**Predicting Hospital Admissions Using Machine Learning Techniques**

**Project Overview**

This project aims to improve the prediction of hospital admissions in emergency departments by addressing limitations in the original study and introducing new methodologies. Key enhancements include the integration of temporal analysis using ARIMA to capture seasonal and time-based patterns, incorporating external factors such as weather data, and improving data processing through advanced missing value imputation techniques like KNN. The project also tackles class imbalance using SMOTE and ADASYN to enhance model performance for underrepresented cases. By employing temporal data splitting and optimizing models like XGBoost and DNN, this research provides a more comprehensive and accurate framework for predicting hospital admissions, ultimately supporting better resource allocation and operational efficiency in emergency departments.

**Setup Instructions**

**1. Required Software**

* **Python: Version 3.8 or higher.**
* **Recommended IDE: VS Code, Jupyter Notebook.**

**2. Required Python Libraries**

Before running any script, make sure the following libraries are installed :

**Required Libraries:**

* pandas: For data manipulation and analysis.
* numpy: For numerical operations and arrays.
* scikit-learn: For preprocessing, model training, and evaluation tools.
* imbalanced-learn: For data balancing techniques such as SMOTE and ADASYN.
* xgboost: For implementing the XGBoost model.
* tensorflow: For building and training Deep Neural Networks (DNNs).

**To install libraries one by one, use the following commands:**

pip install pandas

pip install numpy

pip install scikit-learn

pip install imbalanced-learn

pip install xgboost

pip install tensorflow

pip install matplotlib

pip install seaborn

**3. Folder Structure**

The project package is organized into the following folders:

* **README.md**: Documentation file.
* **data/**: Contains dataset files.
  + Data.csv: Raw dataset.
* **scripts/**: Python scripts for all tasks.
* **outputs/**: Stores results such as models and evaluation metrics.
  + processed\_knn\_data.csv: Data processed using KNN imputation.
  + processed\_median\_data.csv: Data processed using Median imputation.
  + balanced\_smote\_data.csv: Balanced dataset using SMOTE.
  + balanced\_adasyn\_data.csv: Balanced dataset using ADASYN.
  + split\_standard\_\*: Standard split files.
  + split\_temporal\_\*: Temporal split files.

**Scripts:**

1. **Preprocessing and Data Cleaning**

Purpose: Handle missing values and prepare data for modeling.

* **Techniques Used:**
  1. KNN Imputation: Fills missing values based on the nearest neighbors.
  2. Median Imputation: Replaces missing values with the median.
* **Scripts**:
  1. scripts/preprocess\_knn.py: Handles missing values using KNN.
  2. scripts/preprocess\_median.py: Handles missing values using Median.
* **Outputs**:
  1. data/processed\_knn\_data.csv
  2. data/processed\_median\_data.csv

1. **Data Balancing**

Purpose: Balance the target classes to improve model performance.

* **Techniques Implemented**:
  1. **SMOTE (Synthetic Minority Oversampling Technique)**:
     + Generates synthetic samples for the minority class.
  2. **ADASYN (Adaptive Synthetic Sampling)**:
     + Focuses on challenging samples for effective balancing.
* **Scripts**:
  1. scripts/class\_balancing\_smote.py: Implements SMOTE for balancing.
  2. scripts/class\_balancing\_adasyn.py: Implements ADASYN for balancing.
* **Outputs**:
  1. data/balanced\_smote\_data.csv
  2. data/balanced\_adasyn\_data.csv

1. **Data Splitting**

Purpose: Split the data into training, validation, and testing sets.

* **Techniques Implemented**:
  1. **Standard Splitting**:
     + 80% training, 10% validation, 10% testing.
  2. **Temporal Splitting**:
     + Uses historical data for training and future data for testing.
* **Scripts**:
  1. scripts/split\_standard.py: Implements standard splitting.
  2. scripts/split\_temporal.py: Implements temporal splitting.
* **Outputs**:
  1. data/split\_standard\_\*: Standard split files.
  2. data/split\_temporal\_\*: Temporal split files.

1. **Model Implementation**

Purpose: Train, tune, and evaluate predictive models.

* **Models Used**:
  1. **Logistic Regression (LR)**: Baseline linear model.
  2. **XGBoost**: Decision tree-based model optimized for accuracy.
  3. **Deep Neural Network (DNN)**: Advanced multi-layer perceptron for prediction.
* **Scripts**:
  1. scripts/train\_lr.py: Trains Logistic Regression.
  2. scripts/train\_xgboost.py: Trains XGBoost.
  3. scripts/train\_dnn.py: Implements and tunes DNNs.
* **Outputs**:
  1. Models and evaluation metrics saved in outputs/.

1. **ARIMA Model and Time Series Analysis**

Purpose: Analyze temporal patterns in data.

* **Tasks**:
  + Prepare weather data .
  + Train ARIMA for time series analysis and prediction.
* **Scripts**:
  + scripts/arima\_analysis.py: Implements ARIMA.

1. **Feature Enrichment**

Purpose: Enhance predictive power by adding new features.

* **Techniques**:
  1. Integrate **weather data**.
  2. Add **seasonal information** (e.g., month, quarter).
* **Scripts**:
  1. scripts/feature\_enrichment.py: Merges and cleans additional features.