# ECE 180- Probability Theory and Stochastic Processes

## Project Report: Building a CGPA Predictor: Merging Probability and Data Visualization

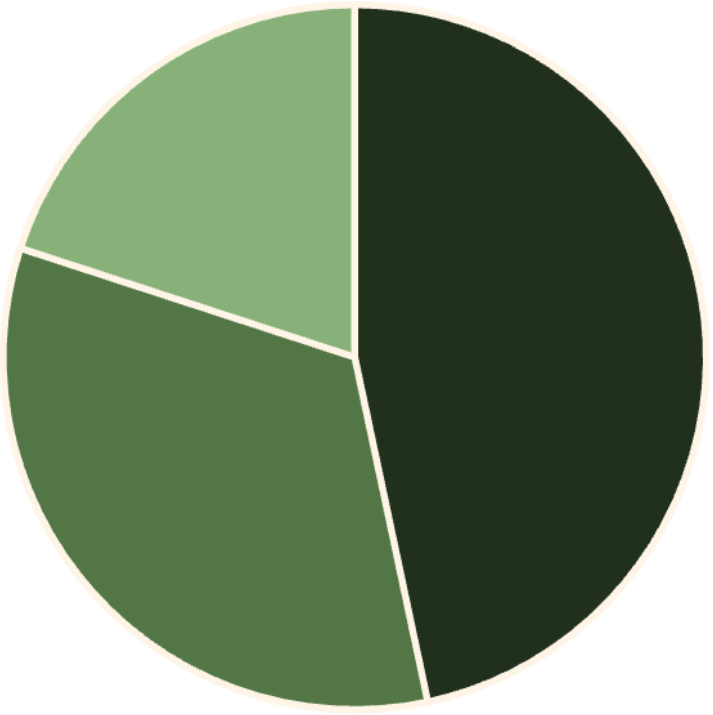
| |
|---|
| Date- 16/1/24 |
| Submitted By- Mohammad Sharique Arshad |
| Section- E2301- G2 |
| Roll No.- 33 |
| Registration No.- 12310194 |
| Submitted To- Dr. Manoj Singh Adhikari |

# Index

# Introduction

Welcome to an exciting journey into the world of academic performance prediction! In this project, I'll create a CGPA (Cumulative Grade Point Average) Predictor that combines probability modeling with data visualization. This tool will not only calculate a student's current CGPA but also forecast future academic performance, presenting the results through engaging pie charts.

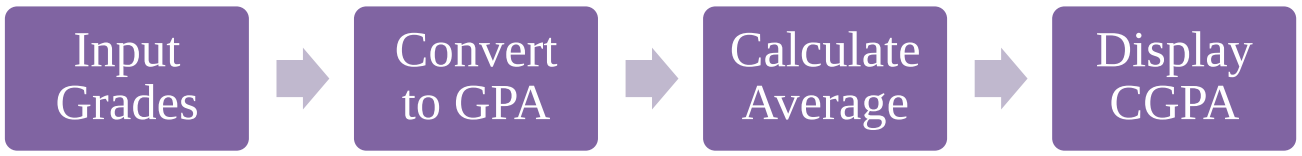
As we dive into this project, we'll explore key concepts in Python programming, probability theory, and data visualization using matplotlib. Whether you're a budding data scientist or a curious student, this project will provide valuable insights into practical applications of these fields. Let's embark on this educational adventure and unlock the power of predictive analytics!



# Understanding the CGPA Calculation

But we start to discuss prediction, so the mechanism for calculating a cumulative grade point average needs to be understood, as that will be at the core of measuring how an individual did with respect to academics overall. The CGPA is an average performance by a student for all the courses undertaken until then, thereby giving a simple, integrated measure of achievements. To compute it, each letter grade a student gets, say A, B, or C, is translated into an equivalent numerical value within a given scale. These numerical values or grade points are, therefore, a reflection of the performance of each student in every course she or he takes.

In our project, we will use a commonly accepted grade-to-GPA conversion scale. Each grade will be assigned a specific point value, such as an 'A' grade might be 10.0 points, and a 'B' might be 7.0 points. This system of conversion allows for the standardization of academic performance to be represented numerically in order to compare across different courses and semesters.



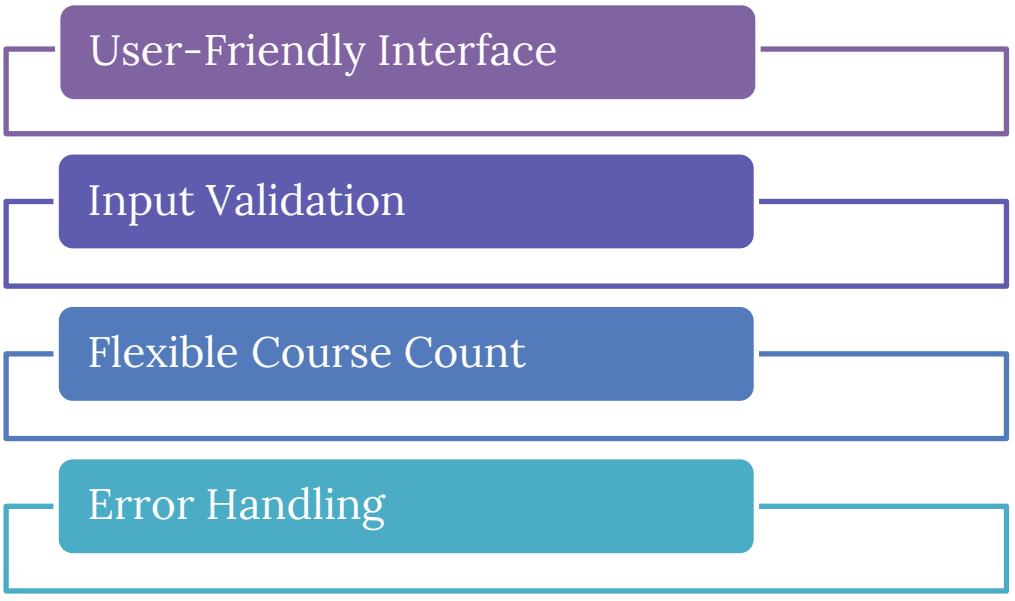

We then determine grade points for every grade calculated. We further proceed with the calculation of averages using all these points. Then the CGPA will emerge since such an average would imply summation of student's performances over time and forms an ideal basis for the forecasted performances in our project.

# Implementing the Grade Input System

Our CGPA Predictor will begin with a reliable grade input system designed to be flexible and accurate. We will develop a function, `get_grades()`, which prompts the user to input his grades for multiple courses. This function will accept the grade inputs until the user signals that he has done so for any number of courses.

For error checking on the accuracy of the data entry, each grade is allowed to have a mechanism in place to ensure an entry is correct before acceptance and calculation for the CGPA. Whenever the invalid grade is entered, it then asks the user to correct the entry of a grade. That way, it prevents incorrect entries from skewing the calculation in the computation of CGPA.

Thus, with input flexibility and strong error checks, the `get_grades()` function makes for a convenient process of providing input on grades from students with as much accuracy as possible to form a foundation for our CGPA Predictor in giving correct calculations. That way, careful input checking makes each grade the true reflection of performance during academics, resulting in an accurate prediction for the final CGPA as possible.



# Designing the Probability Model

Our CGPA Predictor is built around a probability model that estimates the chances of achieving various CGPA ranges in the future. While simplified for educational purposes, this model illustrates how previous performance can help forecast future outcomes.

The core function, `predict_cgpa_range()`, accepts the current CGPA as input and returns probabilities for specific CGPA ranges. These ranges are typically categorized as follows: high (8.5–10.0), medium-high (7.0–8.5), medium (5.0–7.0), and low (3.5–5.0). This categorization allows us to quantify the likelihood of reaching each range based on past performance trends, providing students with insights into their potential academic trajectory.

| Current CGPA: 8.5+ | Current CGPA: 7.0-8.49 | Current CGPA: Below 7.0 |
|---|---|---|
| High chance of maintaining excellence<br>• 70% for 8.5-10.0<br>• 25% for 7.0-8.49<br>• 5% for 5.5-7.0 | Good chance of improvement<br>• 10% for 8.5-10.0<br>• 60% for 7.0-8.49<br>• 25% for 5.5-7.0<br>• 5% for 4.0-5.5 | Higher chance of lower ranges<br>• 5% for 7.0-8.49<br>• 20% for 5.5-7.0<br>• 60% for 4.0-5.5<br>• 15% below 4.0 |

# Implementing the Python Code:

```
import numpy as np
import matplotlib.pyplot as plt
# Grade to GPA conversion
grade_to_gpa = {
    'A+': 10, 'A': 9, 'A-': 8,
    'B+': 7, 'B': 6, 'B-': 5,
    'C+': 4, 'C': 3, 'C-': 2,
    'D+': 1, 'D': 0, 'F': 0
}
def get_grades():
    grades = []
    subjects = ['Probability and Statistics', 'Analog Electronics', 'Digital Electronics', 'PEL',
'Signals and Systems']
    for subject in subjects:
        while True:
            grade = input(f"Enter grade for {subject} (A+, A, A-, B+, B, B-, C+, C, C-, D+, D,
F): ").upper()
            if grade in grade_to_gpa:
                grades.append(grade)
                break
            else:
                print("Invalid grade. Please try again.")
    return grades
def calculate_cgpa(grades):
    total_points = sum(grade_to_gpa[grade] for grade in grades)
    return total_points / len(grades)
def predict_cgpa_range(current_cgpa):
    # Simple probability model
    if current_cgpa >= 8:
        return [0.7, 0.25, 0.05, 0]
    elif current_cgpa >= 6:
        return [0.1, 0.6, 0.25, 0.05]
    elif current_cgpa >= 4:
        return [0.05, 0.2, 0.6, 0.15]
    else:
        return [0, 0.1, 0.3, 0.6]
def plot_grade_distribution(grades):
    grade_counts = {grade: grades.count(grade) for grade in set(grades)}
```

```
    plt.figure(figsize=(10, 5))
    plt.pie(grade_counts.values(), labels=grade_counts.keys(), autopct='%1.1f%%')
    plt.title("Grade Distribution")
    plt.axis('equal')
    plt.show()
def plot_cgpa_prediction(probabilities):
    labels = ['8-10', '6-7.9', '4-5.9', '0-3.9']
    plt.figure(figsize=(10, 5))
    plt.pie(probabilities, labels=labels, autopct='%1.1f%%')
    plt.title("CGPA Range Probability Prediction")
    plt.axis('equal')
    plt.show()
# Main program
grades = get_grades()
cgpa = calculate_cgpa(grades)
probabilities = predict_cgpa_range(cgpa)
print(f"\nCalculated CGPA: {cgpa:.2f}")
print("\nProbability of achieving CGPA ranges:")
for range, prob in zip(['8-10', '6-7.9', '4-5.9', '0-3.9'], probabilities):
    print(f"{range}: {prob*100:.1f}%")
plot_grade_distribution(grades)
plot_cgpa_prediction(probabilities)
```

# Putting It All Together

After constructing all the parts of our CGPA Predictor, the next step is to put them all together to form a complete program. We first invoke get_grades() for user input, after which we call calculate_cgpa() to find out the current CGPA. Following that, we will include this CGPA into predict_cgpa_range() in order to obtain our predictions. At last, we will present the outputs and turn on the visualisation functions that will help generate the pie charts.

The final part of our programs - script - will be most concerned with how the data flows from one function we've written to other functions - for example, from the input function to the calculation and to the drawing function.

| Step | Function | Purpose |
|---|---|---|
| **1** | get_grades() | Collect grade inputs from user |
| **2** | calculate_cgpa() | Compute current CGPA |
| **3** | predict_cgpa_range() | Generate probability predictions |
| **4** | plot_grade_distribution() | Create grade distribution chart |
| **5** | plot_cgpa_prediction() | Create CGPA prediction chart |

# Output Obtained

## Input given by user

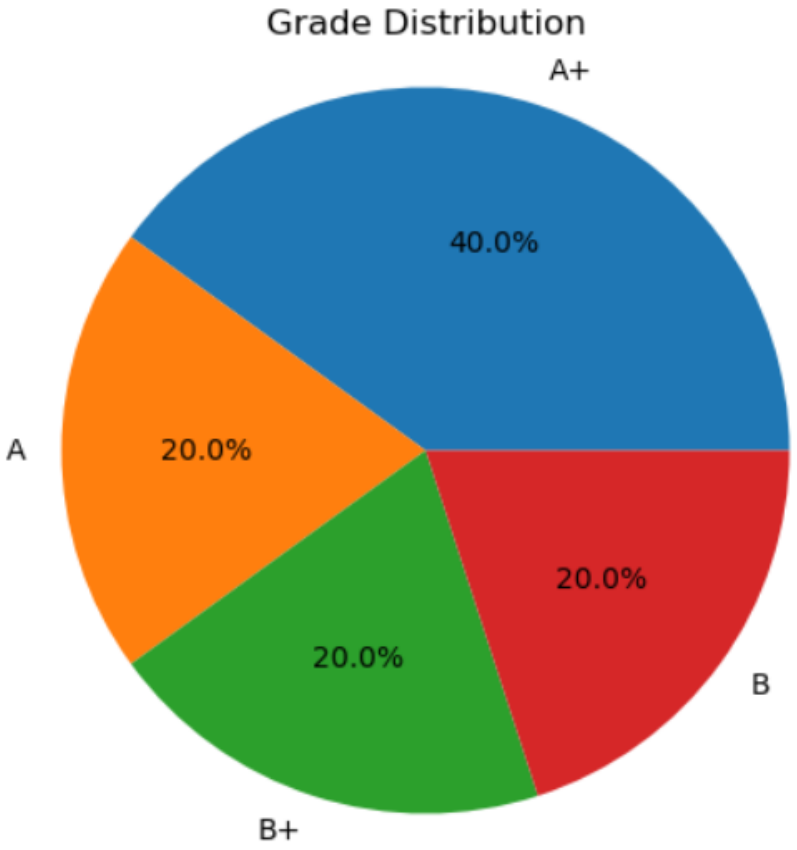
```
Enter grade for Probability and Statistics (A+, A, A-, B+, B, B-, C+, C, C-, D+, D, F): A+
Enter grade for Analog Electronics (A+, A, A-, B+, B, B-, C+, C, C-, D+, D, F): A
Enter grade for Digital Electronics (A+, A, A-, B+, B, B-, C+, C, C-, D+, D, F): A+
Enter grade for PEL (A+, A, A-, B+, B, B-, C+, C, C-, D+, D, F): B
Enter grade for Signals and Systems (A+, A, A-, B+, B, B-, C+, C, C-, D+, D, F): B+
```

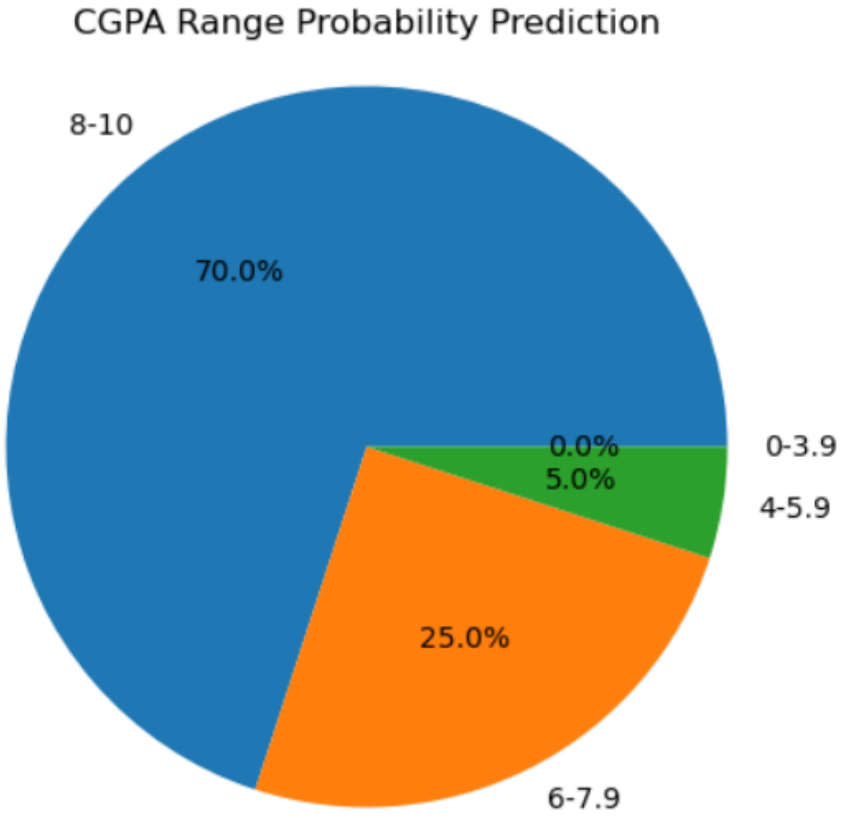
## Output Calculated

```
Calculated CGPA: 8.40

Probability of achieving CGPA ranges:
8-10: 70.0%
6-7.9: 25.0%
4-5.9: 5.0%
0-3.9: 0.0%
```

# Pie Chart Obtained From Output

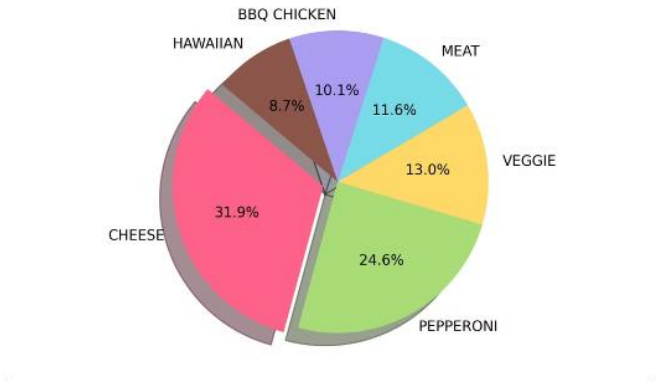


# Visualizing Grade Distribution

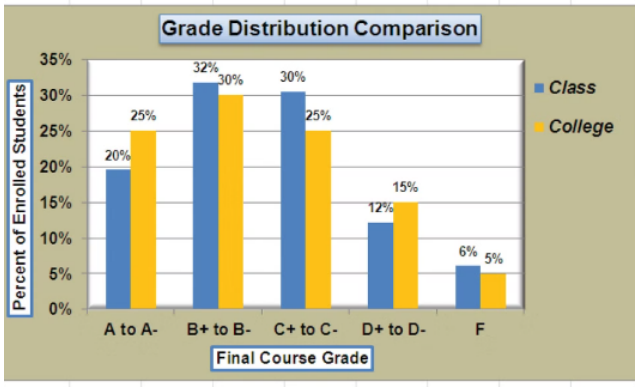
Visualization is key to understanding data patterns. In our CGPA Predictor, we'll create a pie chart to display the distribution of grades entered by the user. This visual representation allows students to quickly grasp their overall performance across different courses.

We'll use matplotlib, a powerful plotting library in Python, to create this chart. The plot_grade_distribution() function will take the list of grades as input and generate a colorful pie chart showing the percentage of each grade received.




Matplotlib Code
The Python code snippet using matplotlib to create a pie chart of grade distribution


Resulting Pie Chart
The colorful pie chart showing the percentage breakdown of different grades




Data Analysis
A student examining their grade distribution to identify strengths and areas for improvement

# Future Enhancements

While our CGPA Predictor is already a powerful tool, there's always room for improvement and expansion. As you become more comfortable with the concepts and code, consider these potential enhancements to take your project to the next level:

You could implement a more sophisticated probability model that takes into account factors like course difficulty or historical grade trends. Another exciting addition would be a graphical user interface (GUI) using a library like Tkinter, making the application more user-friendly and visually appealing.

**Advanced Probability Model**
Incorporate machine learning algorithms to improve prediction accuracy based on larger datasets and more variables

**Graphical User Interface**
Develop a user-friendly GUI using Tkinter or PyQt to make the application more accessible to non-technical users

**Broaden Subject and Course Offerings**
Expand the platform’s library to cover a wider variety of subjects and courses, catering to different fields of study and professional interests

**Additional Visualizations**
Create more advanced charts and graphs, such as line plots showing grade trends or heatmaps of performance across different subjects

# Future Uses of This Model

**Academic Planning for Students**

Course and Module Selection

Goal Setting and Progress Tracking

**Career Counselling and Guidance**

Career Path Exploration

Skills and Competency Development

**Research in Education**

Understanding Performance Influencers

Educational Policy and Program Design

**Personalized Learning and Academic Support**

Customized Study Recommendations

Resource Allocation and Tutoring

**Long-term Academic and Career Planning Tools**

Advanced Data for Academic Advisors

Predictive Analytics for Lifelong Learning

**Institutional Performance and Curriculum Enhancement**

Faculty Development and Training

Curriculum Design and Improvement

With these varied applications, CGPA prediction tools have the potential to transform both individual learning experiences and broader educational systems, fostering a culture of personalized, data-informed education.

# Conclusion

As we wrap up our journey with the CGPA Predictor project, it’s valuable to consider the broader impact of this tool. More than a simple grade calculator, the CGPA Predictor introduces users to the realm of predictive analytics and its promising applications in education.

By integrating probability models and data visualization, the CGPA Predictor empowers students to understand their academic performance better and set realistic improvement goals. This tool not only provides a clear picture of where students currently stand but also helps them identify areas that may need additional focus, offering guidance on their academic journey.

Additionally, this project serves as a practical entry point into essential topics like programming, statistics, and data science. These fields are increasingly relevant in today’s data-focused world, where decision-making is often driven by analytical insights.

The CGPA Predictor project combines these disciplines to create a meaningful and functional tool that highlights the practical use of data science in education. It’s a stepping stone for students and educators alike, inspiring them to explore and leverage data in ways that support personal and academic growth.