Report: Predictive Analysis for Academic Performance

Abstract:

This predictive assessment project aims to evaluate learning outcomes through a comprehensive analysis of various machine learning algorithms. The review covers critical issues in predictive analytics, including data preprocessing, exploratory data analysis (EDA), model development, data validation, and deployment.

The initial phase consists of careful data preprocessing to ensure data quality and accuracy. Subsequently, exploratory data analysis reveals meaningful insights into the relationships among the variables. In the core predictive modeling phase, comparative analysis is performed on at least 4-5 AI/ML algorithms. Successful data validation procedures are used to build predictions and emphasize the strength of the models . The final step is to develop a graphical user interface (GUI) that facilitates testing new data. Users can enter context, select the desired prediction model, and see the predicted results, specifically focusing on SGPA and CGPA for the 5th grade.

Introduction:

In higher education, understanding the factors that influence academic performance is of paramount importance. Academic institutions continually seek to enhance educational experiences and provide targeted support to students, fostering an environment conducive to success. Predictive analysis emerges as a powerful tool in this pursuit, offering insights into the drivers of academic achievement and enabling proactive measures to be taken.

This predictive analysis for academic performance delves into the intricate relationship between various factors and the academic outcomes of students, with a particular focus on predicting SGPA (Semester Grade Point Average) and CGPA (Cumulative Grade Point Average) for the 5th semester. By leveraging data-driven methodologies, this study aims to unearth patterns, trends, and potential predictors that can significantly impact a student's performance.

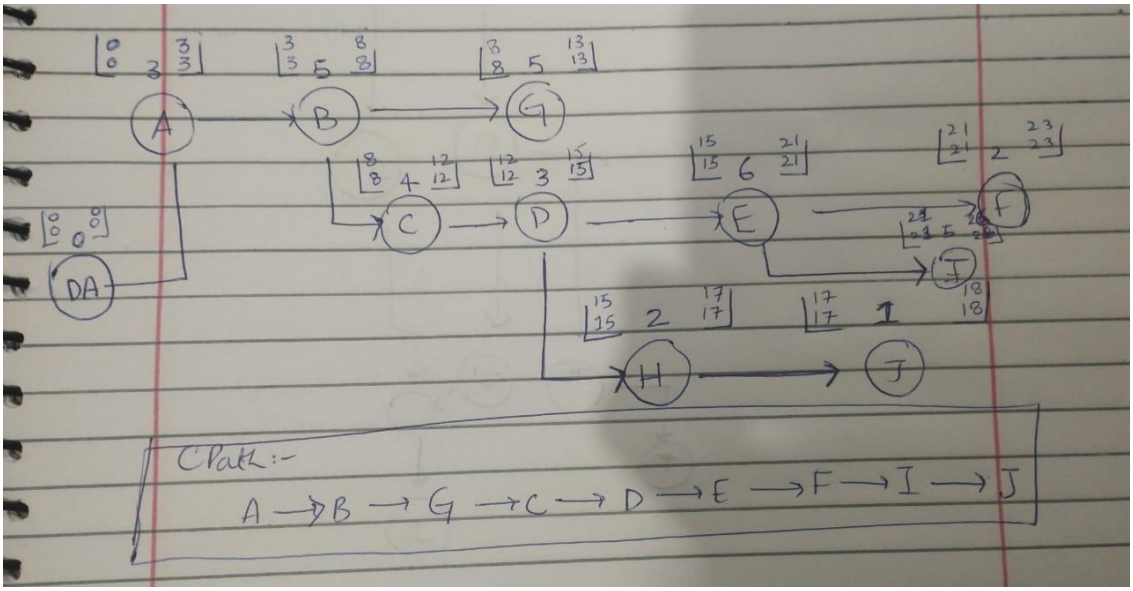
The methodology involves a systematic approach to data cleaning, including the removal of irrelevant and subjective information, meticulous error handling to ensure data integrity, and the identification and elimination of duplicate entries. With a cleaned and refined dataset, the subsequent steps will explore the presence of outliers, further enhancing the robustness of the predictive model.

Activity On Node Diagram:

Activity	Duration (Days)	Immediate Predecessor Activities
A: Data Collection	3	-
B: Data Preprocessing	5	A
C: EDA - Part 1	4	B
D: EDA - Part 2	3	C
E: Model Training	6	C
F: Model Evaluation	2	E
G: GUI Development	5	B
H: Intermediate Report	2	D, G
I: Final Documentation	3	F, H
J: Submission	1	I
Dummy Activity 1	0	C, D

Project Tasks:

A: Data Collection (3 days)
B: Data Preprocessing (5 days, follows A)
C: EDA - Part 1 (4 days, follows B)
D: EDA - Part 2 (3 days, follows C)
E: Model Training (6 days, follows C)
F: Model Evaluation (2 days, follows E)
G: GUI Development (5 days, can start after B)
H: Intermediate Report (2 days, follows D and G)
I: Final Documentation (3 days, follows F and H)
J: Submission (1 day, follows I)
Dummy Activity 1 (0 days, to synchronize EDA parts)



Critical Path :

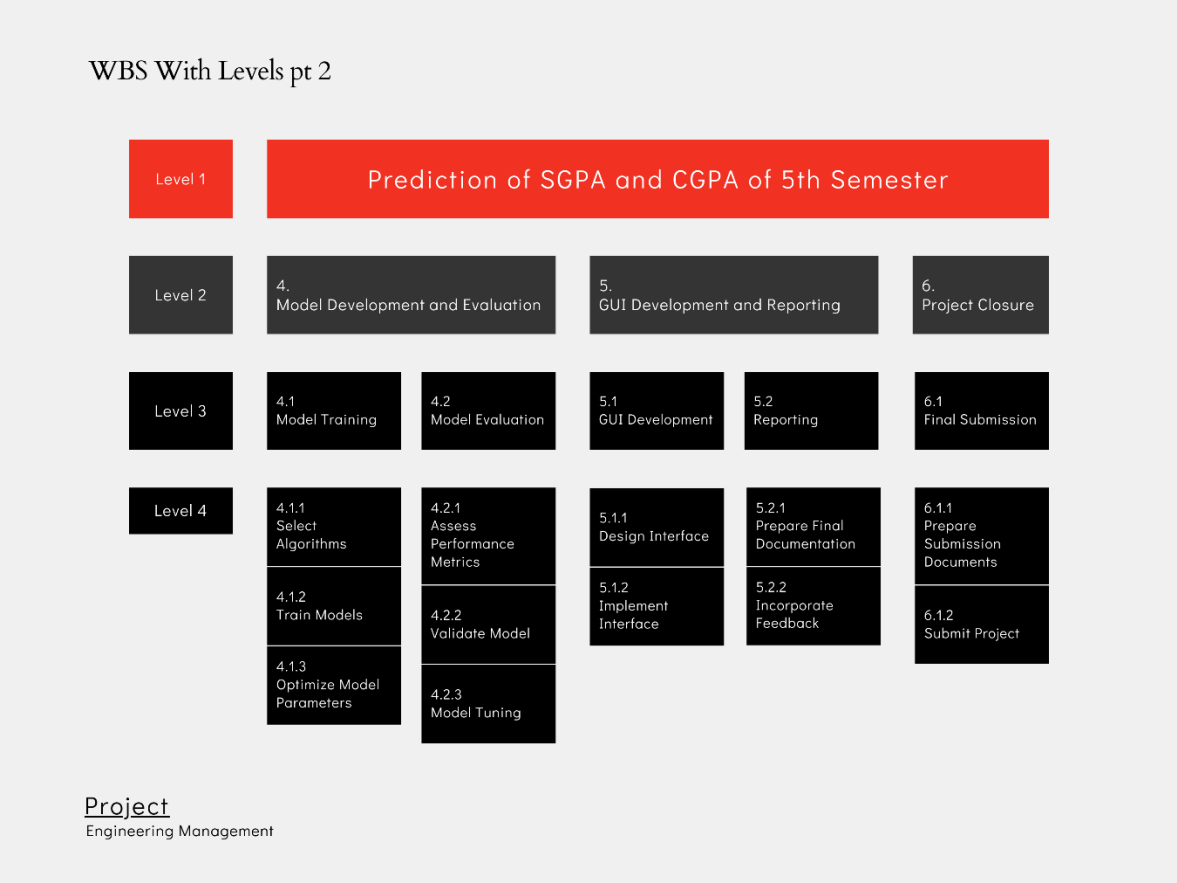
A→B→G→C→D→E→F→I→J

Work Break Structure:

WBS With Levels

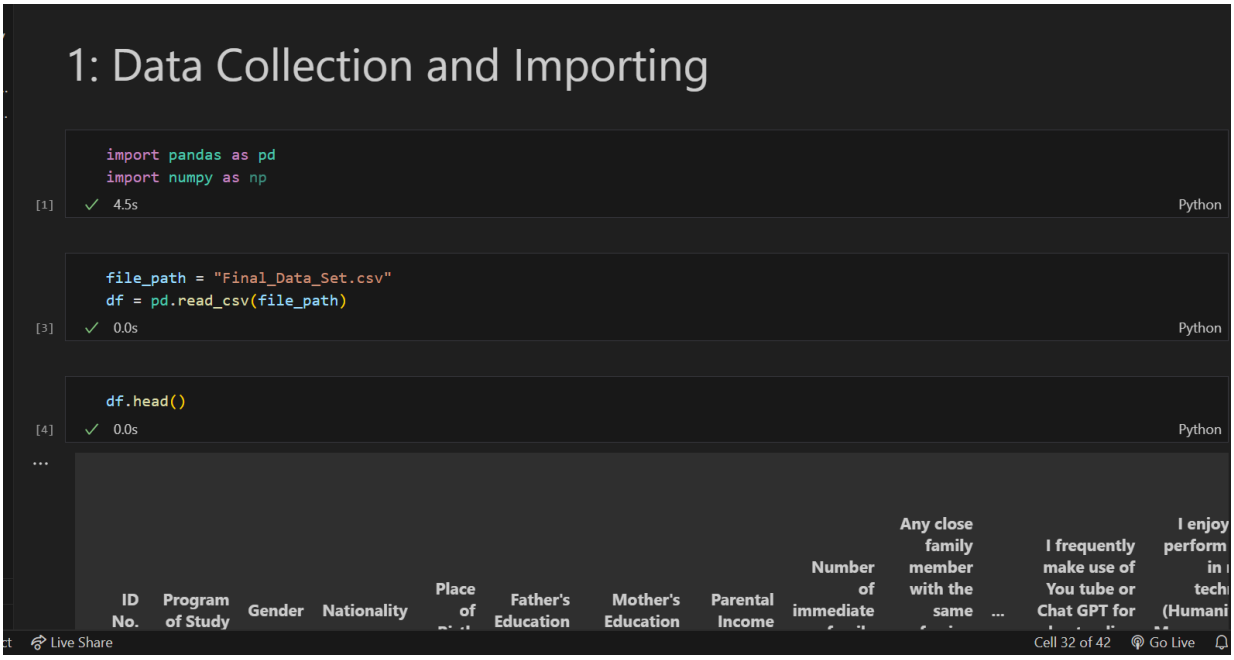
Level 1	Prediction of SGPA and CGPA of 5th Semester			
Level 2	1. Project Management	2. Data Handling	3. Exploratory Data Analysis (EDA)	
Level 3	1.1 Initiation and Planning	1.2 Monitoring and Controlling	2.1 Data Collection	2.2 Data Preprocessing	3.1 EDA - Part 1	3.2 EDA - Part 2
Level 4	1.1.1 Define Project Scope	1.2.1 Track Project Progress	2.1.1 Identifying Data Sources	2.2.1 Clean Data	3.1.1 Conduct Preliminary Analysis	3.2.1 Perform Advanced Analysis
	1.1.2 Develop Project Plan	1.2.2 Adjust Plans as Needed	2.1.2 Collect Data	2.2.2 Transform Data	3.1.2 Visualize Data	3.2.2 Interpret Results
	1.1.3 Identify Stakeholders				3.1.3 Identify Patterns	

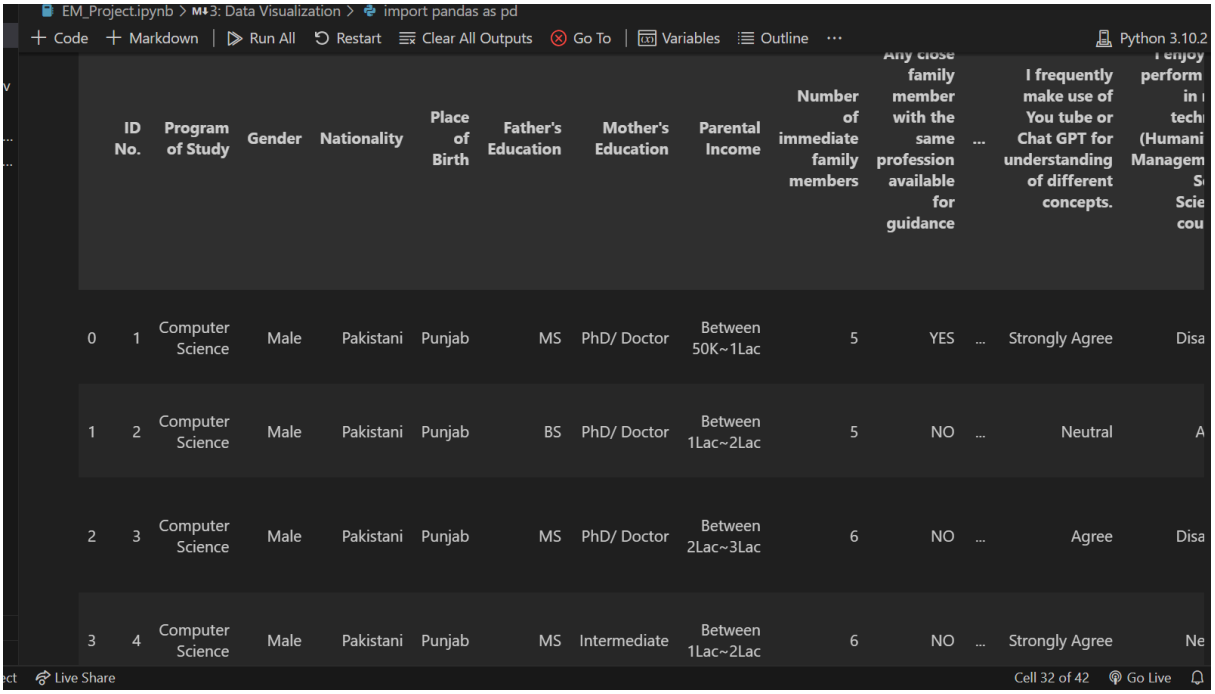
Project
Engineering Management



Data Preprocessing:

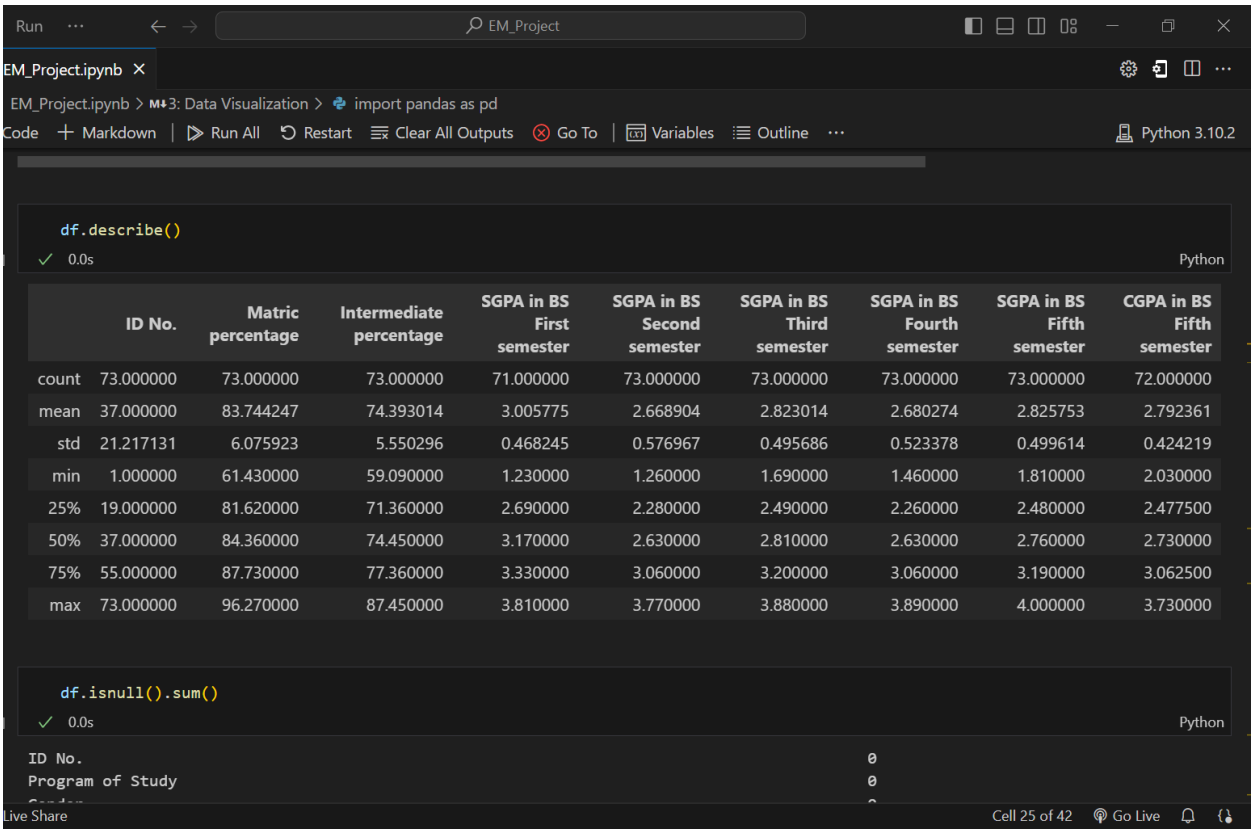
Initially, data cleaning is the start of the process For, that we first load the dataset.





	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	...	I frequently make use of You tube or Chat GPT for understanding of different concepts.	I enjoy perform in tech (Humani Managem Scie cou
0	1	Computer Science	Male	Pakistani	Punjab	MS	PhD/ Doctor	Between 50K~1Lac	5	YES	...	Strongly Agree	Disa
1	2	Computer Science	Male	Pakistani	Punjab	BS	PhD/ Doctor	Between 1Lac~2Lac	5	NO	...	Neutral	A
2	3	Computer Science	Male	Pakistani	Punjab	MS	PhD/ Doctor	Between 2Lac~3Lac	6	NO	...	Agree	Disa
3	4	Computer Science	Male	Pakistani	Punjab	MS	Intermediate	Between 1Lac~2Lac	6	NO	...	Strongly Agree	Ne

In Semester 5, columns deemed not significant for predicting academic achievement were identified and removed. These lines contained ideas about personal and individual preferences, which were not thought to be relevant to the mainstream research. Error handling was used to ensure robustness in column name processing. This includes troubleshooting potential problems such as leading or trailing positions or hidden characters in character names, which can lead to parsing errors. Verified for missing values in the remaining lines of the DataFrame. Fortunately, no missing values were identified, eliminating the need for additional processing or imputation of missing data. Duplicate rows have been detected and removed from the DataFrame.



	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
count	73.000000	73.000000	73.000000	71.000000	73.000000	73.000000	73.000000	73.000000	72.000000
mean	37.000000	83.744247	74.393014	3.005775	2.668904	2.823014	2.680274	2.825753	2.792361
std	21.217131	6.075923	5.550296	0.468245	0.576967	0.495686	0.523378	0.499614	0.424219
min	1.000000	61.430000	59.090000	1.230000	1.260000	1.690000	1.460000	1.810000	2.030000
25%	19.000000	81.620000	71.360000	2.690000	2.280000	2.490000	2.260000	2.480000	2.477500
50%	37.000000	84.360000	74.450000	3.170000	2.630000	2.810000	2.630000	2.760000	2.730000
75%	55.000000	87.730000	77.360000	3.330000	3.060000	3.200000	3.060000	3.190000	3.062500
max	73.000000	96.270000	87.450000	3.810000	3.770000	3.880000	3.890000	4.000000	3.730000

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 73 entries, 0 to 72
Data columns (total 76 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
 15  Intermediate percentage
 16  SGPA in BS First semester
 17  SGPA in BS Second semester
 18  SGPA in BS Third semester
 19  SGPA in BS Fourth semester
```

This step was necessary to ensure that each student's data was unique and prevented distortions in subsequent analyses. Graphical techniques such as box plots or histograms were suggested to identify potential redundancies in the statistical scores. The decision to deal with outliers is left open, as the specific treatment will depend on domain knowledge and the quality of the data.

2.1: Handle missing values

```python
import pandas as pd
import numpy as np

# Define a list of columns with missing values
columns_with_missing = df.columns[df.isnull().any()]

# Impute missing values for numerical columns with mean
for col in columns_with_missing:
    if df[col].dtype == 'float64':
        df[col].fillna(df[col].mean(), inplace=True)

# Impute missing values for numerical columns with median
for col in columns_with_missing:
    if df[col].dtype == 'float64':
        df[col].fillna(df[col].median(), inplace=True)

# Impute missing values for categorical columns with mode
for col in columns_with_missing:
    if df[col].dtype == 'object':
        df[col].fillna(df[col].mode()[0], inplace=True)
```

2.2: Remove duplicate

```python
# Remove duplicate rows
df.drop_duplicates(inplace=True)
```

[10] ✓ 0.0s Python

2.3: Outlier detection and treatment

```python
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats import zscore

# Assuming df is your DataFrame
# Step 1: Visual Exploration
# Use box plots for numerical variables
numerical_columns = df.select_dtypes(include=['float64']).columns
plt.figure(figsize=(12, 8))
sns.boxplot(data=df[numerical_columns])
plt.title('Boxplot for Numerical Variables')
plt.show()
```

+ Code + Markdown

```python
z_scores = zscore(df[numerical_columns])
abs_z_scores = np.abs(z_scores)
threshold = 3  # Adjust this threshold as needed
outliers = df[abs_z_scores > threshold].index
```

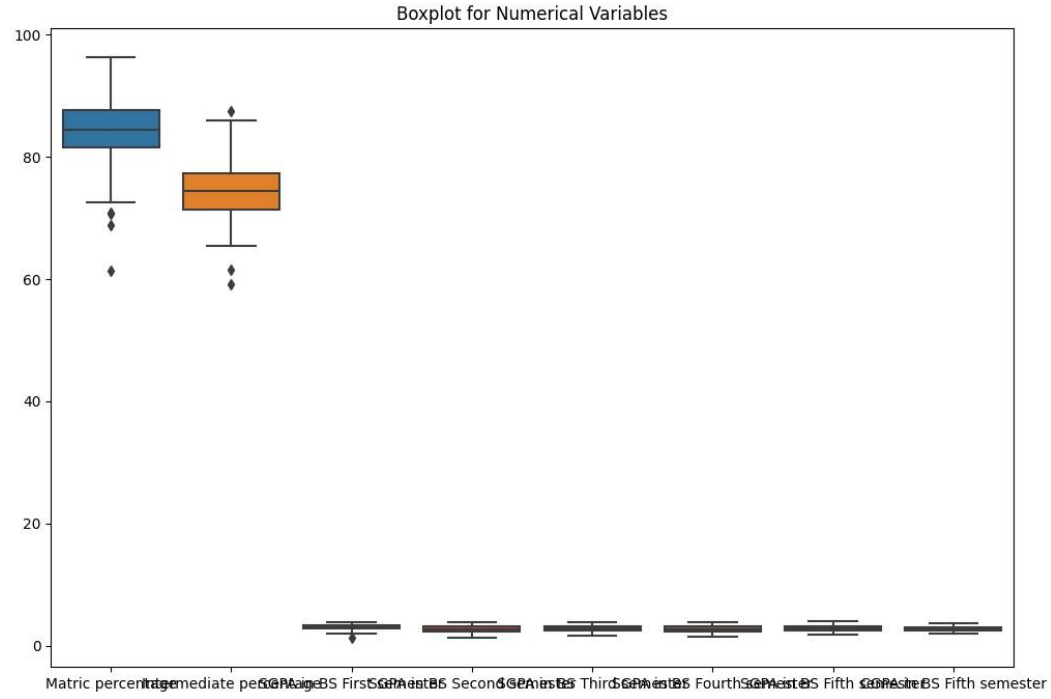
✓ 0.0s Python

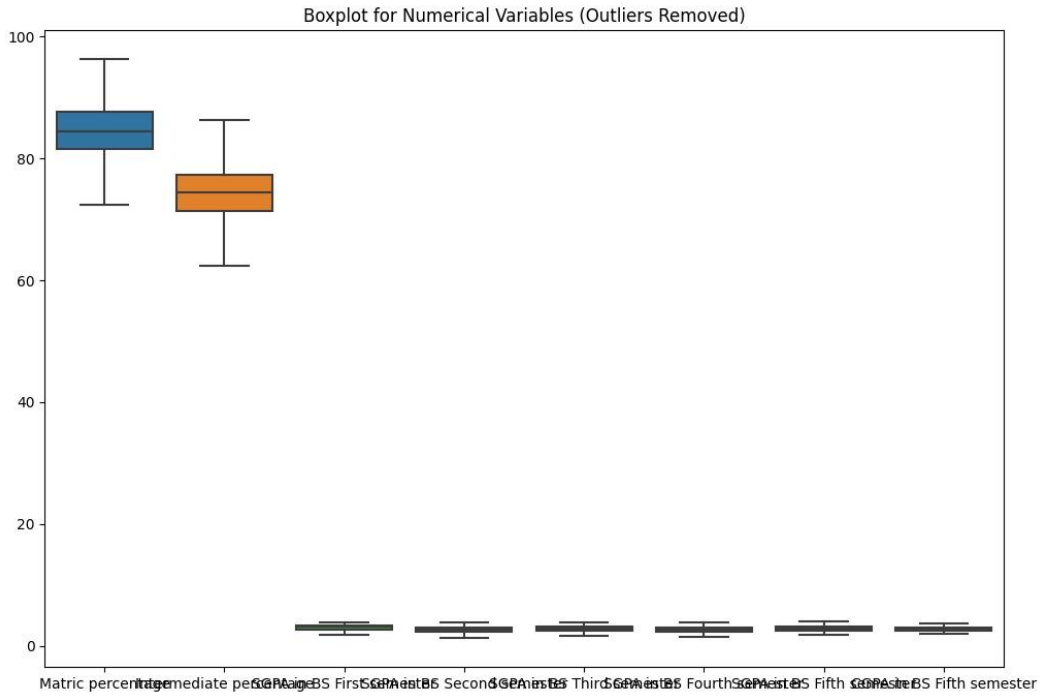
```python
df_clean = df.drop(outliers)
```

✓ 0.0s Python

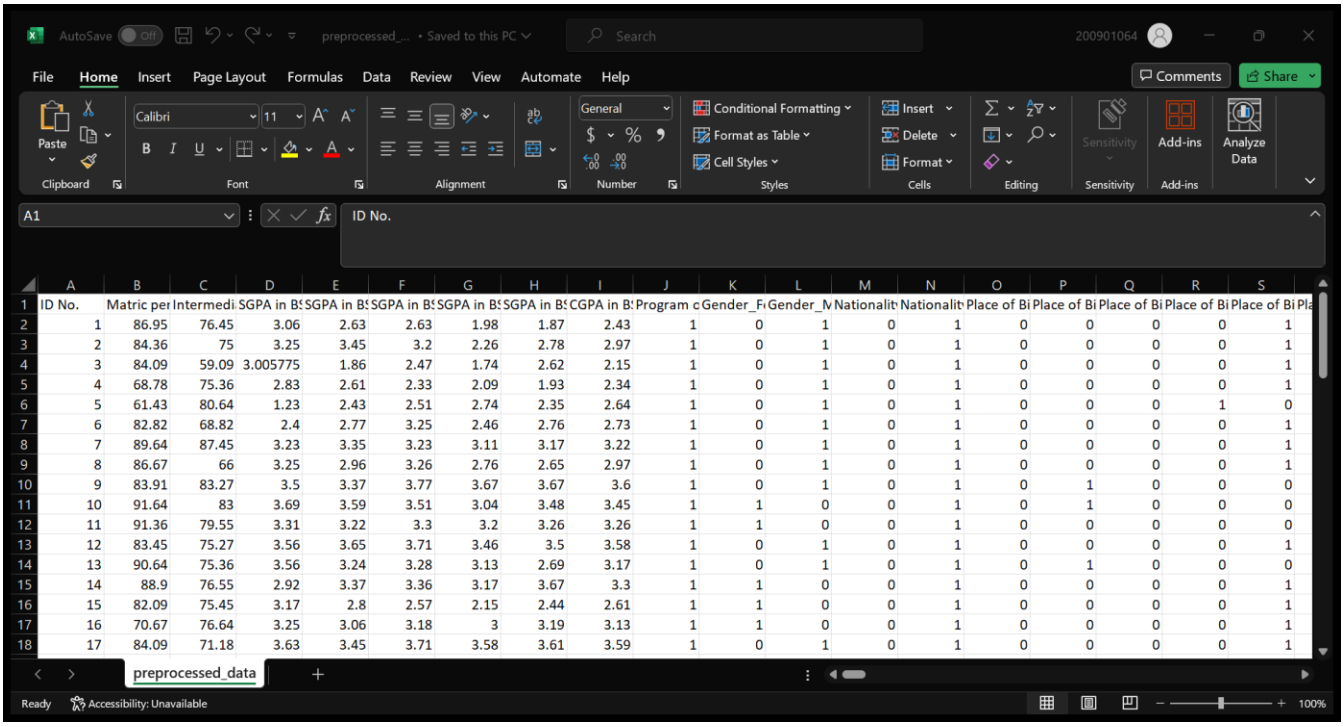
```python
for col in numerical_columns:
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    df_clean[col] = df[col].clip(lower_bound, upper_bound)
```

✓ 0.0s Python

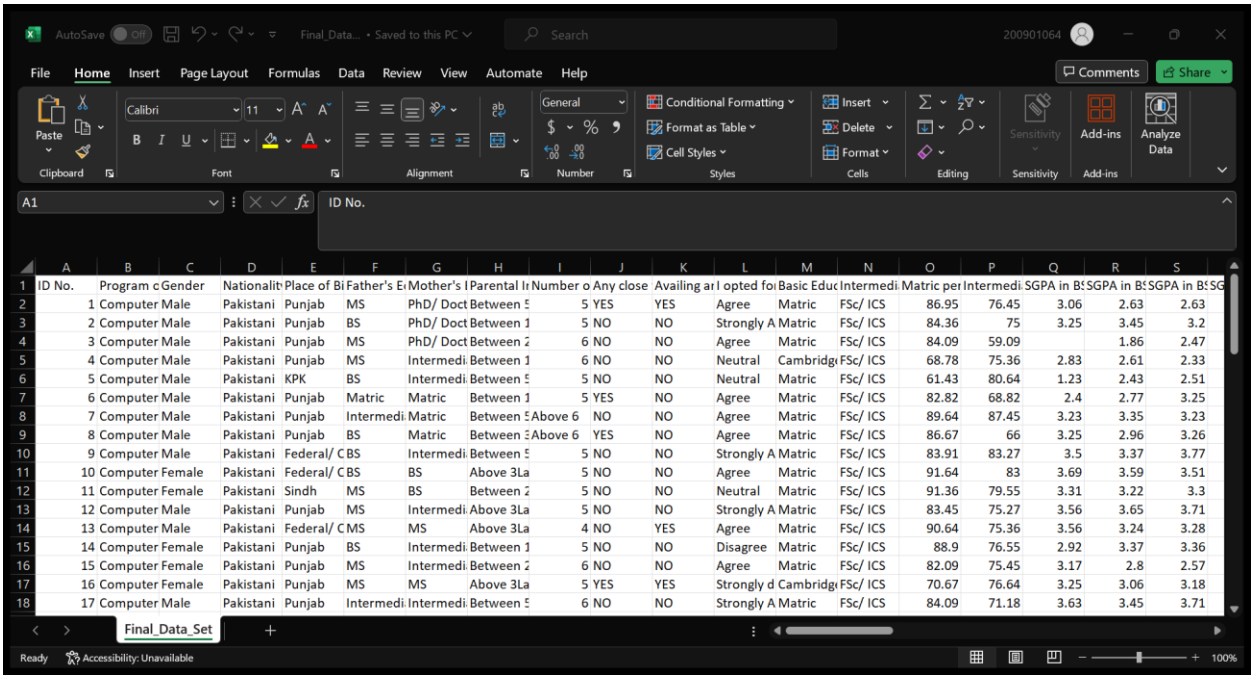




The uncleaned dataset looks like this.



After cleaning the dataset looks like.



ID No.	Program c	Gender	Nationalit	Place of Bi	Father's E	Mother's I	Parental Ir	Number o	Any close	Availing ar	I opted fo	Basic Educ	Intermedi	Matric per	Intermedi	SGPA in B	SGPA in B	SGPA in B	SG
1	Computer	Male	Pakistani	Punjab	MS	PhD/ Doct	Between 5	5	YES	YES	Agree	Matric	FSc/ ICS	86.95	76.45	3.06	2.63	2.63	
2	Computer	Male	Pakistani	Punjab	BS	PhD/ Doct	Between 1	5	NO	NO	Strongly A	Matric	FSc/ ICS	84.36	75	3.25	3.45	3.2	
3	Computer	Male	Pakistani	Punjab	MS	PhD/ Doct	Between 2	6	NO	NO	Agree	Matric	FSc/ ICS	84.09	59.09		1.86	2.47	
4	Computer	Male	Pakistani	Punjab	MS	Intermedi	Between 1	6	NO	NO	Neutral	Cambridge	FSc/ ICS	68.78	75.36	2.83	2.61	2.33	
5	Computer	Male	Pakistani	KPK	BS	Intermedi	Between 5	5	NO	NO	Neutral	Matric	FSc/ ICS	61.43	80.64	1.23	2.43	2.51	
6	Computer	Male	Pakistani	Punjab	Matric	Matric	Between 1	5	YES	NO	Agree	Matric	FSc/ ICS	82.82	68.82	2.4	2.77	3.25	
7	Computer	Male	Pakistani	Punjab	Intermedi	Matric	Between 5	Above 6	NO	NO	Agree	Matric	FSc/ ICS	89.64	87.45	3.23	3.35	3.23	
8	Computer	Male	Pakistani	Punjab	BS	Matric	Between 5	Above 6	YES	NO	Agree	Matric	FSc/ ICS	86.67	66	3.25	2.96	3.26	
9	Computer	Male	Pakistani	Federal/ C	BS	Intermedi	Between 5	5	NO	NO	Strongly A	Matric	FSc/ ICS	83.91	83.27	3.5	3.37	3.77	
10	Computer	Female	Pakistani	Federal/ C	BS	BS	Above 3La	5	NO	NO	Agree	Matric	FSc/ ICS	91.64	83	3.69	3.59	3.51	
11	Computer	Female	Pakistani	Sindh	MS	BS	Between 2	5	NO	NO	Neutral	Matric	FSc/ ICS	91.36	79.55	3.31	3.22	3.3	
12	Computer	Male	Pakistani	Punjab	MS	Intermedi	Above 3La	5	NO	NO	Strongly A	Matric	FSc/ ICS	83.45	75.27	3.56	3.65	3.71	
13	Computer	Male	Pakistani	Federal/ C	MS	MS	Above 3La	4	NO	YES	Agree	Matric	FSc/ ICS	90.64	75.36	3.56	3.24	3.28	
14	Computer	Female	Pakistani	Punjab	BS	Intermedi	Between 1	5	NO	NO	Disagree	Matric	FSc/ ICS	88.9	76.55	2.92	3.37	3.36	
15	Computer	Female	Pakistani	Punjab	MS	Intermedi	Between 2	6	NO	NO	Agree	Matric	FSc/ ICS	82.09	75.45	3.17	2.8	2.57	
16	Computer	Female	Pakistani	Punjab	MS	MS	Above 3La	5	YES	YES	Strongly d	Cambridge	FSc/ ICS	70.67	76.64	3.25	3.06	3.18	
17	Computer	Male	Pakistani	Punjab	Intermedi	Intermedi	Between 5	6	NO	NO	Strongly A	Matric	FSc/ ICS	84.09	71.18	3.63	3.45	3.71	

The rows and columns that were needed for prediction analysis were kept during cleaning.

Exploratory Data Analysis (EDA):

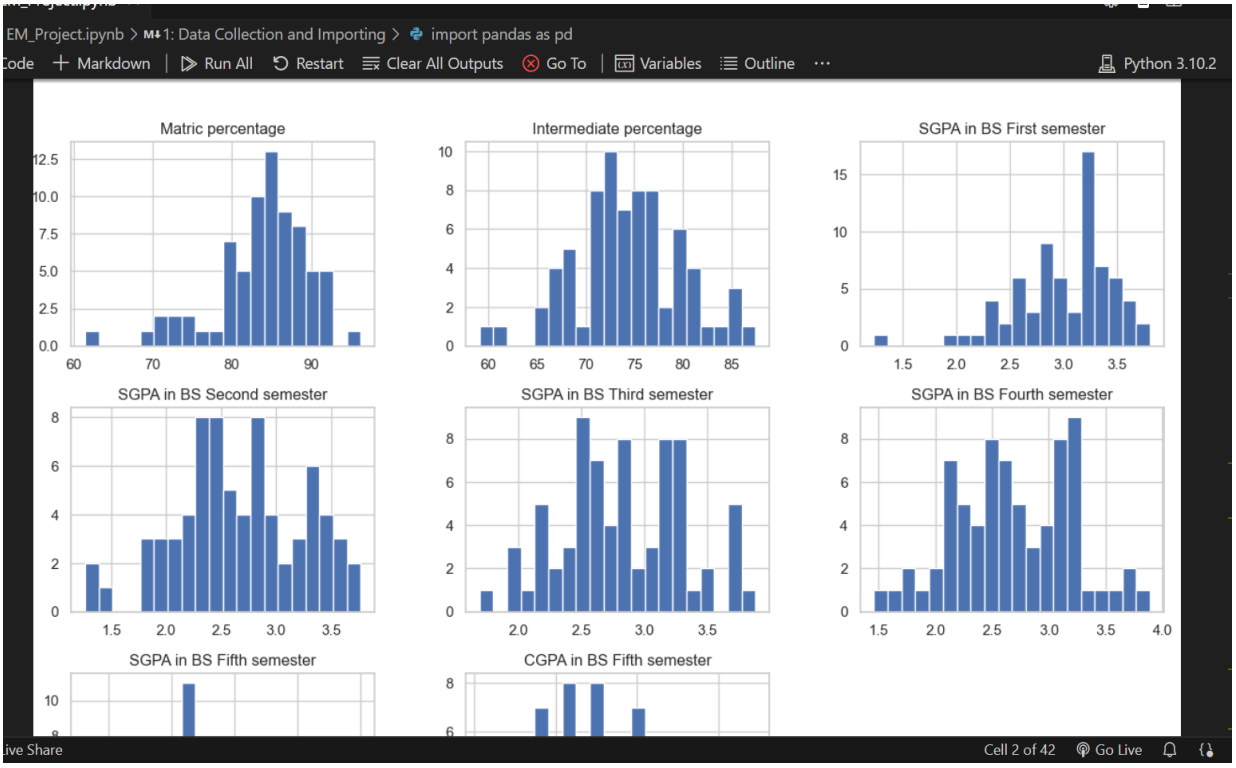
Exploratory Data Analysis (EDA) is a crucial step in the data science and machine learning pipeline as it helps you understand your data better and make informed decisions throughout the modeling process. Following is the pair plot of the analysis.

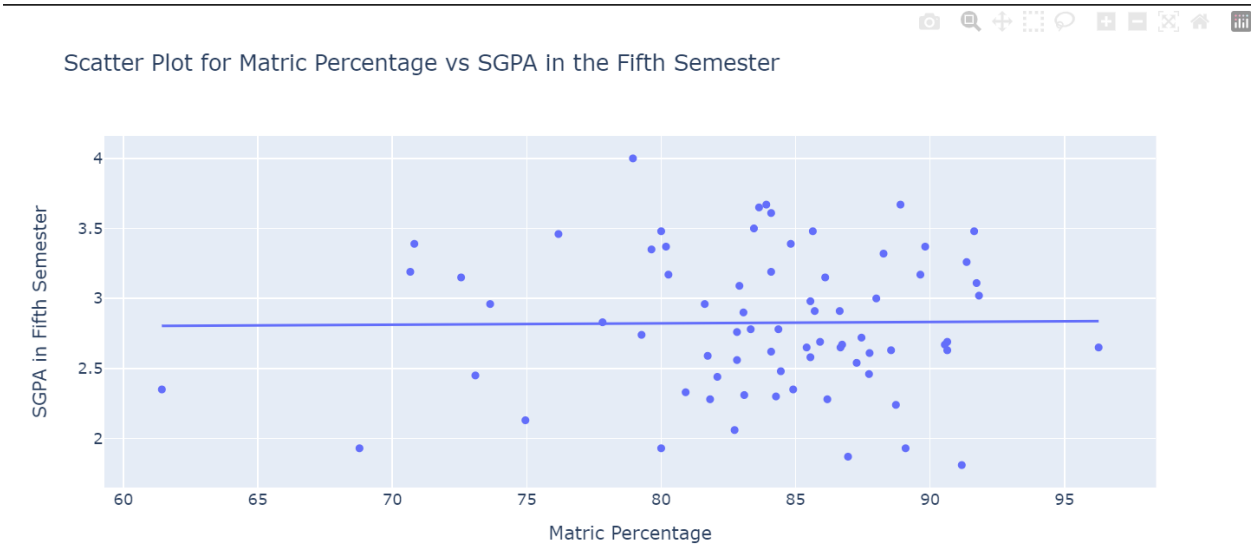
```python
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Set the style of seaborn for better visualization
sns.set(style="whitegrid")

# Select numerical columns for visualization
numerical_columns = df.select_dtypes(include=['float64']).columns
```

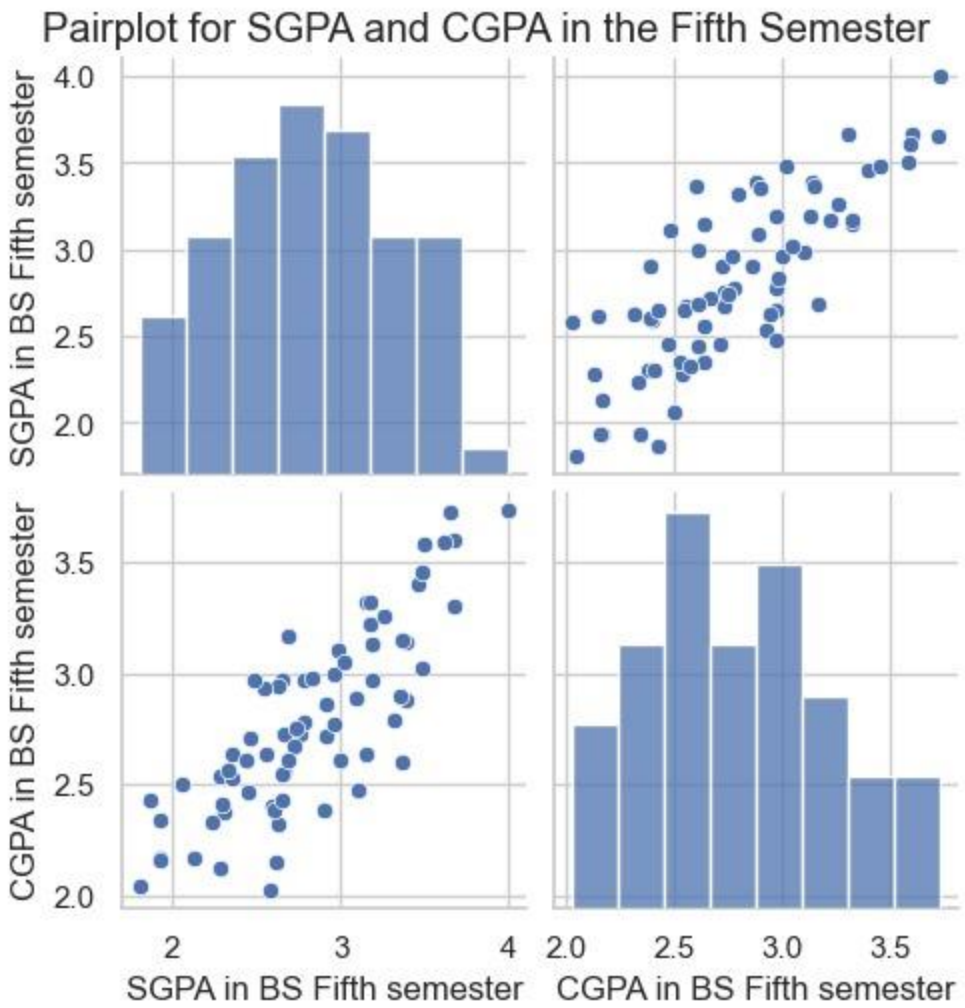
✓ 0.0s Python

```python
# Histograms for Numerical Variables
df[numerical_columns].hist(bins=20, figsize=(15, 10))
plt.suptitle('Histograms for Numerical Variables', y=1.02)
plt.show()
```

✓ 2.6s Python

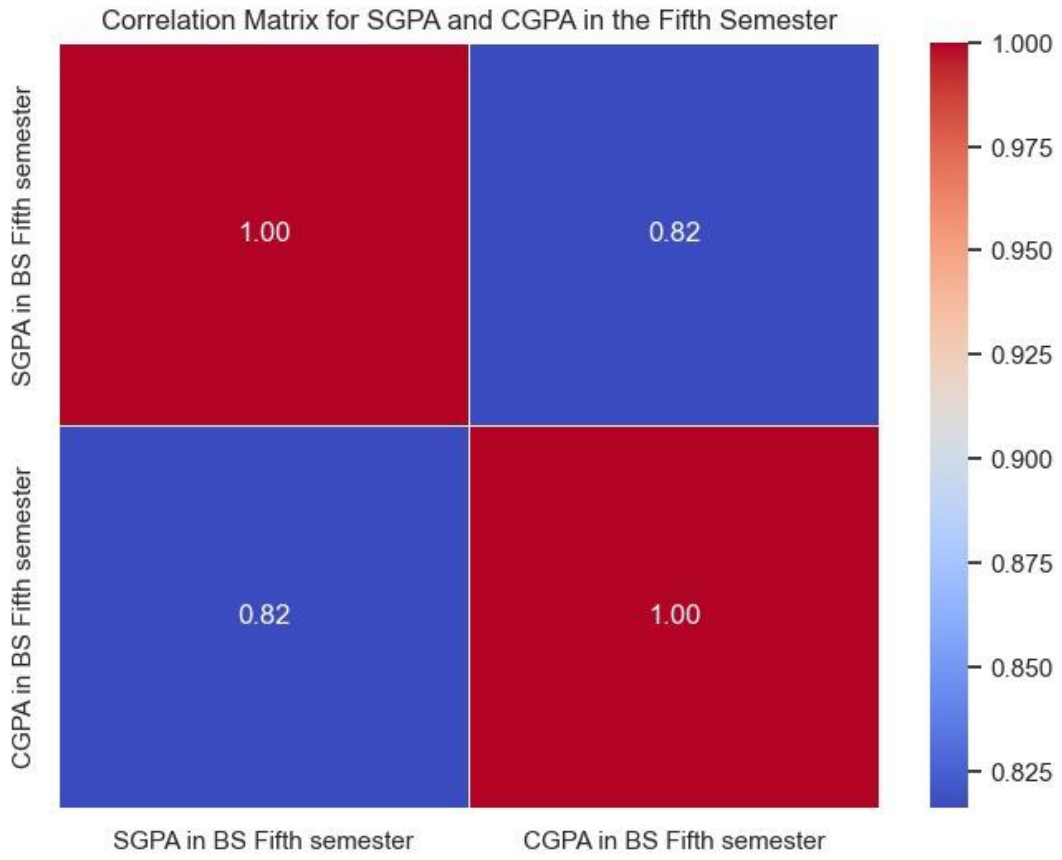

EM_Project.ipynb > 1: Data Collection and Importing > import pandas as pd
Code + Markdown | Run All Restart Clear All Outputs Go To | Variables Outline ... Python 3.10.2
Matric percentage
Intermediate percentage
SGPA in BS First semester
SGPA in BS Second semester
SGPA in BS Third semester
SGPA in BS Fourth semester
SGPA in BS Fifth semester
CGPA in BS Fifth semester
Live Share
Cell 2 of 42
Go Live


Scatter Plot for Matric Percentage vs SGPA in the Fifth Semester
SGPA in Fifth Semester
Matric Percentage



The horizontal axis of the points indicates the student's SGPA, and the horizontal axis indicates the student's CGPA. The diagonal line from bottom left to top right shows where the points would be if each student's SGPA and CGPA were exactly the same.

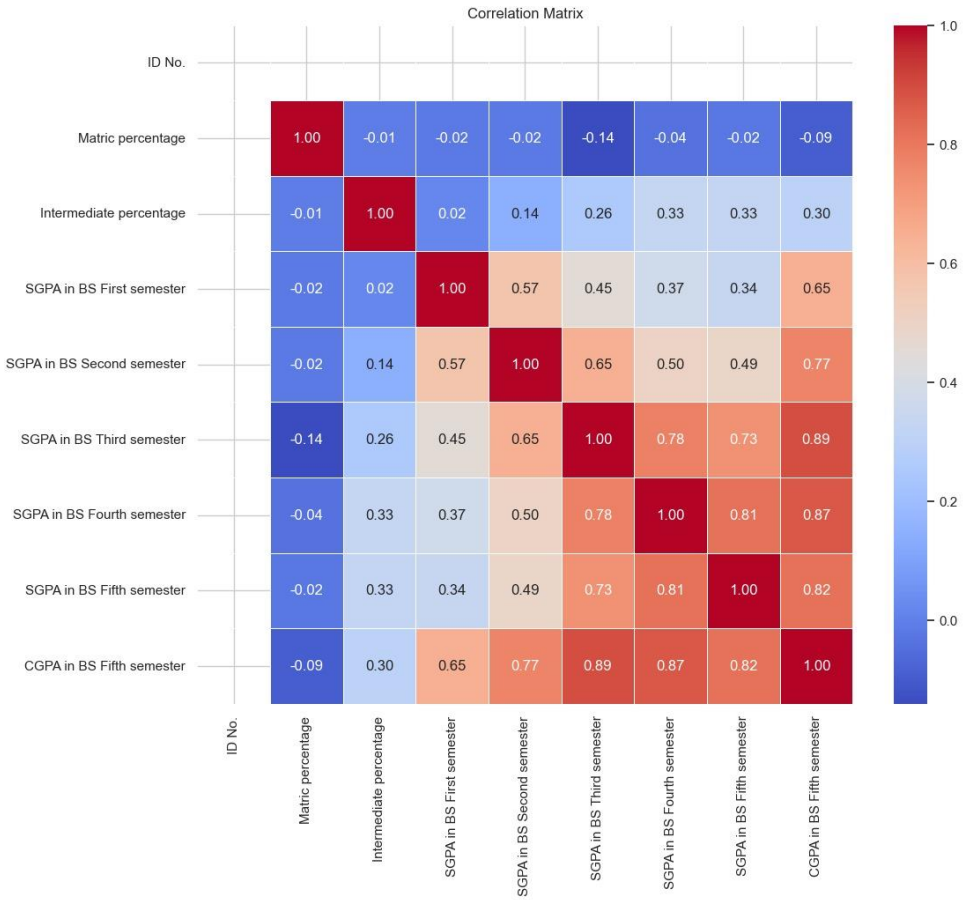
The plot shows that there is a positive correlation between SGPA and CGPA. This means that students with higher SGPA in fifth semester also have higher CGPA. However, the relationship is not perfect. There are some students who have high SGPA but low CGPA, and vice versa. In fifth semester, the average SGPA is higher than the average CGPA. This means that students perform better in their individual sessions than overall.
SGPA has broader features than CGPA. This indicates that there is greater variation in performance from any semester than the performance of the students as a whole.




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

# Assuming df is your DataFrame

# Selecting only numeric columns for the correlation matrix
numeric_columns = df.select_dtypes(include=['float64']).columns

# Creating the correlation matrix
correlation_matrix = df[numeric_columns].corr()

# Plotting the correlation matrix using a heatmap
plt.figure(figsize=(12, 10))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f', linewidths=.5)
plt.title('Correlation Matrix')
plt.show()
```

The upper left panel shows the correlation between SGPA and CGPA, which is 1.000. This shows an overall positive relationship, indicating that students with higher SGPAs also have higher CGPAs, and vice versa. The bottom right box also shows the same correlation coefficient of 1.000. This is because the matrix is symmetric, that is, the relationship between the two variables is the same in both directions. The boxes along the diagonal show the correlation of each variable with itself, which is always 1.000. The boxes on the diagonal show the relationship between SGPA and CGPA across scores. For

example, the box in the second column and the second column show the relationship between students with SGPA between 0.900 and 0.925 and students with CGPA between 0.875 and 0.900 The correlation coefficient in this box is 0.950, which is also a very strong positive correlation.



Fifth-year BS SGPA and fifth-year BS CGPA: The value of this cell is 1.000, as expected, indicating a perfectly positive relationship between these two measures of academic performance.

Matriculation percentage and CGPA in fifth year of BS: The value of this cell is -0.09, indicating a very weak negative relationship. This means that there is little or no correlation between a student's performance in matriculation examination and their overall CGPA in the BS programme.

Mean percentage and CGPA in fifth year of BS: The value of this cell is 0.30, indicating a weak positive relationship. This suggests that students' performance in intermediate examinations may have some positive correlation with their overall CGPA in BS programme.

```python
import pandas as pd
from sklearn.preprocessing import OneHotEncoder

# Identify categorical columns (you might want to adjust this based on your specific dataset)
categorical_columns = df.select_dtypes(include=['object', 'category']).columns

# Initialize OneHotEncoder
encoder = OneHotEncoder(sparse=False)

# Apply OneHotEncoder to the categorical columns
encoded_data = encoder.fit_transform(df[categorical_columns])

# Getting new column names for the encoded features
encoded_columns = encoder.get_feature_names_out(categorical_columns)

# Creating a DataFrame with the encoded data
encoded_df = pd.DataFrame(encoded_data, columns=encoded_columns)

# Concatenate the encoded DataFrame with the original DataFrame (excluding the original categorical columns)
final_df = pd.concat([df.drop(columns=categorical_columns), encoded_df], axis=1)
```

```
...    ID No.  Matric percentage  Intermediate percentage  \
0       1              86.95                    76.45
1       2              84.36                    75.00
2       3              84.09                    59.09
3       4              68.78                    75.36
4       5              61.43                    80.64
5       6              82.82                    68.82
6       7              89.64                    87.45
7       8              86.67                    66.00
8       9              83.91                    83.27
9      10              91.64                    83.00

   SGPA in BS First semester  SGPA in BS Second semester  \
0                   3.060000                        2.63
1                   3.250000                        3.45
2                   3.005775                        1.86
3                   2.830000                        2.61
4                   1.230000                        2.43
5                   2.400000                        2.77
6                   3.230000                        3.35
7                   3.250000                        2.96
8                   3.500000                        3.37
9                   3.690000                        3.59

   SGPA in BS Third semester  SGPA in BS Fourth semester  \
...
```

Predictive Models:

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, Ridge, Lasso, ElasticNet
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from xgboost import XGBRegressor    Import "xgboost" could not be resolved
from lightgbm import LGBMRegressor    Import "lightgbm" could not be resolved
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt

# Separate the features and target variables
X = data.drop(columns=['SGPA in BS Fifth semester', 'CGPA in BS Fifth semester'])
y_sgpa = data['SGPA in BS Fifth semester']  # Target variable SGPA
y_cgpa = data['CGPA in BS Fifth semester']  # Target variable CGPA

# Split the data into training and testing sets
X_train, X_test, y_train_sgpa, y_test_sgpa = train_test_split(
    X, y_sgpa, test_size=0.2, random_state=42)
X_train, X_test, y_train_cgpa, y_test_cgpa = train_test_split(
    X, y_cgpa, test_size=0.2, random_state=42)

# Initialize the models
models = {
    "Linear Regression": LinearRegression(),
    "Ridge Regression": Ridge(),
    "Lasso Regression": Lasso(),
    "ElasticNet Regression": ElasticNet(),
    "Random Forest": RandomForestRegressor(),
    "Gradient Boosting": GradientBoostingRegressor(),
    "XGBoost": XGBRegressor(),
    "LightGBM": LGBMRegressor(),
    "Decision Tree": DecisionTreeRegressor()
}

# Train and evaluate each model
results = {}

for model_name, model in models.items():
    # Fit the model for SGPA
    model.fit(X_train, y_train_sgpa)

    # Predict SGPA
    y_pred_sgpa = model.predict(X_test)

    # Calculate RMSE for SGPA
    rmse_sgpa = mean_squared_error(y_test_sgpa, y_pred_sgpa, squared=False)

    # Fit the model for CGPA
    model.fit(X_train, y_train_cgpa)

    # Predict CGPA
```

1. Linear Regression:

Linear Regression is a simple and widely used regression model that establishes a linear relationship between the input features and the target variable. It aims to minimize the difference between actual and predicted values by finding the best-fit line.

RMSE for SGPA: 0.14

RMSE for CGPA: 0.07

2. Ridge Regression:

Ridge Regression is a regularization technique applied to Linear Regression. It adds a penalty term to the linear regression objective function, preventing overfitting by discouraging large coefficients. It's particularly useful when dealing with multicollinearity.
RMSE for SGPA: 0.13
RMSE for CGPA: 0.07

3. Lasso Regression:
Lasso Regression, like Ridge, is a regularization method for Linear Regression. It adds a penalty term that encourages sparsity in the model by driving some coefficients to exactly zero. This feature selection property makes it useful in high-dimensional datasets.
RMSE for SGPA: 0.23
RMSE for CGPA: 0.27

4. ElasticNet Regression:
ElasticNet Regression combines the penalties of Ridge and Lasso. It is effective when there are multiple features and some of them are highly correlated. This model addresses their limitations individually and provides a balanced approach.
RMSE for SGPA: 0.23
RMSE for CGPA: 0.27

5. Random Forest:
Random Forest is an ensemble learning model that builds a multitude of decision trees and merges them to improve predictive accuracy and control overfitting. It is versatile, handles both regression and classification tasks, and provides feature importance.
RMSE for SGPA: 0.16
RMSE for CGPA: 0.08

6. Gradient Boosting:
Gradient Boosting is an ensemble technique that builds decision trees sequentially, with each tree correcting the errors of the previous one. It is powerful and often achieves high predictive accuracy. Common implementations include XGBoost, LightGBM, and Scikit-Learn's GradientBoostingRegressor.
RMSE for SGPA: 0.18
RMSE for CGPA: 0.09

7. XGBoost:
XGBoost (Extreme Gradient Boosting) is an optimized implementation of gradient boosting. It excels in speed and performance, using techniques like pruning and

regularization. XGBoost is widely used in machine learning competitions and real-world applications.
RMSE for SGPA: 0.19
RMSE for CGPA: 0.11

8. LightGBM:
LightGBM is a gradient boosting framework developed by Microsoft. It is designed for efficient training with large datasets and features a novel tree-growing algorithm, making it faster than many other boosting implementations.
RMSE for SGPA: 0.16
RMSE for CGPA: 0.11

9. Decision Tree:
Decision Tree is a tree-like model where each node represents a decision based on input features. It recursively splits the data to create a structure that leads to a decision or prediction. Decision trees are interpretable and widely used in machine learning and data analysis.
RMSE for SGPA: 0.20
RMSE for CGPA: 0.11

Hybrid Model:
A hybrid model can combine different machine learning models or algorithms to take advantage of their individual strengths and overcome weaknesses. It often combines various approaches, such as integrating rule-based systems with machine learning models or integrating predictive analytics or hybrid models with the aim of increasing and achieving overall performance a difficult prediction.

```
]

# Initialize the meta-learner (you can choose a different model if needed)
meta_learner = LinearRegression()

# Create the stacked model
stacked_model_sgpa = StackingRegressor(estimators=base_models, final_estimator=meta_learner)
stacked_model_cgpa = StackingRegressor(estimators=base_models, final_estimator=meta_learner)

# Train the stacked models
stacked_model_sgpa.fit(X_train, y_train_sgpa)
stacked_model_cgpa.fit(X_train, y_train_cgpa)

# Predict SGPA and CGPA using the stacked models
y_pred_sgpa_stacked = stacked_model_sgpa.predict(X_test)
y_pred_cgpa_stacked = stacked_model_cgpa.predict(X_test)

# Calculate RMSE for SGPA and CGPA predictions
rmse_sgpa_stacked = mean_squared_error(y_test_sgpa, y_pred_sgpa_stacked, squared=False)
rmse_cgpa_stacked = mean_squared_error(y_test_cgpa, y_pred_cgpa_stacked, squared=False)

print(f"Stacked Model (SGPA) - RMSE: {rmse_sgpa_stacked:.2f}")
print(f"Stacked Model (CGPA) - RMSE: {rmse_cgpa_stacked:.2f}")
```

1. **Stacked Model:**

Stacked Model, or Stacking, combines predictions from multiple machine learning models and trains a meta-model on their results. Instead of using homogeneous analyzes (as in clusters), the meta-model learns to calibrate the predictions of the base models based on their performance. Stacking aims to capture overlapping features in models, which can improve overall prediction accuracy and robustness.

Stacked Model (SGPA) - RMSE: 0.15
Stacked Model (CGPA) - RMSE: 0.07

2. **Parallel Model:**

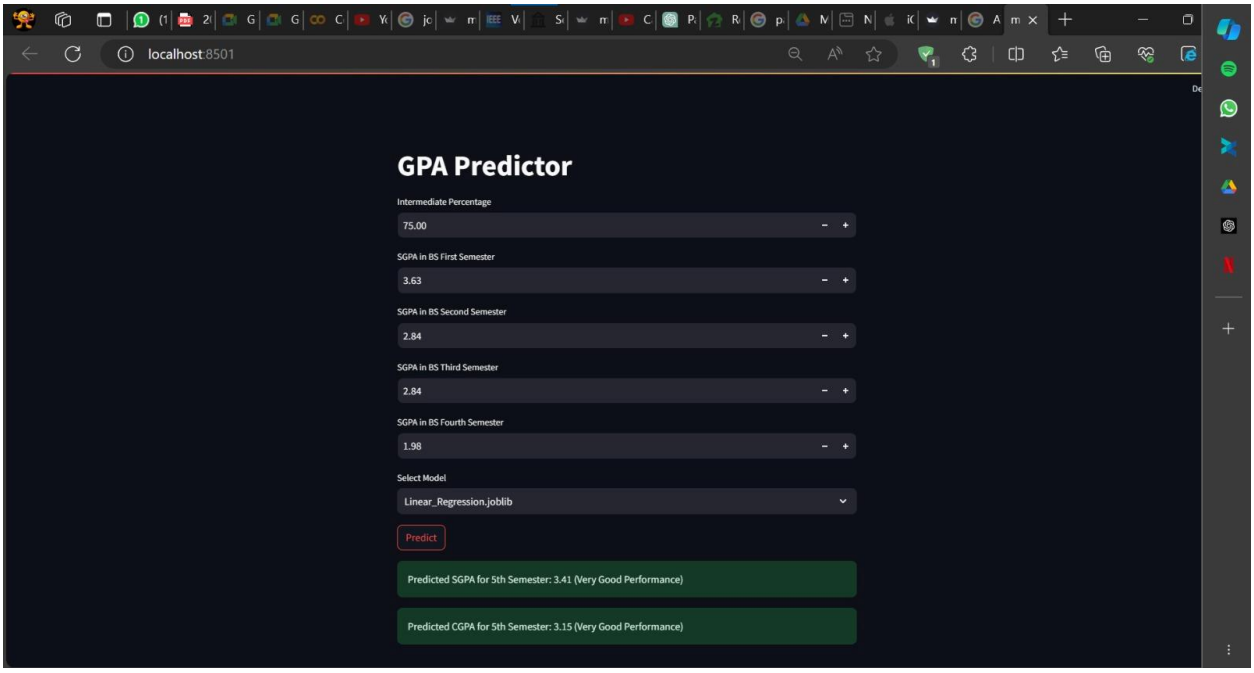
Parallel models involve running multiple machine learning models simultaneously on the same data set or task. Each model works independently, and the results together make predictions or decisions. Parallelization speeds up computation, allowing for faster training and analysis. It is often used in batch processes where models contribute to the final output.

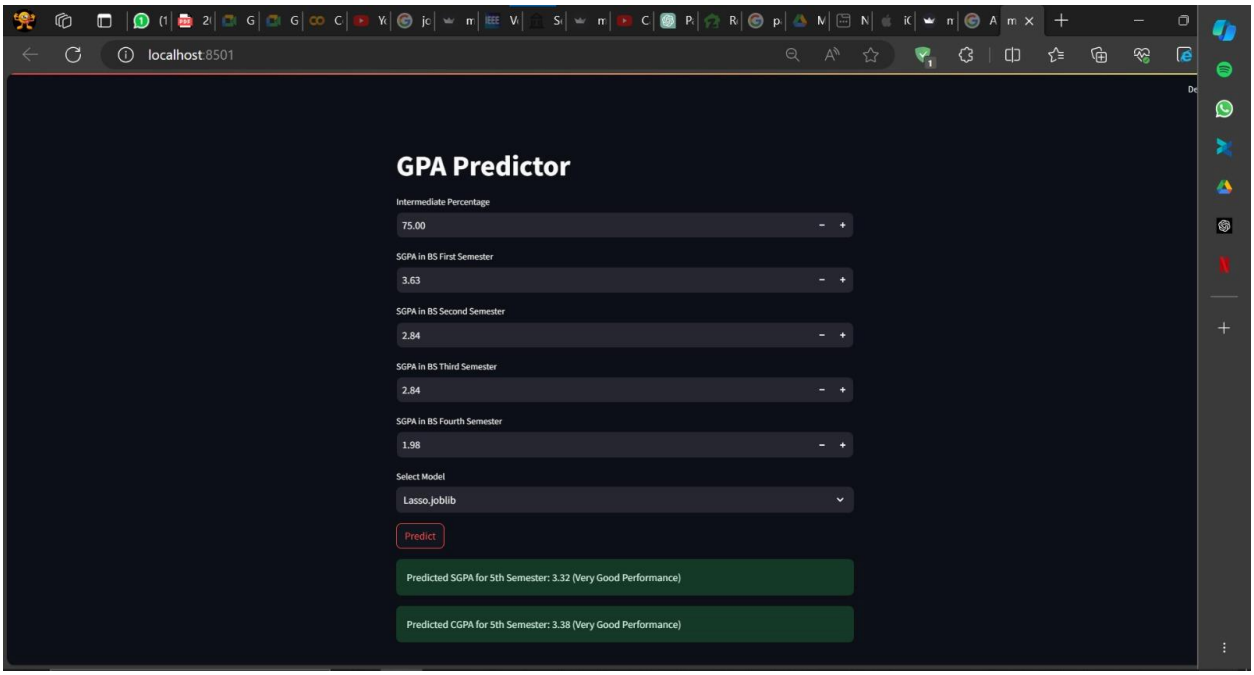
Parallel Ensemble Model (SGPA) - RMSE: 0.15
Parallel Ensemble Model (CGPA) - RMSE: 0.11

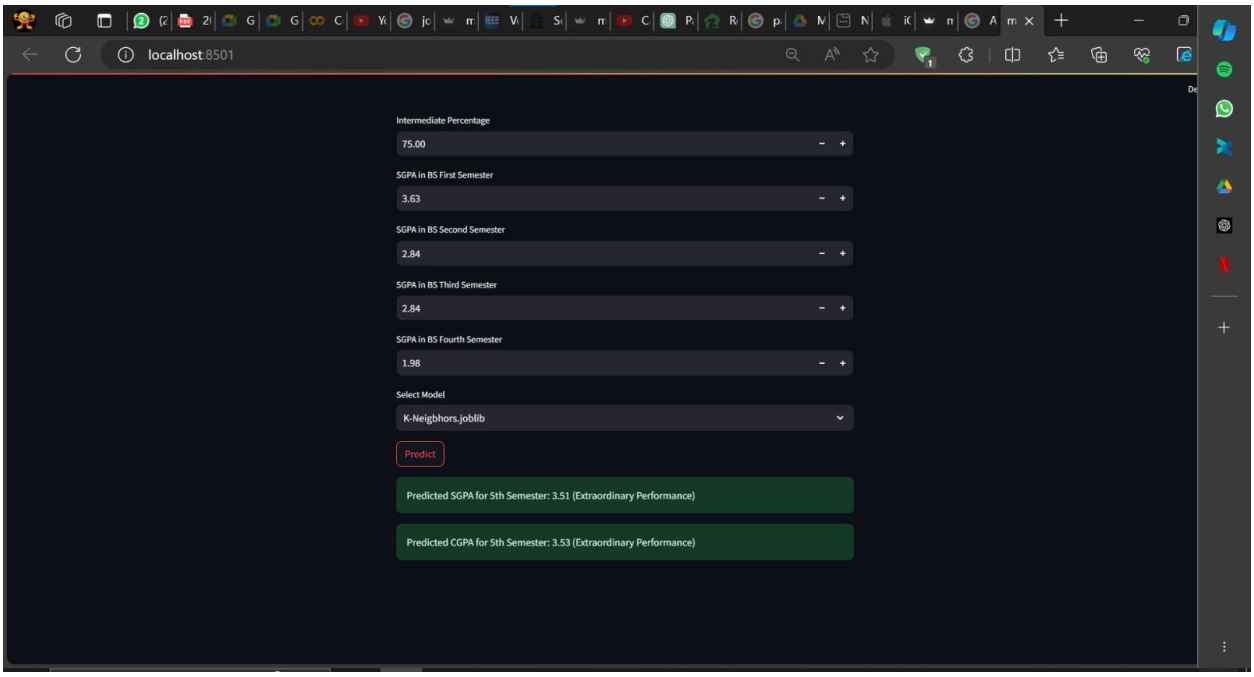
```python
# Create the parallel ensemble model for SGPA
ensemble_model_sgpa = VotingRegressor(estimators=base_models, n_jobs=-1)

# Create the parallel ensemble model for CGPA
ensemble_model_cgpa = VotingRegressor(estimators=base_models, n_jobs=-1)

# Train the parallel ensemble models
ensemble_model_sgpa.fit(X_train, y_train_sgpa)
ensemble_model_cgpa.fit(X_train, y_train_cgpa)

# Predict SGPA and CGPA using the parallel ensemble models
y_pred_sgpa_ensemble = ensemble_model_sgpa.predict(X_test)
y_pred_cgpa_ensemble = ensemble_model_cgpa.predict(X_test)

# Calculate RMSE for SGPA and CGPA predictions
rmse_sgpa_ensemble = mean_squared_error(y_test_sgpa, y_pred_sgpa_ensemble, squared=False)
rmse_cgpa_ensemble = mean_squared_error(y_test_cgpa, y_pred_cgpa_ensemble, squared=False)

print(f"Parallel Ensemble Model (SGPA) - RMSE: {rmse_sgpa_ensemble:.2f}")
print(f"Parallel Ensemble Model (CGPA) - RMSE: {rmse_cgpa_ensemble:.2f}")
```

5. Deployment:

Streamlit is a Python library designed for rapid development and deployment of interactive web applications. It simplifies the process of turning data scripts into shareable web applications with minimal effort. With its intuitive and user-friendly syntax, Streamlit allows developers and data scientists to create interactive dashboards and visualizations effortlessly. Leveraging Streamlit, one can seamlessly integrate data exploration, analysis, and visualization into a web-based format, making it an ideal choice for quick and efficient deployment of machine learning models and data-driven applications.

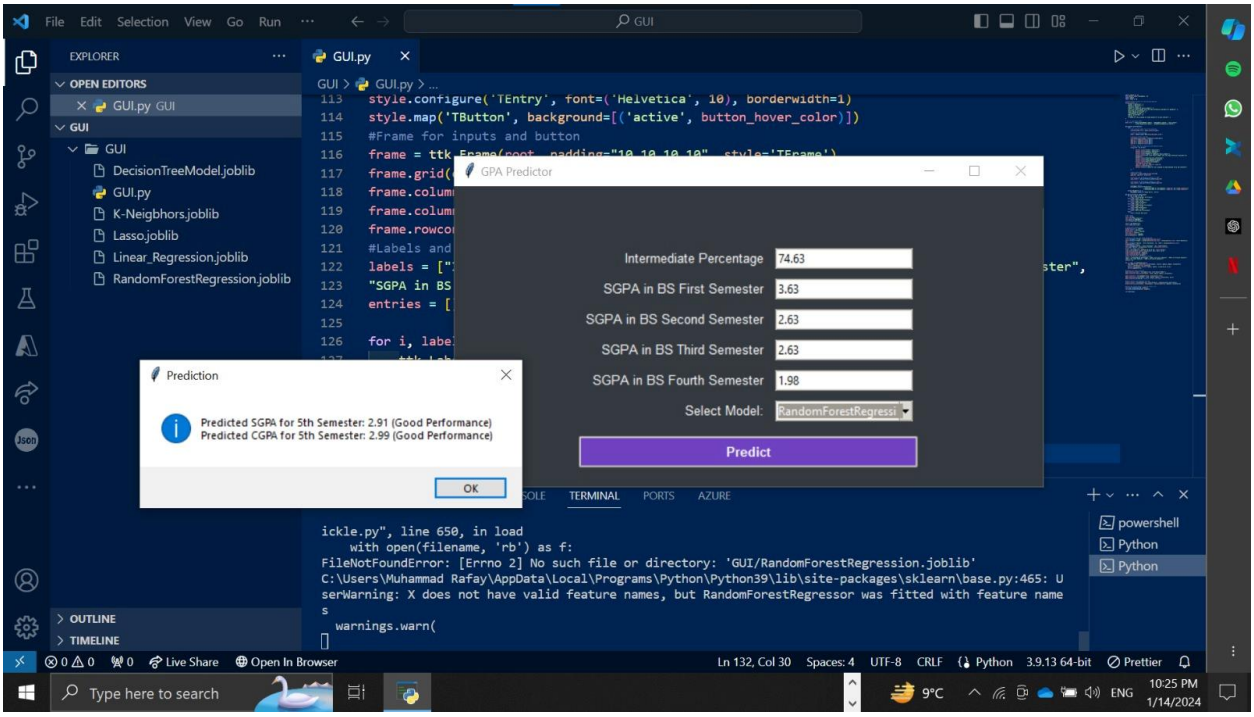

localhost:8501
GPA Predictor
Intermediate Percentage
75.00
SGPA in BS First Semester
3.63
SGPA in BS Second Semester
2.84
SGPA in BS Third Semester
2.84
SGPA in BS Fourth Semester
1.98
Select Model
Linear_Regression.joblib
Predict
Predicted SGPA for 5th Semester: 3.41 (Very Good Performance)
Predicted CGPA for 5th Semester: 3.15 (Very Good Performance)

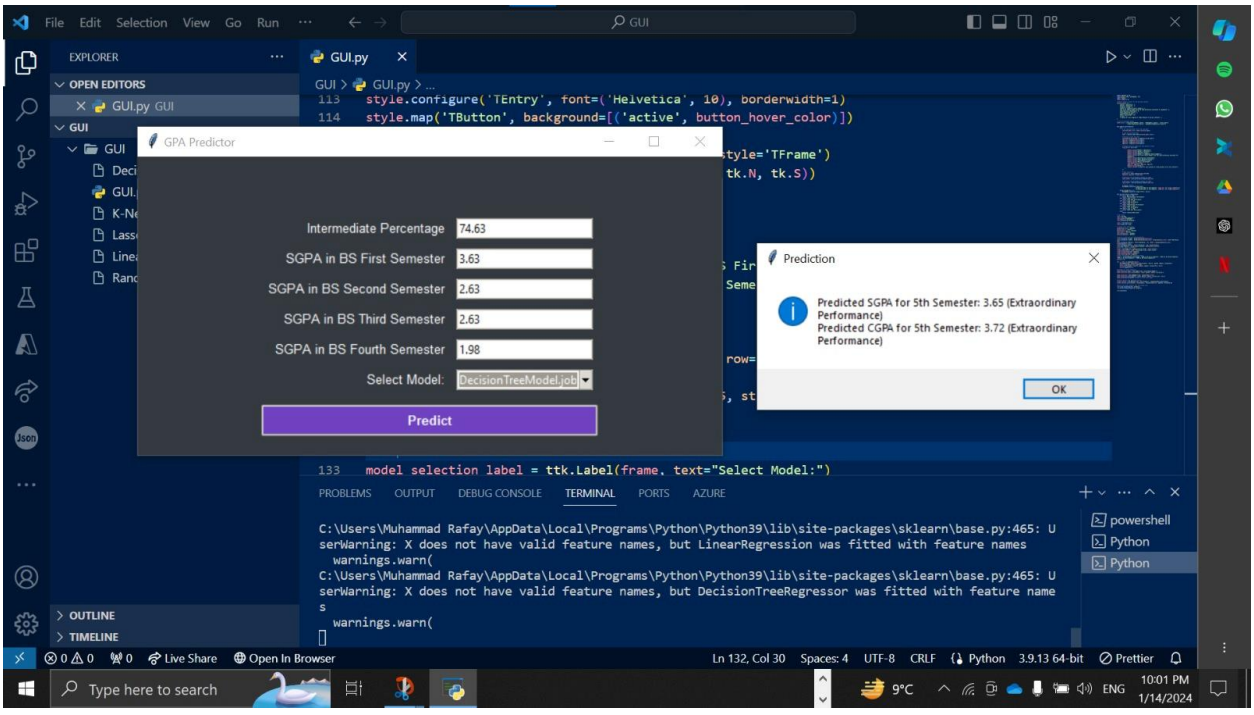

localhost:8501
GPA Predictor
Intermediate Percentage
75.00
SGPA in BS First Semester
3.63
SGPA in BS Second Semester
2.84
SGPA in BS Third Semester
2.84
SGPA in BS Fourth Semester
1.98
Select Model
Lasso.joblib
Predict
Predicted SGPA for 5th Semester: 3.32 (Very Good Performance)
Predicted CGPA for 5th Semester: 3.38 (Very Good Performance)

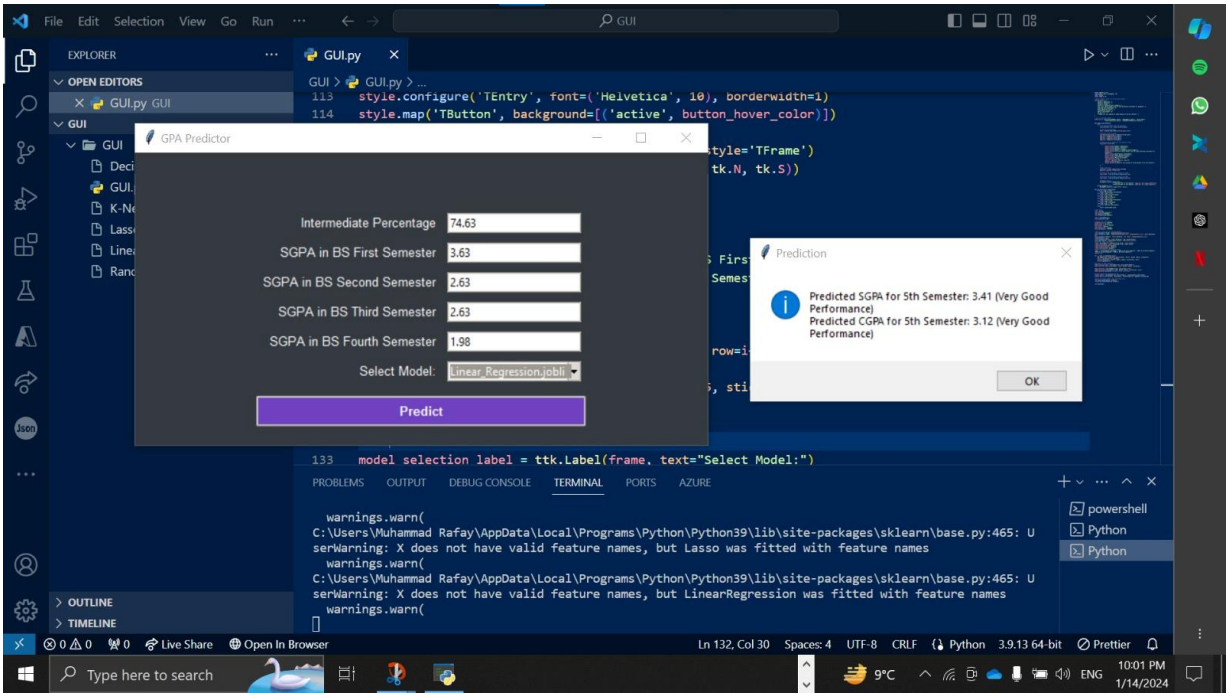


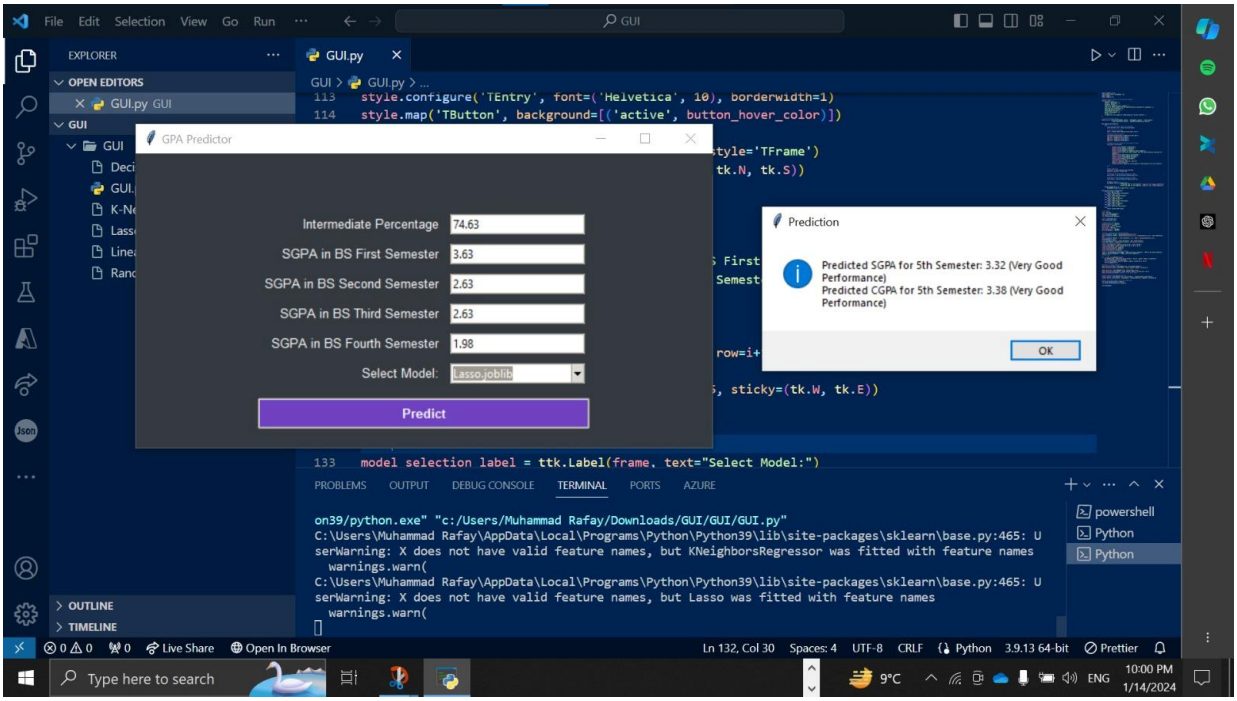
6. GUI Development:

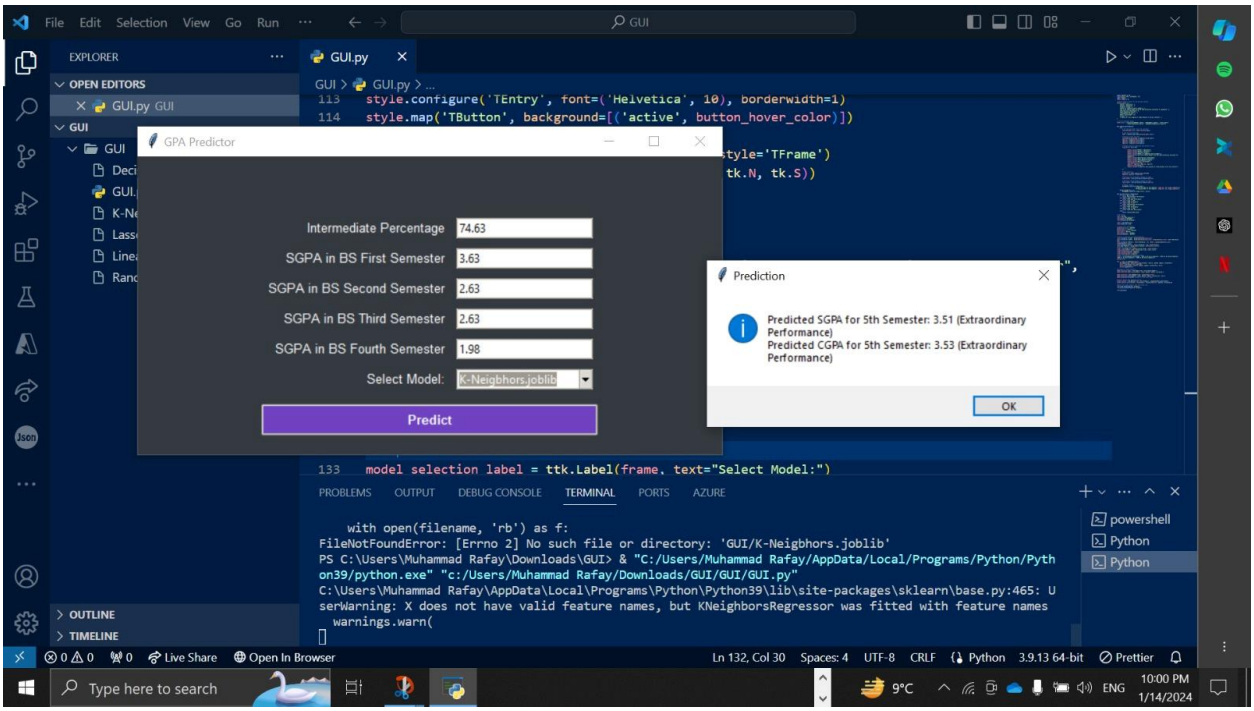
This is a Python-based GUI application designed for predicting a student's SGPA and CGPA for the fifth semester. The user inputs data such as SGPA in previous semesters and intermediate percentage. The application employs machine learning models like Decision Tree, K-Neighbors, Lasso, Linear Regression, and Random Forest Regression for predictions. A "Predict" button initiates the process, gathering input data, loading the selected model, and displaying predicted SGPA and CGPA values. The GUI also offers file navigation and code editing features.


GPA Predictor
Intermediate Percentage 74.63
SGPA in BS First Semester 3.63
SGPA in BS Second Semester 2.63
SGPA in BS Third Semester 2.63
SGPA in BS Fourth Semester 1.98
Select Model: RandomForestRegressi
Predict
Prediction
Predicted SGPA for 5th Semester: 2.91 (Good Performance)
Predicted CGPA for 5th Semester: 2.99 (Good Performance)
OK
FileNotFoundError: [Errno 2] No such file or directory: 'GUI/RandomForestRegression.joblib'
C:\Users\Muhammad Rafay\AppData\Local\Programs\Python\Python39\lib\site-packages\sklearn\base.py:465: UserWarning: X does not have valid feature names, but RandomForestRegressor was fitted with feature names
warnings.warn(


GPA Predictor
Intermediate Percentage 74.63
SGPA in BS First Semester 3.63
SGPA in BS Second Semester 2.63
SGPA in BS Third Semester 2.63
SGPA in BS Fourth Semester 1.98
Select Model: DecisionTreeModel.job
Predict
Prediction
Predicted SGPA for 5th Semester: 3.65 (Extraordinary Performance)
Predicted CGPA for 5th Semester: 3.72 (Extraordinary Performance)
OK
C:\Users\Muhammad Rafay\AppData\Local\Programs\Python\Python39\lib\site-packages\sklearn\base.py:465: UserWarning: X does not have valid feature names, but LinearRegression was fitted with feature names
warnings.warn(
C:\Users\Muhammad Rafay\AppData\Local\Programs\Python\Python39\lib\site-packages\sklearn\base.py:465: UserWarning: X does not have valid feature names, but DecisionTreeRegressor was fitted with feature names
warnings.warn(


GPA Predictor
Intermediate Percentage 74.63
SGPA in BS First Semester 3.63
SGPA in BS Second Semester 2.63
SGPA in BS Third Semester 2.63
SGPA in BS Fourth Semester 1.98
Select Model: Linear_Regression.joblib
Predict
Prediction
Predicted SGPA for 5th Semester: 3.41 (Very Good Performance)
Predicted CGPA for 5th Semester: 3.12 (Very Good Performance)
OK
model selection label = ttk.Label(frame, text="Select Model:")
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS AZURE
warnings.warn(
C:\Users\Muhammad Rafay\AppData\Local\Programs\Python\Python39\lib\site-packages\sklearn\base.py:465: UserWarning: X does not have valid feature names, but Lasso was fitted with feature names
warnings.warn(
C:\Users\Muhammad Rafay\AppData\Local\Programs\Python\Python39\lib\site-packages\sklearn\base.py:465: UserWarning: X does not have valid feature names, but LinearRegression was fitted with feature names
warnings.warn(
powershell
Python
Python
Ln 132, Col 30 Spaces: 4 UTF-8 CRLF Python 3.9.13 64-bit Prettier
Type here to search
9°C ENG 10:01 PM 1/14/2024


GPA Predictor
Intermediate Percentage 74.63
SGPA in BS First Semester 3.63
SGPA in BS Second Semester 2.63
SGPA in BS Third Semester 2.63
SGPA in BS Fourth Semester 1.98
Select Model: Lasso.joblib
Predict
Prediction
Predicted SGPA for 5th Semester: 3.32 (Very Good Performance)
Predicted CGPA for 5th Semester: 3.38 (Very Good Performance)
OK
model selection label = ttk.Label(frame, text="Select Model:")
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS AZURE
on39/python.exe" "c:/Users/Muhammad Rafay/Downloads/GUI/GUI/GUI.py"
C:\Users\Muhammad Rafay\AppData\Local\Programs\Python\Python39\lib\site-packages\sklearn\base.py:465: UserWarning: X does not have valid feature names, but KNeighborsRegressor was fitted with feature names
warnings.warn(
C:\Users\Muhammad Rafay\AppData\Local\Programs\Python\Python39\lib\site-packages\sklearn\base.py:465: UserWarning: X does not have valid feature names, but Lasso was fitted with feature names
warnings.warn(
powershell
Python
Python
Ln 132, Col 30 Spaces: 4 UTF-8 CRLF Python 3.9.13 64-bit Prettier
Type here to search
9°C ENG 10:00 PM 1/14/2024



7. Conclusion:

This study employs a systematic approach to assess and predict academic performance based on various factors. Through careful data preprocessing and exploratory data analysis, the project identifies patterns and correlations in the dataset. The machine learning models employed demonstrate varying levels of accuracy in predicting SGPA and CGPA, with Linear Regression and Ridge Regression showing lower RMSE values compared to Lasso Regression and ElasticNet Regression. Random Forest and Gradient Boosting, as ensemble models, provide competitive performance. The GUI developed facilitates user interaction for testing new data and predicting outcomes. Overall, this project contributes valuable insights into academic predictive analytics, showcasing the potential for proactive interventions to enhance student success in higher education.