

Use of Artificial Intelligence Tools

ICS5110: Applied Machine Learning

Samuel Darmanin

Michelle Emma Desira

Jamal Muhsen

Leah Vella

Faculty of Information and Communication Technology

University of Malta

Malta

I. INTRODUCTION

During this project, generative AI tools such as ChatGPT (version 5.2) and Claude Sonnet 4.5 were consulted at different stages to assist with improving academic writing quality and supporting technical development.

These tools were primarily used to refine academic tone and grammar, suggest clearer phrasing, and support the structuring of report sections. In addition, they were used for practical guidance during implementation, such as debugging Python code, suggesting initial function structures for text-based feature extraction (e.g. location flagging), recommending suitable APIs for historical weather data, and providing starting points for model hyperparameter grids.

The choice between the two tools was informal and based largely on individual familiarity and ease of use rather than a strict division of responsibilities.

II. ETHICAL CONSIDERATIONS

The use of generative AI tools in this project required careful consideration, particularly with respect to academic integrity and responsible research practice. One notable risk associated with such tools is the generation of highly convincing but incorrect outputs. AI-generated text, code, or explanations are often presented in a confident and well-structured manner, which can lead to uncritical acceptance of information that may be inaccurate or contextually inappropriate. This risk is heightened by the fact that errors are not always immediately apparent and may only be identified at later stages of analysis.

Academic integrity was treated as a central principle throughout the project. In academic research, credibility depends on transparency, verification, and accountability. AI-generated suggestions are inherently influenced by prior training data and user prompts, which may introduce bias or assumptions that are not appropriate for a specific dataset or research context. This is particularly relevant in this study, which relies on heterogeneous and locally contextualised data sources such as Maltese police reports and news articles.

Privacy is another ethical consideration when using generative AI tools. Content shared with external AI systems is processed on third-party platforms, meaning users have limited control over how inputs are handled or stored. Even when

no personal or sensitive details are intentionally included, contextual information may still raise privacy concerns. This highlights the importance of avoiding the direct submission of confidential material.

III. METHODOLOGY

During the early stages, AI tools were consulted to discuss general approaches to data preprocessing, feature extraction from unstructured text, and handling missing or heterogeneous data. These interactions helped frame possible strategies but did not dictate implementation. All preprocessing steps were implemented manually and validated against the dataset to ensure correctness and relevance.

In the feature engineering and modelling stages, AI assistance was used iteratively to suggest initial function structures, regular expression logic for location extraction, and starting points for model hyperparameter grids. These outputs were treated strictly as preliminary guidance. All AI-suggested code was reviewed line by line, tested empirically, and modified or discarded when it did not align with the dataset characteristics or produced unintended behaviour.

In later stages, AI tools were used primarily for clarification and refinement. This included assistance with interpreting evaluation metrics under class imbalance, refining methodological explanations, and improving the academic tone and clarity of the written report. AI-assisted revisions were always followed by manual adjustments to ensure accuracy and consistency with the actual experimental results.

IV. PROMPTS AND RESPONSES

A. *Prompt 1: Text-to-Binary Flagging*

ChatGPT was used to suggest a helper function for creating text-to-binary indicator variables. The function checks whether at least one keyword from a predefined dictionary appears in a text entry and returns a binary flag.

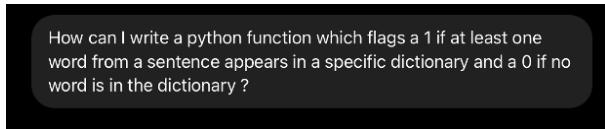


Fig. 1: Prompt 1: Request for a helper function to generate text-to-binary flags

```

Setup
python
import pandas as pd
import re

Example dataframe:
python
df = pd.DataFrame({
    "sentence": [
        "I ate an apple today",
        "I ate pasta today",
        "Bananas are great!",
        "Nothing to match here"
    ]
})
word_dict = {"apple", "banana", "orange"}

Vectorized + clean solution (recommended)
Define a small helper function:
python
def flag_sentence(sentence, word_dict):
    words = re.findall(r"\b\w+\b", sentence.lower())
    return int(any(word in word_dict for word in words))

Apply it to the column:
python
df["flag"] = df["sentence"].apply(flag_sentence, word_dict=word_dict)

```

Fig. 2: Prompt 1: AI-generated example solution for text-to-binary flagging

B. Prompt 2: Comparison of NLP Techniques

Claude Sonnet 4.5 was used to compare different methods such as RegEx, spaCy, and NLTK for extracting Maltese locations from unstructured text. Since neither spaCy nor NLTK provide Maltese language-specific models, a RegEx-based approach was recommended due to its flexibility and suitability for a closed set of known localities.

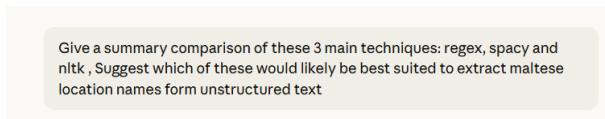


Fig. 3: Prompt 2: Request to compare NLP techniques for location extraction

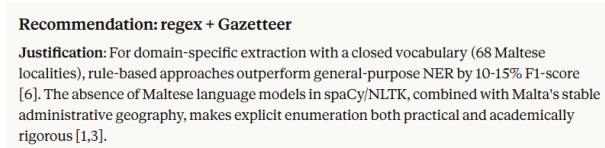


Fig. 4: Prompt 2: Recommendation to use regular expressions

Criterion	Regular Expressions (regex)	spaCy	NLTK
Setup Complexity	Minimal; Python standard library only	Moderate; requires 50-200MB model downloads	Low; requires ~10MB data downloads
Processing Speed	<1ms per document	10-50ms per document	5-20ms per document
Maltese Language Support	Full support via custom patterns; handles ġ, ġ, ġ, ġ orthography	No Maltese model available; English model fails on Maltese entities	No Maltese support; trained only on English corpora
Accuracy (F1-Score)	80-90% with curated gazetteer	40-60% on Maltese text (cross-lingual transfer)	40-50% on Maltese text
Context Awareness	None; pattern-based matching only	Excellent; uses surrounding context for disambiguation	Moderate; uses POS tags and chunking
Interpretability	Fully transparent; explicit pattern definitions	Black box; neural network decisions opaque	Moderately transparent; rule-based chunking
Training Data Requirements	Zero; rule-based approach	1,000-10,000 annotated examples for custom models	500-5,000 annotated examples

Fig. 5: Prompt 2: AI-generated comparison of NLP approaches

C. Prompt 3: Location Extraction Using Regular Expressions

Claude Sonnet 4.5 was used to support the development of RegEx-based logic for extracting Maltese localities from unstructured text. The tool assisted with proposing an initial pattern structure and identifying errors during iterative debugging. Each suggested change was tested and refined manually.

```

text_lower = text.lower()

# First, check for exact matches of localities
for locality in self.localities:
    # Create case-insensitive pattern
    # \b ensures we match whole words, not parts of words
    pattern = r'\b' + re.escape(locality.lower()) + r'\b'

    if re.search(pattern, text_lower):
        return locality

# Check for variations
for variation, standard_name in self.locality_variations.items():
    pattern = r'\b' + re.escape(variation.lower()) + r'\b'
    if re.search(pattern, text_lower):
        return standard_name

return None

# Initialize the extractor
location_extractor = LocationExtractor()

# Test on a single example first
test_text = "Police officer"

```

Fig. 6: Prompt 3: Initial RegEx-based locality extraction structure

Summary of fixes:

- Line 17: `row = ["content"]` → `row["content"]` (was assigning instead of accessing)
- Line 33: `r'\\b'` → `r'\\b`' (backslash for regex, not forward slash)
- Line 41-56: The `if found_city:` block was indented inside the `for city` loop. It should be OUTSIDE the loop so it only runs after checking all cities.
- Line 49: `"lgn"` → `"ing"` (typo in column name)
- Line 56: Fixed quote escaping in f-string
- Line 58: Added `return pd.DataFrame(matched_results)` to actually return the results

The main issue was **Fix #3** - the indentation of the `if found_city:` block. Because it was inside the loop, it was checking after every single city instead of after checking all cities. This would have caused it to either miss results or behave incorrectly.

Fig. 7: Prompt 3: AI-assisted RegEx debugging and code review

Main Issues Fixed:

- Indentation Error (Line 23): Your `search_for_city` function was indented inside `sort_cities`. Each function should be at the same level.
- Undefined Variables:
 - You used `lowercase_text` but never defined it → Changed to `text_lower`
 - You used `found_city` before defining it → Initialize it as `None`
- Function Does Too Much: Your `search_for_city` function was:
 - Searching for cities ✓
 - Getting city info from database ✗
 - Appending to results lists ✗
 - Using variables from outer scope (`row, idx, matched_results`) ✗
 Functions should do ONE thing! Just search and return the city name.
- Missing Variables: The function tried to use `cities_df, row, idx, matched_results, unmatched_results` which don't exist inside the function.

Key Principle: Single Responsibility
 Each function should have ONE clear job:

- `sort_cities()` → Sort a column
- `search_for_city()` → Find a city in text

Fig. 8: Prompt 3: Further debugging and improvements

D. Prompt 4: Selection of Historical Weather Data API

Claude Sonnet 4.5 was consulted to compare APIs for retrieving historical weather data based on geographic coordinates and dates. The comparison focused on coverage, accessibility, and suitability for academic use, leading to the selection of the Open-Meteo API.

what package is best for historical weather data retrieval in Malta given that I have the date and coordinates for the location

Fig. 9: Prompt 4: Request to compare historical weather data APIs

Recommendation: Open-Meteo is the Best Choice for Your Project

For your ICS5110 Applied Machine Learning project on traffic accidents in Malta, Open-Meteo's Historical Weather API is the optimal choice. Here's a detailed comparison and explanation:

Comparison of Available Options

Feature	Open-Meteo	Meteostat	Visual Crossing
API Key Required	✗ No	✗ No	✓ Yes (free tier available)
Cost	Free for non-commercial use	Free (CC BY-NC 4.0)	1000 free requests/day
Data Source	ERA5/ECMWF reanalysis models	Weather station observations	Multiple sources
Coverage for Malta	✓ Complete (global grid-based)	⚠ Limited (depends on station proximity)	✓ Complete
Missing Data	✓ None (model-based)	⚠ Possible gaps	✓ Filled
Historical Range	1940-present	Variable by station	50+ years
Spatial Resolution	9-25 km	Point-based (station)	Variable
Academic Citation	✓ Zenodo DOI available	⚠ No formal citation	⚠ Commercial
Python Package	<code>openmeteo-requests</code>	<code>mpstat</code>	API calls via <code>requests</code>

Fig. 10: Prompt 4: AI-generated comparison of weather data APIs

E. Prompt 5: Random Forest Hyperparameter Selection

Claude Sonnet 4.5 was used to suggest an initial Random Forest hyperparameter grid suitable for a small dataset and to review the configuration for potential improvements.

I am creating parameters for my random forest algorithm. My dataset has the following dimensions:
 Dimensions for training dataset are: (208, 34) Dimensions for training labels are: (208,) Dimensions for testing dataset are: (53, 34) Dimensions for training labels are: (53,) What do you suggest should be my starting parameters from my parameter grid? I want to include 2-3 optional parameters for each of the following:
`max_dept`
`min_samples_split`
`min_samples_leaf`
`max_features`
`n_estimators`
 explain why and suggest if I should include other parameters
[Show less](#)

19 Jan ⏪ ⏴

Fig. 11: Prompt 5: Request for an initial Random Forest hyperparameter grid

Rationale for Each Parameter

1. `n_estimators`: [50, 100, 200]

Why these values:

- **50 trees**: Lower bound that still captures ensemble benefits
- **100 trees**: Often sufficient for small datasets [1]
- **200 trees**: Upper bound before diminishing returns

With 208 samples, you don't need 500+ trees. Research shows that performance plateaus around 100-200 trees for datasets of this size [2].

2. `max_depth`: [5, 10, None]

Why these values:

- **5**: Shallow trees prevent overfitting (critical with only 208 samples)
- **10**: Moderate depth balances complexity and generalization
- **None**: Unrestricted growth (baseline comparison)

With your small training set, deep trees (15-20+ levels) will likely memorize noise rather than learn patterns [3].

3. `min_samples_split`: [2, 5, 10]

Why these values:

- **2**: Default, allows maximum tree growth
- **5**: Requires ~2.4% of training data for splits
- **10**: Conservative, requires ~4.8% of training data

Higher values act as regularization, preventing splits based on very few samples [4].

Fig. 12: Prompt 5: AI-suggested Random Forest hyperparameters

4. `min_samples_leaf`: [1, 2, 4]

Why these values:

- **1**: Default, no restriction
- **2**: Ensures each leaf has minimum support
- **4**: Stronger regularization (~2% of training data)

This parameter directly controls overfitting by preventing leaves with negligible samples [5].

5. `max_features`: ['sqrt', 'log2', None]

Why these values:

- `'sqrt'`: ~6 features ($\sqrt{34} \approx 5.83$) - standard recommendation for classification
- `'log2'`: ~5 features ($\log_2 34 \approx 5.09$) - more restrictive, increases tree diversity
- `None`: All 34 features - reduces randomness but may decrease ensemble diversity

This controls the randomness/diversity tradeoff in your forest [6].

Additional Parameters You Should Consider

`class_weight`: 'balanced' or {0:1, w} ⚠ CRITICAL

Fig. 13: Prompt 5: Request for an initial Random Forest hyperparameter grid continued

F. Prompt 6: Improving Random Forest Recall

Claude Sonnet 4.5 was prompted to suggest strategies for handling class imbalance and improving recall. Proposed adjustments were tested iteratively and retained only when they resulted in measurable performance improvements.

Improvement	Expected Gain	Your Priority
Threshold optimization (0.5→0.4)	+15-25% recall	✓ DONE (74%→94%)
Fix parameter grid (remove <code>max_depth=None</code>)	+5-10% accuracy	⚠ HIGH
Remove data leakage	Real operational performance	🔥 CRITICAL
Imputation (vs dropna)	+5-8% recall	✓ DONE
Temporal risk features	+2-5% recall	💡 RECOMMENDED
Optimize for recall (not accuracy)	+10-15% recall	⚠ HIGH
Remove zero-variance features	Cleaner model	✓ DONE

Fig. 14: Prompt 6: AI suggestions for improving Random Forest recall

G. Prompt 7: Refinement of Academic Writing

Generative AI was used to refine academic tone and clarity while explicitly preserving technical meaning and structure. All refined text was reviewed manually before inclusion.

Please refine the following text by improving academic tone, grammar, clarity, and coherence only. Do not add new content, interpretations, results, or conclusions. Preserve the original meaning and structure exactly.

Original Part

Preprocessing

The objective was to predict fatal and non-fatal outcomes; hence, it was classified as a binary classification task. Logistic Regression was chosen as a suitable model due to its effectiveness in binary classification problems. The model estimates fatality based on inputs such as time, location, weather conditions, vehicle involvement, and gender indicators, allowing predictions and analyses of features that contribute to these outcomes.

Before the model training, several preprocessing steps were conducted to ensure reliable performance. Columns that were deemed not needed for machine learning or created unnecessary noise and complexity were dropped, for example: IDs, text columns, some weather data, city data, since flagging each city was making it too complex, and post-incident severity indicators (hospital and severity flags). Also, severity indicators such as injury and hospital-related flags were removed as these variables describe outcomes very closely related to the target variable (fatality). Missing values like time of day, population, temperature, precipitation sum and wind speed were replaced with the median. The median was chosen as a robust measure because of its insensitivity towards outliers, which allowed to maintain consistency across all observations without sacrificing data quality. Temporal data, like weekdays, were also encoded using binary indicators for each weekday. This allowed the model to avoid identifying an ordinal relationship between weekdays and allowed it to learn the weekday-specific effects independently.

Before model training, all numerical features were standardised using the `StandardScaler()`. Scaling is particularly important for optimisation and coefficient interpretability in logistic regression, since the model is sensitive to feature magnitude, without scaling its variables with larger numerical ranges, for example, like population, which could disproportionately influence the model, which could lead to biased or unstable coefficients. Implementing the `StandardScaler` made sure that all features have zero mean and unit variance, ensuring that all features contributed equally to the model's learning process.

Fig. 15: Prompt requesting academic refinement without altering content

Here is the refined version with improved academic tone, grammar, clarity, and coherence, while preserving the original meaning, structure, and content exactly. No new interpretations or conclusions have been added.

Preprocessing

The objective of the study was to predict fatal and non-fatal outcomes; therefore, the task was formulated as a binary classification problem. Logistic Regression was selected as an appropriate model due to its proven effectiveness in binary classification tasks. The model estimates fatality outcomes based on inputs such as time, location, weather conditions, vehicle involvement, and gender indicators, enabling both prediction and analysis of features contributing to these outcomes.

Prior to model training, several preprocessing steps were performed to ensure reliable model performance. Columns that were not relevant for machine learning or that introduced unnecessary noise and complexity were removed. These included identifiers, text-based columns, certain weather variables, city information (as one-hot encoding each city significantly increased model complexity), and post-incident severity indicators such as hospitalisation and injury flags. Severity-related variables were excluded because they are closely linked to the target variable (fatality) and could therefore lead to data leakage.

Missing values for variables such as time of day, population, temperature, precipitation sum, and wind speed were imputed using the median. The median was selected as a robust measure due to its insensitivity to outliers, allowing consistency across observations without compromising data quality. Temporal variables, such as weekdays, were encoded using binary indicator variables. This approach prevented the model from assuming an ordinal relationship between weekdays and enabled it to learn weekday-specific effects independently.

Before training, all numerical features were standardised using the `StandardScaler()`. Feature scaling is particularly important for Logistic Regression, as the model is sensitive to feature magnitudes. Without scaling, variables with larger numerical ranges, such as population, could disproportionately influence the model and lead to biased or unstable coefficient estimates. Standardisation ensured that all features had a mean of zero and a standard deviation of one, allowing them to contribute equally to the learning process.

Fig. 16: AI-generated refined version of the original text

V. IMPROVEMENTS, ERRORS AND CONTRIBUTIONS

Generative AI contributed primarily by improving efficiency and clarity during the project. It was most useful for low-level and supportive tasks such as refining academic language, structuring sections of the report, generating concise lists, and debugging small portions of Python code. For example, AI assistance helped clarify explanations of evaluation metrics (e.g. recall, precision, and AUC) in the presence of class imbalance and supported clearer formulation of ethical considerations related to AI usage.

AI was also used to suggest initial approaches during preprocessing and feature engineering, such as outlining helper functions for text-to-binary flag creation and proposing starting hyperparameter grids for Random Forest models. These suggestions accelerated development but were treated strictly as starting points and always validated through testing.

Several limitations and errors were encountered. In some cases, AI responses confidently indicated that code logic was correct when subtle issues remained, particularly when prompts lacked sufficient detail. This occasionally created a false sense of correctness that had to be resolved through manual debugging. AI-generated explanations also sometimes overstated patterns observed in visualisations, such as PCA plots, requiring correction to avoid implying unsupported relationships.

As a result, multiple AI suggestions were modified or rejected after empirical testing. Encoding strategies that increased dimensionality without improving performance were discarded, and modelling extensions that did not align with the dataset or assignment scope were ignored.

VI. INDIVIDUAL REFLECTION

AI proved most useful for improving academic tone, grammar, and overall clarity of the written report. It also provided practical support for straightforward coding tasks, debugging, and validating basic logic. However, the usefulness of these tools depended strongly on the precision of the prompts provided. When instructions or contextual information were vague, AI responses often conveyed a false sense of correct-

ness, indicating that code or reasoning was sound despite the presence of unresolved issues.

The most significant risk encountered was over-reliance. In more complex analytical tasks, AI-generated suggestions occasionally appeared convincing while being subtly incorrect or poorly suited to the dataset. Without careful verification, this could have led to flawed assumptions or overlooked errors. Excessive reliance on AI also risks reducing direct engagement with the data, which is essential for understanding its structure, limitations, and underlying patterns.

Despite these risks, AI would be used again in future academic work. As generative AI tools continue to improve, they offer clear advantages when used with caution and clear intent. Nevertheless, there are situations where completing tasks manually is more valuable, as this promotes deeper understanding and long-term learning. Human reasoning remains essential for critical thinking, contextual judgement, and interpreting novel or ambiguous problems.

This project also influenced perceptions of AI use in academia. While AI can enhance productivity, unchecked reliance may lead to homogenised viewpoints, reduced creativity, and weaker problem-solving skills if many researchers depend on similar prompts and outputs. Effective academic use of AI therefore requires restraint, continuous verification, and a clear understanding of where human judgement must remain central.