University of Malta

Department of Artificial Intelligence

Generative AI Journal

ICS5110 – Applied Machine Learning

Coursework Submission

Authors:
Paul Said
Michael Mizzi
Michael Vella
Isaac Cutajar

January 30, 2026

Contents

1 Introduction

Generative Artificial Intelligence (GenAI) models were utilised during this project to support research, software development, and documentation tasks. The primary tools used were Gemini, ChatGPT and GitHub Copilot. ChatGPT and Gemini were selected due to their ability to explain machine learning concepts, assist with academic writing, and generate LaTeX code. GitHub Copilot was selected to assist with software development by suggesting code snippets and implementation patterns.

These tools were chosen as they enhance productivity while allowing the authors to maintain full control and responsibility over the final outputs. The use of GenAI was supplementary and did not replace independent analysis or implementation.

2 Ethical Considerations

The use of generative AI introduces ethical considerations including data bias, privacy, and academic integrity. AI models may generate biased or incorrect outputs due to biases present in their training data. Therefore, all AI-generated content was critically reviewed and verified against lecture notes and reputable academic sources.

Privacy concerns were mitigated by ensuring that no personal, sensitive, or confidential data was shared with AI tools. Transparency was maintained by documenting AI usage in this journal, ensuring accountability and adherence to academic integrity guidelines.

3 Methodology

Generative AI tools were integrated interactively throughout the project. ChatGPT/Gemini were used for conceptual explanations, academic writing assistance, and LaTeX code generation. GitHub Copilot was used during software development to suggest boilerplate code and implementation structures.

The workflow involved manually drafting content, consulting AI tools for clarification or formatting, and manually reviewing and modifying all generated outputs. No automated AI pipeline was created. Instead, AI tools were used selectively at different stages to support learning and productivity.

4 Prompts and Responses

This section documents notable prompts and responses used during the project.

4.1 Example Prompt 1

Prompt: "Explain the hyperparameters of Logistic Regression in Scikit-learn."

Response Summary: ChatGPT explained parameters such as regularisation strength, penalty types, and solver options.

Contribution: This reinforced understanding of model configuration and supported correct implementation.

4.2 Example Prompt 2

Prompt: "Convert this mathematical expression into LaTeX."

Response Summary: ChatGPT generated LaTeX code for equations and tables.

Contribution: This reduced time spent on LaTeX syntax and improved document formatting.

4.3 Example Prompt 3

Prompt: "Update this chart from a pie chart to a bar chart."

Response Summary: ChatGPT modified the existing code to replace the pie chart with an equivalent bar chart.

Contribution: This reduced the time required to consult documentation in order to identify the appropriate parameters and plotting functions.

4.4 Example Prompt 4

Prompt: "Polish this README.md file."

Response Summary: ChatGPT suggested structural improvements and provided revised content to enhance the clarity and completeness of the README file.

Contribution: This reduced the time spent on creating and refining project documentation, helping to other team members in cloning and using the GitHub repository.

5 Improvements, Errors and Contributions

Generative AI contributed to improving writing clarity, code structuring, and conceptual understanding of machine learning methods. It significantly reduced the time required for LaTeX formatting and provided explanations aligned with course content.

However, some AI-generated outputs were generic or contained inaccuracies. These errors were identified by cross-referencing with lecture notes and academic sources. All AI-generated code and explanations were manually validated and corrected where necessary.

6 Individual Reflection

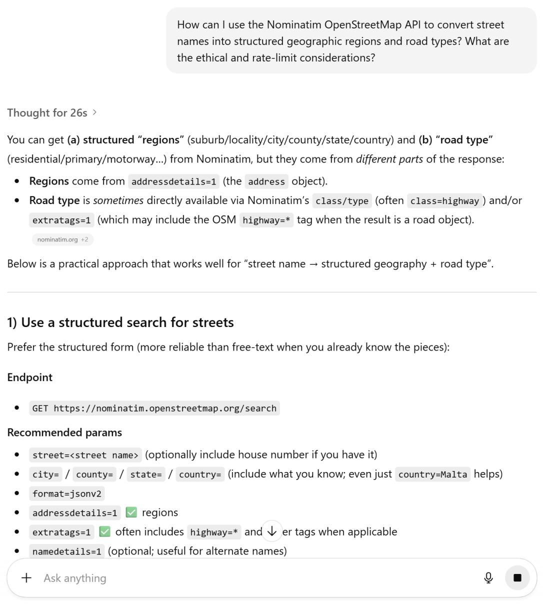
6.1 Paul Said

Generative AI tools were used as supportive learning and productivity aids throughout this project, primarily ChatGPT and GitHub Copilot. Their role was advisory rather than authoritative, and all outputs were critically reviewed, validated, and adapted before being incorporated into the project.

ChatGPT was used to explore strategies for transforming unstructured accident narratives into a structured dataset, including feature extraction, deduplication strategies, and categorical encoding design. These discussions informed the design of the preprocessing pipeline. However, all extraction logic, validation procedures, and data cleaning steps were implemented manually.

To enhance spatial feature engineering, I investigated the use of the Nominatim OpenStreetMap API to geocode accident locations and derive structured geographic features such as regions and road classifications. ChatGPT was consulted to understand reliable query patterns (e.g., structured street search versus free-text), response fields relevant to administrative geography, rate-limiting constraints, and ethical considerations when querying a public geospatial service (Figure 1). This informed the decision to treat geocoding as a deterministic enrichment step and to cache results rather than repeatedly querying the public endpoint.

The Google Routes API was also explored as a potential source of traffic intensity proxies. ChatGPT was used to confirm what the API can and cannot provide, particularly the limitation that historical traffic intensity is not exposed as a directly queryable dataset for past departure times. This supported the methodological decision to treat traffic level as an approximate proxy (or to rely on temporal and road-type features as indirect exposure indicators) rather than presenting it as historical ground truth (Figure 2).



How can I use the Nominatim OpenStreetMap API to convert street names into structured geographic regions and road types? What are the ethical and rate-limit considerations?

Thought for 26s ›

You can get **(a) structured "regions"** (suburb/locality/city/county/state/country) and **(b) "road type"** (residential/primary/motorway...) from Nominatim, but they come from *different parts* of the response:

- **Regions** come from `addressdetails=1` (the `address` object).
- **Road type** is *sometimes* directly available via Nominatim's `class/type` (often `class=highway`) and/or `extratags=1` (which may include the OSM `highway=*` tag when the result is a road object). nominatim.org +2

Below is a practical approach that works well for "street name → structured geography + road type".

1) Use a structured search for streets

Prefer the structured form (more reliable than free-text when you already know the pieces):

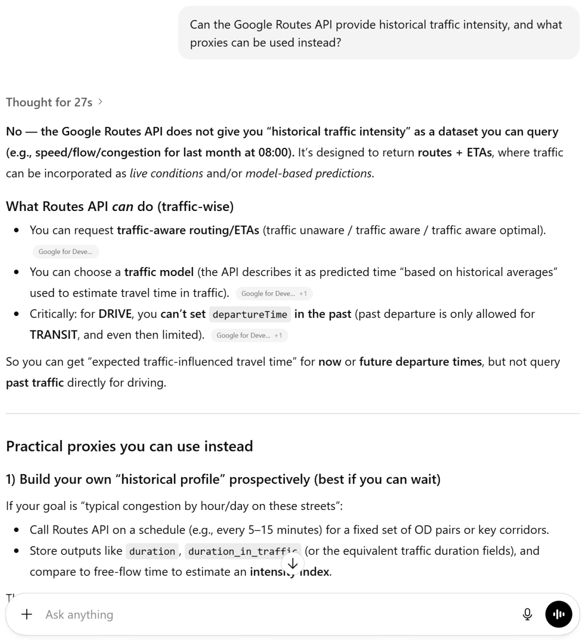
Endpoint

- `GET https://nominatim.openstreetmap.org/search`

Recommended params

- `street=<street name>` (optionally include house number if you have it)
- `city=` / `county=` / `state=` / `country=` (include what you know; even just `country=Malta` helps)
- `format=jsonv2`
- `addressdetails=1` ✅ regions
- `extratags=1` ✅ often includes `highway=*` and er tags when applicable
- `namedetails=1` (optional; useful for alternate names)

Figure 1: Prompt used to explore the Nominatim OpenStreetMap API for geocoding and extracting structured administrative geography and road-type information.



Can the Google Routes API provide historical traffic intensity, and what proxies can be used instead?

Thought for 27s ›

No — the Google Routes API does not give you "historical traffic intensity" as a dataset you can query (e.g., speed/flow/congestion for last month at 08:00). It's designed to return **routes + ETAs**, where traffic can be incorporated as *live conditions* and/or *model-based predictions*.

What Routes API *can* do (traffic-wise)

- You can request **traffic-aware routing/ETAs** (traffic unaware / traffic aware / traffic aware optimal). Google for Deve...
- You can choose a **traffic model** (the API describes it as predicted time "based on historical averages" used to estimate travel time in traffic). Google for Deve... +1
- Critically: for **DRIVE**, you **can't set** `departureTime` **in the past** (past departure is only allowed for **TRANSIT**, and even then limited). Google for Deve... +1

So you can get "expected traffic-influenced travel time" for **now** or **future departure times**, but not query **past traffic** directly for driving.

Practical proxies you can use instead

1) Build your own "historical profile" prospectively (best if you can wait)

If your goal is "typical congestion by hour/day on these streets":

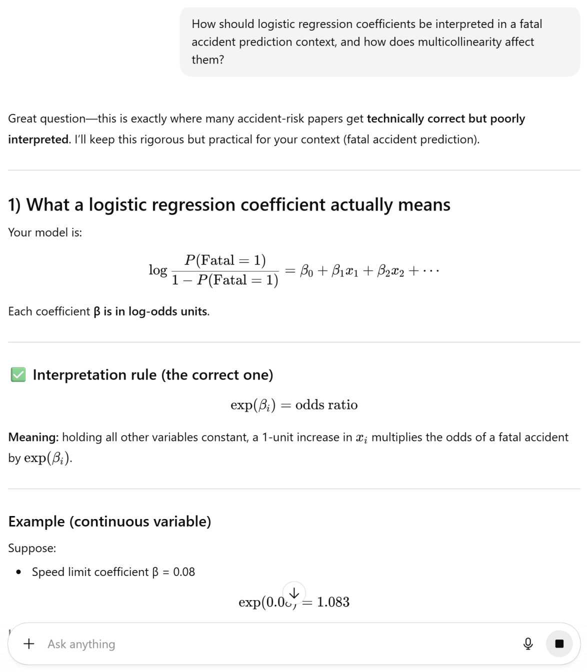
- Call Routes API on a schedule (e.g., every 5–15 minutes) for a fixed set of OD pairs or key corridors.
- Store outputs like `duration`, `duration_in_traffic` (or the equivalent traffic duration fields), and compare to free-flow time to estimate an **intensity index**.

Figure 2: Prompt used to investigate whether the Google Routes API provides historical traffic intensity and to identify defensible proxy variables when historical data is unavailable.

ChatGPT was used to clarify modelling concepts relevant to the chosen evaluation framework, particularly the interpretation of logistic regression coefficients in a fatality prediction context and the effect of multicollinearity on coefficient stability. These explanations were used to inform how coefficient tables and interpretability commentary were written, while avoiding causal claims and maintaining a distinction between association and causation (Figure 3). Outputs were cross-checked against lecture material and official documentation before inclusion.

Given the severe class imbalance in the fatality label, ChatGPT was also used to understand why a default probability threshold of 0.5 is often inappropriate in rare-event classification and how threshold selection can be tuned in a defensible way (e.g., optimising an F1-based operating point while monitoring recall). This supported the evaluation design and the decision to discuss precision–recall trade-offs explicitly rather than relying on accuracy as a primary metric (Figure 4). Explanations were verified against course content and official library references before application.

Generative AI was also used for LaTeX formatting, converting drafted technical content into IEEE-compliant structures, and improving academic writing clarity while preserving meaning.


How should logistic regression coefficients be interpreted in a fatal accident prediction context, and how does multicollinearity affect them?

Great question—this is exactly where many accident-risk papers get **technically correct but poorly interpreted**. I'll keep this rigorous but practical for your context (fatal accident prediction).

1) What a logistic regression coefficient actually means

Your model is:

$$\log \frac{P(\text{Fatal}=1)}{1-P(\text{Fatal}=1)} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots$$

Each coefficient **β is in log-odds units.**

✅ **Interpretation rule (the correct one)**

$$\exp(\beta_i) = \text{odds ratio}$$

Meaning: holding all other variables constant, a 1-unit increase in x_i multiplies the odds of a fatal accident by $\exp(\beta_i)$.

Example (continuous variable)

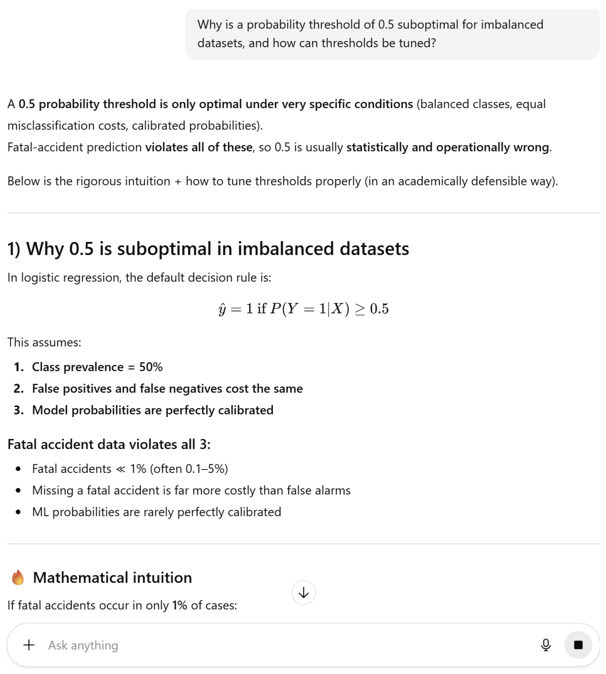
Suppose:

- Speed limit coefficient β = 0.08

$$\exp(0.08) = 1.083$$

Ask anything

Figure 3: Prompt used to understand the interpretation of logistic regression coefficients (log-odds and odds ratios) and how multicollinearity can affect coefficient stability.


Why is a probability threshold of 0.5 suboptimal for imbalanced datasets, and how can thresholds be tuned?

A **0.5 probability threshold is only optimal under very specific conditions** (balanced classes, equal misclassification costs, calibrated probabilities).
Fatal-accident prediction **violates all of these**, so 0.5 is usually **statistically and operationally wrong**.

Below is the rigorous intuition + how to tune thresholds properly (in an academically defensible way).

1) Why 0.5 is suboptimal in imbalanced datasets

In logistic regression, the default decision rule is:

$$\hat{y} = 1 \text{ if } P(Y=1|X) \geq 0.5$$

This assumes:

1. **Class prevalence = 50%**
2. **False positives and false negatives cost the same**
3. **Model probabilities are perfectly calibrated**

Fatal accident data violates all 3:

- Fatal accidents ≪ 1% (often 0.1–5%)
- Missing a fatal accident is far more costly than false alarms
- ML probabilities are rarely perfectly calibrated

🔥 **Mathematical intuition**

If fatal accidents occur in only **1%** of cases:

Ask anything

Figure 4: Prompt used to understand probability threshold selection for imbalanced datasets and the implications for precision–recall trade-offs.

GitHub Copilot assisted with boilerplate code generation for data pipelines, preprocessing routines, and plotting functions. All generated code was manually reviewed, tested, and modified where necessary.

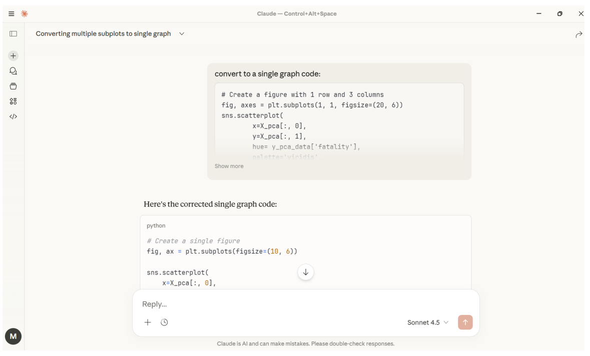
Although Generative AI accelerated development, some responses were generic or required correction for methodological precision. AI explanations occasionally simplified statistical assumptions, reinforcing the importance of verification against authoritative sources. Overall, GenAI tools were most valuable as learning assistants and coding accelerators rather than as sources of final answers.

6.2 Michael Mizzi

The use of GenAI in this context was a new venture for me as this technology was not available during past academic work both during under-graduate and also at post-graduate experiences. Following the guidance offered through this course I have calibrated myself by asking four questions whenever submitting a prompt:

- Am I asking for ideas rather solutions?
- Am I avoiding copy / paste?
- Am I using multiple gen AI sources to minimise risk of bias and hallucination?
- Am I using gen AI tools to challenge my work rather than doing it?

During this project Google scholar lab was used to search and focus on past research papers related to our work. The 'lab' function was used to extract similar works over the past few years. In addition, Gemini was used two-fold; first to ask any generic queries about the subject, KNN model in particular, secondly to code using Colab. Claude and Perplexity AI were both used as cross-checking tools in order to compare and contrast responses to identical prompts fed to Gemini. Of particular interest was the use of Gemini's built-in function in Colab to fix


Claude — Control+Alt+Space
Converting multiple subplots to single graph
Show more
Reply...
Sonnet 4.5
Claude is AI and can make mistakes. Please double-check responses.

convert to a single graph code:

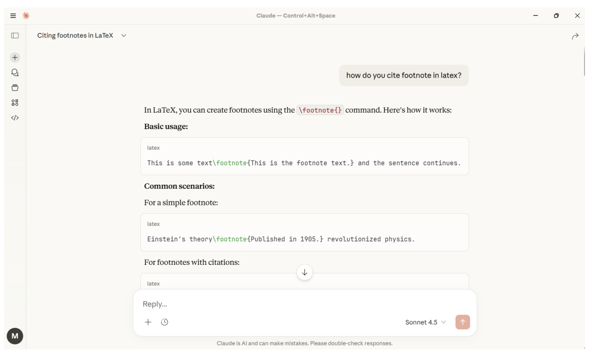
```
# Create a figure with 1 row and 3 columns
fig, axes = plt.subplots(1, 1, figsize=(20, 6))
sns.scatterplot(
        x=X_pca[:, 0],
        y=X_pca[:, 1],
        hue= y_pca_data['fatality'],
```

Here's the corrected single graph code:

```python
# Create a single figure
fig, ax = plt.subplots(figsize=(10, 6))

sns.scatterplot(
    x=X_pca[:, 0],
```

Figure 5: Claude support to rewrite code


Claude — Control+Alt+Space
Citing footnotes in LaTeX
Reply...
Sonnet 4.5
Claude is AI and can make mistakes. Please double-check responses.

how do you cite footnote in latex?

In LaTeX, you can create footnotes using the `\footnote{}` command. Here's how it works:

Basic usage:

```latex
This is some text\footnote{This is the footnote text.} and the sentence continues.
```

Common scenarios:

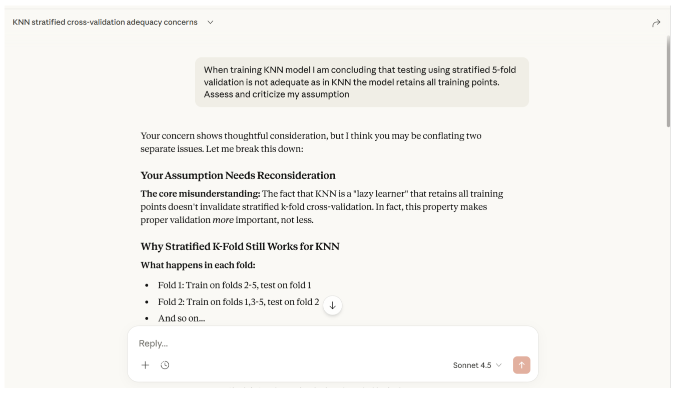
For a simple footnote:

```latex
Einstein's theory\footnote{Published in 1905.} revolutionized physics.
```

For footnotes with citations:

Figure 6: Claude as a Latex tutor

issues with the Python code and to explore improved methodologies to visualize the data, in particular using the Matplotlib[1] library. Similarly Claude was engaged while learning the syntax for LaTeX, this being the first time I was actively working using Overleaf/LaTeX.



When training KNN model I am concluding that testing using stratified 5-fold validation is not adequate as in KNN the model retains all training points. Assess and criticize my assumption

Your concern shows thoughtful consideration, but I think you may be conflating two separate issues. Let me break this down:

Your Assumption Needs Reconsideration

The core misunderstanding: The fact that KNN is a "lazy learner" that retains all training points doesn't invalidate stratified k-fold cross-validation. In fact, this property makes proper validation *more* important, not less.

Why Stratified K-Fold Still Works for KNN

What happens in each fold:

- Fold 1: Train on folds 2-5, test on fold 1
- Fold 2: Train on folds 1,3-5, test on fold 2
- And so on...

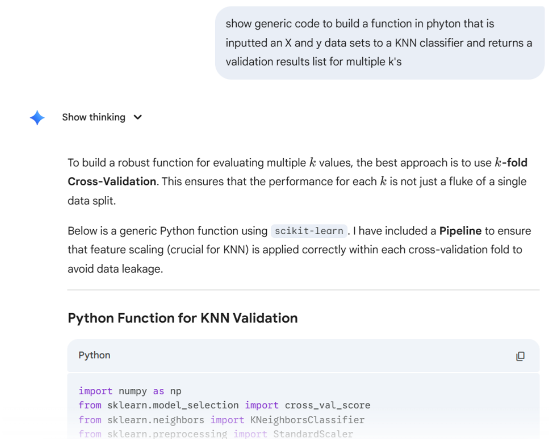
Figure 7: Using Claude to interpret results

Another value-adding tool that I used for the first time was NotebookLM. This practical google-powered RAG proved to be efficient in various aspects. Of most importance was the aspect of summarization of the literature review and the ability to extract comparative context between different research papers. In addition, NotebookLM was also engaged to generate a high-level diagram of our data pipeline. This was processed by running the infographic function on a text source that was purposely compiled to describe the steps used to preprocess, engineer and extract the features and finally perform EDA.

In essence, using these large-language generative models was perceived as a time efficient resource. While one can argue that similar results can be achieved using traditional methodologies, (that is; searching, screening and evaluating information), with genAI one can speed up the process. In our context, the known issue of hallucination could be mitigated by cross-referencing feedback from different tools, such as Claude and Gemini.

List of screenshot representing actual prompts and feedback:

[1] https://matplotlib.org/



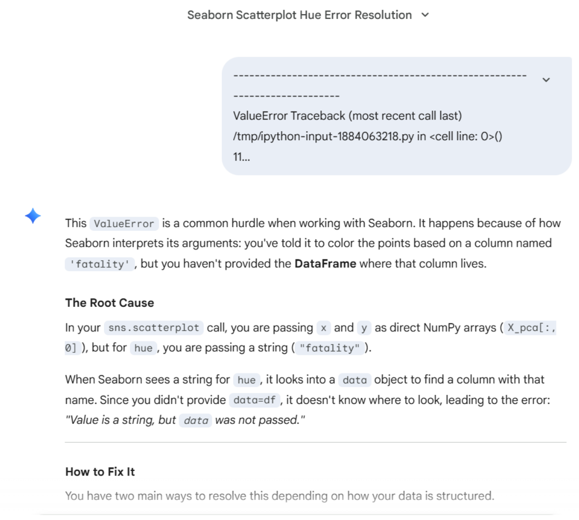
show generic code to build a function in phyton that is inputted an X and y data sets to a KNN classifier and returns a validation results list for multiple k's

To build a robust function for evaluating multiple k values, the best approach is to use k-**fold Cross-Validation**. This ensures that the performance for each k is not just a fluke of a single data split.

Below is a generic Python function using `scikit-learn`. I have included a **Pipeline** to ensure that feature scaling (crucial for KNN) is applied correctly within each cross-validation fold to avoid data leakage.

Python Function for KNN Validation

```python
import numpy as np
from sklearn.model_selection import cross_val_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler
```



Seaborn Scatterplot Hue Error Resolution

```
-------------------------------------------------------
--------------------
ValueError Traceback (most recent call last)
/tmp/ipython-input-1884063218.py in <cell line: 0>()
11...
```

This `ValueError` is a common hurdle when working with Seaborn. It happens because of how Seaborn interprets its arguments: you've told it to color the points based on a column named `'fatality'`, but you haven't provided the **DataFrame** where that column lives.

The Root Cause

In your `sns.scatterplot` call, you are passing `x` and `y` as direct NumPy arrays (`X_pca[:, 0]`), but for `hue`, you are passing a string (`"fatality"`).

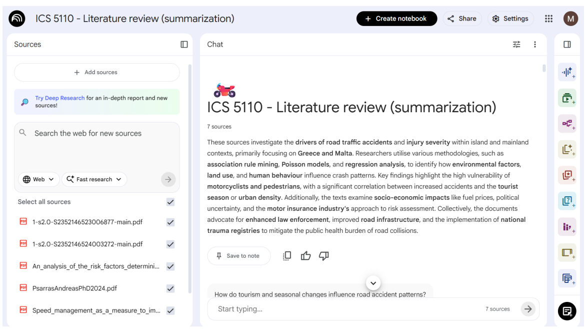
When Seaborn sees a string for `hue`, it looks into a `data` object to find a column with that name. Since you didn't provide `data=df`, it doesn't know where to look, leading to the error: *"Value is a string, but `data` was not passed."*

How to Fix It

You have two main ways to resolve this depending on how your data is structured.

Figure 8: Working with Gemini to develop code

Figure 9: Gemini as a phyton debugger



ICS 5110 - Literature review (summarization)

Create notebook
Share
Settings
Sources
Add sources
Try Deep Research for an in-depth report and new sources!
Search the web for new sources
Web
Fast research
Select all sources

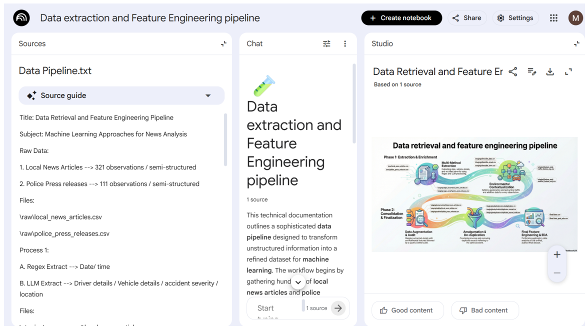
1-s2.0-S2352146523006877-main.pdf
1-s2.0-S2352146524003272-main.pdf
An_analysis_of_the_risk_factors_determini...
PsarrasAndreasPhD2024.pdf
Speed_management_as_a_measure_to_im...

ICS 5110 - Literature review (summarization)

7 sources

These sources investigate the **drivers of road traffic accidents** and **injury severity** within island and mainland contexts, primarily focusing on **Greece and Malta**. Researchers utilise various methodologies, such as **association rule mining**, **Poisson models**, and **regression analysis**, to identify how **environmental factors**, **land use**, and **human behaviour** influence crash patterns. Key findings highlight the high vulnerability of **motorcyclists and pedestrians**, with a significant correlation between increased accidents and the **tourist season** or **urban density**. Additionally, the texts examine **socio-economic impacts** like fuel prices, political uncertainty, and the **motor insurance industry's** approach to risk assessment. Collectively, the documents advocate for **enhanced law enforcement**, improved **road infrastructure**, and the implementation of **national trauma registries** to mitigate the public health burden of road collisions.

How do tourism and seasonal changes influence road accident patterns?



Data extraction and Feature Engineering pipeline

Data Pipeline.txt

Title: Data Retrieval and Feature Engineering Pipeline

Subject: Machine Learning Approaches for News Analysis

Raw Data:

1. Local News Articles --> 321 observations / semi-structured

2. Police Press releases --> 111 observations / semi-structured

Files:

\raw\local_news_articles.csv

\raw\police_press_releases.csv

Process 1:

A. Regex Extract --> Date/ time

B. LLM Extract --> Driver details / Vehicle details / accident severity / location

Files:

Data extraction and Feature Engineering pipeline

1 source

This technical documentation outlines a sophisticated **data pipeline** designed to transform unstructured information into a refined dataset for **machine learning**. The workflow begins by gathering hund of **local news articles** and **police**

Start
1 source
Studio
Data Retrieval and Feature Er
Based on 1 source
Data retrieval and feature engineering pipeline
Phase 1: Extraction & Enrichment
Multi-Method Extraction
Environmental Contextualization
Phase 2: Consolidation & Finalization
Data Augmentation & Audit
Amalgamation & De-duplication
Final Feature Engineering & EDA
Good content
Bad content

Figure 10: NotebookLM as a tool to summarize Literature

Figure 11: NotebookLM as a diagram generator

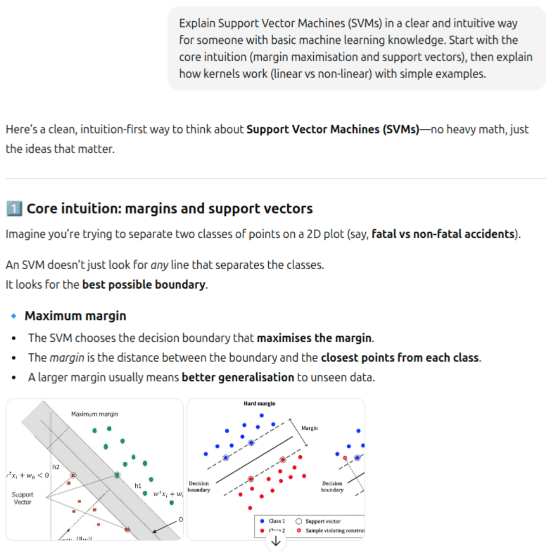
6.3 Michael Vella

In this project, I used ChatGPT as a supportive tool throughout the development and evaluation of my machine learning workflow. Its main purpose was to help with conceptual clarification, debugging, and improving the clarity of my written explanations.

Whenever I had a question or needed clarification, I used ChatGPT to get an initial explanation, which I then cross-checked against lecture notes or other resources. Although I am already familiar with Python and use it regularly for work, I also used ChatGPT to help debug, refactor, and occasionally generate code when needed. In addition, it was useful for refining academic writing, particularly when rephrasing explanations or improving structure while keeping the technical meaning intact. Importantly though, all ideas and final decisions in the project were my own.

Figures 12 and 13 depict two use cases of how I used ChatGPT for this assignment.

While this assignment could have been completed without generative AI, using these tools helped speed up both the research and development process. That said, the most important takeaway from this is that ultimately, all of these are tools and as users of these tools it is up to us, the user, to make responsible and ethical choices in how we apply them.


Explain Support Vector Machines (SVMs) in a clear and intuitive way for someone with basic machine learning knowledge. Start with the core intuition (margin maximisation and support vectors), then explain how kernels work (linear vs non-linear) with simple examples.

Here's a clean, intuition-first way to think about **Support Vector Machines (SVMs)**—no heavy math, just the ideas that matter.

1️⃣ **Core intuition: margins and support vectors**

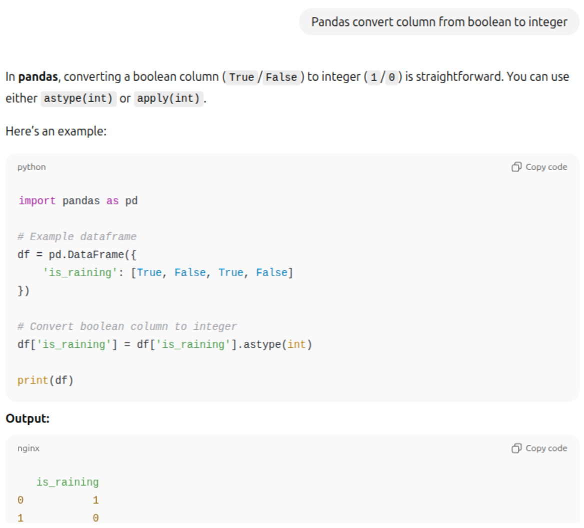
Imagine you're trying to separate two classes of points on a 2D plot (say, **fatal vs non-fatal accidents**).

An SVM doesn't just look for *any* line that separates the classes.
It looks for the **best possible boundary**.

🔹 **Maximum margin**

- The SVM chooses the decision boundary that **maximises the margin**.
- The *margin* is the distance between the boundary and the **closest points from each class**.
- A larger margin usually means **better generalisation** to unseen data.

Figure 12: Using ChatGPT to get an explanation of SVMs.


Pandas convert column from boolean to integer

In **pandas**, converting a boolean column (`True` / `False`) to integer (`1` / `0`) is straightforward. You can use either `astype(int)` or `apply(int)`.

Here's an example:

```python
import pandas as pd

# Example dataframe
df = pd.DataFrame({
    'is_raining': [True, False, True, False]
})

# Convert boolean column to integer
df['is_raining'] = df['is_raining'].astype(int)

print(df)
```

Output:

```nginx
   is_raining
0           1
1           0
```

Figure 13: Using ChatGPT as a coding assitant.

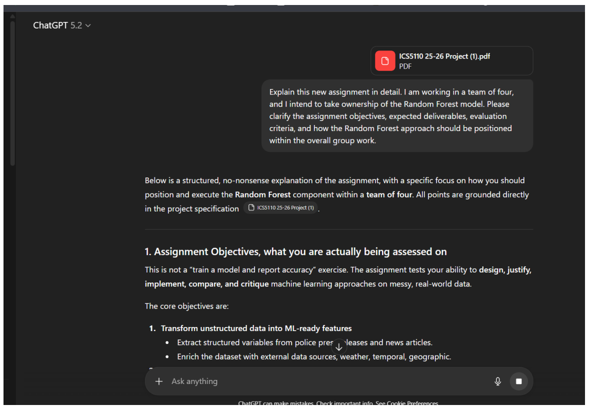
6.4 Isaac Cutajar

As AI is a relatively new area for me, given my stronger background in data engineering and data analytics through both academic study and professional practice, GenAI tools were used primarily to accelerate my upskilling for this ML project and to support a clearer understanding of the assignment requirements.

GenAI was initially leveraged to clarify the project scope, expected deliverables, and best-practice methodologies. When interacting with these tools, I consistently aimed to provide full contextual information within each prompt, including dataset characteristics, constraints, and my ideas. The responses were then understood and evaluated, rather than adopted verbatim.

One notable observation was that, at times, it was necessary to explicitly request simplified explanations, particularly when dealing with model training concepts or when interpreting unfamiliar code structures. This highlighted both the usefulness of GenAI as a learning aid and the importance of actively guiding the level of complexity in its responses.

ChatGPT was primarily used during the early stages of the project, especially for understanding the assignment requirements, identifying appropriate ML approaches, and supporting the data preparation and cleaning phase. Typical prompts focused on best practices for feature engineering, handling class imbalance, and structuring exploratory data analysis. Relevant examples of these interactions are provided through screenshots and conversation references below:


ChatGPT 5.2

ICS5110 25-26 Project (1).pdf
PDF

Explain this new assignment in detail. I am working in a team of four, and I intend to take ownership of the Random Forest model. Please clarify the assignment objectives, expected deliverables, evaluation criteria, and how the Random Forest approach should be positioned within the overall group work.

Below is a structured, no-nonsense explanation of the assignment, with a specific focus on how you should position and execute the **Random Forest** component within a **team of four**. All points are grounded directly in the project specification ICS5110 25-26 Project (1).

1. Assignment Objectives, what you are actually being assessed on

This is not a "train a model and report accuracy" exercise. The assignment tests your ability to **design, justify, implement, compare, and critique** machine learning approaches on messy, real-world data.

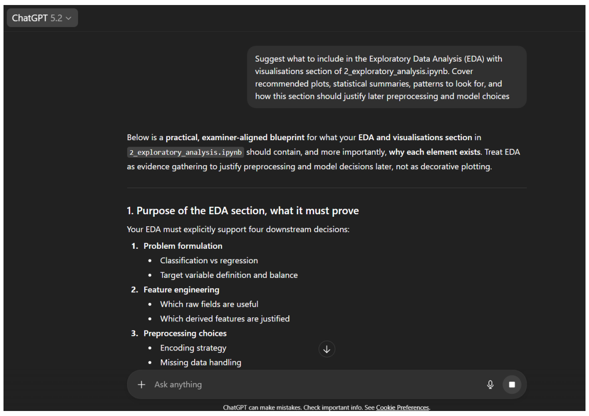
The core objectives are:

1. **Transform unstructured data into ML-ready features**
   - Extract structured variables from police pres leases and news articles.
   - Enrich the dataset with external data sources, weather, temporal, geographic.

Ask anything

ChatGPT can make mistakes. Check important info. See Cookie Preferences.

Figure 14: Understanding the Project.


ChatGPT 5.2

Suggest what to include in the Exploratory Data Analysis (EDA) with visualisations section of 2_exploratory_analysis.ipynb. Cover recommended plots, statistical summaries, patterns to look for, and how this section should justify later preprocessing and model choices

Below is a **practical, examiner-aligned blueprint** for what your **EDA and visualisations section** in `2_exploratory_analysis.ipynb` should contain, and more importantly, **why each element exists**. Treat EDA as evidence gathering to justify preprocessing and model decisions later, not as decorative plotting.

1. Purpose of the EDA section, what it must prove

Your EDA must explicitly support four downstream decisions:

1. **Problem formulation**
   - Classification vs regression
   - Target variable definition and balance
2. **Feature engineering**
   - Which raw fields are useful
   - Which derived features are justified
3. **Preprocessing choices**
   - Encoding strategy
   - Missing data handling

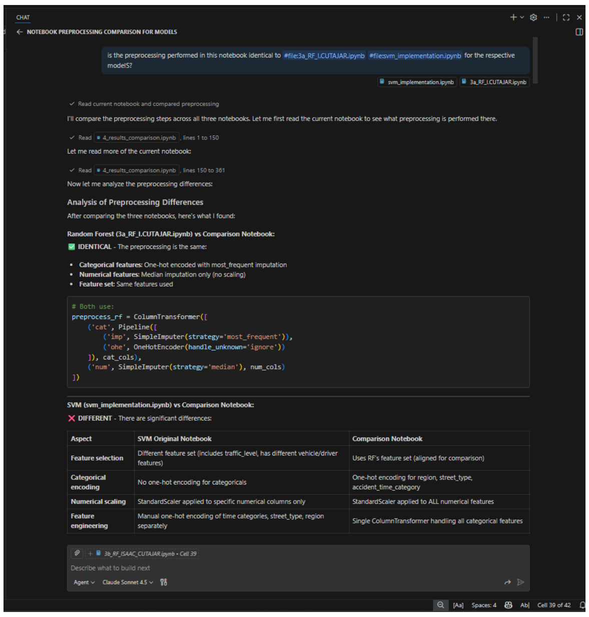
Ask anything

ChatGPT can make mistakes. Check important info. See Cookie Preferences.

Figure 15: Getting inspiration for EDA notebook.

For the exact prompts and outputs make reference to the shared ChatGPT conversation [2].

For the second phase of the assignment, which focused on code development and experimentation, GitHub Copilot (specifically the Claude Sonnet 4.5 model), accessed via a student licence, was used directly within Visual Studio Code. This tool proved particularly effective due to its ability to leverage the full context of the existing codebase within the notebook environment. In addition to code completion, it supported lightweight code reviews (see Figure 16), data quality checks (see Figure 17), and enabled efficient data investigations, as illustrated in Figure 18.


CHAT

NOTEBOOK PREPROCESSING COMPARISON FOR MODELS

is the preprocessing performed in this notebook identical to #file:3a_RF_I.CUTAJAR.ipynb #file:svm_implementation.ipynb for the respective modelS?

svm_implementation.ipynb 3a_RF_I.CUTAJAR.ipynb

✓ Read current notebook and compared preprocessing

I'll compare the preprocessing steps across all three notebooks. Let me first read the current notebook to see what preprocessing is performed there.

✓ Read 4_results_comparison.ipynb, lines 1 to 150

Let me read more of the current notebook:

✓ Read 4_results_comparison.ipynb, lines 150 to 361

Now let me analyze the preprocessing differences:

Analysis of Preprocessing Differences

After comparing the three notebooks, here's what I found:

Random Forest (3a_RF_I.CUTAJAR.ipynb) vs Comparison Notebook:

✅ **IDENTICAL** - The preprocessing is the same:

- **Categorical features:** One-hot encoded with most_frequent imputation
- **Numerical features:** Median imputation only (no scaling)
- **Feature set:** Same features used

```
# Both use:
preprocess_rf = ColumnTransformer([
    ('cat', Pipeline([
        ('imp', SimpleImputer(strategy='most_frequent')),
        ('ohe', OneHotEncoder(handle_unknown='ignore'))
    ]), cat_cols),
    ('num', SimpleImputer(strategy='median'), num_cols)
])
```

SVM (svm_implementation.ipynb) vs Comparison Notebook:

❌ **DIFFERENT** - There are significant differences:

Aspect	SVM Original Notebook	Comparison Notebook
Feature selection	Different feature set (includes traffic_level, has different vehicle/driver features)	Uses RF's feature set (aligned for comparison)
Categorical encoding	No one-hot encoding for categoricals	One-hot encoding for region, street_type, accident_time_category
Numerical scaling	StandardScaler applied to specific numerical columns only	StandardScaler applied to ALL numerical features
Feature engineering	Manual one-hot encoding of time categories, street_type, region separately	Single ColumnTransformer handling all categorical features

3b_RF_ISAAC_CUTAJAR.ipynb • Cell 39

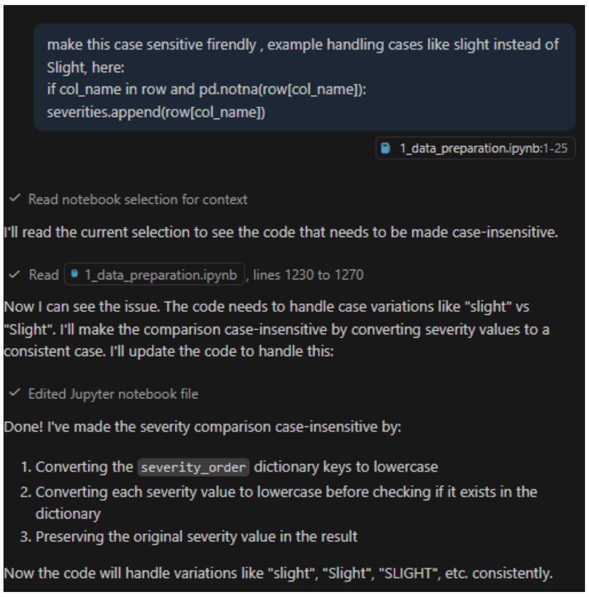
Describe what to build next

Agent Claude Sonnet 4.5

[Aa] Spaces: 4 Ab| Cell 39 of 42

Figure 16: Comparing preprocessing pipelines across model-specific notebooks.

[2] *ChatGPT conversation:* https://chatgpt.com/share/69792082-497c-8002-bf4e-6775b7b0c7e2



make this case sensitive firendly , example handling cases like slight instead of Slight, here:
if col_name in row and pd.notna(row[col_name]):
severities.append(row[col_name])

1_data_preparation.ipynb:1-25

✓ Read notebook selection for context

I'll read the current selection to see the code that needs to be made case-insensitive.

✓ Read 1_data_preparation.ipynb , lines 1230 to 1270

Now I can see the issue. The code needs to handle case variations like "slight" vs "Slight". I'll make the comparison case-insensitive by converting severity values to a consistent case. I'll update the code to handle this:

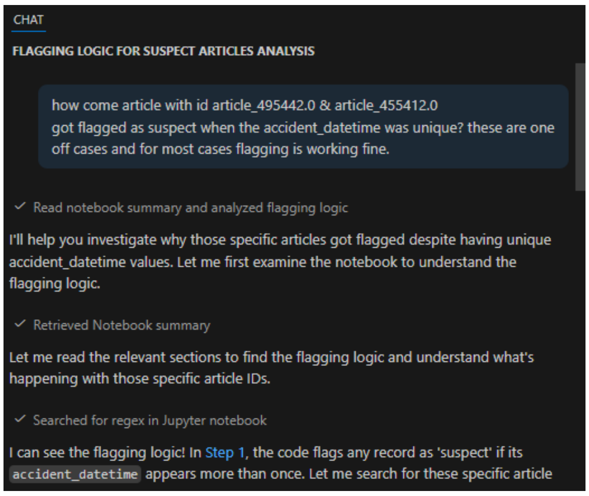
✓ Edited Jupyter notebook file

Done! I've made the severity comparison case-insensitive by:

1. Converting the `severity_order` dictionary keys to lowercase
2. Converting each severity value to lowercase before checking if it exists in the dictionary
3. Preserving the original severity value in the result

Now the code will handle variations like "slight", "Slight", "SLIGHT", etc. consistently.

Figure 17: Introducing case-insensitive handling of severity labels.



CHAT

FLAGGING LOGIC FOR SUSPECT ARTICLES ANALYSIS

how come article with id article_495442.0 & article_455412.0 got flagged as suspect when the accident_datetime was unique? these are one off cases and for most cases flagging is working fine.

✓ Read notebook summary and analyzed flagging logic

I'll help you investigate why those specific articles got flagged despite having unique accident_datetime values. Let me first examine the notebook to understand the flagging logic.

✓ Retrieved Notebook summary

Let me read the relevant sections to find the flagging logic and understand what's happening with those specific article IDs.

✓ Searched for regex in Jupyter notebook

I can see the flagging logic! In Step 1, the code flags any record as 'suspect' if its `accident_datetime` appears more than once. Let me search for these specific article

Figure 18: Analyse and debug the suspect-article flagging logic.

Overall, the use of GenAI tools contributed positively to both efficiency and understanding throughout the project, supporting not only technical development but also tasks such as LaTeX formatting and document structuring. Nevertheless, these tools were used as supportive aids rather than authoritative sources, with all final design choices taken after understanding the scenario in question and discussing them as a team if needed.

7 References and List of Resources Used

- OpenAI ChatGPT: https://chat.openai.com
- GitHub Copilot: https://github.com/features/copilot
- Course Lecture Notes for ICS5110 – Applied Machine Learning
- Scikit-learn Documentation: https://scikit-learn.org
- University of Malta AI and Academic Integrity Guidelines