AIT511-MT Course Project-1 Report

Saatvik Sinha (MT2025722) and Affan Shaikh (MT2025016)

October 2025

1 Introduction

All experiments for this project were conducted in Google Colab. The complete implementation, including all notebooks and scripts, is available at the following GitHub repository: https://github.com/Saatvik-Sinha/AIT-511-MT-Project1

A total of four notebooks were created to perform different experimental setups for this project:

- **OG Notebook:** This notebook contains the initial implementation involving installation of required Python libraries, dataset fetching, exploratory data analysis (EDA), and preprocessing of training and testing datasets without any assumptions or alterations. Multiple models were implemented and trained using grid-search-based hyperparameter tuning. The XGBoost model, for specific hyperparameter values, achieved the best overall performance.
- **Concat Notebook:** In this notebook, the training dataset was merged with the original dataset it was sampled from, followed by removal of duplicate records. Grid search with XGBoost was repeated on this extended dataset, resulting in a noticeable improvement in performance compared to the initial setup.
- **RobustPCA Notebook:** This experiment applied a robust form of Principal Component Analysis (PCA), resistant to outliers, on the continuous features of the concatenated dataset. The top principal components with the highest variance were used to replace the original continuous features. Multiple models were retrained on this transformed dataset, leading to the following observations:
  - Performance significantly decreased for tree-based classifiers.
  - Performance showed negligible improvement for the Naive Bayes classifier.
  - Performance significantly improved for KNN and BallTree classifiers.

  However, none of these models surpassed the XGBoost results obtained in the “OG” and “Concat” notebooks.

- **Oversampling Notebook:** This notebook applied the Synthetic Minority Oversampling Technique (SMOTE) on the concatenated training dataset to address class imbalance by generating additional samples for minority classes. XGBoost was retrained on the balanced dataset, but contrary to expectation, the model performance declined.

2 Setup

Install all necessary Python libraries using `pip`.

Upload the `kaggle.json` credentials file from the Kaggle account.

Import and unzip the dataset files directly from Kaggle using the API and the uploaded credentials.

Import all required Python libraries for experimentation.

Read the original, training, and testing CSV files into pandas dataframes.

3 Exploratory Data Analysis

The following exploratory data analyses were performed:

- **Univariate analysis** of categorical features to determine the frequency distribution of each category.
- Examination of **categorical feature distributions** with respect to the target variable.
- Examination of **continuous feature distributions** with respect to the target variable.

4 Data Processing Steps

- In all notebooks except "OG", the training dataset was concatenated with the original dataset, followed by removal of duplicate records.
- Categorical variables with two classes were label-encoded into binary (0/1) values.
- Categorical variables with four ordered categories `[no, sometimes, frequently, always]` were ordinally encoded as {`no:0, sometimes:1, frequently:2, always:3`} to preserve their inherent frequency order.
- The `MTRANS` column contained five modes of transportation. Based on EDA, the modes `[Walking, Bike, Motorbike]` and `[Automobile, Public Transportation]` showed similar target distributions. Hence, they were grouped into two bins labeled as 0 and 1 respectively.

- Standard scaling was applied to continuous features. In the "RobustPCA" experiment, only mean-centering was performed since standard deviation is not robust to outliers.
- In the "RobustPCA" notebook, singular vectors of continuous columns were computed and used to transform them into principal components. The top four components (out of eight) were retained and added to the feature set, replacing the original continuous columns.
- The target variable `WeightCategory` with seven classes was label-encoded.
- In the "Oversampling" notebook, SMOTE was applied to handle class imbalance. The `k_neighbors` parameter was tuned, and the default value of 5 yielded the best result.

5 Models Used

The following machine learning models were implemented and compared:

- K-Nearest Neighbors (KNN)
- BallTree
- Naive Bayes
- Decision Tree
- Random Forest
- AdaBoost
- Gradient Boost (only in the "OG" notebook, as it was computationally expensive and did not outperform XGBoost)
- XGBoost

6 Hyperparameter Tuning

Grid Search with Stratified K-Fold Cross-Validation ($K = 5$) was employed for hyperparameter optimization across models.

After each full grid search run, the hyperparameter ranges were iteratively refined to narrow down optimal values. Model evaluation relied on the mean validation accuracy across folds and overall training accuracy.

Additionally, confusion matrices were visualized, and detailed classification reports including precision, recall, F1-score, and support were generated for each target category.

7 Comparative Analysis

From the conducted experiments, the following conclusions were drawn:

- XGBoost, with optimally tuned hyperparameters, consistently delivered the best performance across all experiments.
- After applying Robust PCA:
  - Tree-based classifiers experienced a significant drop in performance.
  - Naive Bayes classifier performance showed negligible improvement.
  - KNN and BallTree classifiers exhibited notable performance gains.

  Nevertheless, none of the models surpassed the performance of XGBoost trained in the “OG” and “Concat” notebooks.