	1	8.12	0	5.6	9.1	
...	...	...	...	...	...	...	
99484	1	2	7.51	1	5.7	5.0	
99591	0	1	8.21	3	1.0	3.5	
99797	6	1	6.26	0	5.1	7.7	

99808	6	2	6.04	0	10.0	5.7
99925	3	2	4.39	1	8.1	3.7

	aptitude_score	communication_skills	ml_knowledge	system_design	...	\
60	86.0	6.3	3.7	3.2	...	
216	48.4	4.2	5.0	3.2	...	
255	91.8	3.6	4.7	9.4	...	
278	74.6	5.6	5.8	9.6	...	
326	27.8	6.1	5.5	5.1	...	
...	...	...	...	...	...	
99484	85.5	7.0	4.1	10.0	...	
99591	84.4	6.8	2.5	0.9	...	
99797	64.8	6.2	1.8	9.5	...	
99808	25.2	6.9	7.2	1.5	...	
99925	75.1	4.5	3.1	5.8	...	

	communication_skills_Outlier	ml_knowledge_Outlier	\
60	False	False	
216	False	False	
255	False	False	
278	False	False	
326	False	False	
...	...	...	
99484	False	False	
99591	False	False	
99797	False	False	
99808	False	False	
99925	False	False	

	system_design_Outlier	internships_Outlier	projects_count_Outlier	\
60	False	False	False	
216	False	False	False	
255	True	False	False	
278	True	False	False	
326	False	False	False	
...	...	...	...	
99484	True	False	False	
99591	False	False	False	
99797	True	False	False	
99808	False	False	False	
99925	False	False	False	

	certifications_Outlier	hackathons_Outlier	\
60	False	False	
216	False	False	
255	False	False	
278	False	False	

326	False	False
...	...	...
99484	False	False
99591	False	False
99797	False	False
99808	False	False
99925	False	False

	open_source_contributions_Outlier	extracurriculars_Outlier	\
60	False	False	
216	False	False	
255	False	False	
278	False	False	
326	False	False	
...	...	...	
99484	False	False	
99591	False	False	
99797	False	False	
99808	False	False	
99925	False	False	

	placement_status_Outlier
60	False
216	False
255	False
278	False
326	False
...	...
99484	False
99591	False
99797	False
99808	False
99925	False

[825 rows x 34 columns]

0.0.5 Handling the Outliers

```
[104]: data=df.copy()

def replace_outliers_with_nan(column):
    Q1 = column.quantile(0.25)
    Q3 = column.quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    # Replace outliers with NaN
```

```

    return column.where((column >= lower_bound) & (column <= upper_bound), np.
↪nan)

# Replace outliers with NaN for each feature
for col in data.columns:
    data[col] = replace_outliers_with_nan(data[col])

null_counts = data.isnull().sum()
print("Sum of null values in each column:")
print(null_counts)

```

Sum of null values in each column:

```

branch          0
college_tier    0
cgpa            686
backlogs        4981
coding_skills   382
dsa_score       0
aptitude_score  680
communication_skills 362
ml_knowledge    332
system_design   364
internships     0
projects_count  0
certifications  4946
hackathons      4916
open_source_contributions 0
extracurriculars 0
placement_status 0
dtype: int64

```

```

[105]: # Remove outliers
       # Using the Interquartile Range (IQR) Method
       data=df.copy()

       def remove_outliers_iqr(data):
           # Calculate Q1 (25th percentile) and Q3 (75th percentile) for each feature
           Q1 = data.quantile(0.25)
           Q3 = data.quantile(0.75)
           IQR = Q3 - Q1
           # Define lower and upper bounds
           lower_bound = Q1 - 1.5 * IQR
           upper_bound = Q3 + 1.5 * IQR
           # Filter the dataset to remove rows with outliers in any column
           data_cleaned = data[~((data < lower_bound) | (data > upper_bound)).
↪any(axis=1)]

```

```
    return data_cleaned

cleaned_data = remove_outliers_iqr(data)
print("Original Data:")
print(len(data))
print("\nCleaned Data (without outliers):")
print(len(cleaned_data))
```

Original Data:

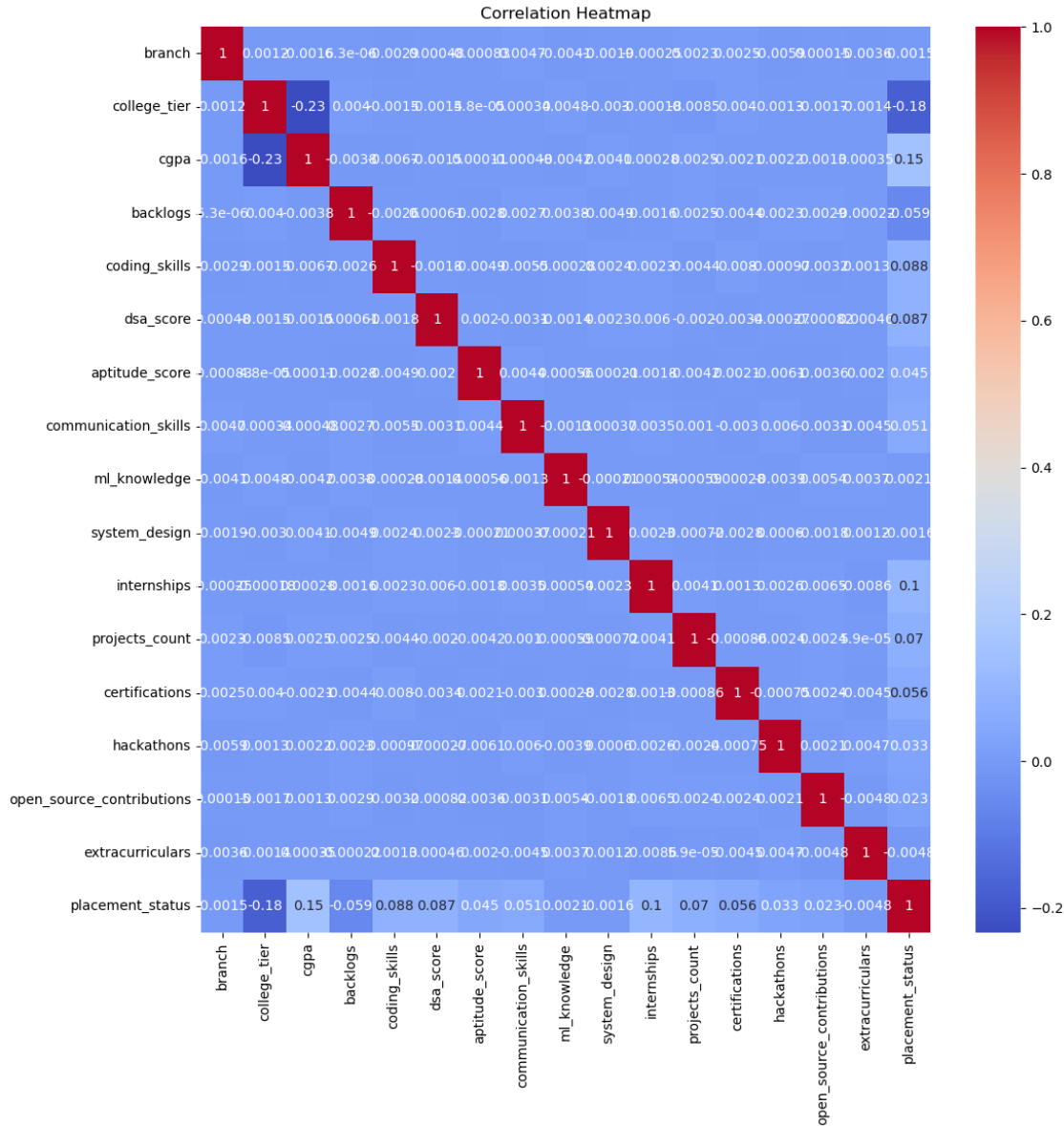
100000

Cleaned Data (without outliers):

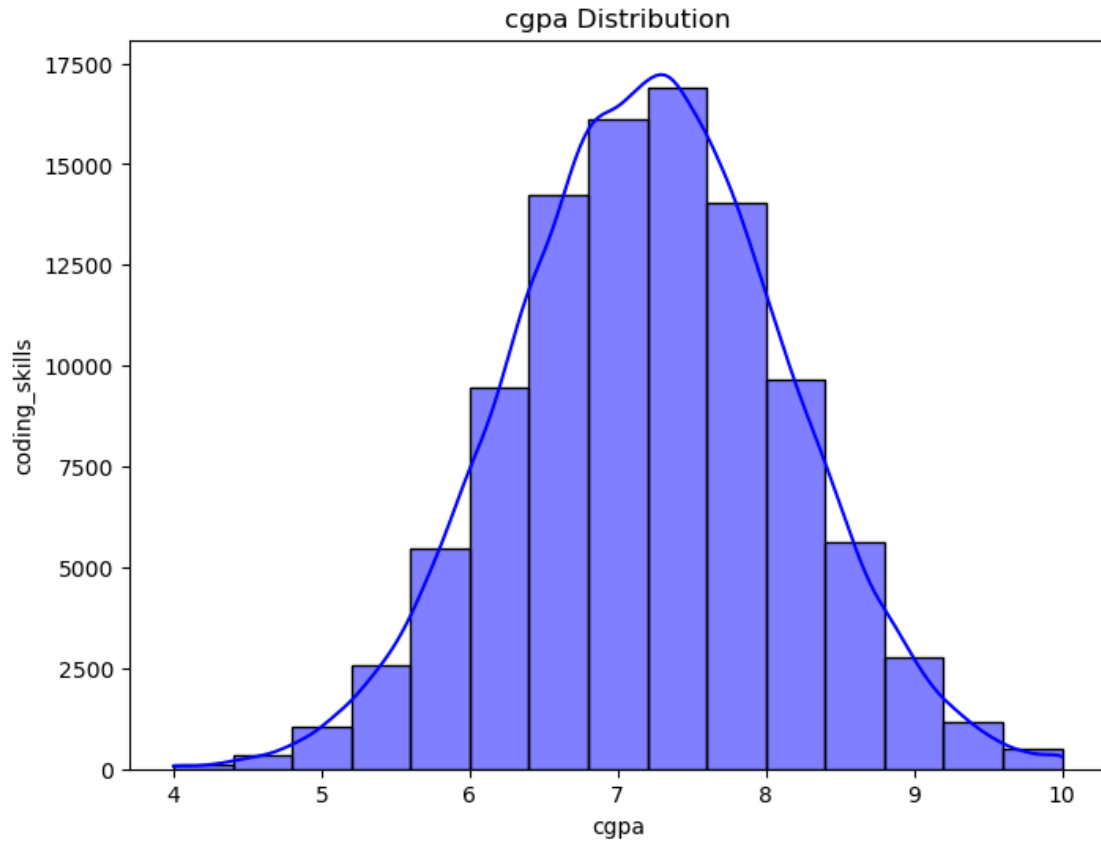
83475

0.0.6 Data Visualizations

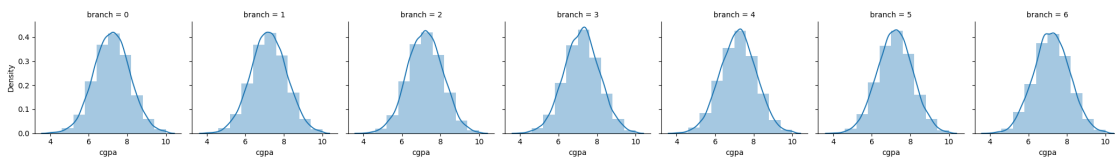
```
[106]: plt.figure(figsize=(12, 12))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```



```
[107]: cat_cols = [col for col in df.columns if df[col].dtype=='object']
num_cols = [col for col in df.columns if df[col].dtype!='object']
for col in cat_cols:
    print(f'{col} has {df[col].unique()} values')
# 1. Distribution of branch
plt.figure(figsize=(8, 6))
sns.histplot(df['cgpa'], bins=15, kde=True, color='blue')
plt.title('cgpa Distribution')
plt.xlabel('cgpa')
plt.ylabel('coding_skills')
plt.show()
```

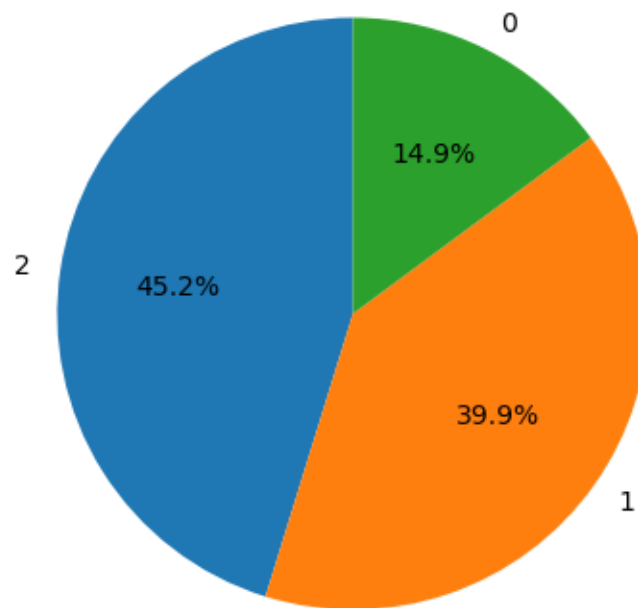


```
[108]: g = sns.FacetGrid(df,col="branch")
g.map(sns.distplot,"cgpa", bins=10)
plt.show()
```

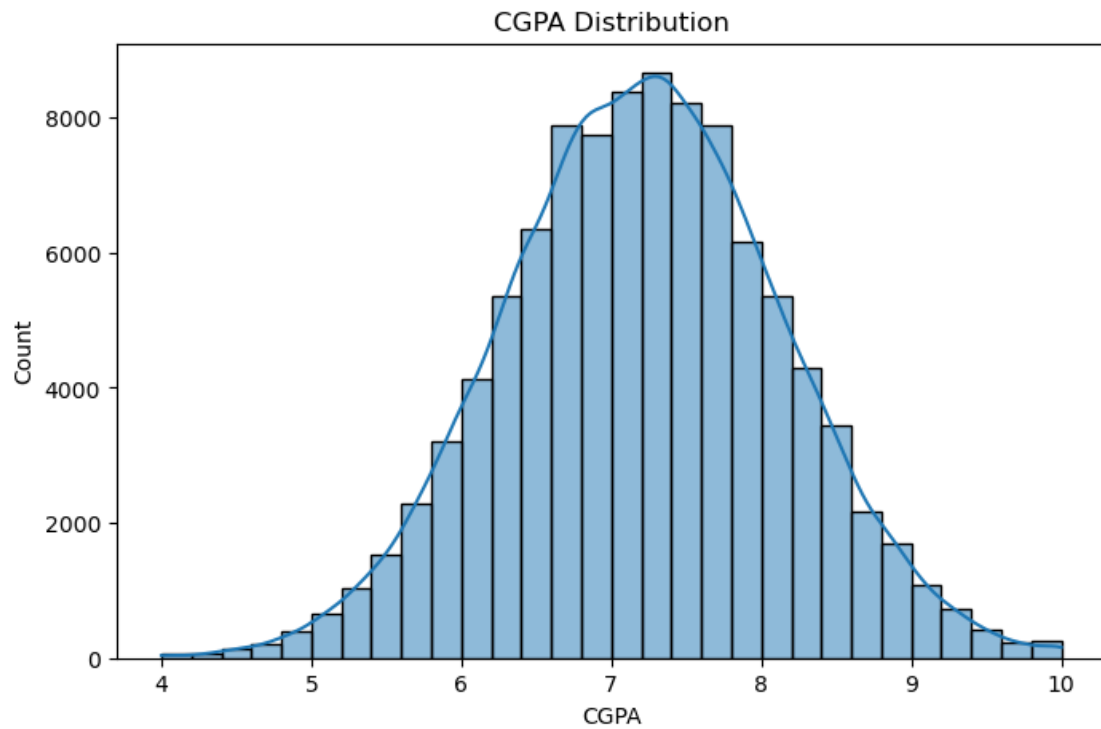


```
[109]: appet_counts = df['college_tier'].value_counts()
plt.figure(figsize=(5,5))
plt.pie(appet_counts, labels=appet_counts.index, autopct='%1.1f%%',
startangle=90)
plt.title("college_tier Distribution")
plt.show()
```

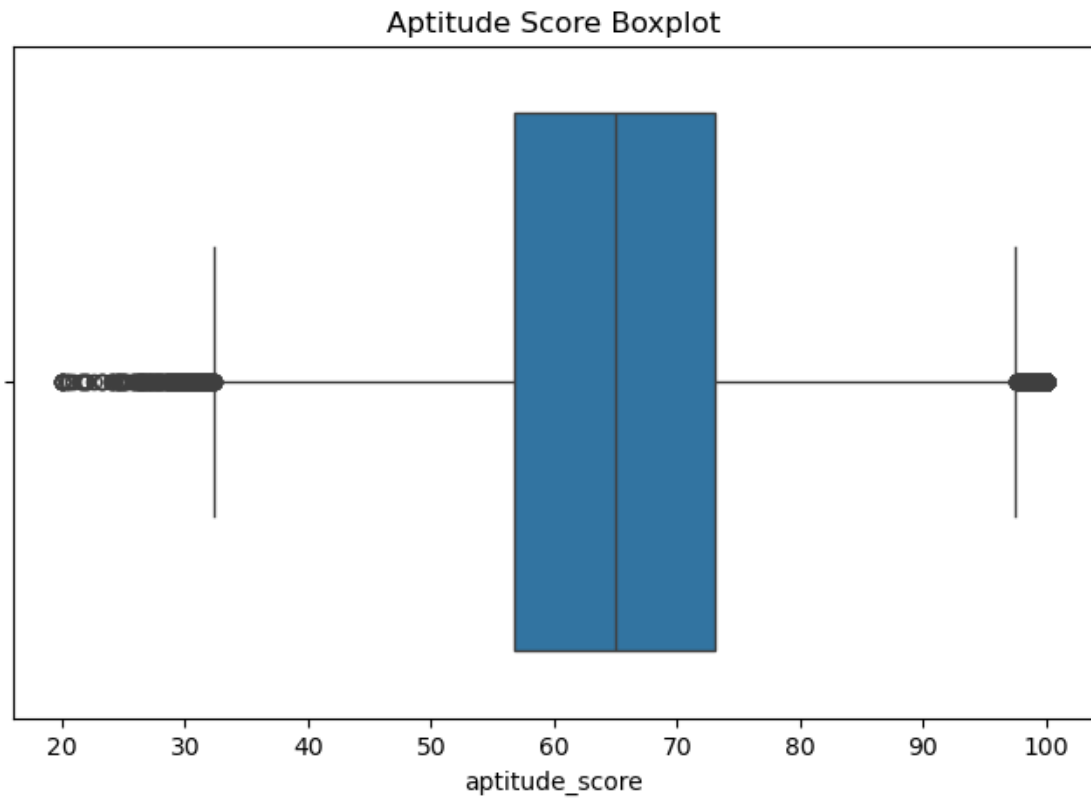
college_tier Distribution



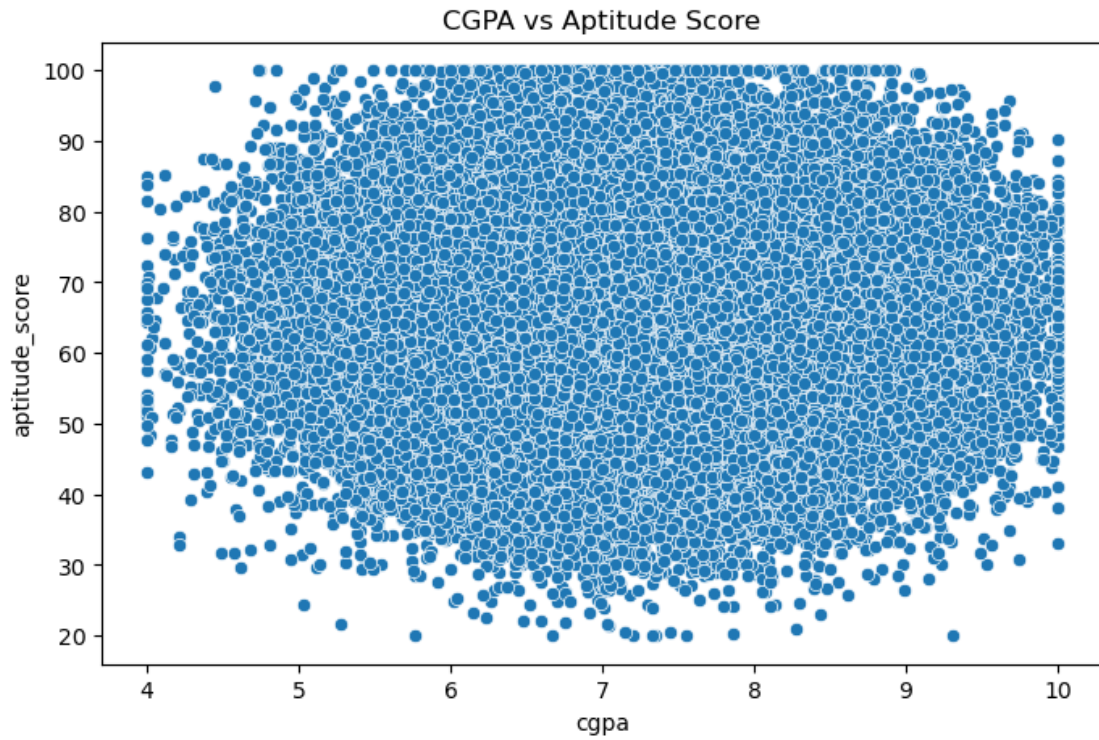
```
[110]: plt.figure(figsize=(8,5))
sns.histplot(df["cgpa"], bins=30, kde=True)
plt.title("CGPA Distribution")
plt.xlabel("CGPA")
plt.ylabel("Count")
plt.show()
```



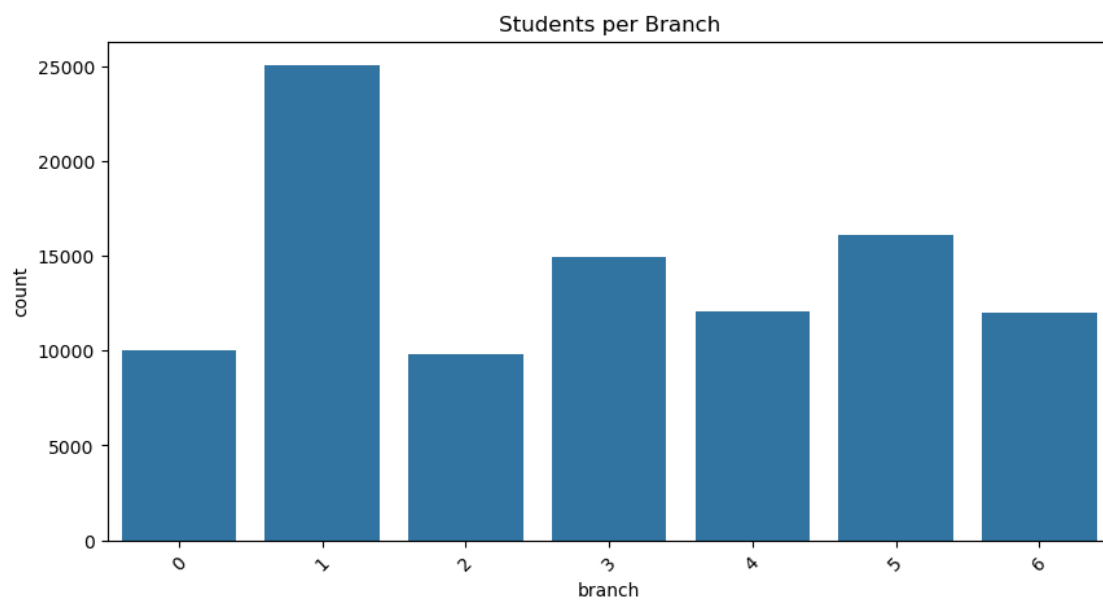
```
[111]: plt.figure(figsize=(8,5))
sns.boxplot(x=df["aptitude_score"])
plt.title("Aptitude Score Boxplot")
plt.show()
```



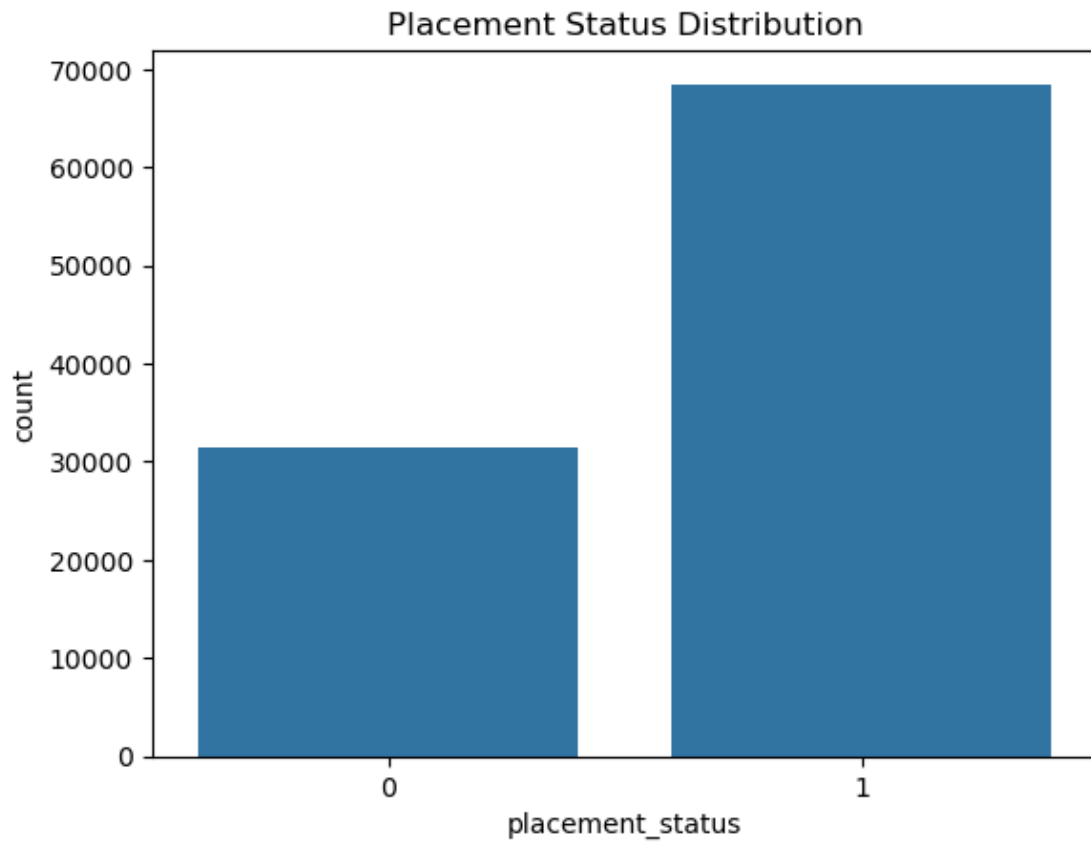
```
[112]: plt.figure(figsize=(8,5))
sns.scatterplot(x="cgpa", y="aptitude_score", data=df)
plt.title("CGPA vs Aptitude Score")
plt.show()
```



```
[113]: plt.figure(figsize=(10,5))
sns.countplot(x="branch", data=df)
plt.title("Students per Branch")
plt.xticks(rotation=45)
plt.show()
```



```
[114]: sns.countplot(x="placement_status", data=df)
plt.title("Placement Status Distribution")
plt.show()
```



```
[115]: sns.pairplot(df[["cgpa", "aptitude_score", "coding_skills", "dsa_score"]])
plt.show()
```

