STROKE PREDICTION

ATISH , PRANAV , RHEA , HARI , RAVINDAR ,NIKSHIPTHA , CHARAN[*]

April 21, 2024

Abstract

This paper reports our experience with building a Stroke Prediction classifier with Machine Learning Algorithms. We employ a dataset encompassing various parameters, including gender, age, lifestyle choices, presence of different diseases, like hypertension and/or heart diseases, and smoking status, to predict the likelihood of a patient to experience a stroke. Our model employs machine learning techniques to analyze input features and generate predictions. By accurately identifying potential stroke patients, this model can significantly contribute to preventive healthcare strategies and improve patient outcomes.

Keywords: Machine Learning, Stroke, Healthcare, K-Nearest Neighbours, Decision Trees, Random Forest, Artificial Neural Network, Naive Bayes, SMOTE, hyper-tuning of parameters

Contents

[*]

1 Introduction

Stroke is the second leading cause of death globally, accounting for approximately 11% of total deaths. According to World Health Organization (WHO), strokes are defined as an acute, global disturbance or dysfunctions of the blood vessels supplying the causes of limb paralysis, severe morbidity, and unconsciousness. Early prediction of stroke occurrence is crucial for preventive healthcare. Understanding and predicting stroke risk can aid in preventive measures and early intervention, potentially saving lives. Researchers have studied different methodologies such as prospective cohort studies, case-control studies, and case series in the past several decades and identified nonmodifiable risk markers (genetic factors, male gender, older age) and modifiable risk factors (Hypertension, Cigarette smoking, Diabetes Mellitus).

The mortality rate and the number of people affected by this disease are expected to increase with the world's population. However, this mortality rate can be prevented by early treatment and prediction. Tools like the Cox proportional hazard model are used to predict strokes, but they struggle with high-dimensional data due to their traditional nature. In such cases, machine learning can significantly improve stroke prediction efficiently and cost-effectively. Various machine learning classifiers, such as Random Forest, Decision Tree, Naive Bayes, K-Nearest Neighbours and Artificial Neural Network, have been used for this project.

This report presents an analysis of a dataset aimed at predicting stroke occurrence based on those factors. We will pre-process this data in order to apply our chosen Machine Learning algorithms. we will be able to compare all these algorithms with respect to their accuracy in predicting data, and testing performance on different benchmark scores such as precision, recall and F1- score. This report concludes determining which method would be the best to predict the likelihood of patient to suffer through a stroke.

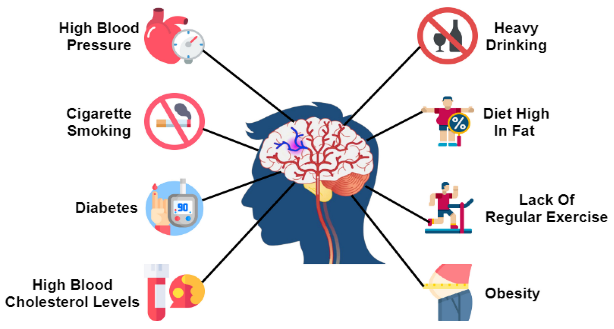
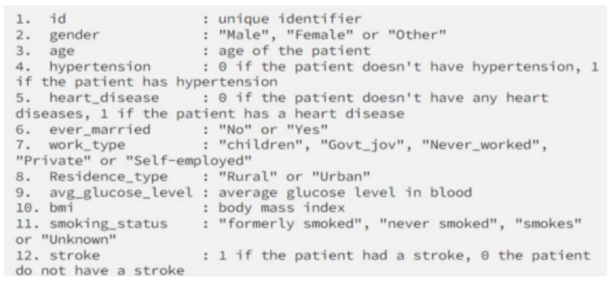


Figure 1: Factors contributing to Stroke

Dataset Link - https://www.kaggle.com/fedesoriano/stroke-prediction-dataset

1.1 Figures



```
1.  id                : unique identifier
2.  gender            : "Male", "Female" or "Other"
3.  age               : age of the patient
4.  hypertension      : 0 if the patient doesn't have hypertension, 1
if the patient has hypertension
5.  heart_disease     : 0 if the patient doesn't have any heart
diseases, 1 if the patient has a heart disease
6.  ever_married      : "No" or "Yes"
7.  work_type         : "children", "Govt_jov", "Never_worked",
"Private" or "Self-employed"
8.  Residence_type    : "Rural" or "Urban"
9.  avg_glucose_level : average glucose level in blood
10. bmi               : body mass index
11. smoking_status    : "formerly smoked", "never smoked", "smokes"
or "Unknown"
12. stroke            : 1 if the patient had a stroke, 0 the patient
do not have a stroke
```

Figure 2: Data Features

1.2 Data Visualisation and Data Description

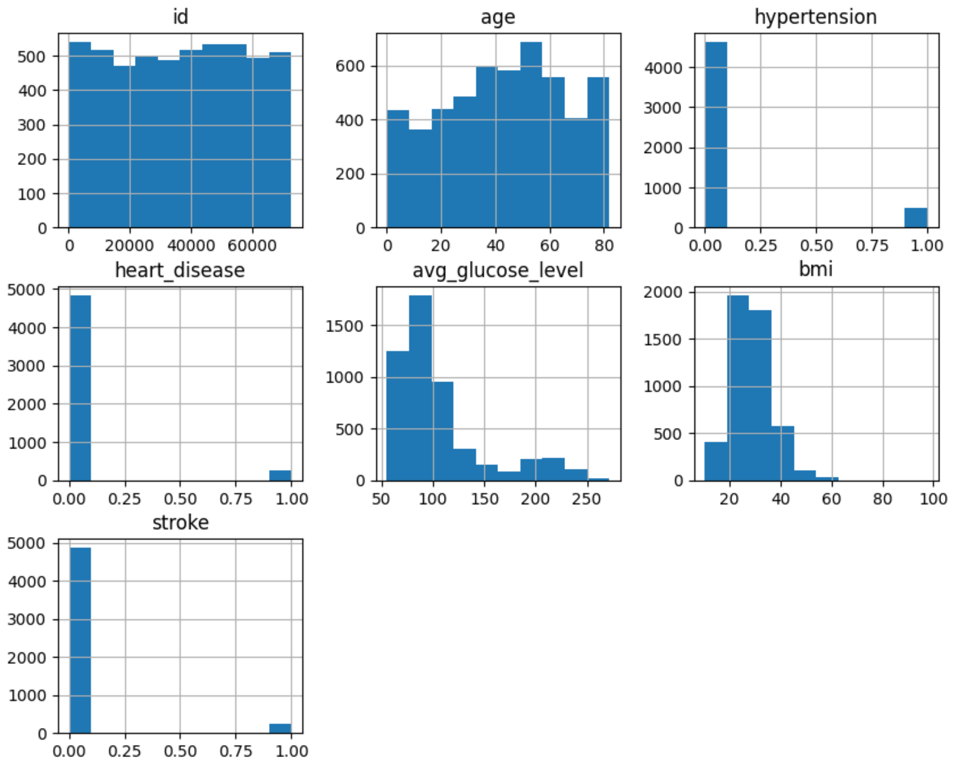


Figure 3: Data Visualisation

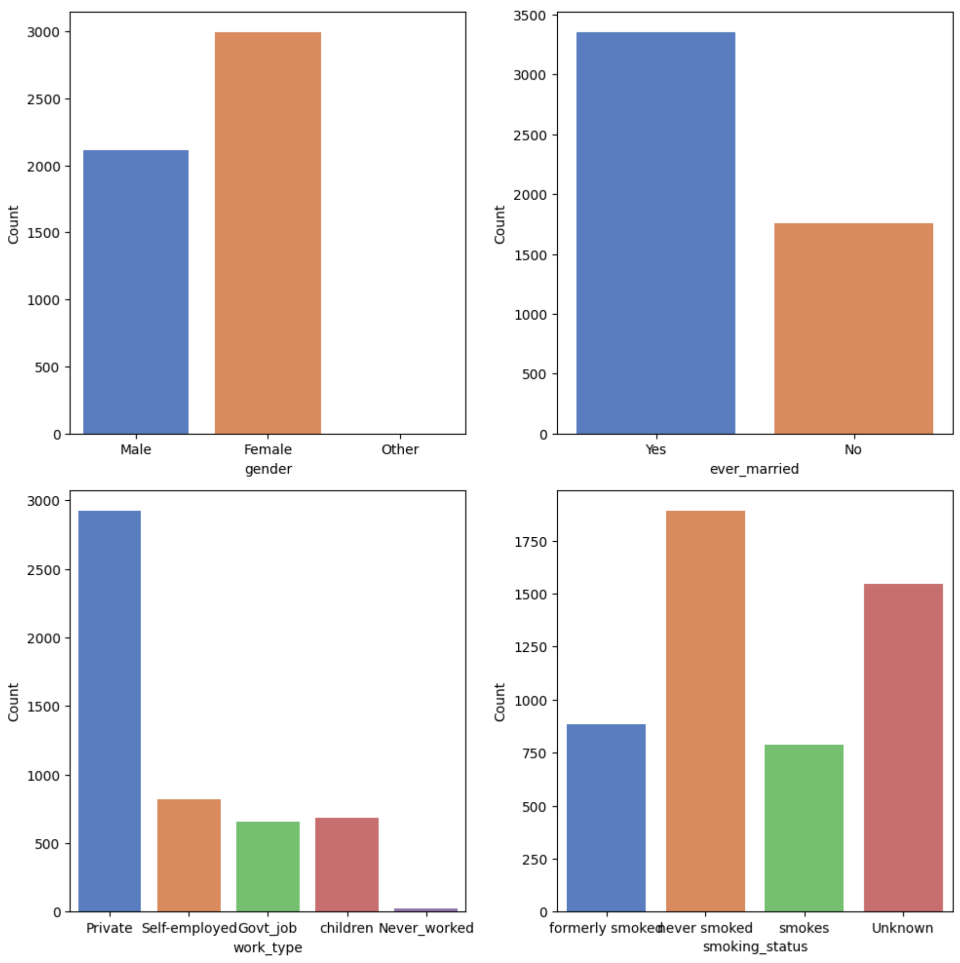
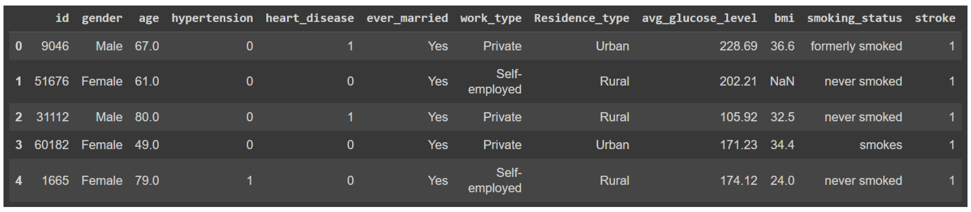
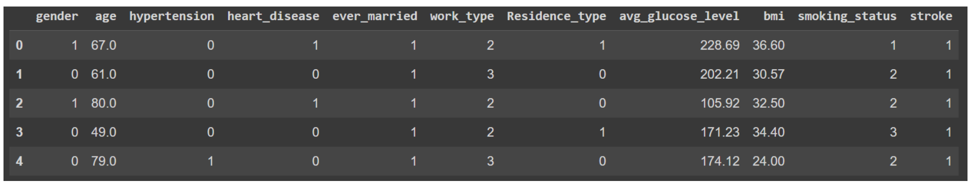


Figure 4: Data Visualisation



	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	9046	Male	67.0	0	1	Yes	Private	Urban	228.69	36.6	formerly smoked	1
1	51676	Female	61.0	0	0	Yes	Self-employed	Rural	202.21	NaN	never smoked	1
2	31112	Male	80.0	0	1	Yes	Private	Rural	105.92	32.5	never smoked	1
3	60182	Female	49.0	0	0	Yes	Private	Urban	171.23	34.4	smokes	1
4	1665	Female	79.0	1	0	Yes	Self-employed	Rural	174.12	24.0	never smoked	1

Figure 5: Head of Data before encoding



	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	1	67.0	0	1	1	2	1	228.69	36.60	1	1
1	0	61.0	0	0	1	3	0	202.21	30.57	2	1
2	1	80.0	0	1	1	2	0	105.92	32.50	2	1
3	0	49.0	0	0	1	2	1	171.23	34.40	3	1
4	0	79.0	1	0	1	3	0	174.12	24.00	2	1

Figure 6: Head of Data after Encoding

1.3 Dataset Description

The following are the attributes of the dataset and the data they contain and represent. You may also check Figure 2 for reference.

- **gender:** Categorical (Male/Female)
- **age:** Numerical (Age of the patient)
- **hypertension:** Binary (0: No, 1: Yes)
- **heart_disease:** Binary (0: No, 1: Yes)
- **ever_married:** Categorical (Yes/No)
- **work_type:** Categorical (Private, Self-employed, Govt_job, children, Never_worked)
- **Residence_type:** Categorical (Urban/Rural)
- **avg_glucose_level:** Numerical (Average glucose level)
- **bmi:** Numerical (Body Mass Index)
- **smoking_status:** Categorical (formerly smoked, never smoked, smokes, Unknown)
- **stroke:** Binary (0: No, 1: Yes)

1.4 Data Preparation

In this subsection, we describe the steps taken to prepare the dataset for model fitting.

1.4.1 Data Cleaning and Imputation

One of the initial challenges was dealing with missing values, particularly in the Body Mass Index (BMI) feature. To address this, we employed a strategy that involved imputing missing BMI values based on the average BMI within groups defined by gender and age. This approach leverages the assumption that BMI varies significantly across different demographic groups. By calculating the mean BMI within these groups, we obtained a more accurate estimate for filling in missing BMI values, thereby minimizing potential biases introduced by simple imputation methods.

1.4.2 SMOTE Implementation

Given the class imbalance inherent in the dataset, with strokes being a relatively rare occurrence compared to non-strokes, we implemented Synthetic Minority Over-sampling Technique (SMOTE). SMOTE works by generating synthetic samples for the minority class, thereby balancing the class distribution. This technique helps prevent the model from being biased towards the majority class and improves its ability to accurately predict strokes. By synthesizing new instances of the minority class based on feature similarities with existing instances, SMOTE effectively augments the dataset and enhances the model's performance.

1.4.3 Outlier Removal

Outliers can have a detrimental effect on model performance, particularly in algorithms sensitive to data distribution. We identified two features, average glucose level and BMI, where outliers were likely to have a significant impact. To address this, we applied outlier detection techniques and subsequently removed the outliers from the dataset. By focusing on these specific features, we aimed to improve the robustness of the model by ensuring that it is not unduly influenced by extreme values that may skew the underlying patterns.

2 Approaches Tried

2.1 Decision Tree Approach

- **Methodology:** Implemented a decision tree classifier from scratch using a top-down, recursive approach. The decision tree aims to partition the feature space into regions where instances share similar characteristics, ultimately assigning a class label to each region.
- **Implementation Details:**
  - **Initialization:** The DecisionTree class is initialized with parameters, such as the maximum depth of the tree (`max_depth`).
  - **Tree Construction:** The `fit()` method is responsible for building the decision tree. It calls the `_build_tree()` function, which recursively constructs the tree. At each node, the algorithm selects the feature that maximizes information gain to split the data.
  - **Information Gain:** Information gain is used as the criterion for selecting the best split. It measures the reduction in entropy (or increase in purity) achieved by splitting the data on a particular feature.
  - **Leaf Node Determination:** Leaf nodes are determined based on two conditions: either reaching the maximum depth specified by `max_depth` or achieving purity (i.e., all instances in the node belong to the same class).
  - **Prediction:** The `predict()` method traverses the decision tree to assign class labels to test instances. It starts at the root node and follows the appropriate branches based on the values of the features until reaching a leaf node.

2.2 K-Nearest Neighbors (KNN) Approach:

- **Methodology:** Implemented the K-Nearest Neighbors algorithm to predict stroke risk based on the similarity of feature vectors. KNN is a non-parametric, instance-based learning algorithm that classifies instances based on their proximity to other instances in the feature space.
- **Implementation Details:**
  - **Initialization:** The KNN class is initialized with a parameter k, representing the number of neighbors to consider.
  - **Training:** The `fit()` method stores the training data (`X_train` and `y_train`) for later use in prediction.
  - **Distance Calculation:** Euclidean distance is computed between each test sample and all training samples to measure their similarity.
  - **Nearest Neighbors:** The `nearest_neighbors()` method identifies the k nearest neighbors of a test sample based on calculated distances.
  - **Voting:** Predictions are made by a majority vote among the labels of the nearest neighbors.
  - **Prediction:** The `predict()` method predicts the class labels for the test set by iterating over each test sample, finding its nearest neighbors, and determining the most common class label among them.

2.3 Naive Bayes Approach:

- **Methodology:** Utilized the Gaussian Naive Bayes algorithm for stroke risk prediction. Naive Bayes is a probabilistic classifier based on Bayes' theorem with the "naive" assumption of feature independence, which simplifies computation by assuming that the presence of a particular feature is unrelated to the presence of any other feature.
- **Implementation:**
  - **Initialization:** The `GaussianNB` class is imported from scikit-learn, which provides an implementation of Gaussian Naive Bayes suitable for continuous features.
  - **Training:** The model is trained using the `fit()` method on the training data (`x_train` and `y_train`).
  - **Prediction:** Predictions are made on the test set (`x_test`) using the `predict()` method.

2.4 Random Forest Approach:

- **Methodology:** Utilized the Random Forest algorithm for stroke risk prediction. Random Forest is an ensemble learning method that constructs multiple decision trees during training and outputs the mode of the classes (classification) or mean prediction (regression) of the individual trees.
- **Implementation:**
  * **Initialization:** A RandomForest class is defined with parameters such as the number of estimators (decision trees), maximum depth of each tree, and whether to use bootstrapping for sampling.
  * **Training:** The `fit()` method is called with the training data (`X_train` and `y_train`). This method fits each decision tree in the forest to a bootstrap sample of the training data.
  * **Prediction:** Predictions are made on the test set (`X_test`) using the `predict()` method, which aggregates predictions from all trees in the forest.

2.5 Artificial Neural Network Approach

- **Methodology:** Utilized a Artificial Neural Network for Stroke Risk Prediction
- **Approach:**
  * Employed a Artificial Neural Network architecture for stroke risk prediction, leveraging PyTorch for model implementation.
  * Utilized a wrapper class compatible with scikit-learn's estimator interface for seamless integration with GridSearchCV for hyperparameter tuning.
- **Implementation:**
  * **Data Preprocessing:** Converted input features and labels into PyTorch tensors.
  * **Neural Network Architecture:** Defined a simple feedforward neural network with configurable hidden layer dimensions.
  * **Model Training:** Utilized the BCEWithLogitsLoss loss function and the Adam optimizer for model training.
  * **Hyperparameter Tuning:** Conducted hyperparameter tuning using GridSearchCV to find the optimal hidden layer dimension.

3 Experiments and Results

The dataset used in this project consists of 5110 instances and 12 attributes, including both numerical and categorical features. Each instance represents information about a patient's demographic characteristics, lifestyle factors, and medical history, with the target variable indicating whether the patient has experienced a stroke.

3.1 Experimental Setting

- **Approaches Tested:**
  - Decision Tree
  - K-Nearest Neighbors (KNN)
  - Naive Bayes (GaussianNB)
  - Artificial Neural Network
- **Evaluation Metrics:**
  - **Accuracy:** Proportion of correctly classified instances.
  - **Precision:** Proportion of true stroke cases among the predicted stroke cases.
  - **Recall:** Proportion of true stroke cases that were correctly identified.
  - **F1-score:** Harmonic mean of precision and recall, providing a balance between the two.
- **Training and Testing:**
  - The dataset was split into training and testing sets (using a 70-30) to train the models and evaluate their performance.
  - Models were trained on the training set and evaluated on the unseen test set to assess their generalization ability.

3.2 Comparison of Results

- **Decision Tree:**
  - Achieved certain accuracy, precision, recall, and F1-score on the test set.
  - Limited by potential overfitting, especially with a fixed depth.
  - **Accuracy:** 0.7060755336617406
  - **Precision:** 0.683105981112277
  - **Recall:** 0.7347629796839729
  - **F1-score:** 0.7079934747145188
- **K-Nearest Neighbors (KNN):**
  - Showed effectiveness in predicting stroke risk, but computational expense may be an issue, particularly with large datasets.
  - Choice of hyperparameter 'k' can significantly impact model performance.
  - **Accuracy:** 0.873015873015873
  - **Precision:** 0.8090737240075614
  - **Recall:** 0.9661399548532731
  - **F1-score:** 0.8806584362139918
- **Naive Bayes (GaussianNB):**
  - Demonstrated effectiveness in predicting stroke risk, leveraging probabilistic reasoning.
  - Assumption of feature independence may not hold true for all datasets, affecting performance.
  - **Accuracy:** 0.7679255610290093
  - **Precision:** 0.727810650887574
  - **Recall:** 0.8329571106094809
  - **F1-score:** 0.7768421052631579
- **Random Forest:**

    - Random Forest is an ensemble learning algorithm that builds multiple decision trees and combines their predictions to improve accuracy and reduce overfitting.
    - It is particularly effective for classification and regression tasks, offering robustness and scalability while providing insights into feature importance.
    - **Accuracy:** 0.8637110016420362
    - **Precision:** 0.8377518557794273
    - **Recall:**0.891647855530474
    - **F1-score:** 0.8638600328048114
- **Artificial Neural Network:**
    - Performance depends on the architecture, number of layers, neurons, and activation functions chosen.
    - Can potentially capture complex relationships in the data but requires more computational resources and tuning.
    - **Accuracy:** 0.7766830870279147
    - **Precision:** 0.7385229540918163
    - **Recall:** 0.835214446952596
    - **F1-score:** 0.7838983050847458

3.2.1 Findings and Limitations

- **Decision Tree:**
    - The decision tree achieved moderate performance metrics on the test set. However, it may be limited by potential overfitting, especially when the tree depth is fixed.
- **K-Nearest Neighbors (KNN):**
    - KNN showed promising results in predicting stroke risk, but its computational expense may be prohibitive, especially with large datasets. Additionally, the choice of the hyperparameter 'k' can significantly affect model performance.
- **Naive Bayes (GaussianNB):**
    - Gaussian Naive Bayes demonstrated effectiveness in predicting stroke risk, leveraging probabilistic reasoning. However, the assumption of feature independence may not hold true for all datasets, potentially impacting performance.
- **Random Forest:**
    - Random Forest exhibited robust performance with high accuracy and precision. It is particularly effective for classification tasks, providing insights into feature importance. However, it may require more computational resources compared to simpler algorithms.
- **Artificial Neural Network:**
    - Artificial Neural Network showed promising results in capturing complex relationships in the data. However, its performance heavily depends on the chosen architecture and requires careful tuning. Additionally, it may require more computational resources compared to traditional machine learning algorithms.

4 Summary

We've developed a stroke prediction model using machine learning, employing several algorithms like Decision Tree, K-Nearest Neighbors (KNN), Artificial Neural Network (ANN), Naive Bayes, and Random Forest. The aim is to forecast the likelihood of someone having a stroke based on certain risk factors.

Five machine learning approaches were explored:

- **Decision Tree:** Decision trees resemble trees where each internal node represents a test on an attribute, branches show the outcome of the test, and leaves indicate a class label. They're easy to grasp but might overfit and lack generalization. depth.But in our case this didn't happened as we handled outliers and hyper tuned the parameter.

- **K-Nearest Neighbors (KNN):** KNN is a simple algorithm for classification and regression, predicting a data point's class by examining the 'k' closest data points and taking a majority vote. It's picked for our demo due to its simplicity,effectiveness, and interpretability. KNN doesn't need training, making it fast and suitable for small datasets, and it handles noisy data well.This gave the best accuracy and recall scores.
- **Naive Bayes (GaussianNB):** Employed the Gaussian Naive Bayes algorithm for stroke risk prediction, leveraging probabilistic reasoning. Although effective, the assumption of feature independence may not hold true for all datasets.
- **Random Forest :**can perform classification and regression tasks by constructing multiple decision trees and aggregating their predictions, resulting in improved accuracy and robustness.
- **Artificial Neural Network :**Inspired by the brain's structure, ANN comprises interconnected nodes or neurons in layers. It's great at capturing complex patterns but needs lots of data and is computationally expensive. ANN may overfit and can be tough to interpret.

The experimental setting included splitting the dataset into training and testing sets, evaluating models using metrics such as accuracy, precision, recall, and F1-score. Results were compared across the different approaches, highlighting their strengths and limitations.We observed that the best approach was the KNN approach. Although computationally expensive, it gave the best results in terms of precision, accuracy and recall. Future steps may involve further experimentation with hyperparameter tuning, feature selection, and model optimization, as well as exploration of ensemble methods or advanced neural network architectures for improved performance.

A Contribution of Each Team Member

1. ATISH: Implemented the demo approach using Decision tree/KNN approach and contributed to make the report .
2. PRANAV: Implemented the Feedforward Neural Network approach and Gaussian Naive Bayes algorithm. Conducted hyper-parameter tuning on ANN. Contributed in fixing errors in the KNN approach and the SMOTE implementation. Made the Spotlight video and edited the report.
3. RHEA: Performed Data Visualisations and contributed in data preparation by implementing SMOTE and handling outliers. Contributed to the decision tree approach and hypertuned it. Implemented KNN and Random Forest. Reviewed and edited minor details in the report.
4. HARI: Contributed to make the presentation,report and helped in web page.
5. RAVINDAR: Implemented code for web page and contributed to helped in demo code and report.
6. NIKSHIPTHA: Contributed to make the presentation.
7. CHARAN: Helped in presentation.