National College of Ireland

Configuration Manual

MSc Research Project
MSc in Data Analytics

Venkata Rakesh Chowdary Nallagatla
Student ID: 23300400

School of Computing
National College of Ireland

Supervisor: Vikas Tomer

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Venkata Rakesh Chowdary Nallagatla
Student ID:	23300400
Programme:	MSc in Data Analytics
Year:	2025
Module:	MSc Research Project
Supervisor:	Vikas Tomer
Submission Due Date:	15/09/2025
Project Title:	Configuration Manual
Word Count:	723
Page Count:	4

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Rakesh Chowdary
Date:	15/09/2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Venkata Rakesh Chowdary Nallagatla
23300400

1 System Overview

This project develops a fairness-aware and interpretable machine learning pipeline for predicting hospital readmissions among diabetic patients. The system integrates several key components:

- **Data preprocessing:** Includes merging, imputation, encoding, and SMOTE-based class balancing to ensure a clean and equitable dataset.
- **Modeling:** Employs a Transformer-based architecture in combination with a Random Forest classifier, and benchmarks its performance against XGBoost and Logistic Regression models.
- **Evaluation:** Utilizes traditional performance metrics such as AUC-ROC and accuracy, alongside fairness metrics including Disparate Impact Ratio and Equal Opportunity Difference.
- **Interpretability:** Uses SHAP (SHapley Additive exPlanations) to enhance model transparency and provide insight into feature contributions.

The entire pipeline is implemented in Python using the `PyTorch`, `Scikit-learn`, and `SHAP` libraries, and executed within the Google Colab environment to leverage GPU acceleration.

2 Software Requirements

The implementation of the machine learning pipeline relies on the following software stack and computational environment:

- **Python Version:** Python 3.8 or higher.
- **Libraries and Packages:**
  - `PyTorch` – for building and training the Transformer model.
  - `Scikit-learn` – for data preprocessing, Random Forest, and XGBoost models.
  - `Pandas` – for efficient data manipulation and analysis.
  - `NumPy` – for numerical computations.
  - `SHAP` / `LIME` – for model interpretability and explanation.

  - `Imbalanced-learn` – specifically for SMOTE-based class balancing.
  - `Matplotlib` / `Seaborn` – for data and result visualizations.

- **Execution Environment:** Recommended execution via Google Colab for GPU acceleration. Alternatively, a local Jupyter Notebook can be used.

3 Hardware Requirements

To ensure smooth execution and efficient training of the machine learning models, the following hardware specifications are recommended:

- **Minimum Requirements:** 8 GB RAM and a 4-core CPU are sufficient for small-scale experiments and initial data processing.
- **Recommended Setup:** Google Colab Pro with GPU acceleration (Tesla T4 or P100) is preferred for training Transformer-based models, significantly reducing computation time.
- **Storage:** At least 10 GB of free disk space is required to accommodate datasets, temporary files, and saved model checkpoints.

4 Installation Guide

To set up the environment and run the hospital readmission prediction pipeline, follow these steps:

1. **Set up Python environment:**

```
conda create -n readmission_env python=3.8
conda activate readmission_env
```

2. **Install required libraries:**

```
pip install torch scikit-learn pandas shap imbalanced-learn xgboost lime
```

3. **Download the required datasets:**
   - Diabetic Patients Readmission Prediction Dataset (from Kaggle).
   - US Healthcare Data (Hospital Readmission Dataset) from Kaggle.

5 Data Preparation

The following steps were performed to prepare the dataset for model training and evaluation:

1. **Merge Datasets:** Combined multiple data sources using common identifiers such as patient IDs.
2. **Preprocessing:**
   - Imputed missing values using appropriate statistical methods (median for numerical, mode for categorical).
   - Normalized numerical features using min-max scaling to ensure uniformity.
   - One-hot encoded categorical variables such as gender and race to make them suitable for machine learning algorithms.
3. **Balance Classes:** Applied SMOTE (Synthetic Minority Oversampling Technique) to the training dataset to address class imbalance.
4. **Split Data:** Divided the dataset into 80% training and 20% testing sets, using stratified sampling based on readmission status to maintain class proportions.

```
[ ] from imblearn.over_sampling import SMOTE
    from sklearn.model_selection import train_test_split

    # Split first to avoid data leakage from SMOTE
    X = df_encoded.drop(columns=['readmitted_binary'])
    y = df_encoded['readmitted_binary']

    X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=0.2, stratify=y, random_state=42
    )

    # Apply SMOTE to training data

    sm = SMOTE(random_state=42, k_neighbors=1)
    X_train_bal, y_train_bal = sm.fit_resample(X_train, y_train)
```

Figure 1: Code Snippet

6 Model Training and Evaluation

The model training and evaluation pipeline combines a Transformer-based feature extractor with a Random Forest classifier. The process includes the following steps:

1. **Train Transformer:**
   - Configured multi-head self-attention layers with 4 attention heads and an embedding size of 64.
   - Trained the model for 20 epochs using the Adam optimizer with a learning rate of 0.001.
2. **Extract Embeddings:** The output embeddings generated by the trained Transformer model were used as feature inputs to the Random Forest classifier.

3. **Evaluation:**

   - **Metrics:** Evaluated the model using AUC-ROC, accuracy, precision, and recall.
   - **Cross-validation:** Performed 5-fold stratified cross-validation to ensure robustness.
   - **Threshold Tuning:** Adjusted the classification threshold to 0.2 to improve recall performance.

7 Troubleshooting

Common issues encountered during training and deployment, along with their suggested solutions, are summarized below:

Issue	**Solution**
CUDA out of memory	Reduce batch size or use Google Colab Pro with upgraded GPU (e.g., Tesla T4 or P100).
Class imbalance persists	Increase the SMOTE sampling ratio or verify that class balancing is only applied to the training set.
Low test AUC	Check for data leakage or overfitting. Consider tuning hyperparameters such as dropout rate in the Transformer model.

Table 1: Common issues and their solutions during model training and evaluation.