

Project 3

Data Visualization & EDA

Introduction to Data Science with Python

Computer Science Program

Kutaisi International University

Project Information

Total Weight: 6% of Final Grade

Task Breakdown: 3 Tasks \times 2% each

Due Date: End of Week 8 (Sunday, 23:59 GT)

Format: Individual Assignment

Deliverable: Jupyter Notebook (.ipynb) or Python Script (.py)

Contents

1 Project Overview

1.1 Introduction

Data visualization is a critical skill in data science—it's how we communicate insights, discover patterns, and tell compelling stories with data. In this project, you will create a comprehensive set of visualizations to explore student academic performance data, uncover relationships between variables, and present your findings in a clear, professional manner.

You'll work with Matplotlib for foundational plotting, Seaborn for statistical visualizations, and combine both to create a cohesive visual narrative that answers key questions about student success factors.

Important: Data Files

You will be provided with a Python script (`generate_project3_data.py`) that generates a comprehensive student performance dataset with 500 students and 27 features including demographics, academic metrics, study habits, and lifestyle factors.

Download the data generator script from:

- LMS course materials folder
- Or instructor will provide via email/announcement

Run the script: `python generate_project3_data.py`

This will create: `student_performance.csv`

1.2 Learning Objectives

Upon completion of this project, you will be able to:

- **Matplotlib Fundamentals:** Create line plots, scatter plots, bar charts, histograms, and subplots
- **Seaborn Mastery:** Utilize advanced statistical visualizations (heatmaps, violin plots, pair plots)
- **Exploratory Data Analysis:** Systematically investigate data to discover patterns and relationships
- **Statistical Thinking:** Identify correlations, distributions, and group differences
- **Design Principles:** Apply color theory, layout, and labeling for effective visualizations
- **Data Storytelling:** Combine multiple visualizations to communicate insights clearly
- **Critical Analysis:** Interpret visualizations and draw evidence-based conclusions

Task	Focus Area	Weight
Task 1	Matplotlib Fundamentals	2%
Task 2	Seaborn & Statistical Analysis	2%
Task 3	Comprehensive Visual Story	2%
Total		6%

1.3 Project Structure

Critical Deadline Information

Academic Integrity Policy:

- All work must be your own individual effort
- You may consult official documentation (Matplotlib, Seaborn, Pandas) and course materials
- Copying code from online sources, AI tools, or other students will result in zero points
- If you reference any external resources for concepts or visualization techniques, cite them in comments or markdown cells

2 Task Specifications

2.1 Task 1: Matplotlib Fundamentals (2%)

Objective

Master the basics of Matplotlib by creating fundamental visualizations that explore different aspects of the student performance dataset.

Requirements

Part A: Individual Plot Types (40%)

Create the following individual visualizations using Matplotlib:

1. Line Plot:

- Plot the relationship between Study Hours and GPA
- Sort data by study hours before plotting
- Include title, axis labels, and grid
- Use appropriate color and line style

2. Scatter Plot:

- Attendance Rate (x-axis) vs Final Average (y-axis)
- Color points by Major (use different colors for each)
- Include legend, title, and axis labels
- Add transparency (alpha) to see overlapping points

3. Bar Chart:

- Average GPA by Major
- Sort bars from highest to lowest GPA
- Use horizontal bars for better label readability
- Add value labels on each bar

4. Histogram:

- Distribution of Current GPA
- Use 20-30 bins
- Show mean and median as vertical lines
- Add normal distribution curve overlay (optional bonus)

5. Box Plot:

- Compare course scores (5 courses) side by side
- Show outliers clearly
- Include title and axis labels
- Consider rotating x-axis labels if needed

Part B: Subplots and Layout (35%)

Create a single figure with multiple subplots:

1. 2×2 Subplot Grid:

- Top-left: GPA distribution histogram
- Top-right: Study Hours vs GPA scatter
- Bottom-left: Average scores by Year (bar chart)
- Bottom-right: Attendance distribution (histogram)
- Use `plt.subplot()` or `fig.add_subplot()`
- Set overall figure title with `suptitle()`
- Adjust spacing with `tight_layout()` or `subplots_adjust()`

Part C: Customization & Design (25%)

Demonstrate advanced Matplotlib features:

1. Customize one plot extensively:

- Change figure size using `figsize`
- Use a custom color palette
- Adjust font sizes for title and labels
- Add annotations to highlight interesting points
- Include a text box with key statistics
- Style the plot professionally

2. Save figures:

- Save at least 2 plots as PNG files (300 DPI)
- Use descriptive filenames
- Demonstrate `plt.savefig()` usage

Evaluation Criteria

- Correct plot types with appropriate data (35%)
- Proper labeling (titles, axes, legends) (25%)
- Subplot creation and layout (20%)
- Visual design and customization (15%)
- Code organization (5%)

Essential Matplotlib Functions

- Plotting: `plt.plot()`, `plt.scatter()`, `plt.bar()`, `plt.hist()`, `plt.boxplot()`
- Customization: `plt.title()`, `plt.xlabel()`, `plt.ylabel()`, `plt.legend()`
- Layout: `plt.figure()`, `plt.subplot()`, `plt.tight_layout()`
- Styling: `plt.grid()`, `plt.xlim()`, `plt.ylim()`, colors, markers
- Saving: `plt.savefig()`

2.2 Task 2: Seaborn & Statistical Analysis (2%)

Objective

Use Seaborn's powerful statistical visualization capabilities to perform deeper exploratory data analysis and uncover complex relationships in the data.

Requirements

Part A: Distribution Analysis (30%)

Create visualizations to understand data distributions:

1. Distribution Plot:

- Use `sns.histplot()` or `sns.kdeplot()` for GPA distribution
- Show both histogram and KDE curve
- Compare distribution across Gender or Year using `hue`
- Include appropriate title and labels

2. Violin Plot:

- Compare Final Average across different Majors
- Show data points inside violins using `inner`
- Use color palette (e.g., `palette="Set2"`)
- Rotate x-labels if needed for readability

3. Box Plot with Seaborn:

- Study Hours comparison across Academic Status
- Use `sns.boxplot()` with color coding
- Show mean marker in addition to median
- Add swarm plot overlay to show individual points (optional)

Part B: Relationship Analysis (40%)

Explore relationships between variables:

1. Correlation Heatmap:

- Calculate correlation matrix for all numerical variables
- Use `sns.heatmap()` to visualize
- Set `annot=True` to show correlation values
- Use diverging colormap (e.g., `cmap="coolwarm"`)
- Adjust figure size for readability
- Identify top 5 strongest correlations

2. Pair Plot:

- Select 4-5 key numerical variables
- (e.g., GPA, Study Hours, Attendance, Sleep Hours, Previous GPA)
- Use `sns.pairplot()` with `hue` for a categorical variable
- Include both scatter plots and distributions

- Add diagonal KDE plots

3. Regression Plot:

- Use `sns.regplot()` or `sns.lmplot()`
- Show relationship between Study Hours and Current GPA
- Include confidence interval
- Calculate and display R-squared value

Part C: Categorical Analysis (30%)

Analyze categorical variables:

1. Count Plot:

- Distribution of students across Majors
- Use `sns.countplot()`
- Sort by count (descending)
- Add value labels on top of bars

2. Grouped Analysis:

- Use `sns.catplot()` or `sns.barplot()`
- Show average GPA by Major and Year
- Use `col` or `hue` for grouping
- Include error bars showing confidence intervals

3. Facet Grid:

- Create a `sns.FacetGrid()` showing scatter plots
- Rows: Gender, Columns: Has Scholarship
- Plot: Attendance vs GPA in each facet
- Ensures consistent axes across facets

Evaluation Criteria

- Correct use of Seaborn functions (40%)
- Statistical validity of visualizations (25%)
- Interpretation of results (20%)
- Visual design and clarity (10%)
- Code documentation (5%)

Key Seaborn Functions

- Distribution: `sns.histplot()`, `sns.kdeplot()`, `sns.violinplot()`, `sns.boxplot()`
- Relationships: `sns.scatterplot()`, `sns.regplot()`, `sns.heatmap()`, `sns.pairplot()`
- Categorical: `sns.countplot()`, `sns.barplot()`, `sns.catplot()`
- Multi-plot: `sns.FacetGrid()`, `sns.PairGrid()`
- Styling: `sns.set_theme()`, `sns.set_palette()`, `sns.set_style()`

2.3 Task 3: Comprehensive Visual Story (2%)

Objective

Combine your visualization skills to tell a cohesive, data-driven story that answers important questions about student academic success.

Scenario

The university administration wants to understand factors affecting student performance to improve support services. Your task is to create a comprehensive visual analysis that identifies key patterns and provides actionable insights.

Requirements

Part A: Research Questions (15%)

Define and state 3-4 clear research questions such as:

1. What factors most strongly predict student GPA?
2. How do study habits differ across academic performance levels?
3. Does working part-time affect academic success?
4. Are there differences in performance across majors or demographics?

Part B: Visual Dashboard (50%)

Create a comprehensive visual dashboard (6-10 visualizations) that answers your research questions:

1. Overview Section:

- Key statistics summary (text or table)
- Overall GPA distribution
- Student demographic breakdown

2. Performance Factors Analysis:

- Multiple visualizations showing relationships between:
 - Study hours and performance
 - Attendance and grades
 - Sleep and academic success
 - Previous vs current GPA
- Use appropriate plot types for each relationship
- Ensure consistent styling across all plots

3. Comparative Analysis:

- Compare performance across:
 - Different majors
 - Year levels (Freshman through Senior)
 - Students with/without scholarships
 - Students with/without part-time work

- Use grouped visualizations
- Highlight significant differences

4. Integration Requirements:

- Use both Matplotlib and Seaborn
- Create at least one complex multi-panel figure
- Apply consistent color scheme throughout
- Ensure professional layout and spacing
- All plots must be clearly labeled

Part C: Insights & Recommendations (35%)

Provide written analysis (minimum 300 words total) addressing:

1. Key Findings:

- 5-7 major insights from your visualizations
- Support each finding with specific visual evidence
- Quantify findings where possible (e.g., "Students who study >15 hours/week have 0.8 higher GPA on average")

2. Pattern Interpretation:

- Explain unexpected patterns or trends
- Discuss strongest correlations found
- Identify any interesting outliers or subgroups

3. Actionable Recommendations:

- For students: How to improve performance based on data
- For university: Support services to implement
- For specific at-risk groups: Targeted interventions

4. Limitations:

- What questions couldn't be answered with available data?
- What additional data would be helpful?
- Caution about correlation vs causation

Evaluation Criteria

- Clear research questions (10%)
- Comprehensive dashboard with appropriate visualizations (40%)
- Visual design quality and consistency (15%)
- Depth of insights and analysis (25%)
- Quality of recommendations (10%)

Excellence Indicators: To achieve >95%, demonstrate:

- Strategic visualization choices that effectively answer research questions
- Professional design with consistent aesthetics
- Deep insights that go beyond obvious observations
- Evidence-based recommendations with quantitative support
- Creative use of advanced visualization techniques
- Clear, compelling narrative flow

3 Submission Guidelines

3.1 Deliverable Format

Submit **ONE** file in your preferred format:

- Jupyter Notebook: `Project3_FirstName_LastName.ipynb`, OR
- Python Script: `Project3_FirstName_LastName.py`

You may use any IDE (PyCharm, VS Code, Jupyter, Spyder, etc.)

Required File Structure

For Jupyter Notebook (.ipynb):

1. Title Cell (Markdown):

- Project title and number
- Your full name and Student ID
- Submission date
- Honor code: "I certify this work is my own"

2. For Each Task:

- Clear header with task number and title
- Brief markdown explanation before visualizations
- Code cells generating visualizations
- All visualizations must be visible in the notebook
- Markdown cells interpreting each visualization

3. Include Generated Files:

- Original CSV file (from data generator)
- (Optional) Saved PNG files of key visualizations
- Do NOT include the data generator script itself

For Python Script (.py):

1. Header Comment Block:

```
1 """
2 Project 3: Data Visualization & EDA
3 Name: [Your Full Name]
4 Student ID: [Your ID]
5 Date: [Submission Date]
6 Honor Code: I certify this work is my own
7
8 Note: This script generates visualizations.
9     Run it to see all plots.
10 """
```

2. For Each Task:

- Clear section comments

- Code generating visualizations
- Use `plt.show()` to display plots
- Comments explaining what each visualization shows
- Save important figures to files

3.2 Code Quality Standards

Mandatory Standards:

- **Imports:** All at the top: `import matplotlib.pyplot as plt, import seaborn as sns, import pandas as pd, import numpy as np`
- **Plot Labels:** Every plot must have title, axis labels, and legend (if applicable)
- **Readability:** Text must be readable (appropriate font sizes)
- **Colors:** Use colorblind-friendly palettes where possible
- **Layout:** Proper spacing, no overlapping elements
- **Comments:** Explain what each visualization shows and why
- **Execution:** Code must run completely without errors

3.3 Visualization Best Practices

- **Choose the right plot type:** Match visualization to data type
- **Remove chart junk:** Eliminate unnecessary elements
- **Use color purposefully:** Not just for decoration
- **Make text readable:** Font sizes appropriate for viewing
- **Sort data meaningfully:** Order categories by value or logic
- **Include units:** Specify what measurements represent
- **Annotate important points:** Draw attention to key findings
- **Maintain consistency:** Same style across all visualizations

3.4 Submission Methods

Choose **ONE** submission method:

Option 1: Direct LMS Upload (Recommended)

1. Ensure all visualizations display correctly
2. For Jupyter: verify all cells executed and outputs visible
3. Gather all files (notebook/script + CSV file + optional PNG files)
4. Create ZIP file: `Project3_FirstName_LastName.zip`
5. Upload ZIP to LMS before deadline
6. Verify upload was successful

Option 2: GitHub Repository Submission

1. Create a **public** GitHub repository named: `KIU-DS-Project3-FirstName-LastName`
2. Repository structure:

```
1 Project3/
2   Project3_FirstName_LastName.ipynb (or .py)
3   data/
4     student_performance.csv
5   figures/ (optional)
6     plot1.png
7     plot2.png
8     ...
9   README.md
10  requirements.txt
```

3. `README.md` should include:
 - Project description
 - Your name and student ID
 - Instructions to run your code
 - Brief summary of key findings
4. `requirements.txt`:

```
1 pandas==2.0.3
2 numpy==1.24.3
3 matplotlib==3.7.1
4 seaborn==0.12.2
```

5. Submit repository URL via LMS **before deadline**
6. **CRITICAL:** Do not commit after deadline

Critical Deadline Information

Deadline Enforcement:

- **Due Date:** End of Week 8 - Sunday, 23:59 Georgian Time
- **LMS Submissions:** Timestamp automatically recorded, late submissions not accepted
- **GitHub Submissions:** Last commit timestamp verified
- **Any commits after deadline = Automatic 0 points**
- **Technical Issues:** Submit at least 2 hours early
- **No Extensions:** Start early!

3.5 Pre-Submission Checklist

Before submitting, verify:

- All three tasks completed
- Code runs successfully and generates all plots
- For Jupyter: all visualizations visible in notebook
- All plots have titles, labels, and legends
- Text is readable (check font sizes)
- CSV file included
- Filename follows convention
- Header includes name, ID, honor code
- Interpretation/insights written for Task 3
- No errors when running complete file
- (If ZIP) all files included
- (If GitHub) README and requirements.txt present
- Submitting well before deadline

4 Grading Rubric

4.1 Grade Distribution

Component	Points	% of Final Grade
Task 1: Matplotlib Fundamentals	2.0	2%
Task 2: Seaborn & Statistical Analysis	2.0	2%
Task 3: Comprehensive Visual Story	2.0	2%
Project Total	6.0	6%

Table 1: Project Point Distribution

4.2 Detailed Rubric

Criterion	Excellent (90-100%)	Good (70-89%)	Needs Work (<70%)
Plot Correctness	Correct plot types, accurate data representation	Minor errors in implementation	Wrong plot types or major errors
Labeling	All elements properly labeled, clear and readable	Most labels present, some readability issues	Missing labels or illegible text
Design Quality	Professional, consistent aesthetics	Adequate design, some inconsistencies	Poor visual design, hard to interpret
Analysis	Deep insights, quantified findings	Basic interpretation present	Superficial or missing analysis
Code Quality	Clean, well-organized, documented	Functional but could improve	Messy or poorly structured

Table 2: Quality Assessment Rubric

4.3 Deductions

Points will be deducted for:

- **-10%** Missing or incorrect filename
- **-15%** Code doesn't run or produces errors
- **-10%** Missing CSV file
- **-15%** Plots missing titles, labels, or legends
- **-10%** Text too small to read (font size < 10pt)
- **-10%** Inconsistent styling across visualizations
- **-15%** Missing interpretations for Task 3
- **-5%** Poor code organization

- **-100%** Late submission or commits after deadline
- **-100%** Academic integrity violations

4.4 Bonus Opportunities (Up to +10%)

- **+5%**: Exceptional visualization design and creativity
- **+3%**: Outstanding insights with statistical rigor
- **+2%**: Use of advanced techniques (animations, interactive plots with note)

Note: Bonus points apply to this project only, maximum score capped at 6%

5 Resources & Support

5.1 Official Documentation

- Matplotlib Documentation: <https://matplotlib.org/stable/index.html>
- Matplotlib Gallery: <https://matplotlib.org/stable/gallery/index.html>
- Seaborn Documentation: <https://seaborn.pydata.org/>
- Seaborn Gallery: <https://seaborn.pydata.org/examples/index.html>
- Pandas Documentation: <https://pandas.pydata.org/docs/>

5.2 Recommended Study Materials

- Course lecture slides (Weeks 6-8)
- Lab notebooks on visualization
- Python Data Science Handbook - Chapter 4: <https://jakevdp.github.io/PythonDataScienceHandbook/04.ipynb>
- Seaborn Tutorial: <https://seaborn.pydata.org/tutorial.html>

5.3 Color Palette Resources

- ColorBrewer: <https://colorbrewer2.org/>
- Matplotlib Colormaps: <https://matplotlib.org/stable/tutorials/colors/colormaps.html>
- Seaborn Color Palettes: https://seaborn.pydata.org/tutorial/color_palettes.html

5.4 Getting Help

1. **Review Course Materials:** Start with lecture slides and lab notebooks
2. **Check Gallery Examples:** Matplotlib and Seaborn galleries show code examples
3. **Email Instructor:** Contact Nika Gagua at Nika.Gagua@kiu.edu.ge for clarifications
4. **Study Groups:** Discuss visualization approaches with peers (write your own code)

Success Strategies

- **Start Early:** Begin right after Week 7
- **Run Data Generator First:** Execute the provided script before coding
- **Explore the Data:** Understand variables before visualizing
- **Sketch First:** Plan your visualizations on paper
- **Iterate:** First version rarely perfect, refine your plots
- **Check Examples:** Use gallery examples as inspiration
- **Test Readability:** View plots at different sizes
- **Tell a Story:** Connect visualizations with narrative
- **Save Progress:** Save plots as you create them
- **Ask for Feedback:** Show plots to others for clarity check

5.5 Common Pitfalls to Avoid

- Forgetting to run the data generator script
- Choosing wrong plot types for data (e.g., line plot for categories)
- Missing labels, titles, or legends
- Using too many colors that confuse rather than clarify
- Text too small to read
- Overlapping labels or elements
- Inconsistent styling across plots
- Creating plots without interpreting them
- Not using Seaborn's statistical capabilities
- Trying to show too much in one visualization
- Starting too late (visualization takes time!)

6 Frequently Asked Questions

6.1 General Questions

Q: How long should this project take?

A: Plan for 8-10 hours total. Task 1: 2-3 hours, Task 2: 3-4 hours, Task 3: 3-4 hours. Visualization requires iteration and refinement.

Q: Can I use other libraries like Plotly?

A: Focus on Matplotlib and Seaborn as required. Plotly is bonus territory if you want to explore, but not required.

Q: Do my plots need to be publication-quality?

A: They should be clear, readable, and professional. Publication-quality is a bonus.

Q: Where do I get the data generator script?

A: Download `generate_project3_data.py` from LMS course materials or contact instructor.

6.2 Technical Questions

Q: How do I make my plots look better?

A: Use Seaborn themes (`sns.set_theme()`), consistent colors, appropriate sizes, and proper spacing. Check gallery examples for inspiration.

Q: My subplots are overlapping. How do I fix this?

A: Use `plt.tight_layout()` or manually adjust with `plt.subplots_adjust()`.

Q: How do I save a high-quality figure?

A: Use `plt.savefig('filename.png', dpi=300, bbox_inches='tight')`.

Q: Can I create interactive plots?

A: This project focuses on static plots. Interactive plots can be a bonus but aren't required.

Q: How many visualizations for Task 3?

A: 6-10 visualizations that comprehensively answer your research questions.

6.3 Data Questions

Q: Can I create my own dataset?

A: No. All students must use the provided data generator to ensure consistency.

Q: What if I find interesting patterns not in the requirements?

A: Explore them! Bonus points for creative analysis beyond requirements.

Q: Should I clean the data?

A: The generated data is already clean. Focus on visualization, not cleaning.

6.4 Submission Questions

Q: Do visualizations need to be embedded in the notebook?

A: Yes for Jupyter. For Python scripts, they should display when code runs.

Q: Can I submit only PNG files instead of code?

A: No. Code is required. PNG files are optional supplements.

Q: What if my notebook is large due to many plots?

A: That's expected. Ensure it's under 50MB for LMS submission.

Best of Luck!

Visualization is both science and art.
Focus on clarity, tell compelling stories with your data,
and don't be afraid to iterate and improve your plots.

Let your visualizations speak for themselves!

- Your Data Science Instructors