**Stroke Prediction Using Machine Learning Predictive Models**

**1. Project Description**

This project aims to **predict** the likelihood of a **stroke** based on demographic, health, and lifestyle factors. Using **machine learning models**, we processed the dataset, handled **class imbalances**, performed **feature selection**, and optimized **hyperparameters** to achieve better predictive performance.

**The key objectives of this project include:**

* Data Preprocessing
* Feature Selection & Engineering
* Model Training & Evaluation
* Model Hyperparameter Tuning

This README provides an overview of the dataset, preprocessing steps, model development, and final results to help others understand and replicate the process efficiently.

We used **NumPy** and **Pandas** for data manipulation, and **Scikit-learn** for preprocessing, feature selection (**SelectKBest, RFE**), and training models like **Logistic Regression, Random Forest, and Gradient Boosting**. To address class imbalance, we applied **SMOTE** from **imblearn**. **GridSearchCV** optimized hyperparameters, while evaluation metrics (accuracy, precision, recall, F1-score, ROC AUC) helped assess and refine model performance for better stroke risk classification.

**2. Dataset & Preprocessing**

**Feature Selection:**

* Applied SelectKBest (ANOVA F-test) to retain the top 10 most informative features.
* Used RFE (Recursive Feature Elimination) with an SVM to refine the feature set further.

**Data Splitting & Resampling:**

* 80/20 train-test split with stratification to maintain class balance.
* **SMOTE (Synthetic Minority Over-sampling Technique)** was applied to address class imbalance.

**Feature Scaling:**

* Standardized numerical features using StandardScaler for better model performance.

**Models Trained & Evaluated**

We trained the following machine learning models:  
**Logistic Regression** – Baseline model for comparison.  
**Random Forest Classifier** – Showed strong performance after tuning.  
**Gradient Boosting Classifier** – Performed well with feature interactions.  
**K-Nearest Neighbors (KNN)** – Required scaling but was sensitive to data distribution.  
**Naive Bayes** – Worked well with categorical features but had lower overall accuracy.  
**Neural Network (MLPClassifier)** – More complex, but struggled with limited data.

**Model Metrics Evaluation:**

* **Accuracy**
* **Precision & Recall**
* **F1 Score**
* **ROC-AUC Score**

**3. Hyperparameter Tuning**

We performed GridSearchCV on the **Random Forest model**, optimizing:

* n\_estimators (200, 300, 500)
* max\_depth (10, 15, 20)
* min\_samples\_split (5, 10)
* min\_samples\_leaf (2, 3, 5)

**Best Parameters Found:**

n\_estimators = 300, max\_depth = 15, min\_samples\_split = 10, min\_samples\_leaf = 3

**4. Final Model & Adjustments**

* **Training the Final Random Forest Model** with the best parameters.
* **Threshold Adjustment:** Used Precision-Recall Curve to determine the best probability threshold instead of the default 0.5.

**Final Performance Metrics:**

|  |  |
| --- | --- |
| **Metric** | **Score** |
| Accuracy | 76.81% |
| Precision | 16.19% |
| Recall | 57.14% |
| F1 Score | 25.24% |
| ROC AUC | 74.31% |

***Amirhossein Ghalaei***

***Data Analyst***

***2024-03-26***