Amirkabir University of Technology
(Tehran Polytechnic)

Exploring Academic Performance in General Mathematics 1

A Case Study of Students at AmirKabir University of Technology

by Koorosh Komeilizadeh

Project Introduction

In this project, our primary objective is to conduct a meticulous analysis of student and instructor performance during the grading process for General Mathematics 1 at AmirKabir University of Technology for the 1401 academic year. Leveraging a dataset generously provided by the esteemed Faculty of Mathematics and Computer Science, we will employ advanced data retrieval and cleaning techniques to ensure data accuracy and consistency. By scrutinizing the information meticulously, we aim to discern insightful trends and patterns that can shed light on the efficacy of the grading system and identify areas for potential improvement. Our findings will serve as a valuable resource for enhancing the overall educational experience and fostering an environment of academic excellence at the university.

Project Contributors

Koorosh Komeilizadeh - Author and Analyst
DSAI Undergraduate Student at Leiden University

Behzad Najafi Saghezchi - Reviewer
Assistant Professor at AmirKabir University of Technology

Farzad Didehvar - Reviewer
Assistant Professor at AmirKabir University of Technology

Outlines

1- Preview

1.1 Dataset

The Faculty of Mathematics and Computer Science at AmirKabir University of Technology provides reports in Excel format, transparently disclosing students' detailed grades for each semester. Our objective is to import this data into a data frame, perform data cleaning, and eliminate any outliers to conduct appropriate data analysis. The dataset pertains to General Mathematics 1 for the 1401 academic year at AmirKabir University of Technology

The meticulously cleaned and meticulously processed dataset for this project is readily accessible on Kaggle.

General Mathematics 1 Student grades at AUT

1.2 Goals

```
- Understanding Grade Distribution of Students.
- Exploring the Correlation between Instructors and Average Grades.
- Investigating the Relationship Between Students' Majors and Grade Status.
- Finding the relationships between international and native students' grades.
- Predicting Final Grades based on Midterm and Homework Scores.
- Providing a Clean Dataset of Grades for Future Research Purposes.
```

1.3 Tools

- **Pandas**

Utilized to convert the data into a data frame format, enabling efficient data cleaning and storage.

- **Numpy**

  Employed for advanced mathematical functions and array operations.

- **Matplotlib**

  Used for data visualization and generating various types of plots and graphs to better understand the data.

- **Scikit-learn**

  Applied to develop Linear regression models for predictive analysis and Encoding Labels for non-numeric datas

1.4 Author and Analyst

Koorosh Komeilizadeh

Data Science and Artificial Intelligence
undergraduate student at Leiden University

Passionate Data Science & AI student with expertise in Python, Machine Learning, and Data Analysis. Eager to explore innovative applications of AI and committed to driving data-driven solutions for impactful research.

Email : kkomeilizadeh@gmail.com
Website : www.kooroshkz.com
Github & LinkedIn : kooroshkz

2- Data Processing

Data processing is a crucial step in data analysis that plays a pivotal role in ensuring the accuracy, reliability, and meaningfulness of the results. In any data-driven project, raw data often contains noise, inconsistencies, and missing values, making it challenging to draw insightful conclusions directly. Data processing involves a series of techniques, such as data cleaning, transformation, integration, and normalization, aimed at refining the raw data into a usable and consistent format. By addressing data quality issues, data processing enhances the integrity of the analysis, reduces bias, and enables more robust statistical methods.

2.1 Import Data

Pandas

Pandas is a popular Python library widely used for data manipulation and analysis. It provides powerful data structures and functions to efficiently work with structured data. One of its core data structures is the DataFrame, which is essentially a two-dimensional, labeled data structure resembling a table or spreadsheet. The DataFrame allows data to be organized in rows and columns, where each column can have a specific data type. It is versatile and can handle different types of data, including CSV, JSON, SQL databases, and Excel files.

```python
import pandas as pd
```

```python
# Read all sheets from the "grades.xlsx" Excel file into a dictionary of DataFrames
all_sheets = pd.read_excel("grades.xlsx", sheet_name=None)
```

```
# Concatenate all DataFrames in the dictionary into a single DataFrame
merged_df = pd.concat(all_sheets.values(), ignore_index=True)
```

```
# Shows a preview of DataFrame
merged_df
```

	گروه درس	شماره دانشجو	تکلیف1	تکلیف2	تکلیف3	تکلیف4	تکلیف5	تکلیف6	تکلیف7	تکلیف8	...	پایانترم از 9	میانترم از7	نهایی 1	پایانترم از13	میانترم از3
0	1	40027008	40.0	33.0	0.0	35.0	40.0	40	35	0.0	...	0.5	2.2	4.93	0.722222	0.94285
1	1	40113901	37.0	33.0	38.0	0.0	40.0	40	40	40.0	...	3.0	3.1	8.78	4.333333	1.32857
2	1	40122045	24.0	32.0	40.0	20.0	15.0	40	35	18.0	...	1.0	4.3	7.54	1.444444	1.84285
3	1	40122046	0.0	36.0	40.0	40.0	40.0	40	35	40.0	...	1.9	0.1	4.71	2.744444	0.04285
4	1	40122047	0.0	0.0	0.0	0.0	0.0	40	0	0.0	...	0.0	1.1	1.50	0.000000	0.47142
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1276	19	40132012	0.0	40.0	40.0	36.0	35.0	0	35	22.0	...	3.5	3.9	9.48	5.055556	1.67142
1277	19	40132013	25.0	20.0	20.0	37.0	40.0	40	40	0.0	...	6.3	0.0	8.52	9.100000	0.00000
1278	19	40132021	40.0	34.0	40.0	40.0	40.0	40	40	40.0	...	8.6	5.6	17.34	12.422222	2.40000
1279	19	40132033	0.0	0.0	40.0	38.0	35.0	40	40	0.0	...	5.8	2.3	10.03	8.377778	0.98571
1280	19	40132048	37.0	39.0	38.0	38.0	35.0	40	40	40.0	...	8.5	0.0	11.57	12.277778	0.00000

1281 rows × 28 columns

2.2 Data Cleaning

Data cleaning is a critical phase aimed at enhancing the quality and reliability of the dataset. This process involves identifying and rectifying errors, inconsistencies, and missing values present in the raw data. By employing techniques such as outlier detection, handling missing data, and resolving discrepancies, data cleaning ensures that the dataset is accurate, complete, and consistent. Moreover, it plays a crucial role in preparing the data for analysis, preventing misleading or biased results. Through meticulous data cleaning, my project aims to lay a solid foundation for insightful data analysis and ultimately make well-informed decisions based on trustworthy and dependable data.

```
# Get a list of all column names in the DataFrame
columns = merged_df.columns.tolist()

# Define a list of column names that you want to drop
extra_columns = columns[0:1] + columns[10:19] + columns[21:-1]

# Drop the columns specified in the `extra_columns` list from the DataFrame
merged_df.drop(extra_columns, axis=1, inplace=True)
```

```
merged_df
```

	شماره دانشجو	تکلیف1	تکلیف2	تکلیف3	تکلیف4	تکلیف5	تکلیف6	تکلیف7	تکلیف8	میانترم از7	نهایی 1	استاد
0	40027008	40.0	33.0	0.0	35.0	40.0	40	35	0.0	2.2	4.93	دکتر کیانی
1	40113901	37.0	33.0	38.0	0.0	40.0	40	40	40.0	3.1	8.78	دکتر کیانی
2	40122045	24.0	32.0	40.0	20.0	15.0	40	35	18.0	4.3	7.54	دکتر کیانی
3	40122046	0.0	36.0	40.0	40.0	40.0	40	35	40.0	0.1	4.71	دکتر کیانی
4	40122047	0.0	0.0	0.0	0.0	0.0	40	0	0.0	1.1	1.50	دکتر کیانی
...	...	...	...	...	...	...	...	...	...	...	...	...
1276	40132012	0.0	40.0	40.0	36.0	35.0	0	35	22.0	3.9	9.48	دکترشریعتی
1277	40132013	25.0	20.0	20.0	37.0	40.0	40	40	0.0	0.0	8.52	دکترشریعتی
1278	40132021	40.0	34.0	40.0	40.0	40.0	40	40	40.0	5.6	17.34	دکترشریعتی
1279	40132033	0.0	0.0	40.0	38.0	35.0	40	40	0.0	2.3	10.03	دکترشریعتی
1280	40132048	37.0	39.0	38.0	38.0	35.0	40	40	40.0	0.0	11.57	دکترشریعتی

1281 rows × 12 columns

```
#Rename headers
merged_df.columns = ["ID"] + [f"HW{x}" for x in range(1,9)] + ["mid", "final", "prof"]
```

```
# Fix duplicate prof name error
merged_df.replace("دکترخسروی", "دکتر خسروی", inplace=True)
merged_df.replace("شجاعی", "دکترشجاعی", inplace=True)
```

```
# Define a dictionary of prof names in Farsi and English
prof_names = {
    "دکتر ایمانفر": "Dr. Imanfar",
    "دکتر خسروی": "Dr. Khosravi",
    "دکتر رحمتی": "Dr. Rahmati",
    "دکتر رستمی": "Dr. Rostami",
    "دکتر نجفی": "Dr. Najafi",
    "دکتر کیانی": "Dr. Kiani",
    "دکتراخلاقی": "Dr. Akhlaghi",
    "دکتربروجردیان": "Dr. Boroujerdiyan",
    "دکتربیاتی": "Dr. Bayati",
    "دکترتوانا": "Dr. Tavana",
    "دکتررضی": "Dr. Razi",
    "دکترسعیدی مدنی": "Dr. Saeidi Madani",
    "دکترشجاعی": "Dr. Shojai",
    "دکترشریعتی": "Dr. Shariaty"
}

# Rename instructors names to English
merged_df["prof"] = merged_df["prof"].replace(prof_names)
```

Label encoding

Label Encoding in scikit-learn is a method used to convert categorical data into numerical format. It assigns a unique integer to each category, making it compatible with machine learning algorithms that require numeric inputs. However, be cautious with ordinal assumptions, and consider One-Hot Encoding for certain scenarios.

```
from sklearn.preprocessing import LabelEncoder

# Create a LabelEncoder object
label_encoder = LabelEncoder()

# Use the LabelEncoder to encode the 'prof' column in the DataFrame
merged_df['prof'] = label_encoder.fit_transform(merged_df['prof'])

# Create a dictionary to map the original categorical labels to their corresponding encoded numbers
encoded_dict = {label: index for index, label in enumerate(label_encoder.classes_)}

# Convert the dictionary to a DataFrame for easier inspection and further use
encoded_dict = pd.DataFrame(encoded_dict.items(), columns=['profname', 'Encoded number'])
```

```
# A transposed shape of profnames DataFrame
encoded_dict.T
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
profname	Dr. Akhlaghi	Dr. Bayati	Dr. Boroujerdiyan	Dr. Imanfar	Dr. Khosravi	Dr. Kiani	Dr. Najafi	Dr. Rahmati	Dr. Razi	Dr. Rostami	Dr. Saeidi Madani	Dr. Shariaty	Dr. Shojai	Dr. Tavana
Encoded number	0	1	2	3	4	5	6	7	8	9	10	11	12	13

Drop rows with any non-numeric values and reset the indexes.

```
# Convert all columns of the DataFrame to numeric type
# If any non-numeric values are encountered, they will be converted to NaN (Not a Number)
merged_df = merged_df.apply(pd.to_numeric, errors='coerce')
```

```
# Drop rows containing NaN values (rows with non-numeric data)
merged_df = merged_df.dropna()

# Reset the index of the DataFrame after dropping rows with NaN values
merged_df.reset_index(drop=True, inplace=True)
```

```
cleaned_df = merged_df
cleaned_df
```

	ID	HW1	HW2	HW3	HW4	HW5	HW6	HW7	HW8	mid	final	prof
0	40027008	40.0	33.0	0.0	35.0	40.0	40	35.0	0.0	2.2	4.93	5
1	40113901	37.0	33.0	38.0	0.0	40.0	40	40.0	40.0	3.1	8.78	5
2	40122045	24.0	32.0	40.0	20.0	15.0	40	35.0	18.0	4.3	7.54	5
3	40122046	0.0	36.0	40.0	40.0	40.0	40	35.0	40.0	0.1	4.71	5
4	40122047	0.0	0.0	0.0	0.0	0.0	40	0.0	0.0	1.1	1.50	5
...	...	...	...	...	...	...	...	...	...	...	...	...
1267	40132012	0.0	40.0	40.0	36.0	35.0	0	35.0	22.0	3.9	9.48	11
1268	40132013	25.0	20.0	20.0	37.0	40.0	40	40.0	0.0	0.0	8.52	11
1269	40132021	40.0	34.0	40.0	40.0	40.0	40	40.0	40.0	5.6	17.34	11
1270	40132033	0.0	0.0	40.0	38.0	35.0	40	40.0	0.0	2.3	10.03	11
1271	40132048	37.0	39.0	38.0	38.0	35.0	40	40.0	40.0	0.0	11.57	11

1272 rows × 12 columns

2.2 Data Output to CSV

```
# Save the cleaned DataFrame to a CSV file
cleaned_df.to_csv('data_sets/grades.csv', index=False)

# Save the DataFrame containing the encoded mapping to a CSV file
encoded_dict.to_csv('data_sets/prof_dict.csv', index=False)
```

3- Data Visualization

Data visualization is a crucial aspect of data analysis, where complex and voluminous datasets are visually represented using charts, graphs, and other graphical elements. Through visualizations, data analysts can effectively communicate insights, patterns, and trends that might be challenging to discern from raw numbers alone. By presenting data in a visually appealing and intuitive manner, data visualization enables decision-makers to grasp key information quickly, make informed choices, and identify potential opportunities or challenges. It also aids in detecting outliers, understanding data distributions, and validating assumptions. From simple bar charts to intricate interactive dashboards, data visualization empowers analysts to extract meaningful knowledge from data, making it an indispensable tool in the data analysis process.

3.1 Data Overview

An overview of the distribution of grades for students in General Mathematics 1 in comparison between midterm and final exam grades.

The .describe() method is then applied to the selected DataFrame, which generates summary statistics for the columns 'mid' and 'final'. These statistics include count, mean, standard deviation, minimum value, 25th percentile, median, 75th percentile, and maximum value for each column.

```
df = cleaned_df

df[["mid","final"]].describe()
```

	mid	final
count	1272.000000	1272.000000
mean	3.557390	9.860483
std	2.188107	4.952926
min	0.000000	0.000000
25%	1.850000	6.107500

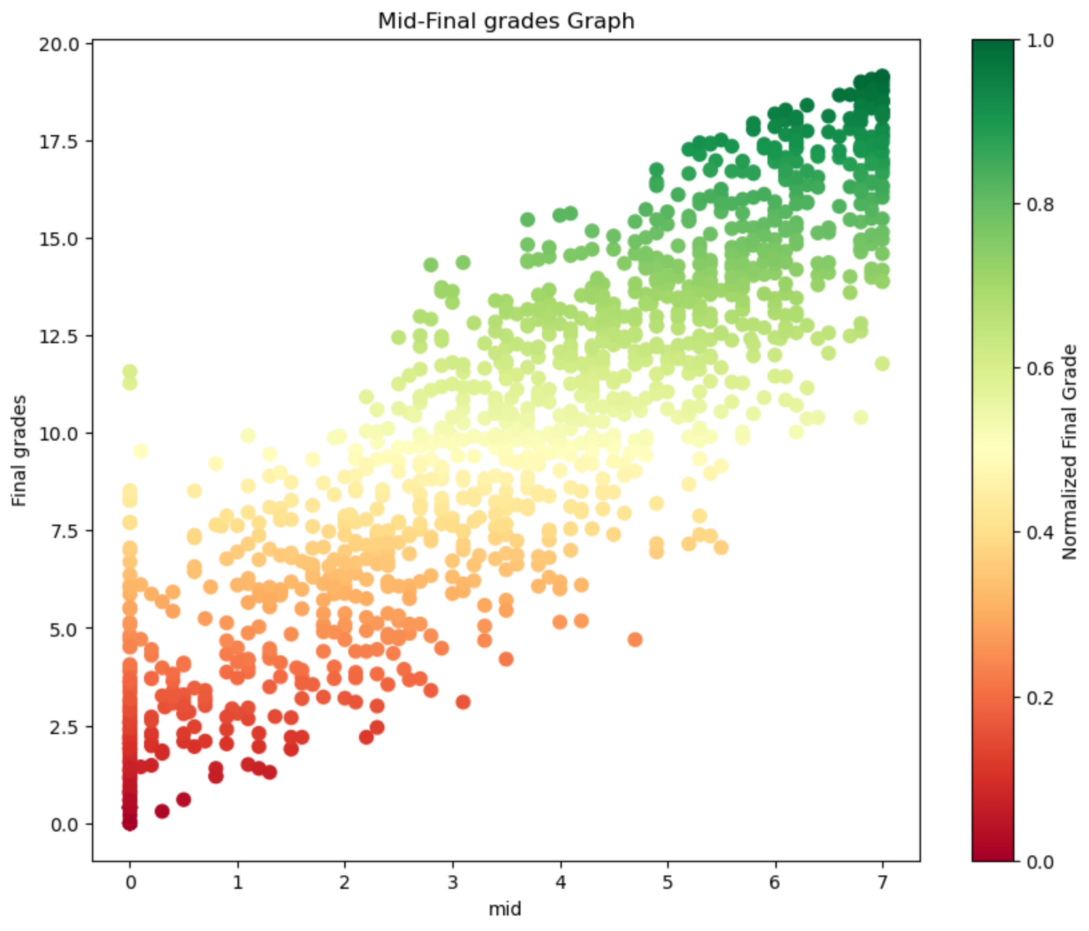
	mid	final
50%	3.700000	10.010000
75%	5.400000	13.802500
max	7.000000	19.140000

Matplotlib

Matplotlib is a Python library for creating high-quality visualizations. It offers a wide range of plot types and customization options, making it ideal for data exploration and presentation. Its straightforward syntax and compatibility with other Python libraries make it a popular choice for data visualization tasks.

```
import matplotlib.pyplot as plt
```

```
max_final = df['final'].max()
cmap = plt.colormaps.get_cmap('RdYlGn')
normalized_final = df['final'] / max_final
plt.figure(figsize=(10, 8))
plt.scatter(df['mid'], df['final'], c=normalized_final, cmap=cmap, marker='o', s=50)
plt.xlabel('mid')
plt.ylabel('Final grades')
plt.title('Mid-Final grades Graph')
plt.colorbar(label='Normalized Final Grade')

plt.show()
```



The resulting scatter plot visualizes the relationship between 'mid' and 'final' grades while providing color-coded insights into the normalized final grades of the students.

3.2 Grades by Professors

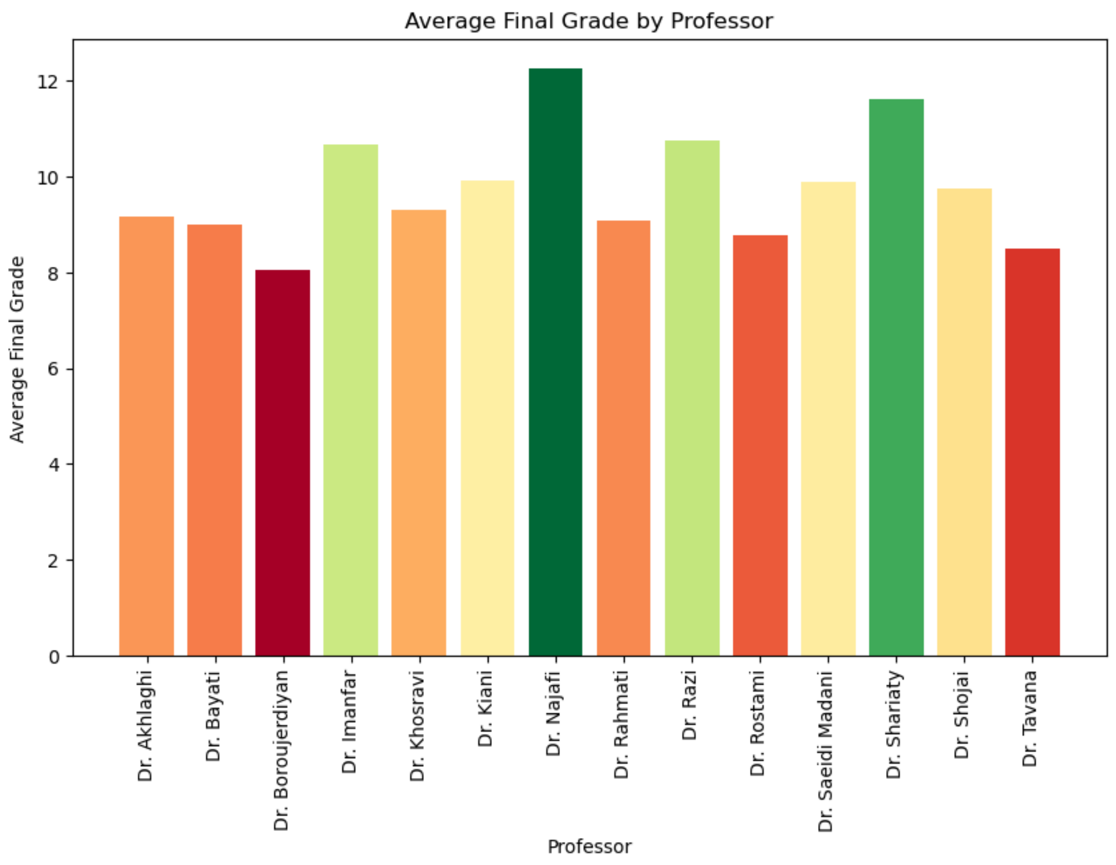
An overview of the distribution of grades for students in General Mathematics 1 will be presented, focusing on the comparison between the midterm and final exam grades. By analyzing and visualizing the grade distributions, we aim to gain insights into the students' performance and understand potential patterns or trends that may have influenced their academic progress throughout the course.

Mean Grades by Professors

```
#Calculate the average final grade for each professor in the DataFrame `df`.
df["prof"] = label_encoder.inverse_transform(df["prof"])
average_final_grade = df.groupby('prof')['final'].mean()

print(average_final_grade)
```

```
prof
Dr. Akhlaghi          9.154219
Dr. Bayati            8.990282
Dr. Boroujerdiyan     8.043429
Dr. Imanfar          10.676857
Dr. Khosravi          9.294724
Dr. Kiani             9.924054
Dr. Najafi           12.252857
Dr. Rahmati           9.070423
Dr. Razi             10.750563
Dr. Rostami           8.762000
Dr. Saeidi Madani     9.897465
Dr. Shariaty         11.624077
Dr. Shojai            9.744924
Dr. Tavana            8.502246
Name: final, dtype: float64
```

```
average_final_grade = df.groupby('prof')['final'].mean()
average_final_grade_dict = average_final_grade.to_dict()
grades = average_final_grade.tolist()

normalized_grades = [(g - min(grades)) / (max(grades) - min(grades)) for g in grades]

cmap = plt.get_cmap('RdYlGn')

plt.figure(figsize=(10, 6))

for professor, norm_grade in zip(average_final_grade.keys(), normalized_grades):
    color = cmap(norm_grade)
    plt.bar(professor, average_final_grade[professor], color=color)

plt.xlabel('Professor')
plt.ylabel('Average Final Grade')
plt.title('Average Final Grade by Professor')
plt.xticks(rotation=90)

sm = plt.cm.ScalarMappable(cmap=cmap, norm=plt.Normalize(vmin=0, vmax=1))
sm.set_array([])

plt.show()
```

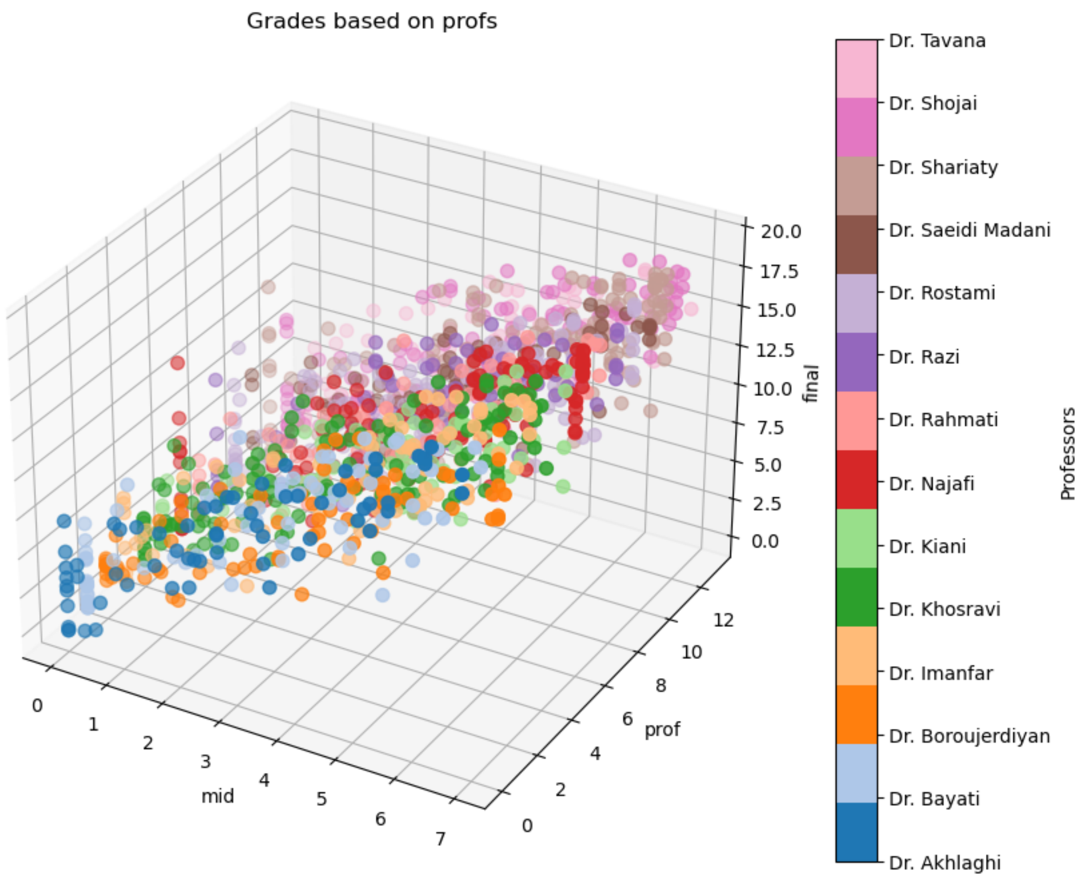


The resulting bar plot visualizes the average final grade for each professor, with colors indicating the relative strength of the normalized grades.

3D vectorization

```
from matplotlib.colors import ListedColormap
```

```
df_prof_named = df

df_prof_named['prof'] = pd.Categorical(df_prof_named['prof'])
num_professors = len(df_prof_named['prof'].cat.categories)
cmap = ListedColormap(plt.cm.tab20.colors[:num_professors])
fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(111, projection='3d')
x, y , z = df_prof_named['mid'], df_prof_named['prof'].cat.codes, df_prof_named['final']
sc = ax.scatter(x, y, z, c=df_prof_named['prof'].cat.codes, cmap=cmap, marker='o', s=50)
ax.set_xlabel('mid')
ax.set_ylabel('prof')
ax.set_zlabel('final')
ax.set_title('Grades based on profs')
cbar = plt.colorbar(sc, ticks=range(num_professors))
cbar.set_label('Professors')
cbar.set_ticklabels(df_prof_named['prof'].cat.categories)

plt.show()
```



The resulting 3D scatter plot visualizes the relationship between 'mid', 'final', and professors. Each professor is represented by a unique color, and their corresponding names are shown on the colorbar for easy identification.

4- Correlation of Clusters with Grades

This study explores the correlation between students' majors and their General Mathematics 1 grades through data plots and statistical analysis. Using tangible visualizations and statistical measures, we aim to identify any potential patterns or trends that link academic disciplines with performance in the course. The results will inform targeted support strategies and curriculum adjustments to enhance students' success in General Mathematics 1.

4.1 Correlation between majors and grades

By student ID, we have categorized grades into majors to find out if there is any correlation between students' majors and their math grades.

```
# Define a dictionary to map numeric major IDs to their corresponding major names
majors = {
    23: "Electrical Engineering",    26: "Mechanical Engineering",
    22: "Chemical Engineering",    33: "Biomedical Engineering",
    34: "Petroleum Engineering",    27: "Mining Engineering",
    39: "Materials Engineering",    24: "Civil Engineering",
    29: "Aerospace Engineering",    11: "Physics and Energy",
    25: "Industrial Engineering",    31: "Computer Engineering",
    13: "Computer Science",    12: "Applied Mathematics",
    30: "Marine Engineering",    32: "Polymer Engineering",
    28: "Textile Engineering"
}

# Extract the two-digit IDs from the 'ID' column and map them to their corresponding major names using the 'majors' dictionary
major_df = df[["ID","mid","final"]]
pd.options.mode.chained_assignment = None
major_df['ID'] = major_df['ID'].astype(str).apply(lambda x: x[3:5])
majors = {str(k): v for k, v in majors.items()}
major_df['ID'] = major_df['ID'].astype(str).replace(majors)

# Calculate the average final grade for each major
major_ranked = major_df.groupby('ID')['final'].mean().reset_index().sort_values(by='final',
ascending=False).reset_index(drop=True)
```

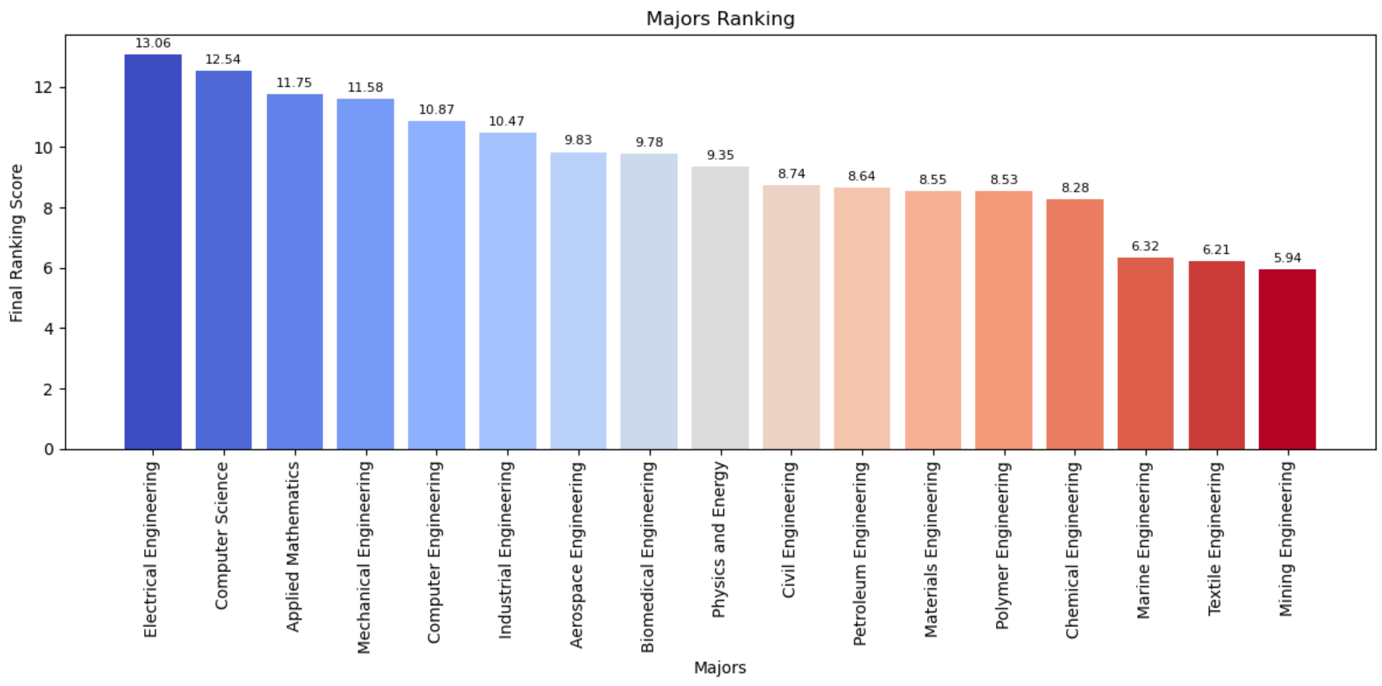
4.2 Mean math grades for majors

Numpy

NumPy is a powerful Python library for numerical computing and data manipulation. It provides efficient multi-dimensional arrays and a vast collection of mathematical functions, making it a foundation for various scientific computing tasks, including data science and machine learning.

```
import numpy as np
```

```
plt.figure(figsize=(12, 6))
#  NumPy is used to create evenly spaced color values for the bars in the bar plot.
colors = plt.cm.coolwarm(np.linspace(0, 1, len(major_ranked)))
bars = plt.bar(major_ranked['ID'], major_ranked['final'], color=colors)
plt.xticks(rotation=90, fontsize=10)
plt.xlabel('Majors')
plt.ylabel('Final Ranking Score')
plt.title('Majors Ranking')
for bar, value in zip(bars, major_ranked['final']):
    plt.annotate(f'{value:.2f}', xy=(bar.get_x() + bar.get_width() / 2, bar.get_height()), xytext=(0, 3),
                 textcoords="offset points", ha='center', va='bottom', fontsize=8)
sm = plt.cm.ScalarMappable(cmap='coolwarm', norm=plt.Normalize(vmin=major_ranked['final'].min(),
vmax=major_ranked['final'].max()))
sm.set_array([])
plt.tight_layout()
plt.show()
```

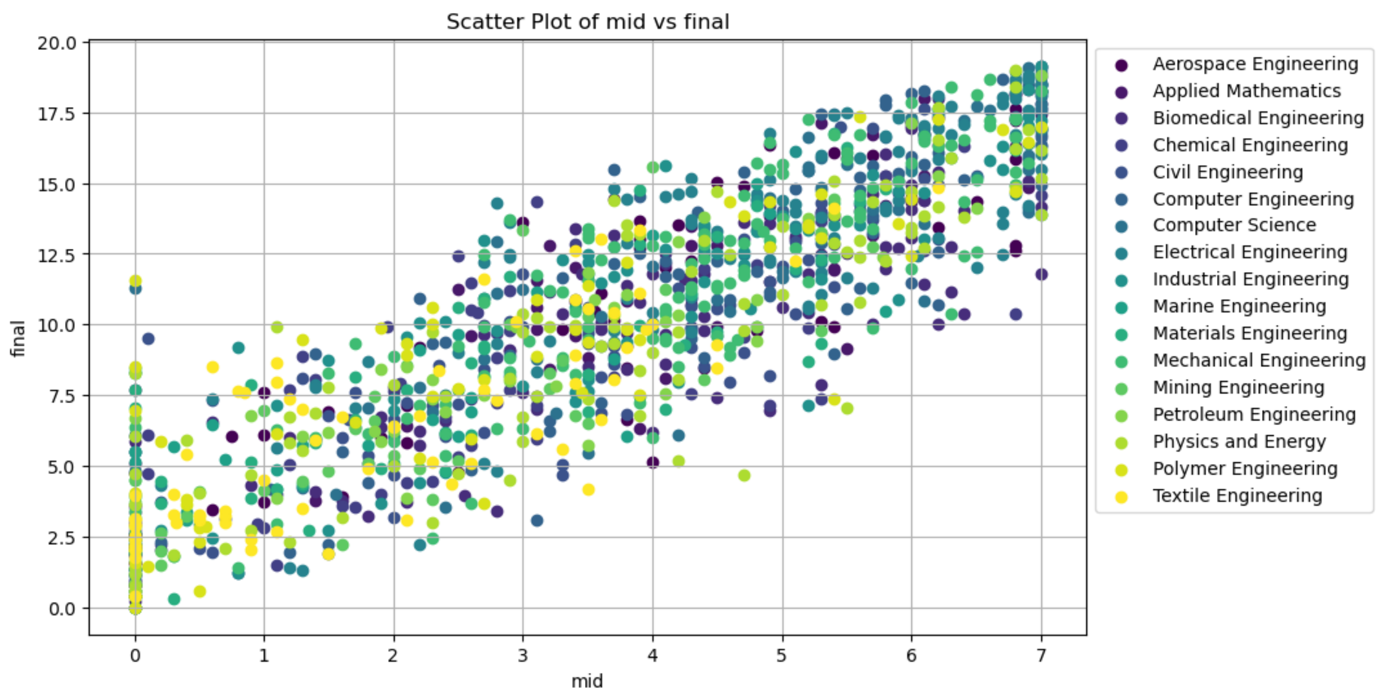


The resulting bar plot visualizes the ranking of majors based on their average final grades. Each major is represented by a colored bar, and the corresponding average final grade is annotated on top of each bar.

4.3 Majors on Mid-Final Plot

The DataFrame presents the mean grades of students in the midterm and final exams for various majors. By exploring this dataset, one can gain valuable insights into the grade distributions across different academic disciplines and identify any trends or patterns in students' performance. Analyzing the mean grades allows for a more comprehensive understanding of how students from different majors performed on average in both exams.

```
# Warnings can help ignore the futher information which ipython can show
import warnings
warnings.filterwarnings('ignore')

cmap = plt.cm.get_cmap('viridis', len(major_df['ID'].unique()))
plt.figure(figsize=(10, 6))
for i, (ID, group) in enumerate(major_df.groupby('ID')):
    plt.scatter(group['mid'], group['final'], color=cmap(i), label=ID)

plt.xlabel('mid')
plt.ylabel('final')
plt.title('Scatter Plot of mid vs final')
plt.legend(loc='upper left', bbox_to_anchor=(1, 1))
plt.grid(True)
plt.show()
```



The resulting scatter plot visualizes the relationship between 'mid' and 'final' grades for each major, with each major's data points represented by a different color from the 'viridis' colormap. The legend allows you to identify each major based on its unique color. The grid lines further aid in interpreting the data.

4. Native and International Students

There can be a correlation between the grades of international and native students. Analyzing this relationship offers insights into their academic performance and helps tailor support programs and teaching methods for improved learning outcomes.

```
# We create int_df DataFrame then replace 0,1,7,9 for Native and 4 for International students
int_df = df[["ID","mid","final"]]
int_df['ID'] = int_df['ID'].astype(str).apply(lambda x: x[5:6])
int_df['ID'] = pd.to_numeric(int_df['ID'], errors='coerce').fillna(0).astype(int)
int_df['ID'].replace([1, 9, 7], 0, inplace=True)
int_df['ID'] = int_df['ID'].replace({0: 'Native', 4: 'International'})
```

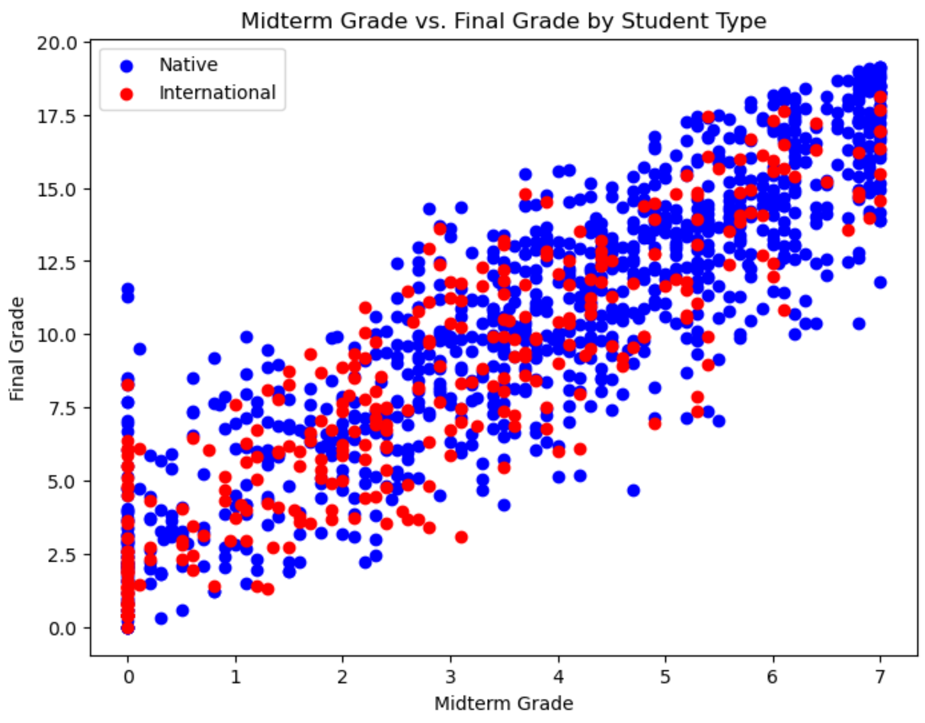
Total amount of Native and International students

```
int_df['ID'].value_counts()
```

```
Native           951
International    321
Name: ID, dtype: int64
```

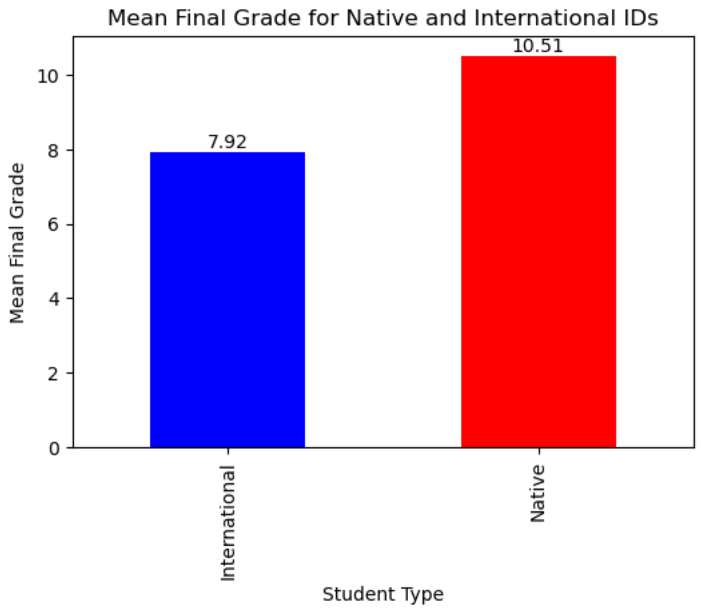
Midterm Grade vs. Final Grade by Student Type

```
native_students = int_df[int_df['ID'] == 'Native']
international_students = int_df[int_df['ID'] == 'International']

plt.figure(figsize=(8, 6))
plt.scatter(native_students['mid'], native_students['final'], color='blue', label='Native')
plt.scatter(international_students['mid'], international_students['final'], color='red', label='International')
plt.xlabel('Midterm Grade')
plt.ylabel('Final Grade')
plt.title('Midterm Grade vs. Final Grade by Student Type')
plt.legend()

plt.show()
```



Mean final grade for Native and International IDs

```
mean_final_grade = int_df.groupby('ID')['final'].mean()

# Plot the bar chart
plt.figure(figsize=(6, 4))
ax = mean_final_grade.plot(kind='bar', color=['blue', 'red'])
plt.xlabel('Student Type')
plt.ylabel('Mean Final Grade')
plt.title('Mean Final Grade for Native and International IDs')
for index, value in enumerate(mean_final_grade):
    ax.text(index, value, f'{value:.2f}', ha='center', va='bottom', fontsize=10)

plt.show()
```



So, it can be concluded that international students may have slightly lower grades in math compared to native students.

5- Machine Learning Model

Machine learning is a branch of artificial intelligence that allows computers to learn from data and make predictions without explicit programming. Simple Linear Regression (SLRM) and Multiple Linear Regression (MLRM) are two common techniques in machine learning. SLRM models the linear relationship between one input and one output variable, while MLRM considers multiple input variables. In educational contexts, SLRM and MLRM can be used to predict a student's final exam score based on their midterm grade, homework grades, or a combination of factors. These predictive models aid educators in identifying at-risk students and providing targeted support, ultimately enhancing academic outcomes through data-driven decisions.

5.1 Data Processing

Data processing in machine learning involves preparing and transforming raw data for model training. It includes data cleaning, feature engineering, encoding categorical variables, data splitting, and scaling. Proper data processing is crucial for improving model performance and ensuring accurate predictions. As we had processed our data before so we can recall the DataFrame by importing `grades.csv`

```
df = pd.read_csv("data_sets/grades.csv")
df.head()
```

	ID	HW1	HW2	HW3	HW4	HW5	HW6	HW7	HW8	mid	final	prof
0	40027008	40.0	33.0	0.0	35.0	40.0	40	35.0	0.0	2.2	4.93	5
1	40113901	37.0	33.0	38.0	0.0	40.0	40	40.0	40.0	3.1	8.78	5
2	40122045	24.0	32.0	40.0	20.0	15.0	40	35.0	18.0	4.3	7.54	5
3	40122046	0.0	36.0	40.0	40.0	40.0	40	35.0	40.0	0.1	4.71	5
4	40122047	0.0	0.0	0.0	0.0	0.0	40	0.0	0.0	1.1	1.50	5

Split data into train and test set

Train-test split is a technique used in machine learning to divide the dataset into training and testing subsets. The training set is used to train the model, while the testing set evaluates its performance on unseen data, ensuring accurate predictions and preventing overfitting.

Scikit-learn

Scikit-learn is a popular Python library for machine learning, offering a wide range of algorithms and tools. It provides the train_test_split function to easily split data into training and testing sets. This helps evaluate model performance, prevent overfitting, and build reliable machine learning models.

```
from sklearn.model_selection import train_test_split
```

```
# For SLRM
slrm_X = df[['mid']]
slrm_y = df['final']

slrm_X_train, slrm_X_test, slrm_y_train, slrm_y_test = train_test_split(slrm_X, slrm_y, test_size=0.2, random_state=42)

# For MLRM
mlrm_X = df[['HW1', 'HW2', 'HW3', 'HW4', 'HW5', 'HW6', 'HW7', 'HW8', 'mid', 'prof']]
mlrm_y = df['final']

mlrm_X_train, mlrm_X_test, mlrm_y_train, mlrm_y_test = train_test_split(mlrm_X, mlrm_y, test_size=0.2, random_state=42)
```

5.2 Simple Linear Regression Model

Simple Linear Regression is a statistical technique to model the linear relationship between a dependent variable $(Y)$ and an independent variable $(X)$ as: $Y = β0 + β1 * X + ε$. It assumes linearity, independence, homoscedasticity, and normality of residuals.

Scikit-learn, a popular Python library, offers an efficient implementation of Simple Linear Regression. It simplifies model training, prediction, and evaluation. By using scikit-learn, you can preprocess data, split it into training and testing sets, and assess the model's accuracy with metrics like Mean Squared Error or R-squared.

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
```

```
# Create a linear regression object and Fit the model to the training data
model = LinearRegression()
model.fit(slrm_X_train, slrm_y_train)

# Predict using the model on the testing set
slrm_y_test_pred = model.predict(slrm_X_test)
```

What we are looking for are the values of $w$ (Weight) and $b$ (Bias) to place in our simple linear regression model:

$$ f_{w,b}(x) = wx + b \tag{1}$$

Scikit-learn will present this values after calculatoin as coefficient for w and intercept for b by $(w,b) = $`(model.coef_ , model.intercept_)`

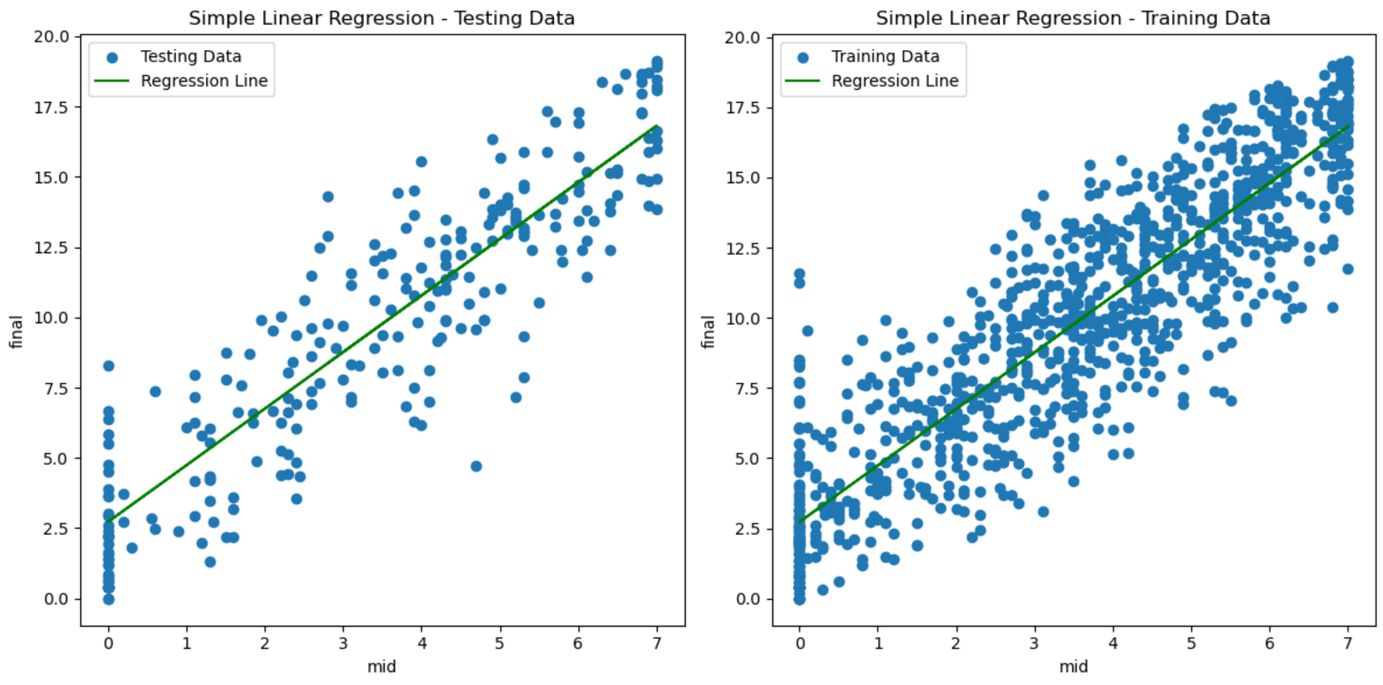
```
print(f"f(x) = {float(model.coef_):.3f}X + {model.intercept_:.3f}")
```

```
f(x) = 2.014X + 2.723
```

$$ f_{w,b}(x) = 2.014x + 2.723 \tag{1}$$

Plot the data and the regression line for the testing set

```
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 6))
ax1.scatter(slrm_X_test, slrm_y_test, label="Testing Data")
ax1.plot(slrm_X_test, slrm_y_test_pred, color='green', label="Regression Line")
ax1.set_xlabel('mid')
ax1.set_ylabel('final')
ax1.legend()
ax1.set_title('Simple Linear Regression - Testing Data')
ax2.scatter(slrm_X_train, slrm_y_train, label="Training Data")
ax2.plot(slrm_X_test, slrm_y_test_pred, color='green', label="Regression Line")
ax2.set_xlabel('mid')
ax2.set_ylabel('final')
ax2.legend()
ax2.set_title('Simple Linear Regression - Training Data')
plt.tight_layout()
plt.show()
```



The resulting figure will show two subplots, one for testing data and the other for training data, with the regression line plotted over the data points in both cases. This allows for a visual comparison between the performance of the simple linear regression model on the testing and training data.

Calculate error metrics on the testing set

Calculating error metrics on the testing set is a crucial step in assessing how well a machine learning model performs on unseen data. Common metrics include Mean Absolute Error (MAE), Mean Squared Error (MSE) and R2-score for regression. These metrics help in model evaluation, comparison, and fine-tuning to build more accurate and reliable machine learning models.

```
mae_test = mean_absolute_error(slrm_y_test, slrm_y_test_pred)
mse_test = mean_squared_error(slrm_y_test, slrm_y_test_pred)
r2_test = r2_score(slrm_y_test, slrm_y_test_pred)
```

Mean Absolute Error (MAE) is the mean of the absolute value of the errors:

$$\frac 1n\sum_{i=1}^n|y_i-\hat{y}_i|$$

```
print("Testing Set - Mean Absolute Error (MAE): {:.2f}".format(mae_test))
```

```
Testing Set - Mean Absolute Error (MAE): 1.75
```

Mean Squared Error (MSE) is the mean of the squared errors:

$$\frac 1n\sum_{i=1}^n(y_i-\hat{y}_i)^2$$

```
print("Testing Set - Mean Squared Error (MSE): {:.2f}".format(mse_test))
```

```
Testing Set - Mean Squared Error (MSE): 4.66
```

R-squared represents the proportion of the variance in the dependent variable

$$1 - \frac{\sum (y_i - \bar{y})^2}{\sum (y_i - \hat{y}_i)^2}$$

```
print("Testing Set - R-squared (coefficient of determination): {:.2f}".format(r2_test))
```

```
Testing Set - R-squared (coefficient of determination): 0.82
```

5.3 Multi Linear Regression Model

MLRM (Multiple Linear Regression Model) is a statistical technique used to model the relationship between a dependent variable and two or more independent variables. The model assumes a linear relationship, and it can be represented as:

$$y = \beta_0 + \beta_1 \cdot x_1 + \beta_2 \cdot x_2 + \ldots + \beta_n \cdot x_n + \varepsilon$$

Where:

- $y$ is the dependent variable.
- $(x_1, x_2, \ldots, x_n)$ are the independent variables.
- $(\beta_0, \beta_1, \beta_2, \ldots, \beta_n)$ are the coefficients.
- $(\varepsilon)$ is the error term.

Scikit-learn, a Python library, provides an easy-to-use implementation of MLRM through the `LinearRegression` class. You can create and train a model, estimate coefficients, and make predictions using this class.

Train the Multiple Linear Regression Model:

```
# Create a Multiple Linear Regression model
mlrm = LinearRegression()

# Fit the Multiple Linear Regression model to the training data
mlrm.fit(mlrm_X_train, mlrm_y_train)

# Calculate the R-squared score for the training data
train_score = mlrm.score(mlrm_X_train, mlrm_y_train)

# Calculate the R-squared score for the testing data
test_score = mlrm.score(mlrm_X_test, mlrm_y_test)
```

Final Function which is in form: $$f_{w,b}(x^{(i)}) = \sum_{n=1}^i(w^{(i)} x^{(i)}) + b $$

Scikit-learn represents coefitients as `mlrm.coef_` and intercept as `mlrm.intercept_` so we can extract $f_{w,b}$ as:

```
coefficients = mlrm.coef_
feature_names = mlrm_X.columns

formula = "Final Grade = "
for i, (coefficient, feature_name) in enumerate(zip(coefficients, feature_names)):
    if i == 0:
        formula += f"{coefficient:.2f} * {feature_name}"
    else:
        formula += f" + {coefficient:.2f} * {feature_name}"
```

```
# Print the clean formula
print(f"{formula} + {mlrm.intercept_:.2f}")
```

```
Final Grade = 0.02 * HW1 + 0.01 * HW2 + 0.01 * HW3 + 0.02 * HW4 + 0.02 * HW5 + 0.02 * HW6 + 0.02 * HW7 + 0.02 * HW8 + 1.79 * mid
+ -0.02 * prof + -0.27
```

$$f_{w^{(i)},b}(x^{(i)}) = 0.02HW^{(1)} + 0.01HW^{(2)} + 0.01HW^{(3)} + 0.02HW^{(4)} + 0.02HW^{(5)} + 0.02HW^{(6)} + 0.02HW^{(7)} + 0.02HW^{(8)} + 1.79mid - 0.27$$

Score the model

R-squared represents the proportion of the variance in the dependent variable

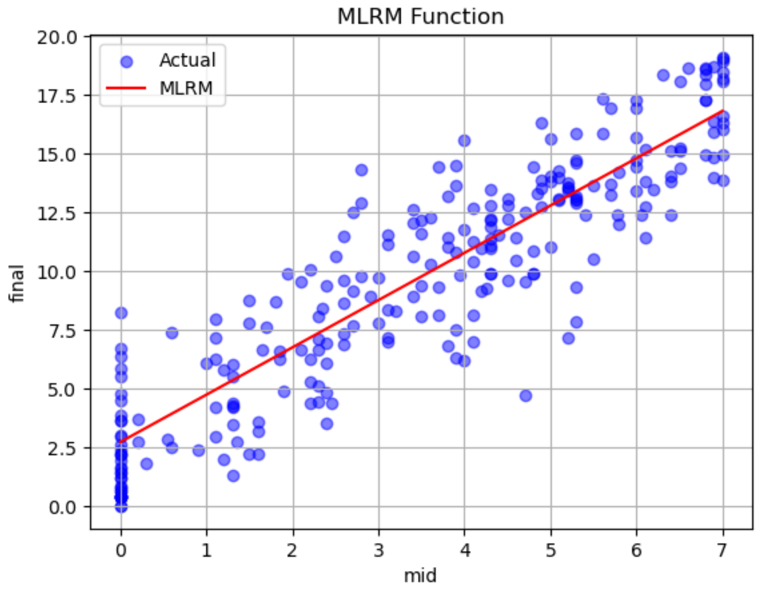
$$1 - \frac{\sum (y_i - \bar{y})^2}{\sum (y_i - \hat{y}_i)^2}$$

```
print("Testing R-squared score:", test_score)
```

```
Testing R-squared score: 0.8826555536699574
```

Plotting the predicted values against the actual values in the test set

```
plt.scatter(mlrm_X_test['mid'], mlrm_y_test, label='Actual', color='blue', alpha=0.5)

# Sort the mid values to create a smooth curve
sorted_mid = mlrm_X_test['mid'].sort_values()

# Predict the final grades using the sorted mid values and plot the curve
plt.plot(sorted_mid, model.predict(pd.DataFrame(sorted_mid, columns=['mid'])), label='MLRM', color='red')

plt.xlabel('mid')
plt.ylabel('final')
plt.title('MLRM Function')
plt.legend()
plt.grid(True)
plt.show()
```



The resulting plot will show the actual final grades as blue points and the MLRM curve in red, representing the predicted final grades based on the 'mid' values. The smooth curve provides a visual representation of how the model predicts the 'final' grades as the 'mid' values change.

6- Conclusion

6.1 Project results

In conclusion, the research sheds light on the complex dynamics of student grades at Amirkabir University of Technology. The significant grade variability indicates the diverse academic abilities among students in general math 1. Although instructors have a limited direct impact, they can influence the likelihood of achieving higher grades, as seen in Dr. Najafi and Shariaty's classes. Additionally, the link between students' majors and grades reveals that Mathematics and challenging engineering disciplines lead to higher average grades, while Mining, Textile, and Marine Engineering exhibit lower averages. Additionally, the average final grade is lower among international students compared to native ones; however, international students with grades above 5 exhibit a well-distributed pattern among higher grades. These findings underline the importance of considering multiple factors, such as instructors and majors, to better understand and support student's academic achievements.

6.2 Answer to goals

Understanding Grade Distribution of Students.

```
At Amirkabir University of Technology, the distribution of student grades in general math 1 shows significant variability,
covering a wide range from 0 to 19.4. The average grade is 9.86, reflecting the central tendency, while the standard deviation
of 4.9 indicates the spread or dispersion of the grades around the mean. This diverse distribution of grades suggests varying
levels of performance and achievement among the students in the course.
```

Exploring the Correlation between Instructors and Average Grades.

```
Although the instructor's direct impact on grading is relatively limited, the chance of passing and achieving higher grades is
easier in Dr. Najafi and Shariaty's classes, while it is more challenging in Dr. Broojerdian and tavana's class.
```

Investigating the Relationship Between Students' Majors and Grade Status.

```
One significant difference found in valuable research achievements was the impact of the field of study on students' average
grades. Students in the Faculty of Mathematics and challenging engineering disciplines had the highest average grades, with
Electrical and Mechanical Engineering leading, while the lowest average grades were observed in Mining, Textile, and Marine
Engineering.
```

Finding the relationships between international and native students' grades.

```
International students tend to have lower grades than native students. However, both groups show similar grade distributions for
scores above 5, suggesting comparable academic performance in higher grades. These findings highlight the need to support
international students initially and promote an inclusive environment for continued success.
```

Predicting Final Grades based on Midterm and Homework Scores.

```
With the assistance of two machine learning models based on linear regression, namely SLRM (Simple Linear Regression) and MLRM
(Multiple Linear Regression), we were able to provide formulas that enable us to predict students' grades before the final exam.
These models have high accuracy in predicting grades, allowing us to make timely decisions regarding whether to drop or continue
their courses.
```

Providing a Clean Dataset of Grades for Future Research Purposes.

```
We have compiled and provided a clean dataset of grades, meticulously organized and free from errors or inconsistencies,
specifically intended for future research purposes. This dataset serves as a valuable resource for researchers and educators
seeking to analyze and explore various aspects of academic performance and related factors.
```

With the help of data processing, including data transfer from Excel to CSV, removal of redundant entries, standardization of the data frame into English, and elimination of non-numeric data, we have facilitated the ability for researchers to further examine the dataset more easily. Those interested in in-depth analysis can now delve into the data with greater ease and efficiency for their research purposes.

In addition to the initial analysis, we can further explore the data with the following potential items:

- **Fraud Detection based on Grade Records:** Implementing algorithms and statistical methods to identify any anomalies or suspicious patterns in the grades data that might indicate academic fraud or misconduct.
- **Student Clustering:** Utilizing clustering techniques to group students based on similar academic performance, study habits, or other relevant factors. This can provide valuable insights into different student profiles and help personalize educational approaches.
- **More Accurate Professors' Performance Analysis:** Conducting a more comprehensive evaluation of professors' performance by identifying and handling unusual grade distributions. This can involve investigating grade inflation, grading consistency, and other factors affecting the quality of education.

- **Trend Analysis over Time:** Analyzing the data across multiple academic years to uncover long-term trends and patterns in student performance, course popularity, and other relevant metrics.
- **Predictive Modeling:** Building predictive models to forecast future student performance, course enrollments, or other key indicators. This can aid in proactive decision-making and resource planning.

6.4 Contact and Update project

To continue contributing to the project or explore opportunities for future collaborations, I invite you to actively participate by creating a pull request on the designated GitHub

Repository: kooroshkz/math_grades_analysis

Your contributions and feedback are highly valued, and together we can enhance the analysis and uncover valuable insights from the data.

If you have any questions, suggestions, or innovative ideas, don't hesitate to reach out to me directly:

Contact:

Email: kkomeilizadeh@gmail.com

Linkedin: Koorosh Komeilizadeh

I look forward to collaborating with you and making meaningful advancements in the project. Let's work together to excel in the realm of data analysis!

Thanks for your attention

Proudly presented by Koorosh Komeilizade

kooroshkz.com