Gen AI Journal

Introduction

Briefly describe the generative AI models you chose to use and the rationale behind that choice. (Maximum of 1 page)

- David
    - ChatGPT was mainly used to help with some help with some specific code examples, both to help as a starting off point as well as a guide for things i was unsure about (such as matplotlib graphs)
- Jan
    - ChatGPT was used to provide a quick and direct answer to a coding query
- Karl
    - ChatGPT was used to assist with implementing K-means clustering on the dataset and to provide some talking points for the ethical review.
- Matthew
    - ChatGPT: general queries about the dataset and code. General ideation like potential correlations between features in the dataset, assistance with gradio, pep8 code.
    - GitHub Copilot: inline coding assistance, especially for repetitive code. Also used to help plotting with plotly.

Ethical Considerations

Discuss the ethical aspects of using generative AI in your project. This should include issues like data bias and privacy. (Maximum of 1 page)

- David
    - Since I only used generative ai to help with general coding advice on different libraries, I did not really face any ethical issues.
- Jan
    - Since I only used GenAI to help with coding, no ethical issues were encountered.
- Karl
    - There were no ethical issues from the coding aspect since ChatGPT was just used as a technical guide. However, it was used to help write the ethical review by providing the dataset and asking for ethical talking points on the data, which could raise questions about the validity of its information.
- Matthew
    - Gen AI could definitely be biased in terms of ideation of which features could be correlated with one another, or whether social status makes a difference in grades. We did not use gen AI to make any conclusion, however, and always based our assessments on statistical analysis

Methodology

Outline the methods and steps to integrate the generative AI model into your work. Which tools did you use, and in which sequence? Did you create your pipeline, or did you just use one tool? There is no wrong answer, and this journal aims for transparency and accountability.

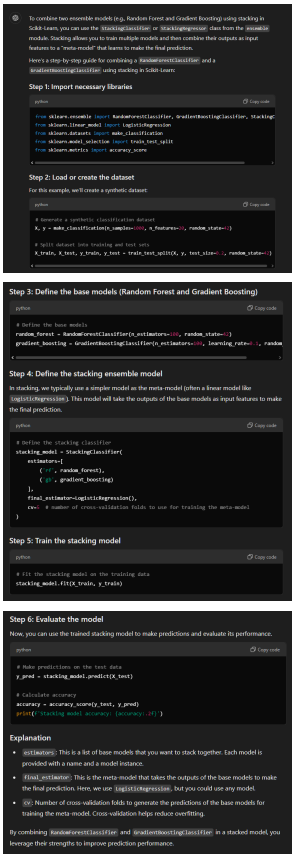
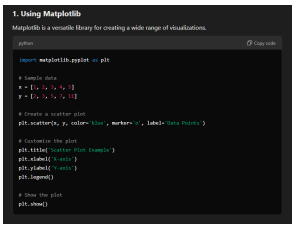
- David
    - I turned to ChatGPT in cases where I needed help with some code, in order to avoid having to directly go over any library documentation or search for online examples.
- Jan
    - I consulted ChatGPT when I needed help to implement an additional feature to my code. ChatGPT was used since it could adapt to the context of the query which was asking how to implement something in code using a specific Python package.
- Karl
    - I used ChatGPT in situations when I was unclear on how to approach the code and couldn't find valid resources quickly enough online. In the ethical review, I merely used ChatGPT to propose some talking points based on the dataset.
- Matthew
    - I would consult with ChatGPT whenever stuck/looking for ideas rather than a continuous implementation throughout the project. GitHub copilot was used inline and thus was integrated throughout, but would help more with practicalities such as variable names, function calls and library imports rather than ideation.

Prompts and Responses

List down the specific prompts that were used with the generative AI model and you found noteworthy. For each prompt, also include the generated response and explain how it contributed to improving your project. It is advisable to use screenshots for this part.

- David

Prompt	**Relevant Response**	**Contribution**

how to combine two ensemble models (random forest and gradient boosting) using stacking in scikit learn https://chatgpt.com/share/6735cd3e-6928-800a-ac06-9a6c0ff9d16b		Gave me an overview of how such a model would be implemented into my code.
scatter plot guide in python https://chatgpt.com/share/678674dd-d71c-800a-8f52-e2c7fdb3436c		Showed me how you would go about creating a scatter plot using matplotlib. (the screenshot provided only shows the relevant section of the response, the rest can be seen following the link)

- Jan

Prompt	Response	Contribution
how to get a single column from a pandas dataframe https://chatgpt.com/share/674dfe2b-20d4-800a-942d-16fc8b03072e	In **pandas**, you can select a single column from a DataFrame using various methods. Here are the most common approaches: **1. Using Square Brackets** This is the most common method. The column name is passed as a string inside the square	Showed me what code I needed to implement for the functionality I wanted

brackets.

df['column_name']

This returns a **Series** object representing the column.

2. Using Dot Notation

If the column name is a valid Python identifier (no spaces, special characters, or conflicts with existing attributes), you can use dot notation.

df.column_name

This also returns a **Series**.
Note that this method is not as flexible as the square brackets method.

3. Using .loc

The .loc indexer allows you to select a column by its name explicitly.

df.loc[:, 'column_name']

This also returns a **Series**, but if you need it as a DataFrame, you can wrap it in a list:

df.loc[:, ['column_name']]

This returns a **DataFrame** with just that column.

4. Using .iloc

If you know the column index (e.g., 0 for the first column), you can use .iloc.

df.iloc[:, column_index]

This returns a **Series**.
If you want it as a DataFrame, wrap the column index in a list:

df.iloc[:, [column_index]]

This returns a **DataFrame**.

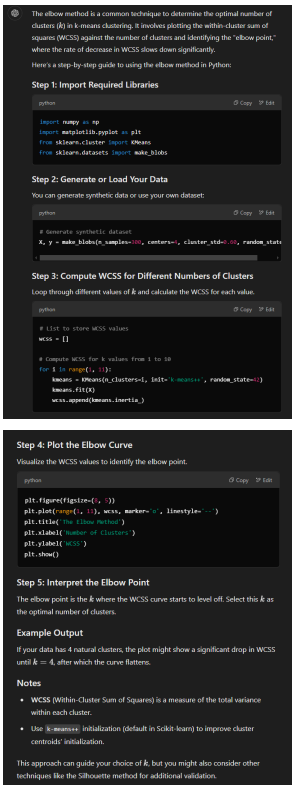
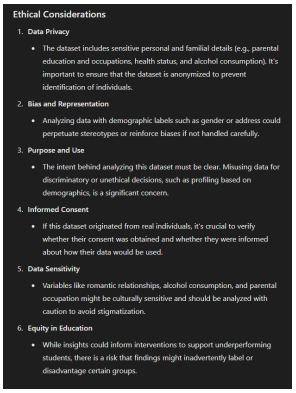
Example

```
import pandas as pd

# Sample DataFrame
data = {'A': [1, 2, 3], 'B':
```

	[4, 5, 6], 'C': [7, 8, 9]}
	df = pd.DataFrame(data)
	
	# Get column 'B'
	col_b = df['B'] # Series
	col_b_df = df[['B']] # DataFrame	

- Karl

Prompt	Response	Contribution
How do I use the elbow method in Python for k-means clustering? https://chatgpt.com/share/678940e3-5364-800d-8f5e-a140477ad3be		This helped me understand how to use the elbow method to find the optimal number of clusters for k-means clustering.
Analyse this dataset which approaches student achievement in secondary education of a Portuguese school and was retrieved from Kaggle. What ethical considerations can be discussed concerning this data? https://chatgpt.com/share/678942ed-168c-800d-8e32-7f55d56dba7e		This provided a helpful starting point for writing the ethical review. Through these talking points, I was able to compile my thoughts in a much quicker and concise manner.

- Matthew

Prompt	Response	Contribution

<We provided ChatGPT with the documentation for the sklearn.decomposition function PCA available at https://scikit-learn.org/dev/modules/generated/sklearn.decomposition.PCA.html>
 	
--- 	
 	
How to select n_components such that information retained is > 90%
	To select n components such that more than 90% of the information (variance) is retained in the principal component analysis (PCA), you can follow these steps:
	
	Fit the PCA model without specifying n_components: This will allow you to examine all possible principal components.
	
	python
	Copy code
	from sklearn.decomposition import PCA
	
	# Assuming X is your data matrix
	pca = PCA() # n_components=None by default
	pca.fit(X)
	Examine the cumulative explained variance: Use the explained_variance_ratio_ attribute, which gives the proportion of variance explained by each component. Calculate the cumulative sum of this attribute to determine how much variance is captured as you add each component.
	
	python
	Copy code
	cumulative_variance = pca.explained_variance_ratio_.cumsum()
	Identify the smallest number of components that retain 90% of variance: Find the index of the first component for which the cumulative variance exceeds 90% (0.90).
	
	python
	Copy code
	import numpy as np
	
	# Identify the number of components that retain >90% variance
	n_components_90 = np.argmax(cumulative_variance ⩾ 0.90) + 1
	Apply PCA with the identified number of components: Now you can create a new PCA model with n_components=n_components_90 to retain at least 90% of the variance.
	
	python
	Copy code
	pca_90 = PCA(n_components=n_components_9
		This prompt provided a useful template which we could use to implement different PCA models which transform the dataset into PCA components with varying levels of explained variance. ChatGPT provided a 90% version, we also added a 95% version.

	0) X_reduced = pca_90.fit_transform(X) This approach provides a data-driven way to set n_components for a specific variance threshold, making it straightforward to choose 90% or any other variance target.	

Use gradio to create an application which allows the user to input all features and makes a prediction using the 3 models. Outputs the results in the same table format	To create a Gradio app that allows users to input features and generate predictions from three models, we'll need to integrate each model into a Gradio interface. Below is an example implementation using Python, assuming you have three machine learning models (model1, model2, model3) already trained and loaded.
	
	Here's the code:
	
	python
	Copy code
	import gradio as gr
	import numpy as np
	
	# Assuming model1, model2, and model3 are pre-loaded
	def load_models():
	    # Load or define your models here
	    # For illustration, assume they have a .predict() method
	    return model1, model2, model3
	
	# Function to predict based on user input
	def predict(input_features):
	    model1, model2, model3 = load_models()
	
	    # Ensure input is in the correct format for prediction
	    features = np.array(input_features).reshape(1, -1)
	
	    # Get predictions from each model
	    prediction1 = model1.predict(features)
	    prediction2 = model2.predict(features)
	    prediction3 = model3.predict(features)
	
	    # Return predictions in table format
	    return {
	        "Model": ["Model 1", "Model 2", "Model 3"],
	        "Prediction": [prediction1[0], prediction2[0], prediction3[0]]
	    }
	
	# Define Gradio interface components
	input_features = [gr.Number(label=f"Feature
		This provided a good template which we could then modify to fit the needs of the 4 ML implementations.

	{i+1}") for i in range(5)] # Adjust the number of features as needed
	output = gr.Dataframe(headers=["Model", "Prediction"], datatype=["str", "number"])
	
	# Create Gradio Interface
	interface = gr.Interface(
	    fn=predict,
	    inputs=input_features,
	    outputs=output,
	    title="Multi-Model Prediction App",
	    description="Enter features to get predictions from three different models."
	)
	
	# Launch the app
	if __name__ == "__main__":
	    interface.launch()
	Explanation of Code:
	Load Models: The load_models function loads each model. Replace model1, model2, and model3 with actual model loading or initialization code.
	Predict Function: The predict function accepts user input, reshapes it for the models, and gets predictions from each model. The results are organized into a table.
	Gradio Interface:
	input_features: Generates a list of input fields for each feature.
	output: Defines a table for displaying model predictions.
	Launch App: Starts the Gradio app, allowing the user to input features and view predictions in real time.
	Ensure your models are compatible with this setup, especially if they require pre-processing or have different feature requirements.	

Improvements, Errors and Contributions

Discuss the areas where generative AI contributed to enhancing your work or instances where the output contained errors. This can include but is not limited to data analysis, formulation of ethical considerations, literature review enhancement, or idea generation. Highlight specific cases where this happened.

- David
    - In both cases where gen AI was used, the material it gave me to work with was always correct. That being said, it should be noted that I only used it to give me basic information as a way to quicken my coding process, rather than giving it prompts specific to this project.
- Jan
    - Due to the simple and direct nature of the request, the AI gave a correct response.
- Karl
    - No anomalies or incorrect data were ever encountered when using ChatGPT. However, when using it to provide talking points for the ethical review, I was mindful of what kind of generated information I used to avoid steering out of context.
- Matthew
    - On the whole gen AI served as a good contribution to data analysis and idea generation. There were some instances where it was clear the model didn't understand the whole context of the dataset, e.g. at some point ChatGPT suggested predicting student dropout likelihood given other features, but this was not a feature in the dataset and thus we could not assess it.

Individual Reflection

Reflect on your personal experience using generative AI in your project. Discuss what you learned, what surprised you, and how your perspective on using AI in academic projects has changed, if at all. Did Gen AI help you be more efficient? Do you feel you wasted more time when you used it? For which part was it most helpful? Literature review, debugging?

- David
    - For my case, ChatGPT was an aid in helping me speed up my coding process, as instead of having to manually search website and tutorials to understand certain concepts, I could easily obtain an explanation and example with a simple prompt.
- Jan
    - ChatGPT was very helpful in saving time as my exact query regarding a specific package was answered with a simple prompt.
- Karl
    - I think ChatGPT, and generative AI in general, can be a very useful tool for students and workers alike. While it has its limitations, constructing a concise prompt and setting expectations for its response will provide helpful

assistance overall. In the context of this assignment, I was quite surprised that ChatGPT was able to analyse the dataset thoroughly and in-depth.

- Matthew
    - With regards to implementing PCA, ChatGPT was particularly useful as it would not only provide Python implementations, but when prompted, would provide justification for why it was doing what it did with reference to the theory behind PCA. Thus I was able to both implement PCA quickly while understanding it. Regardless, it remains invaluable to have a good grasp of the field rather than depending entirely on generative AI, as this allows you to generate more powerful prompts, as well as being able to cross check the results generated by gen AI prior to implementation. Gen AI definitely helped me to code more efficiently, particularly when creating stylesheets for the web page (through GitHub copilot), as well in debugging, particularly the gradio implementations.

References and List of Resources Used

- David
    - https://chatgpt.com
- Jan
    - https://chatgpt.com
- Karl
    - https://chatgpt.com
- Matthew
    - https://chatgpt.com
    - https://github.com/features/copilot