Institute of Space Technology

Predicting S&CGPA of 5th smester

Engineering Management

Semester Project

12 January 2024

Contents

1: Literature Review

The use of machine learning models to forecast academic achievement has gained popularity in recent years, with a particular emphasis on the Cumulative Grade Point Average (CGPA) and Semester Grade Point Average (SGPA). Numerous studies have examined how well machine learning algorithms predict these important educational measures, providing insightful information and possible educational sector applications.

Mohamed Elfaki and Tarig Abdelgadir's "**Predicting Academic Performance using Machine Learning Algorithms (2017)"** was one of the first research to look at the predictive power of machine learning models. To predict CGPA and SGPA, they looked at a number of techniques, such as neural networks, decision trees, and linear regression. The study's conclusions showed that machine learning models have the potential to predict academic performance correctly, which may help educational institutions identify students who are at risk and provide timely interventions.

The significance of forecasting academic success in a university context was highlighted in the 2019 study **"Forecasting Student success: An Application of Machine Learning Algorithms at a UAE University"** by Ahmed Elngar et al. To forecast students' CGPA and SGPA, the researchers used a variety of machine learning methods, including Random Forest, Gradient Boosting, and Support Vector Machines. Their research shown how effective these models are in identifying kids early on who may need more academic help, which eventually leads to better student results.

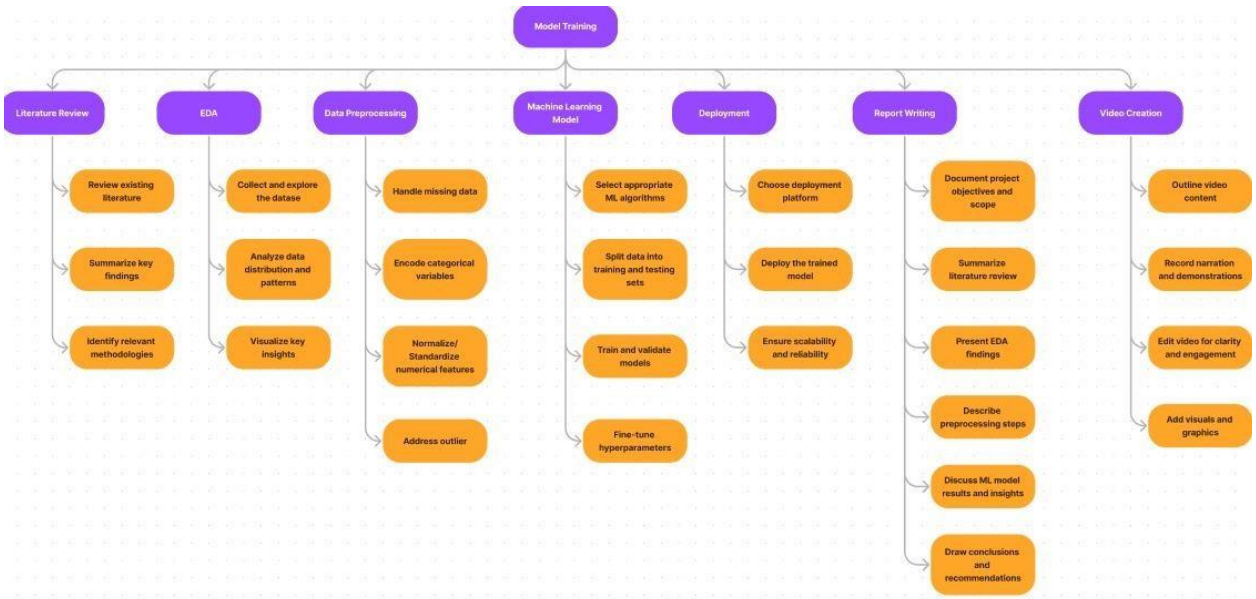
In their 2018 study, "**Predicting University Student Academic Performance: A Comparative Study of Machine Learning Algorithms",** Saeed Ur Rehman et al. compared the effectiveness of machine learning algorithms in predicting the academic performance of university students, taking into account factors like CGPA and SGPA. The research assessed the effectiveness of many algorithms, including logistic regression, decision trees, and k-nearest neighbors. The study found that when it came to prediction accuracy, ensemble methods—Random Forest in particular—performed better than other algorithms.

The use of machine learning methods for forecasting university students' academic success was explored in **"Academic success Prediction Using Machine Learning methods (2020)",** written by Sahar Alharbi and Sameera A. Khan. To predict CGPA and SGPA, factors such prior SGPA, attendance, and coursework grades were taken into account. This study demonstrated how machine learning models may be used to provide useful information on the development and performance of students.

"**Data Mining Techniques for Predicting Student Academic Performance: A Case Study (2014)"** A case study by Mustafa Şahin and Cansu Günay forecasted student academic achievement, including CGPA and SGPA, using data mining methods, such as decision trees and neural networks. This research demonstrated how machine learning models may help educators and organizations pinpoint the variables affecting students' performance, which makes it easier to create successful intervention plans.

When taken as a whole, these papers demonstrate the increasing interest in using machine learning models to predict CGPA and SGPA in educational settings. These models' predictive powers provide chances to enhance academic counselling, provide early intervention for students who are at danger, and improve overall academic results. However, it is crucial to recognize that the accuracy and applicability of these predictions are heavily influenced by the characteristics, algorithms, and dataset quality used.

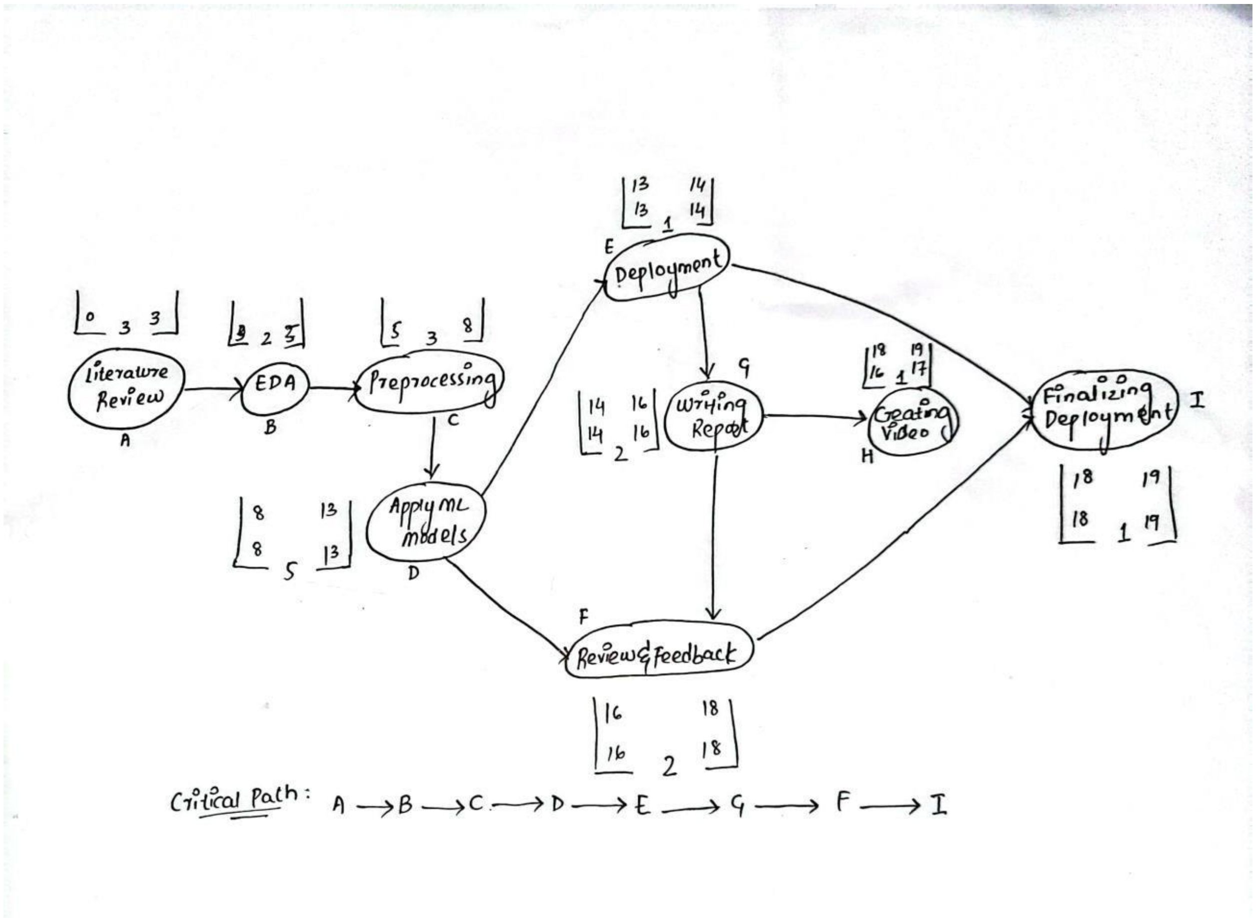
2: Work Breakdown Structure



3: Activity-On-Node

Activity	Duration	Predecessors
Literature Review	3 days	None
EDA	2 days	Literature Review
Preprocessing	3 days	EDA
Applying ML Models	5 days	Preprocessing
Deployment	1 day	Applying ML Models
Writing Report	2 days	Deployment
Creating Video	1 day	Writing Report

Review and Feedback	2 days	Writing Report, Applying ML Models
Finalizing Deployment	1 day	Deployment, Review and Feedback



4: Project Requirements

4.1: Data Preprocessing

This code snippet below performs the following tasks: Initially, it imports necessary libraries, such as Matplotlib with Seaborn for data visualisation, NumPy for mathematical calculations, and Pandas for data processing. Then it reads the 'Final Data Set.csv' CSV dataset from the given file location into a Pandas DataFrame called 'df.' Lastly, it makes use of the df.info() function to provide fundamental details about the dataset, including its size, names of the columns, counts of non-null values in each column, data types, and memory utilization. This first investigation serves as a foundation for further in-depth data analysis and visualization by assisting in comprehending the structure of the dataset and detecting any problems with the data.

```python
# Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Load your dataset
df = pd.read_csv(r'C:\Users\HP\Downloads\assignment3\Final Data Set.csv')

# Display basic information about the dataset
df.info()
```

[3] Python

```
class 'pandas.core.frame.DataFrame'>
angeIndex: 73 entries, 0 to 72
ata columns (total 84 columns):
#   Column
--  ------
0   ID No.
1   Program of Study
2   Gender
3   Nationality
4   Place of Birth
5   Father's Education
6   Mother's Education
7   Parental Income
8   Number of immediate family members
9   Any close family member with the same profession available for guidance
10  Availing any scholarship
11  I opted for this program of study because of my own interest.
12  Basic Education Stream
13  Intermediate Stream
14  Matric percentage
```

0 ⚠ 15 0 Ln 1, Col 71 (70 selected) Spaces: 4 CRLF Cell 14 of 56 Go Live

Anomy in CSV file

```python
# Summary statistics of numerical columns
df.describe()
```

Python

	ID No.	Matric percentage	Intermediate percentage	SGPA in BS First semester	SGPA in BS Second semester	SGPA in BS Third semester	SGPA in BS Fourth semester	SGPA in BS Fifth semester	CGPA in BS Fifth semester	Unnamed: 76	Unnamed: 77	Unnamed: 78	Unnamed: 79	Unnamed: 80	Unnamed: 81	Unnamed: 82	Unn
count	73.000000	73.000000	73.000000	71.000000	73.000000	73.000000	73.000000	73.000000	72.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
mean	37.000000	83.744247	74.393014	3.005775	2.668904	2.823014	2.680274	2.825753	2.792361	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
std	21.217131	6.075923	5.550296	0.468245	0.576967	0.495686	0.523378	0.499614	0.424219	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
min	1.000000	61.430000	59.090000	1.230000	1.260000	1.690000	1.460000	1.810000	2.030000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
25%	19.000000	81.620000	71.360000	2.690000	2.280000	2.490000	2.260000	2.480000	2.477500	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
50%	37.000000	84.360000	74.450000	3.170000	2.630000	2.810000	2.630000	2.760000	2.730000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
75%	55.000000	87.730000	77.360000	3.330000	3.060000	3.200000	3.060000	3.190000	3.062500	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
max	73.000000	96.270000	87.450000	3.810000	3.770000	3.880000	3.890000	4.000000	3.730000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

The above code shows the statistical summary of the dataset and how each feature has what value.

4.2: Data Cleaning

This code takes data csv and cleans it up by removing unnecessary columns. It searches for any columns whose names begin with "Unnamed" and deletes them. This leaves you with a tidier table only containing relevant information for your analysis. **Removing extra columns**

```python
df = df.drop(df.columns[df.columns.str.startswith('Unnamed')], axis=1)
```

Info after removing extra columns

```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 73 entries, 0 to 72
Data columns (total 76 columns):
 #   Column                                                                   Non-Null Count  Dtype
---  ------                                                                   --------------  -----
 0   ID No.                                                                   73 non-null     int64
 1   Program of Study                                                         73 non-null     object
 2   Gender                                                                   73 non-null     object
 3   Nationality                                                              73 non-null     object
 4   Place of Birth                                                           73 non-null     object
 5   Father's Education                                                       73 non-null     object
 6   Mother's Education                                                       73 non-null     object
 7   Parental Income                                                          73 non-null     object
 8   Number of immediate family members                                       73 non-null     object
 9   Any close family member with the same profession available for guidance  73 non-null     object
 10  Availing any scholarship                                                 73 non-null     object
 11  I opted for this program of study because of my own interest.           73 non-null     object
 12  Basic Education Stream                                                   73 non-null     object
 13  Intermediate Stream                                                      73 non-null     object
 14  Matric percentage                                                        73 non-null     float64
 15  Intermediate percentage                                                  73 non-null     float64
 16  SGPA in BS First semester                                                71 non-null     float64
 17  SGPA in BS Second semester                                               73 non-null     float64
 18  SGPA in BS Third semester                                                73 non-null     float64
 19  SGPA in BS Fourth semester                                               73 non-null     float64
...
 74  My overall mobile usage (daily) for non-academic purpose is limited to:  73 non-null     object
 75  Identify any other variable that has negatively impacted on your academic performance.  67 non-null     object
dtypes: float64(8), int64(1), object(67)
memory usage: 43.5+ KB
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```


Checking missing values

The code scans a data table for missing values, counting them in each column and displaying the results for initial data quality assessment. It peeks at only the first chunk to avoid overflow.

```python
# Check for missing values
df.isnull().sum()
```

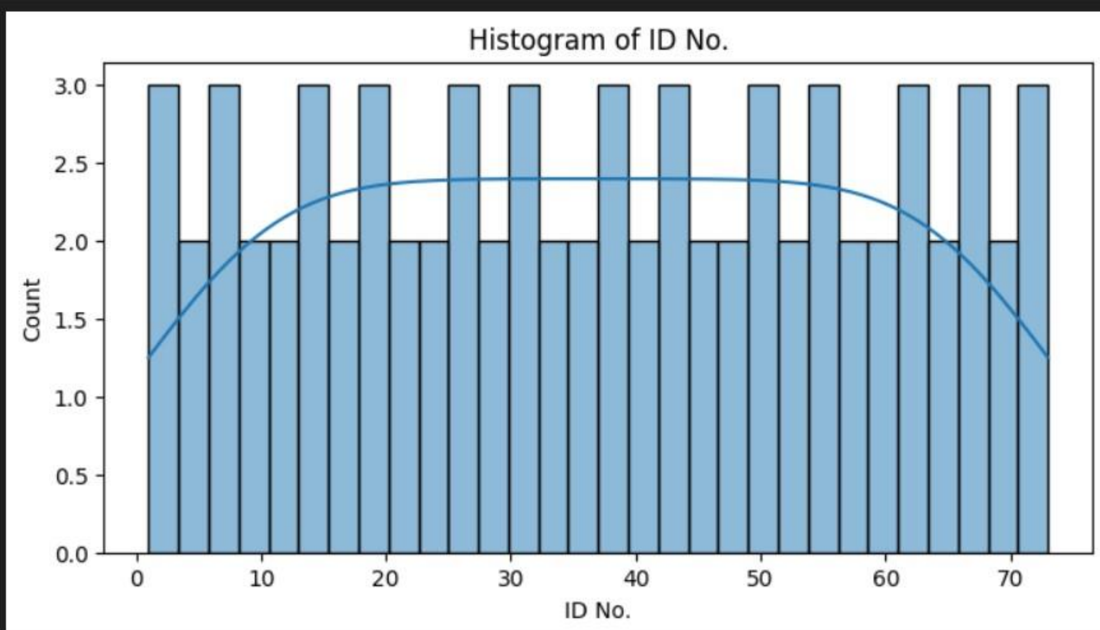
```
ID No.                                                                                    0
Program of Study                                                                          0
Gender                                                                                    0
Nationality                                                                               0
Place of Birth                                                                            0
                                                                                         ..
I often come across health issues that effect my academic performance.                    0
My preferable mode of study is:                                                           0
My preferable time of study is:                                                           0
My overall mobile usage (daily) for non-academic purpose is limited to:                   0
Identify any other variable that has negatively impacted on your academic performance.    6
Length: 76, dtype: int64
```

4.3: Visualizations/ EDA

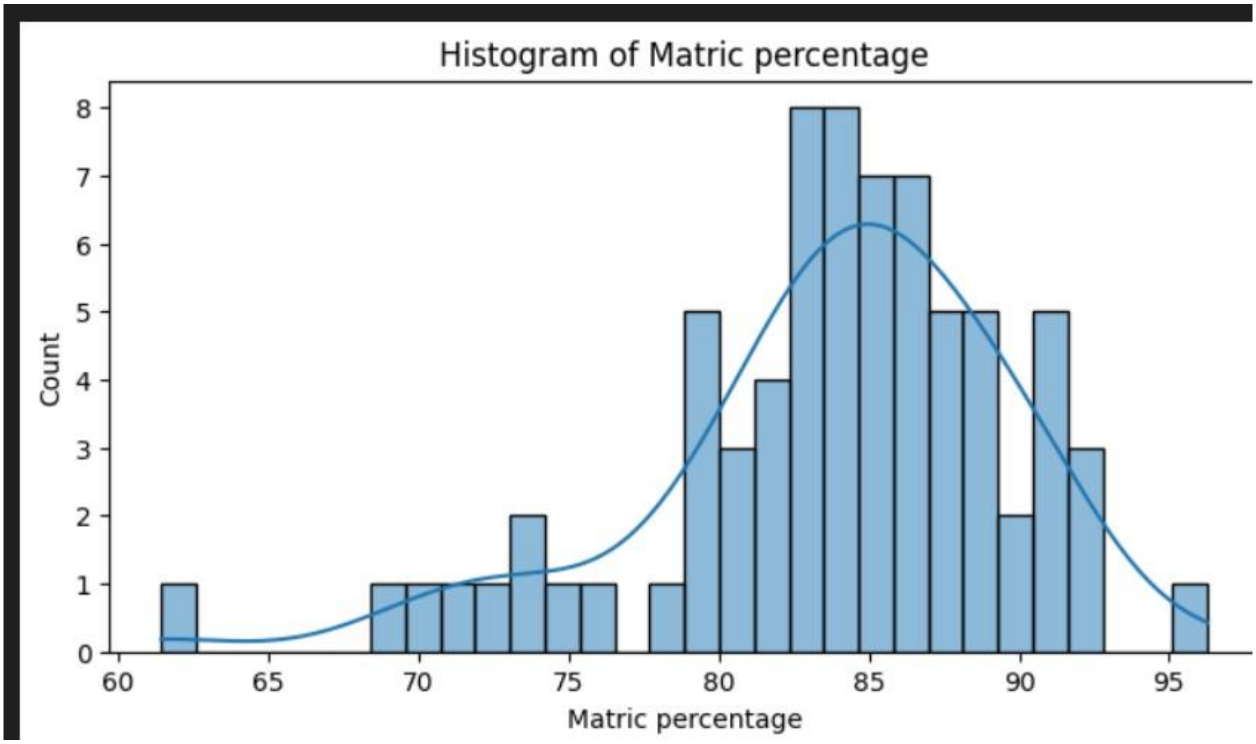
Exploratory Data Analysis (EDA) is a crucial step in understanding the dataset and uncovering insights. In this phase, we analyze the data and visualize relationships between different variables using various graphs and plots.

Histogram for Numerical Columns

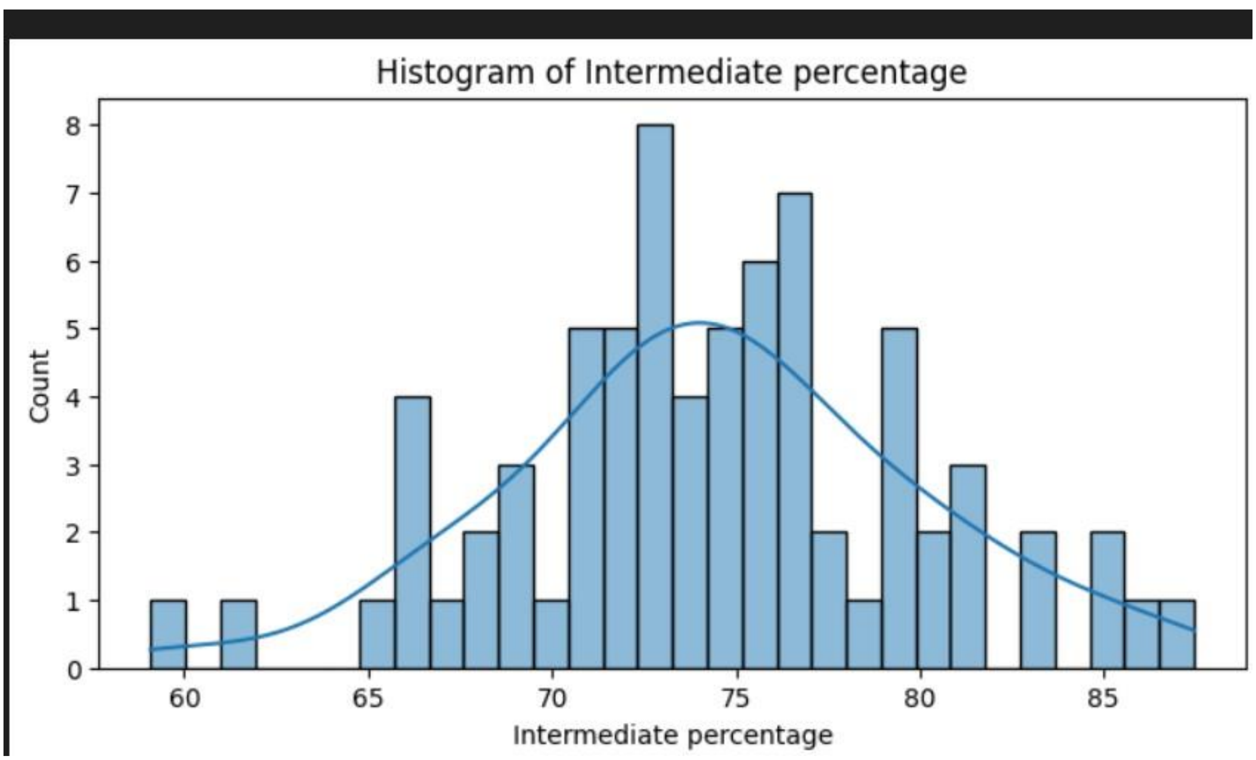
```python
# Histograms for numerical columns
numerical_cols = df.select_dtypes(include=['float64', 'int64']).columns
for col in numerical_cols:
    plt.figure(figsize=(8, 4))
    sns.histplot(df[col], bins=30, kde=True)
    plt.title(f'Histogram of {col}')
    plt.show()
#This code will create histograms for each numerical column to visualize their distributions.
```



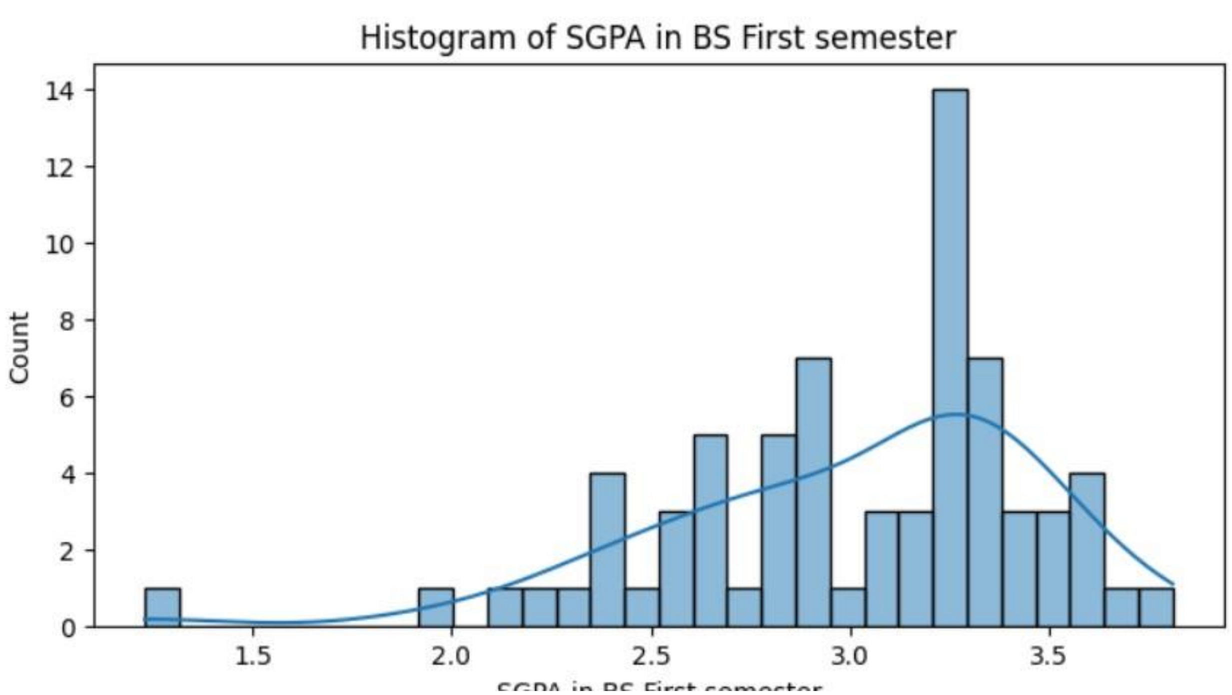
The histogram below shows a relation between matric percentage marks and number of students that lie in that range.



The histogram below shows a relation between intermediate percentage marks and number of students that lie in that range.



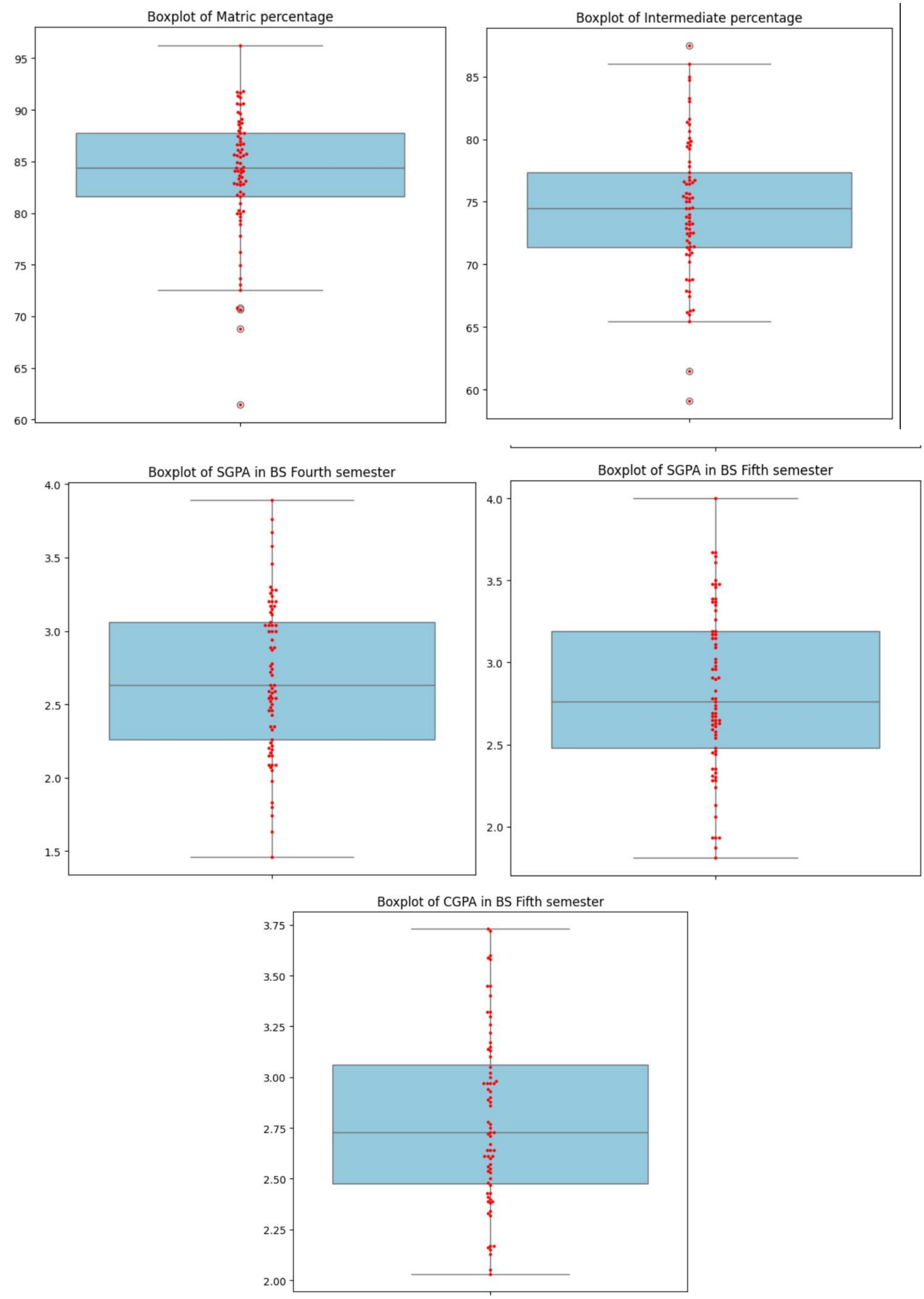
The histogram below shows a relation between SGPA of First semester marks and number of students that lie in that range.



Box plot for attributes

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

# Set a color palette for the plots
sns.set_palette("Set2")

# Set the figure size
plt.figure(figsize=(6, 50))  # Adjusted the figure size for vertical arrangement

# Create box plots for numerical columns
for index, col in enumerate(numerical_cols, start=1):
    plt.subplot(len(numerical_cols), 1, index)  # Changed the subplot arrangement for vertical display
    sns.boxplot(data=df, y=col, color='skyblue')  # Use y=col for vertical orientation
    sns.swarmplot(data=df, y=col, color='red', size=3)  # Show data points as red dots
    plt.title(f'Boxplot of {col}')
    plt.xlabel('')
    plt.ylabel('')

plt.tight_layout()  # Adjust subplot spacing for a better layout
plt.show()
```

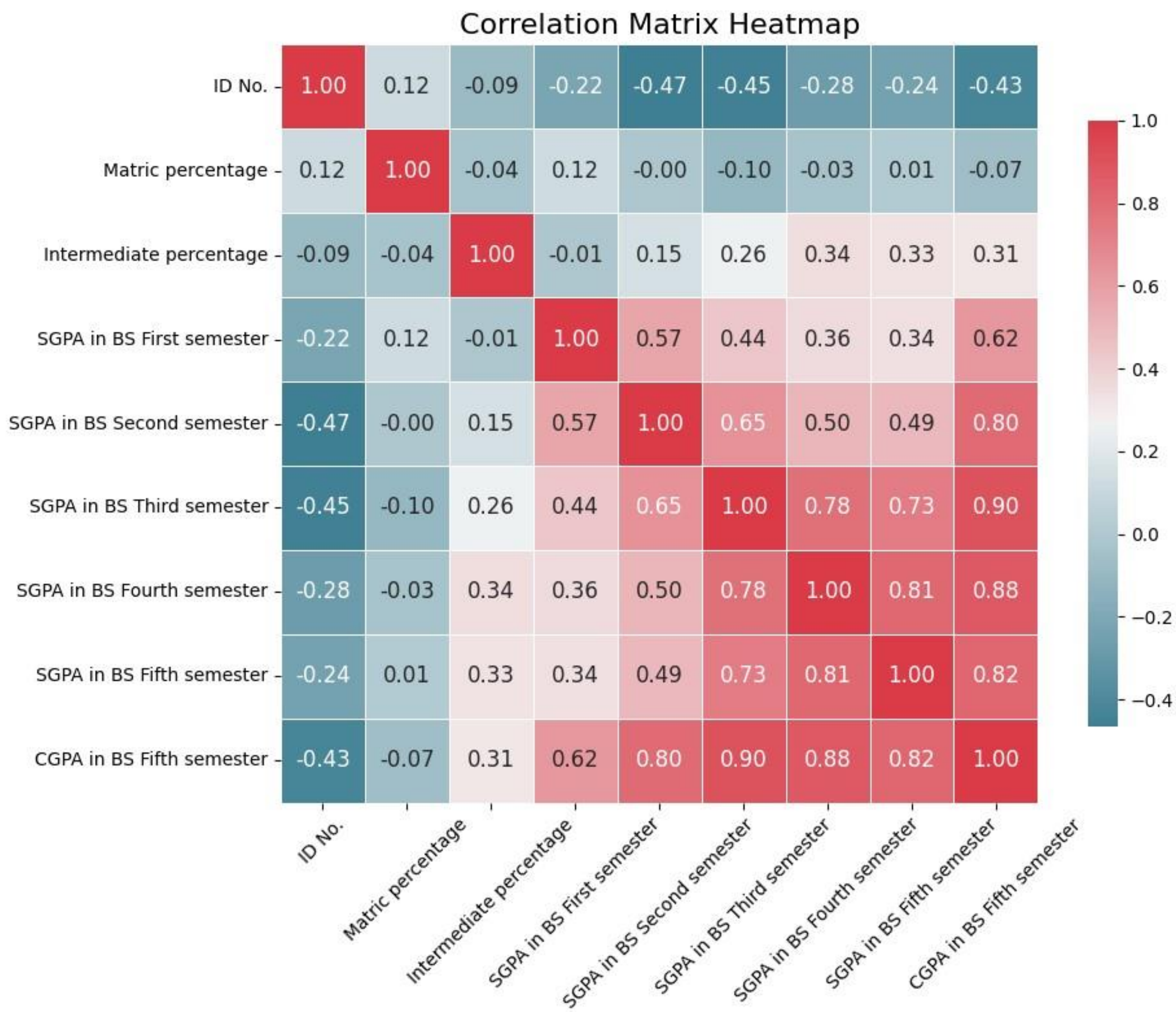
The box plots below show the distribution of numerical data in a certain threshold and display what outliers the data may have, this helps us judge If the data is fit for running model or we need to further preprocess it. As shown below most of the data is within the quartile range of boxplot, so we don’t have outliers that may cause the model to have poor accuracy.


Boxplot of Matric percentage
95
90
85
80
75
70
65
60
Boxplot of Intermediate percentage
85
80
75
70
65
60
Boxplot of SGPA in BS Fourth semester
4.0
3.5
3.0
2.5
2.0
1.5
Boxplot of SGPA in BS Fifth semester
4.0
3.5
3.0
2.5
2.0
Boxplot of CGPA in BS Fifth semester
3.75
3.50
3.25
3.00
2.75
2.50
2.25
2.00

Correlation Matrix

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

# Create a correlation matrix
corr_matrix = df[numerical_cols].corr()

# Set a custom color palette
cmap = sns.diverging_palette(220, 10, as_cmap=True)

# Set the figure size
plt.figure(figsize=(10, 8))

# Create the heatmap with customizations
sns.heatmap(corr_matrix,
            annot=True,
            fmt=".2f",  # Format the annotations to two decimal places
            cmap=cmap,
            square=True,  # Make the cells square
            linewidths=0.5,  # Add white lines between cells
            cbar_kws={'shrink': 0.8},  # Customize colorbar size
            annot_kws={'size': 12},  # Customize annotation font size
           )

plt.title('Correlation Matrix Heatmap', fontsize=16)
plt.xticks(rotation=45)  # Rotate x-axis labels for better readability
plt.yticks(rotation=0)   # Rotate y-axis labels for better readability

plt.tight_layout()
plt.show()
```

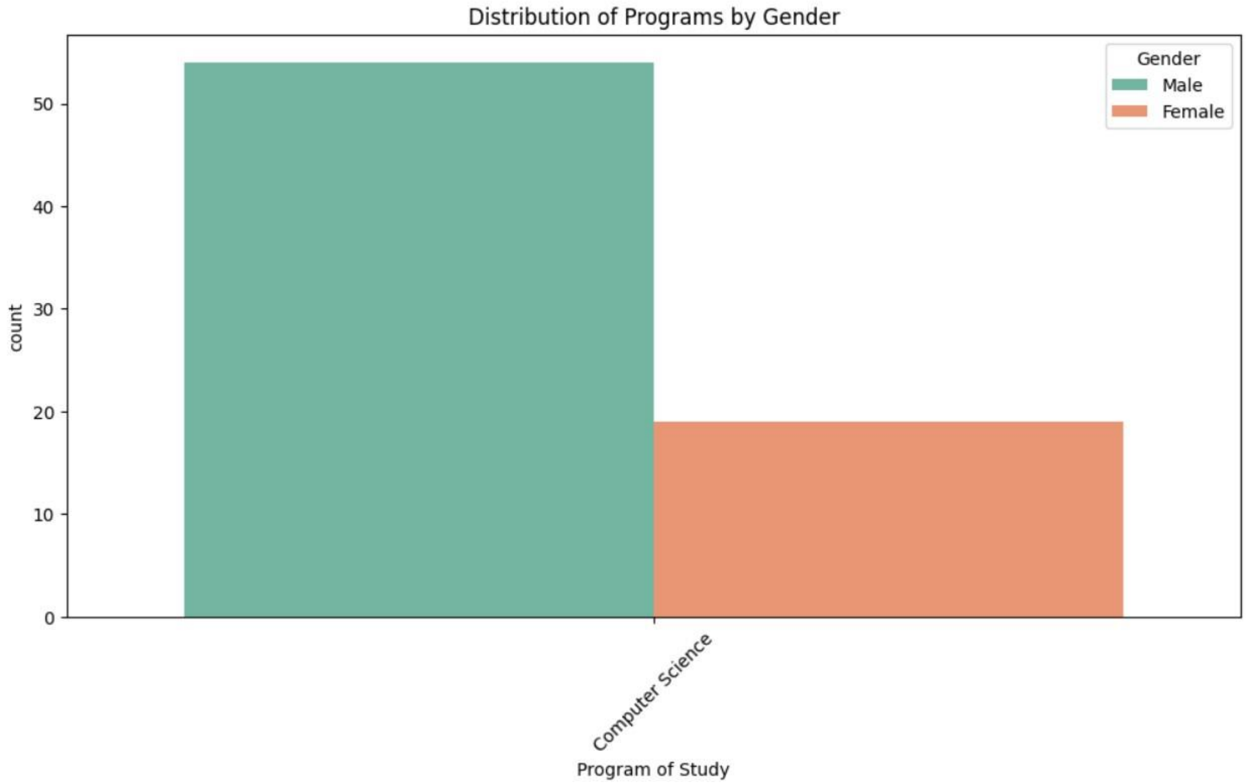
The heatmap shows that there is a positive correlation between the SGPA in all semesters. This means that students who tend to do well in one semester tend to do well in other semesters as well. The strongest correlation is between the SGPA in the fourth and fifth semesters, which is 0.88. This means that there is a very strong positive relationship between the SGPA in these two semesters.



Distribution of program by gender

```
plt.figure(figsize=(12, 6))
sns.countplot(data=df, x='Program of Study', hue='Gender', palette='Set2')
plt.title('Distribution of Programs by Gender')
plt.xticks(rotation=45)
plt.show()
```

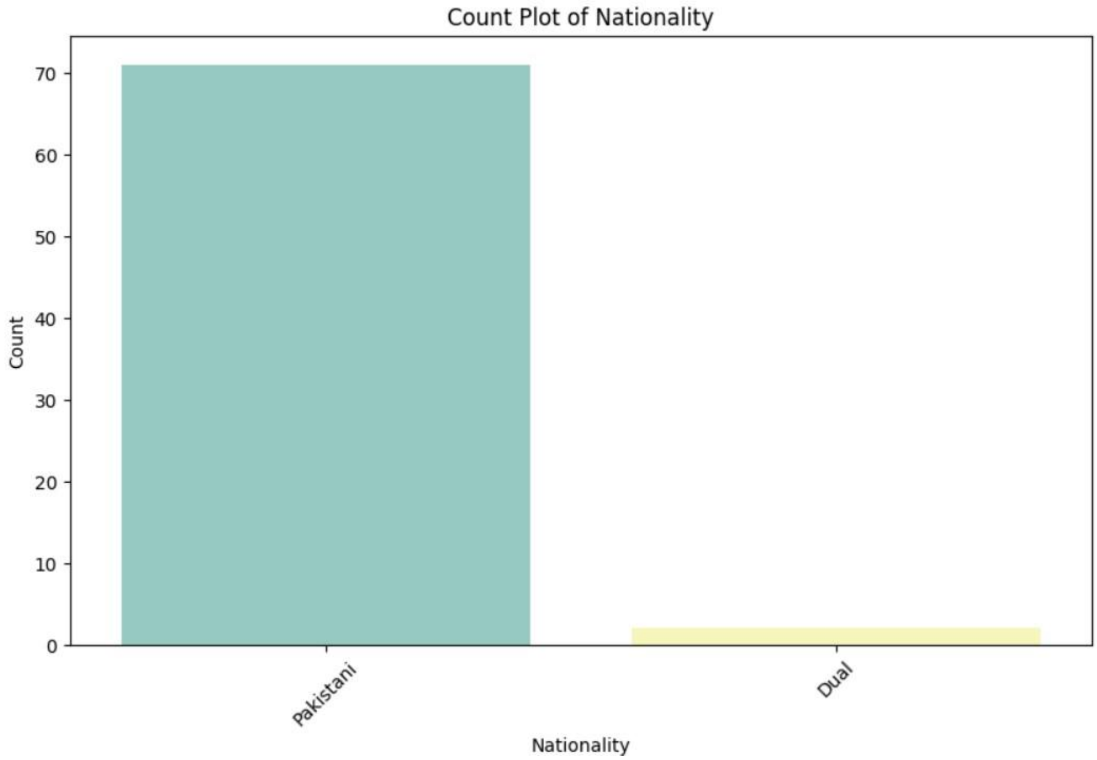
The distribution graph shows that the data is imbalanced and has more male class than female which may cause bias in the prediction.



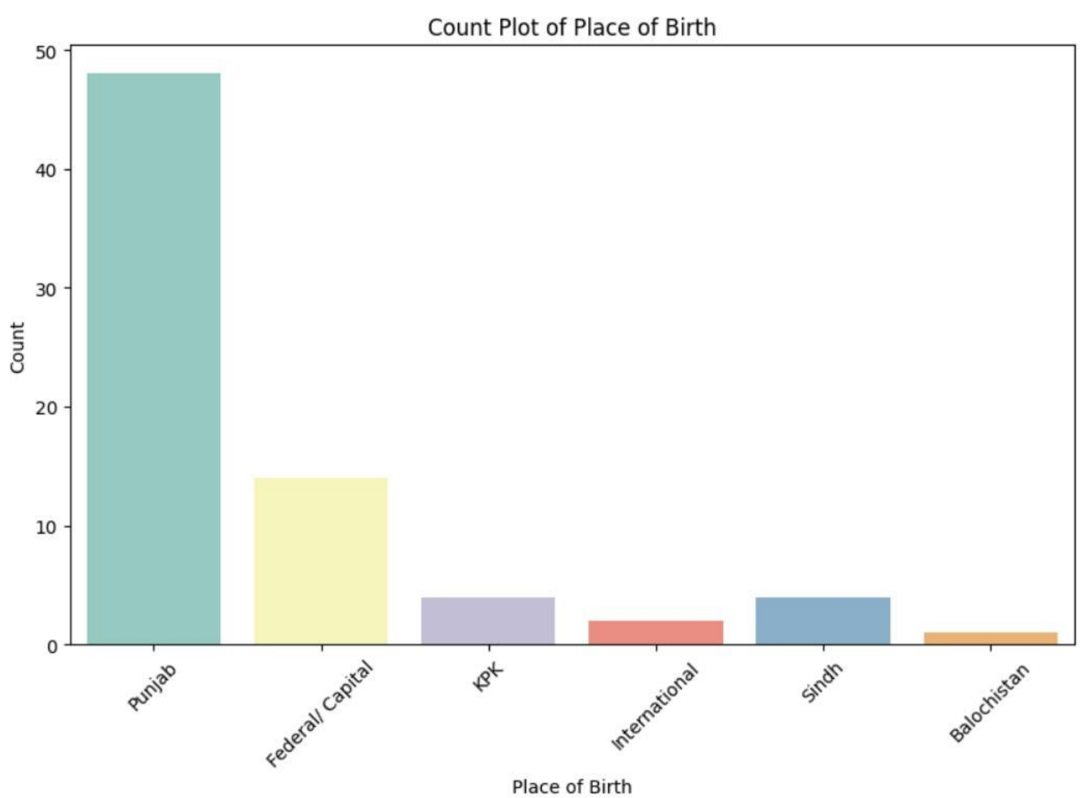
Scholarship availing by gender

```
cross_tab = pd.crosstab(df['Gender'], df['Availing any scholarship'])
cross_tab.plot(kind='bar', stacked=True, figsize=(10, 6), colormap='Set2')
plt.title('Scholarship Availment by Gender')
plt.xlabel('Gender')
plt.ylabel('Count')
plt.xticks(rotation=0)
plt.show()
```

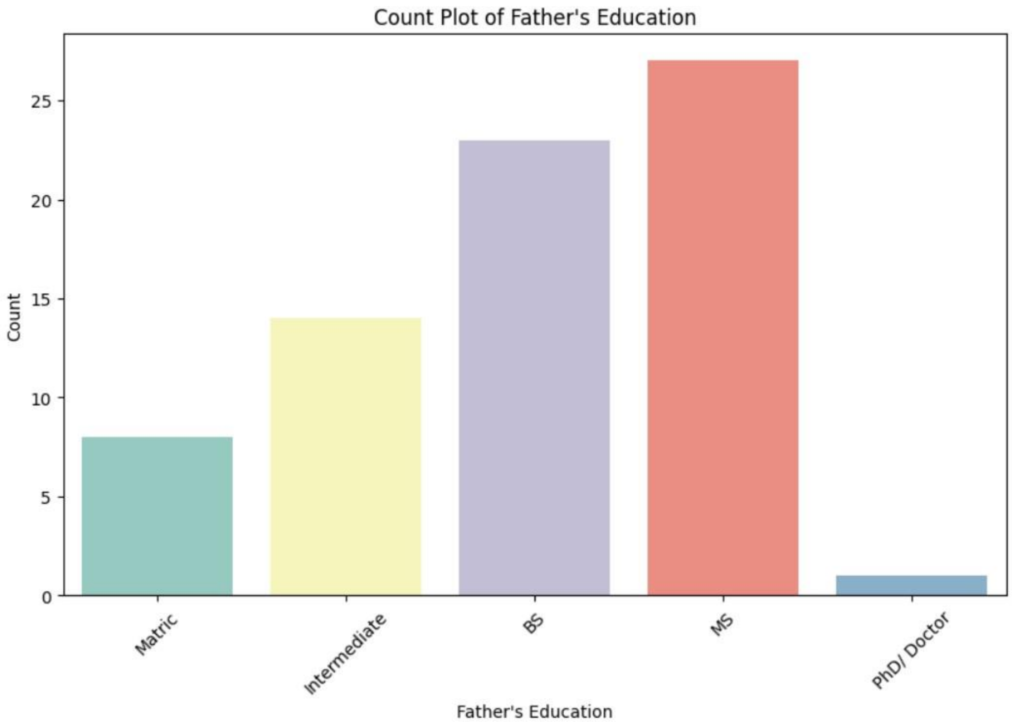
Count plot for Nationality



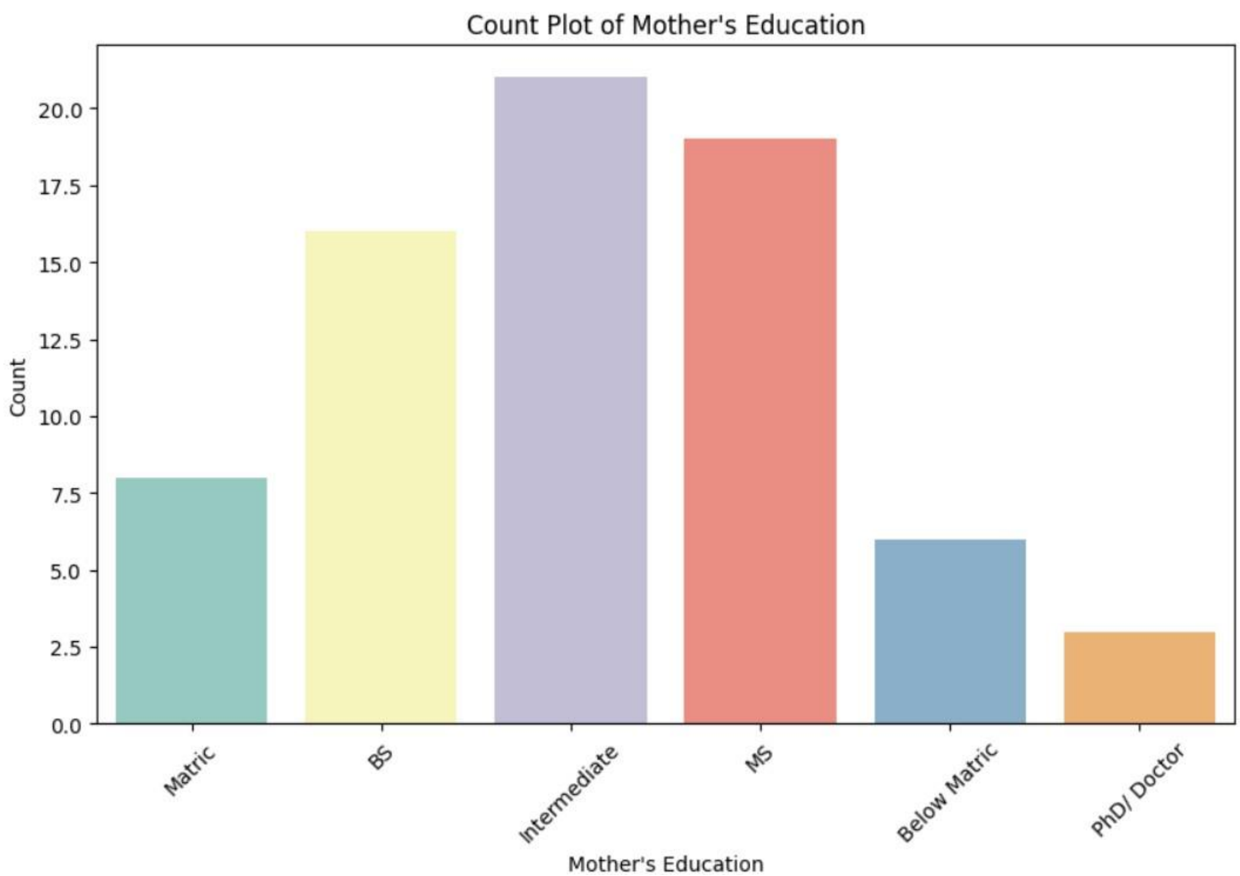
Students Areas



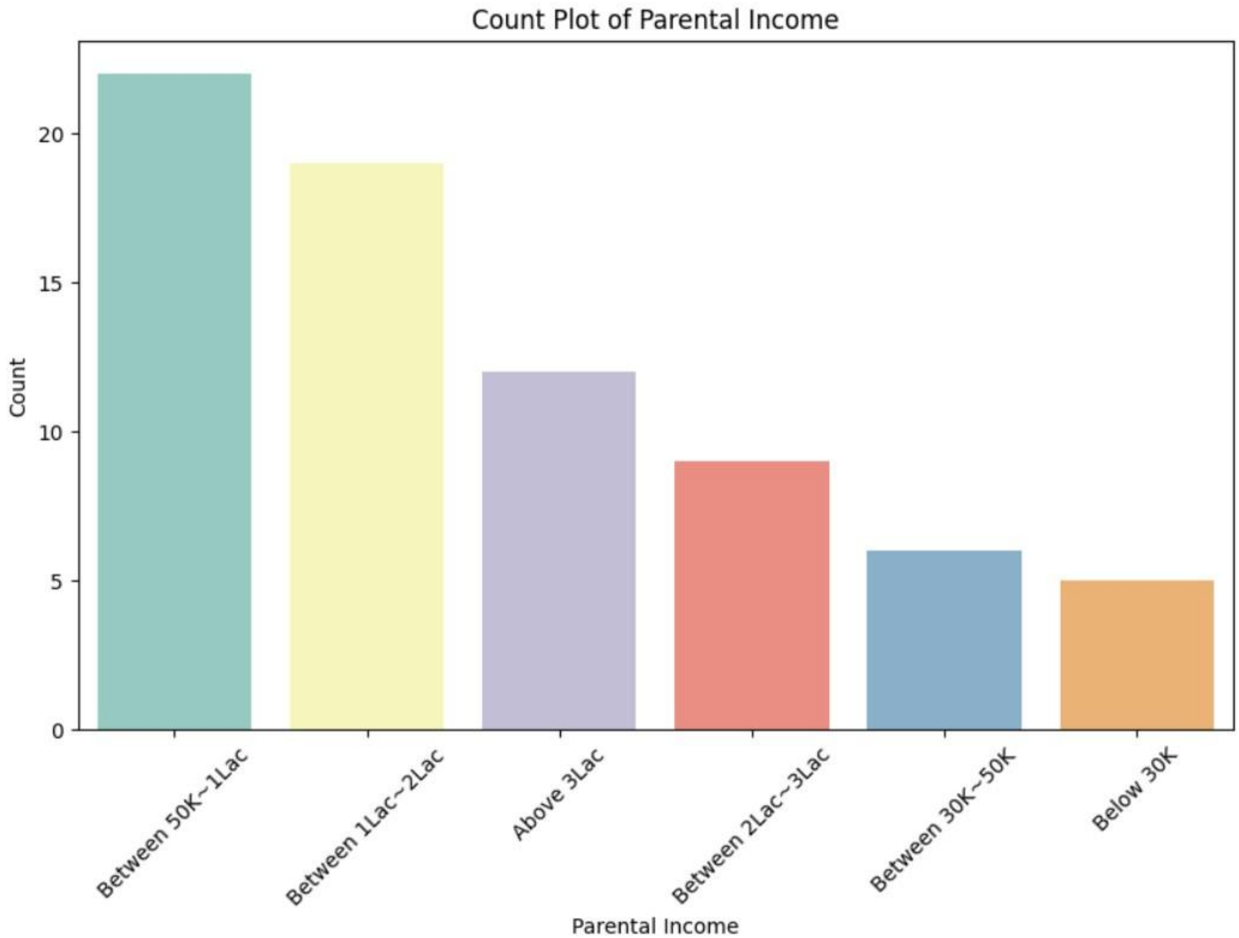
Students Father Education



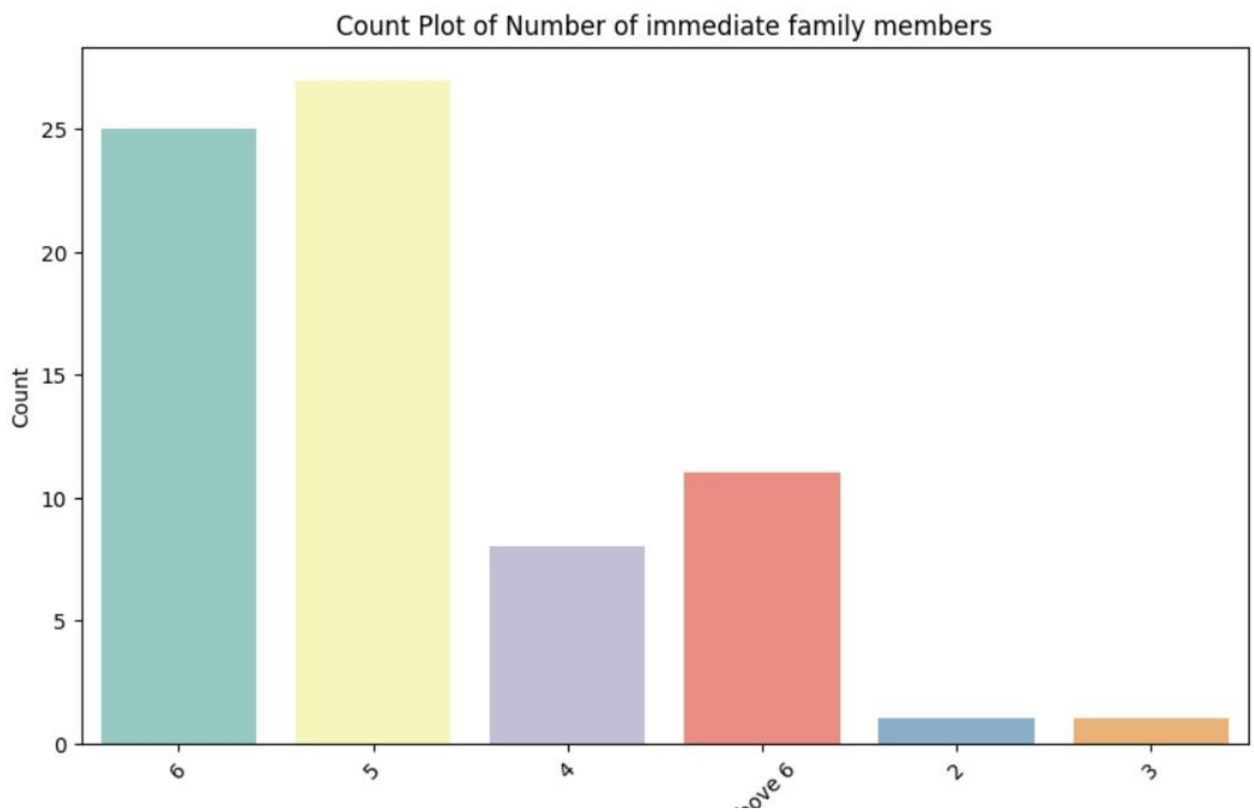
Mothers education of Students



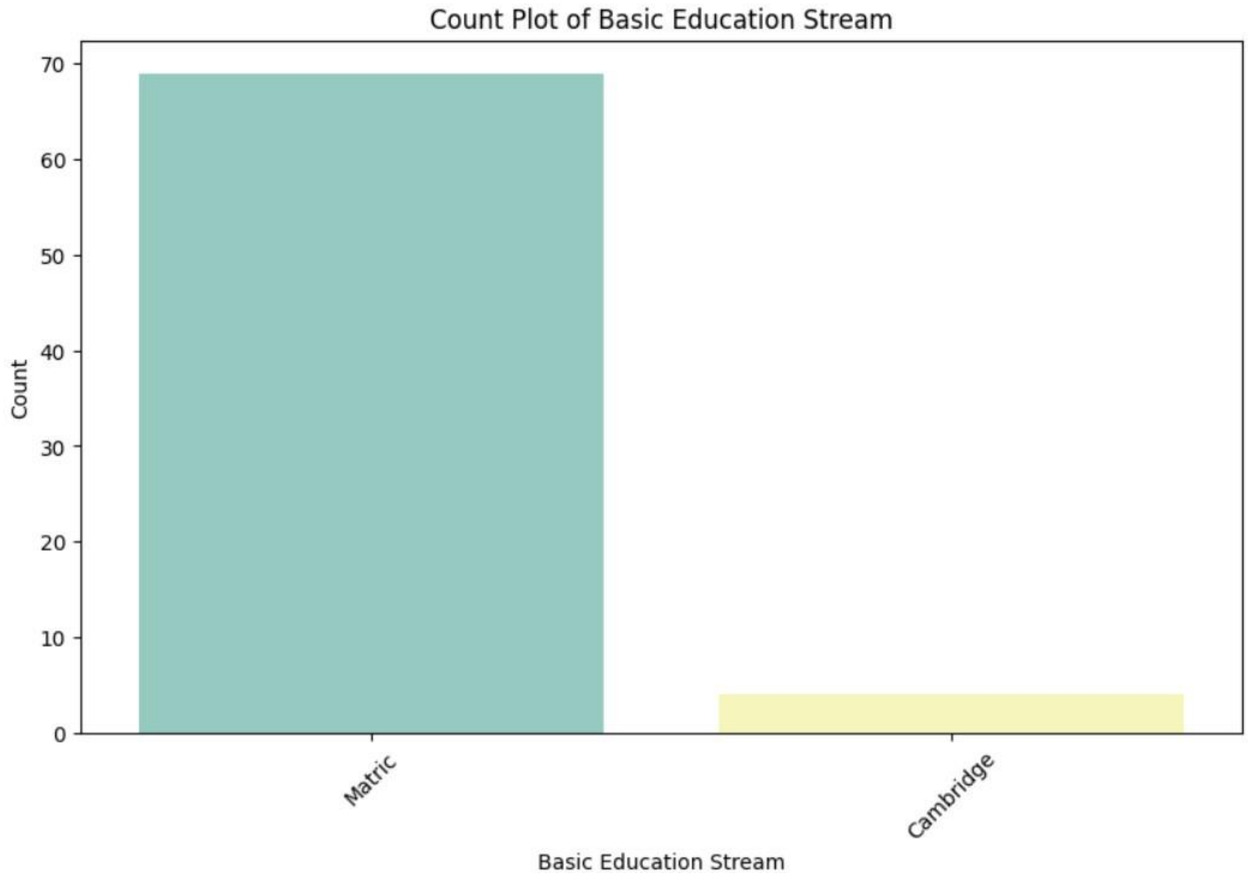
Students parallel income



Students family members count


Count Plot of Number of immediate family members
Count
0
5
10
15
20
25
6
5
4
ove 6
2
3

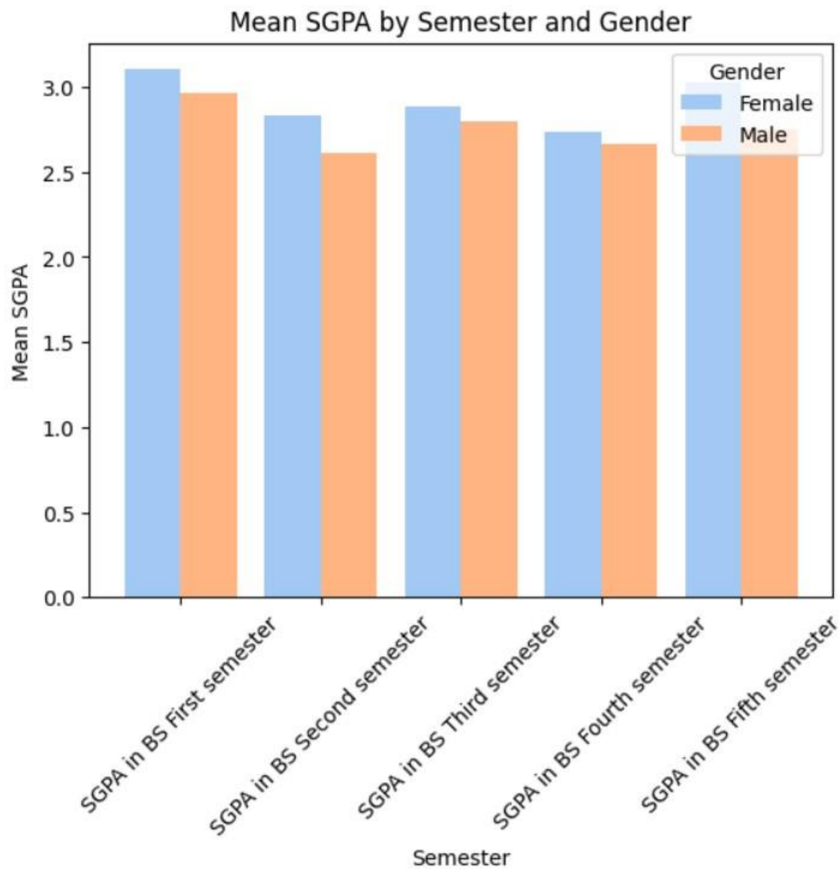
Education stream of students



Gender wise average GPSA till smester 5

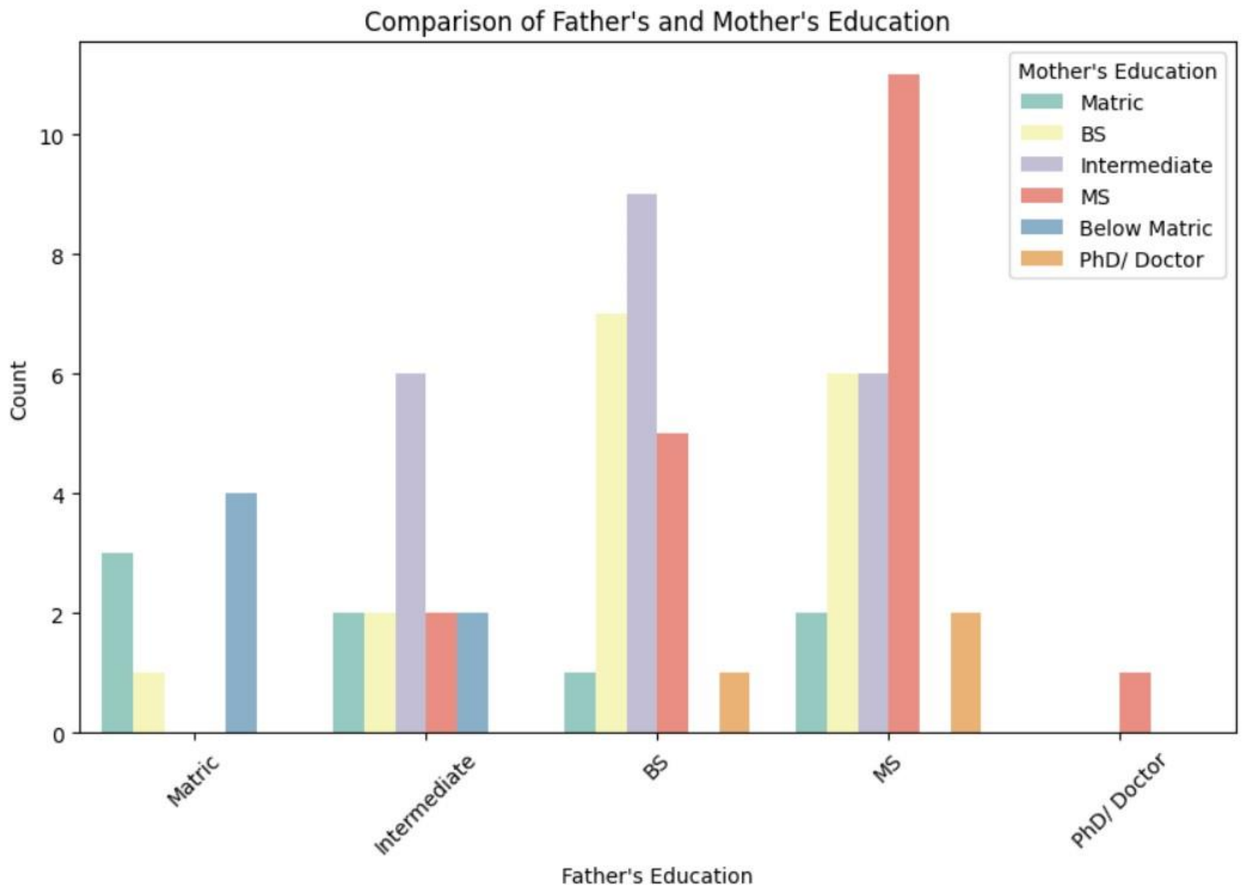
```
semester_columns = ['SGPA in BS First semester', 'SGPA in BS Second semester', 'SGPA in BS Third semester', 'SGPA in BS F

# Create an empty DataFrame to store the results
mean_gpas_by_semester = pd.DataFrame()

# Calculate the mean SGPA for each semester and store it in the DataFrame
for semester in semester_columns:
    mean_gpas_by_semester[semester] = df.groupby('Gender')[semester].mean()

# Display the mean GPAs for each semester by gender
print(mean_gpas_by_semester)
```

```
        SGPA in BS First semester  SGPA in BS Second semester  \
Gender
Female                   3.104737                    2.830000
Male                     2.969615                    2.612222

        SGPA in BS Third semester  SGPA in BS Fourth semester  \
Gender
Female                   2.885263                    2.732105
Male                     2.801111                    2.662037

        SGPA in BS Fifth semester
Gender
Female                   3.024211
Male                     2.755926
```



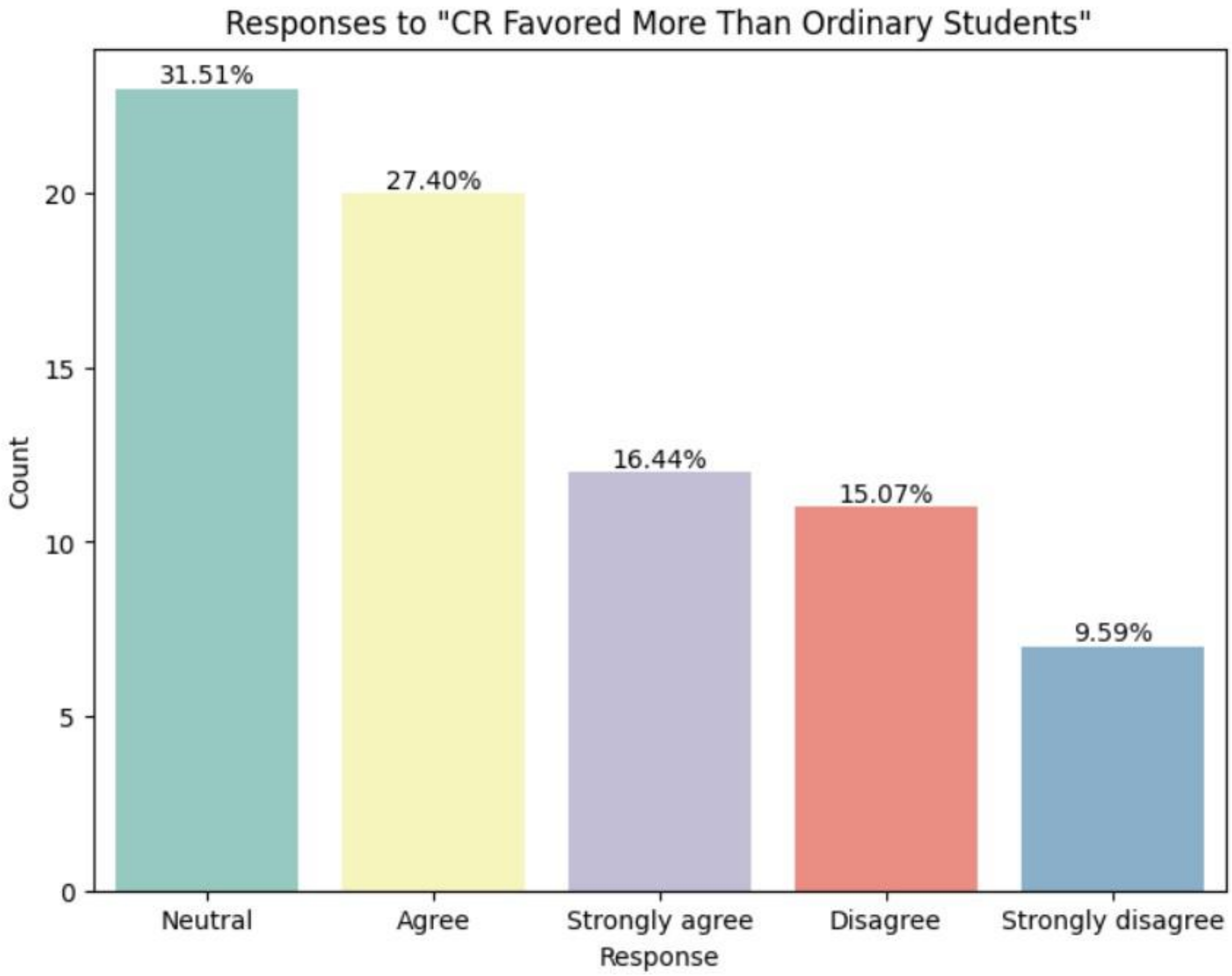
Comparison of mother and father Education

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

# Select the columns for comparison
education_comparison = df[['Father\'s Education', 'Mother\'s Education']]

# Create a count plot to compare father and mother education
plt.figure(figsize=(10, 6))
sns.countplot(data=education_comparison, x='Father\'s Education', hue='Mother\'s Education', palette='Set3')
plt.xlabel('Father\'s Education')
plt.ylabel('Count')
plt.title('Comparison of Father\'s and Mother\'s Education')
plt.xticks(rotation=45)
plt.legend(title='Mother\'s Education', loc='upper right')
plt.show()
```



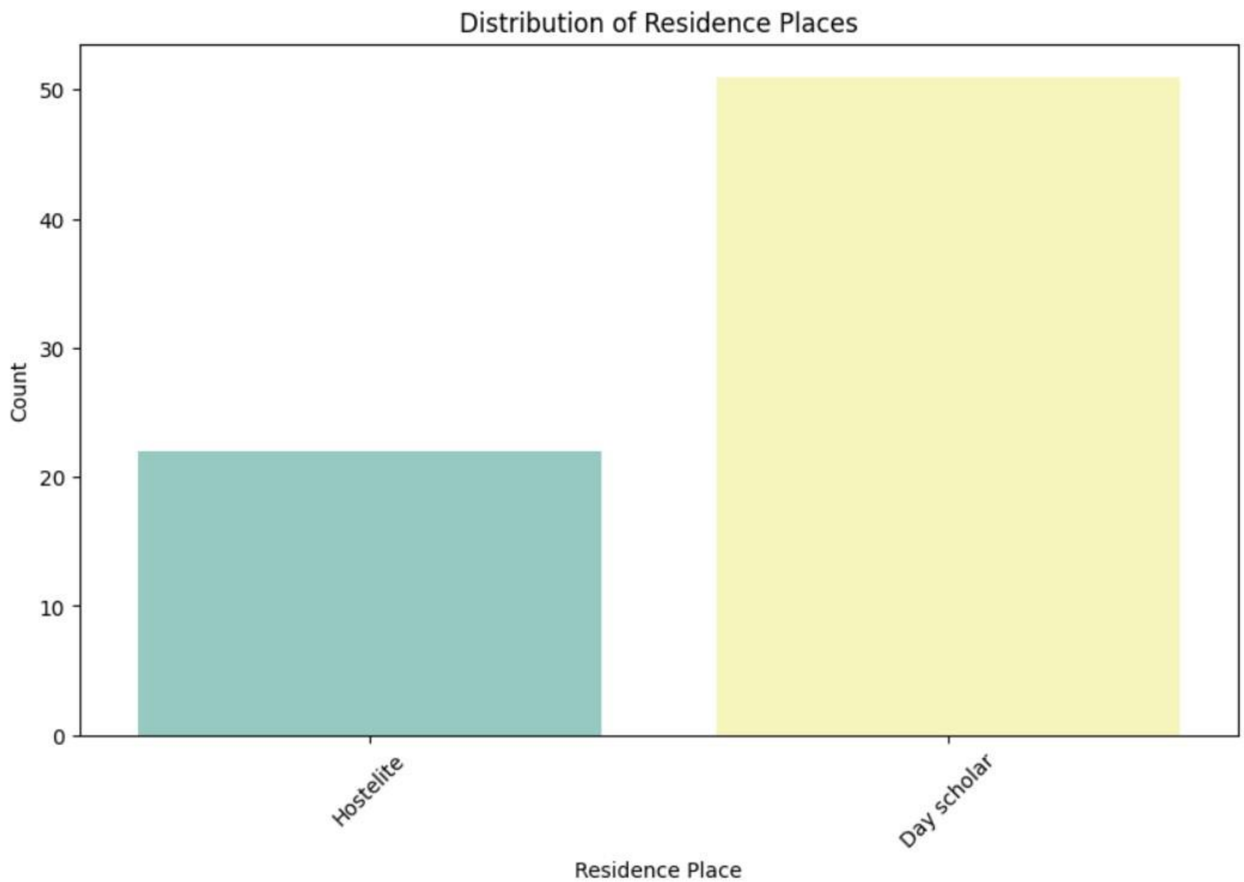
Opinion of students about CR

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

# Count the responses in the column
counts = df["Class Representative (CR) is favored more than any ordinary student as he/ she is in more contact with teachers."].value_counts()

# Create a bar plot
plt.figure(figsize=(8, 6))
sns.barplot(x=counts.index, y=counts, palette='Set3')

# Display percentages on top of the bars
total_count = len(df)
for i, count in enumerate(counts):
    plt.text(i, count, f'{count/total_count*100:.2f}%', ha='center', va='bottom')

plt.xlabel('Response')
plt.ylabel('Count')
plt.title('Responses to "CR Favored More Than Ordinary Students"')
plt.xticks(rotation=0)
plt.show()
```



Students in hostel and day scholars

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

# Create a count plot for "What is your residence place?"
plt.figure(figsize=(10, 6))
sns.countplot(data=df, x='What is your residence place?', palette='Set3')
plt.xlabel('Residence Place')
plt.ylabel('Count')
plt.title('Distribution of Residence Places')
plt.xticks(rotation=45)
plt.show()
```



4.4: Preprocessing

Preprocessing

Step 1: Data Loading

```python
import pandas as pd

# Load the dataset
file_path = r'C:\Users\HP\Downloads\assignment3\Final Data Set.csv'
data = pd.read_csv(file_path)

# Display the first few rows of the dataset
data.head()
```

Python

1. Loading the data to the vscode file in python.
2. Getting head values of data in the csv.

	ID No.	Program of Study	Gender	Nationality	Place of Birth	Father's Education	Mother's Education	Parental Income	Number of immediate family members	Any close family member with the same profession available for guidance	...	My overall mobile usage (daily) for non-academic purpose is limited to:	Identify any other variable that has negatively impacted on your academic performance.
0	33	Computer Science	Male	Pakistani	Punjab	Matric	Matric	Between 50K~1Lac	6	YES	...	7~8 hours	Teachers not in good 'Mood' gives marks withou...
1	32	Computer Science	Male	Pakistani	Federal/ Capital	Intermediate	BS	Between 1Lac~2Lac	5	YES	...	7~8 hours	Financial Issues
2	9	Computer Science	Male	Pakistani	Federal/ Capital	BS	Intermediate	Between 50K~1Lac	5	NO	...	More than 8 hours	NIL
3	17	Computer Science	Male	Pakistani	Punjab	Intermediate	Intermediate	Between 50K~1Lac	6	NO	...	More than 8 hours	not as such everything is covered in the quest...
4	12	Computer Science	Male	Pakistani	Punjab	MS	Intermediate	Above 3Lac	5	NO	...	More than 8 hours	Nothing more.

3. Removing null values from the dataset in categorical columns.

Step 3: Handling Null Values in Categorical Columns

```python
# Handling null values in categorical columns
data_clean_filled = data_clean.fillna('Unknown')
```

4. Label encode the data so that it is easier to process using models.

Step 4: Label Encoding

```python
from sklearn.preprocessing import LabelEncoder

# Label Encoding
label_encoder = LabelEncoder()
for column in categorical_columns:
    data_clean_filled[column] = label_encoder.fit_transform(data_clean_filled[column])
```

5. Standardizing the data so that it is in a range from 0 to 1 and is consistent with the models.

Step 5: Standardization

```python
from sklearn.preprocessing import StandardScaler

# Convert 'Unknown' placeholders to a numeric value (-1)
data_clean_numeric = data_clean_filled.replace('Unknown', -1)

# Identifying features and target variables
target_variables = ['SGPA in BS Fifth semester', 'CGPA in BS Fifth semester']
features = data_clean_numeric.drop(columns=target_variables)

# Standardizing the features
scaler = StandardScaler()
standardized_features = scaler.fit_transform(features)

# Converting the standardized features back to a DataFrame
standardized_features_df = pd.DataFrame(standardized_features, columns=features.columns)

# Reattaching the target variables
preprocessed_data = pd.concat([standardized_features_df, data_clean_numeric[target_variables]], axis=1)
```

6. Saving the preprocessed data to a new csv file to keep changes saved.

saving new CSV file

```python
# Saving the preprocessed data to a new CSV file in the current working directory
output_file_name = 'Preprocessed_Data_Set.csv'
preprocessed_data.to_csv(output_file_name, index=False)

# Output the path for confirmation
print("File saved as:", output_file_name)
```

[72] Python

```
File saved as: Preprocessed_Data_Set.csv
```

+ Code + Markdown

Applying Models(GoTo model.ipynb)

4.4: Model development

The code imports libraries for data manipulation and analysis, loads a preprocessed dataset, splits it into training and testing sets, and defines diverse regression models to compare for predicting fifth-semester grades based on earlier performance. Essentially, it sets the stage for training and evaluating different models to find the best one for grade prediction.

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.svm import SVR
from sklearn.metrics import mean_squared_error, r2_score
import joblib

# Load the preprocessed dataset
file_path = "C:\\Users\\HP\\Downloads\\assignment3\\Processed_Data_Set.csv"
preprocessed_data = pd.read_csv(file_path)

# Splitting the data into features and target variables
features = ['Matric percentage', 'Intermediate percentage',
            'SGPA in BS First semester', 'SGPA in BS Second semester',
            'SGPA in BS Third semester', 'SGPA in BS Fourth semester']

X = preprocessed_data[features]
y_sgpa = preprocessed_data['SGPA in BS Fifth semester']
y_cgpa = preprocessed_data['CGPA in BS Fifth semester']

# Split the data into training and testing sets (80% train, 20% test)
X_train, X_test, y_sgpa_train, y_sgpa_test, y_cgpa_train, y_cgpa_test = train_test_split(
    X, y_sgpa, y_cgpa, test_size=0.2, random_state=42)

# Model selection
models = {
    'Linear Regression': LinearRegression(),
    'Ridge Regression': Ridge(),
    'Lasso Regression': Lasso(),
    'Decision Tree': DecisionTreeRegressor(),
    'Random Forest': RandomForestRegressor(),
    'Support Vector Regression': SVR()
}
```

4.5: Model Training

This code is like a chef preparing various dishes—different machine learning models—to predict fifth grades in a BS program. It cycles through popular recipes like linear regression, decision trees, and random forests. For each "dish," it uses past semesters' grades as ingredients and trains the model to predict both "SGPA" and "CGPA" on unseen data. Then, it meticulously measures the taste (accuracy) of each dish with metrics like R2 and MSE. Finally, it neatly arranges all the flavors (performance numbers) for easy comparison and serves up the most delicious—the best performing model—for future grade predictions.

```python
# Training and evaluation
results = []

for model_name, model in models.items():
    # Train for SGPA
    model.fit(X_train, y_sgpa_train)
    sgpa_predictions = model.predict(X_test)
    sgpa_mse = mean_squared_error(y_sgpa_test, sgpa_predictions)
    sgpa_r2 = r2_score(y_sgpa_test, sgpa_predictions)

    # Train for CGPA
    model.fit(X_train, y_cgpa_train)
    cgpa_predictions = model.predict(X_test)
    cgpa_mse = mean_squared_error(y_cgpa_test, cgpa_predictions)
    cgpa_r2 = r2_score(y_cgpa_test, cgpa_predictions)

    # Store results for each model
    results.append({
        'Model': model_name,
        'SGPA MSE': sgpa_mse,
        'SGPA R2': sgpa_r2,
        'CGPA MSE': cgpa_mse,
        'CGPA R2': cgpa_r2
    })

    # Save each model to a joblib file
    joblib.dump(model, f"{model_name.replace(' ', '_')}_model.joblib")

# Displaying the results
results_df = pd.DataFrame(results)
print(results_df)
```

✓ 6.1s

4.6: Training Results

Below are the results that our multiple models have given along with their evaluation metrics.

```
✓ 6.1s

                     Model  SGPA MSE   SGPA R2  CGPA MSE   CGPA R2
0        Linear Regression  0.055989  0.798984  0.807278  0.319302
1         Ridge Regression  0.055999  0.798950  0.807010  0.319528
2         Lasso Regression  0.361542 -0.298028  1.187117 -0.000979
3            Decision Tree  0.122033  0.561869  1.096620  0.075328
4            Random Forest  0.092558  0.667692  1.034075  0.128066
5 Support Vector Regression  0.130602  0.531106  0.900969  0.240301
```

Random Forest has the highest R2 score for both SGPA and CGPA prediction, suggesting that it explains the most variance in the target variables. However, it also has the highest MSE score for CGPA prediction, which means that there is more error between the predicted and actual values for CGPA.

Linear Regression has the lowest MSE score for CGPA prediction, which means that it has the least error on average. However, it also has the lowest R2 score for both SGPA and CGPA prediction, which means that it explains less of the variance in the target variables.

Ultimately, the best model for you will depend on your specific priorities. If you are more concerned about minimizing error, then Linear Regression might be the best choice for CGPA prediction.

4.7: Visual Interface and Prediction

Developing an intuitive web application that enables people to engage with your machine learning models is a necessary step in deploying Python models to Streamlit. A Python package called Streamlit makes it easier to create web applications for data science and machine learning. You must import your trained models, preprocess input data, and utilise Streamlit's straightforward API to develop an easy-to-use user interface before you can publish Python models to Streamlit. Through the UI, users may enter data, and the app will use your algorithms to provide real-time predictions and insights. When your app is finished, you may host it on other servers such as AWS, Heroku, or Streamlit Sharing, which will allow more people to access your models without having to install dependencies or write code. By improving model usability and democratising data science, this deployment strategy makes data science more approachable to non-technical stakeholders and end users.

```python
import streamlit as st
import joblib
import numpy as np
import pandas as pd

# Load the saved models
models = {
    'Linear Regression': joblib.load('Linear_Regression_model.joblib'),
    'Ridge Regression': joblib.load('Ridge_Regression_model.joblib'),
    'Lasso Regression': joblib.load('Lasso_Regression_model.joblib'),
    'Decision Tree': joblib.load('Decision_Tree_model.joblib'),
    'Random Forest': joblib.load('Random_Forest_model.joblib'),
    'Support Vector Regression': joblib.load('Support_Vector_Regression_model.joblib')
}

# Function to provide performance comments based on GPA
def get_performance_comment(gpa):
    if 3.51 <= gpa <= 4.00:
        return 'Extraordinary Performance'
    elif 3.00 <= gpa < 3.51:
        return 'Very Good Performance'
    elif 2.51 <= gpa < 3.00:
        return 'Good Performance'
    elif 2.00 <= gpa < 2.51:
        return 'Satisfactory Performance'
    elif 1.00 <= gpa < 2.00:
        return 'Poor Performance'
    elif 0.00 <= gpa < 1.00:
        return 'Very Poor Performance'
    else:
        return 'Invalid GPA'

# Streamlit app
st.title('SGPA and CGPA Predictor')

model_names = list(models.keys())
selected_model = st.selectbox('Select a model for prediction:', model_names)
```

```python
# Streamlit app
st.title('SGPA and CGPA Predictor')

model_names = list(models.keys())
selected_model = st.selectbox('Select a model for prediction:', model_names)

st.sidebar.title('Enter Input Data:')

# Input fields for the required data
input_matric_percentage = st.sidebar.number_input('Matric Percentage:', min_value=0.0, max_value=100.0, step=0.1)
input_intermediate_percentage = st.sidebar.number_input('Intermediate Percentage:', min_value=0.0, max_value=100.0, step=0.1)
input_sgpa_1st_sem = st.sidebar.number_input('SGPA in BS First Semester:', min_value=0.0, max_value=4.0, step=0.01)
input_sgpa_2nd_sem = st.sidebar.number_input('SGPA in BS Second Semester:', min_value=0.0, max_value=4.0, step=0.01)
input_sgpa_3rd_sem = st.sidebar.number_input('SGPA in BS Third Semester:', min_value=0.0, max_value=4.0, step=0.01)
input_sgpa_4th_sem = st.sidebar.number_input('SGPA in BS Fourth Semester:', min_value=0.0, max_value=4.0, step=0.01)

# Button to predict
if st.sidebar.button('Predict'):
    model = models[selected_model]

    # Creating a DataFrame for the input data
    input_data = pd.DataFrame(
        [[input_matric_percentage, input_intermediate_percentage, input_sgpa_1st_sem,
          input_sgpa_2nd_sem, input_sgpa_3rd_sem, input_sgpa_4th_sem]],
        columns=["Matric percentage", "Intermediate percentage",
                 "SGPA in BS First semester", "SGPA in BS Second semester",
                 "SGPA in BS Third semester", "SGPA in BS Fourth semester"]
    )

    # Make predictions for 5th semester SGPA
    sgpa_prediction_5th_sem = model.predict(input_data)[0]
```

The screen below is a screenshot of deployed models on streamlit interface. Each model can be run individually on input data and it gives accurate answers.