Machine Learning Foundations
Data Preparation

February 5, 2025

1 Description

In this assignment, you will carry out data preparation tasks using The Complete Titanic Dataset. This is a well-known training dataset about the tragedy of the Titanic, the British ocean liner that sank in the Atlantic Ocean on April 15, 1912. You will acquire hands-on experience with data cleaning, preprocessing, encoding, and feature engineering, which are essential steps in the machine-learning pipeline.

2 Objectives

This assignment gives you the opportunity to:

- handle missing values in the dataset;
- encode categorical variables using one-hot encoding;
- apply feature imputation;
- apply feature scaling to numerical variables;
- split the data into training, validation, and test sets;
- address class imbalance;
- perform feature selection to identify the most relevant features; and
- train a simple Logistic Regression.

3 Dataset

The Complete Titanic Dataset can be downloaded from Kaggle. It is in Excel format (.xls), and you can use the Pandas method `read_excel` to load it into a DataFrame. This dataset includes the following classes:

1. **survival**: Survival (0 = No; 1 = Yes).
2. **class**: Passenger Class (1 = 1st; 2 = 2nd; 3 = 3rd).
3. **name**: Name.
4. **sex**: Sex.
5. **sibsp**: Number of Siblings/Spouses Aboard.
6. **parch**: Number of Parents/Children Aboard.
7. **ticket**: Ticket Number.
8. **fare**: Passenger Fare.
9. **cabin**: Cabin.
10. **embarked**: Port of Embarkation (C = Cherbourg; Q = Queenstown; S = Southampton).
11. **boat**: Lifeboat (if survived).
12. **body**: Body number (if did not survive and the body was recovered).

This data can be utilized to train models to predict various types of questions. For instance, will a passenger survive? What factors influence survival? Can we group passengers? What was the ticket price? Are there any anomalies in the data? For this assignment, we concentrate on data exploration and feature engineering for the following ML problem: **Will a passenger survive?**

4 Instructions

Code the following tasks in Python using a Jupyter Notebook. Create a GitHub (GH) repository for this course and push your notebook to that repository. Name your Jupyter Notebook using the following naming template: `assignment_1_<name_surname[_surname]>.ipynb`. You should not push your notebook only at the end of the assignment, right before submitting it. Instead, get used to pulling and pushing to your GH repository while working on the assignment. This practice will be beneficial once you start collaborating with your team on a shared codebase. See §7 for more information about using GH.

Explain your approach, reasoning for your choices, and the conceptual tools you employ for each of the following tasks. For instance, detail why you selected a particular method to handle missing values, whether you opted to remove a column from the dataset and the reasoning behind this decision, why a particular feature engineering technique was utilized, and the anticipated effect of your feature engineering on the models that could be used to address the stated problem.

You will encounter the following potential challenges:

- Effectively managing missing values: determining when to drop versus when to impute.
- Understanding the rationale for feature engineering: recognizing why certain features should be removed or transformed.
- Correctly balancing the dataset: weighing the trade-offs of oversampling versus undersampling.
- Selecting features beyond correlation: employing automated selection techniques.

Task 1: Data Loading and Initial Exploration

Lecture material: Lecture 3, slides 4–8, 10, and 11.

- Load the dataset into a Pandas `DataFrame`.
- Perform basic exploratory data analysis (EDA) to comprehend the structure and characteristics of the data.

Note: Your analysis should include appropriate exploratory statistics and visualizations.

Task 2: Managing Missing Values

Lecture Material: Lecture 3, slides 22–24.

- Identify the columns containing missing values.
- Develop a strategy to address them.

Task 3: Encoding Categorical Variables

Lecture material: Lecture 4, slides 10–15, 21.

- Identify the categorical variables in the dataset.
- Utilize `OneHotEncoder` to encode them.
- Observe the transformation and discuss its impact on machine learning models.

Task 4: Feature Scaling

Lecture material: Lecture 5, slides 14–20.

- Standardize the numerical variables using `StandardScaler`.
- Normalize the numerical variables using `MinMaxScaler`.
- Discuss the differences between standardization and normalization, along with their importance.

Task 5: Data Splitting

Lecture material: Lecture 2, slides 4–7.

- Split the dataset into training, validation, and test sets.
- Ensure that the split reflects the original distribution of the target variable using stratification.

Note: a good strategy is to first split the dataset into 'training' and 'others', and then split 'others' into equally sized 'validation' and 'test' sets. When splitting sets, consider the argument `stratify` of the `train_test_split` method.

Task 6: Addressing Class Imbalance

Lecture material: Lecture 3, slides 25–27; Lecture 4, slides 4–5.

- Apply a method to address class imbalance (e.g., Oversampling Technique (SMOTE), Adaptive Synthetic Sampling Method (ADASYN)).

Note: You can load a SMOTE and/or ADASYN implementation from the Python module `imblearn`.

Task 7: Feature Selection

Lecture material: Lecture 5, slides 10–14, 19.

- Eliminate low variance and highly correlated features.
- Why do we carry out tasks 6 and 7 after splitting the dataset into training, validation, and test sets? Could we have conducted them on the entire dataset instead? Please elaborate on your answer.

Task 8: Training a Logistic Regression Model

Lecture material: Lecture 6, slides 5–9.

- Train a Logistic Regression Model to predict whether a passenger survives.

Note: Use the method `predict` from the class `LogisticRegression` with the validation set. Have fun finding a visually appealing way to display the results of the predictions on the validation set. An analysis of model performance is not required and will not affect your final grade for the assignment. However, I won't stop you from including it. 8-)

5 Deliverables

A Jupyter Notebook containing code and a detailed explanation for tasks 1–8. The explanation should approximately equate to a standalone document of at least two pages.

6 Evaluation Criteria

- **Correct Implementation of Tasks** (40%): Ensure that all steps (data loading, preprocessing, encoding, scaling, feature selection) are properly executed.
- **Explanation and Rationale** (30%): Justify your specific choices and explain the methods, techniques, and parameter selections that informed those decisions.
- **Code Organization and Cleanliness** (10%): Your code should be modular, include meaningful and informative comments (for instance, the comment 'this is a variable definition' is not helpful!), and properly utilize GitHub.
- **Depth of Analysis** (20%): Evaluate how thoroughly you explored and interpreted results instead of merely applying techniques without understanding.

WARNING: Code that fails to compile will be deemed invalid and contribute 0 to the overall grade, regardless of how trivial the error may be. **You must test the entire notebook before submission**. To perform a complete test, execute 'Restart the kernel and run all cells' right before submitting the assignment and ensure that every cell in the notebook compiles successfully. One last thing... Did I mention that you must test the entire notebook before submission?!

7 Minimal Everyday GitHub Guide

Follow this workflow to create and maintain your code in a course repository. This assumes you are using Git from a shell, but the same process can be followed when using an integrated development environment with GitHub integration or module.

1. Log into GitHub and select New Repository.
2. Assign the name `ML-fundamentals-2025`; do not change this name.
3. Check 'Add a README file.'
4. Select Python under 'Add .gitignore.'
5. Choose an MIT license.
6. Go into your new repository and copy the clone string under 'Clone.' **Note**: I assume you have already configured GitHub with your SSH key. If not, please follow a suitable tutorial for that.
7. In a terminal on your computer, use the cloning command you just copied to clone your repository locally. **Note**: this assumes you have already installed the 'git' command on your local workstation or laptop. If not, please follow a suitable tutorial for that.
8. Create your `assignment_1_<name_surname[_surname]>.ipynb` notebook file within the directory `ML--fundamentals-2025`.
9. Start editing your notebook by executing the assignment's tasks.
10. When you finish a task, take a break, or are done for the day, issue the following commands from the terminal, inside your repository directory:
    - `git add assignment_1_<name_surname[_surname]>.ipynb`
    - `git commit -m 'Implementing Task n'`
    - `git pull`
    - `git push`
11. It is **essential** that you get into the habit of issuing a `git pull` command **before** you start editing your notebook. This will help establish a good habit and avoid conflicts when working in a team on the same codebase.