HEART STROKE PREDICTOR

Anmol Mani Dubey

(SESSION 2022-26)

(Minor Project Report)

INTRODUCTION

Heart stroke, also known as cerebrovascular accident (CVA), is a medical emergency that occurs when blood flow to a part of the brain is interrupted or severely reduced, depriving brain tissue of oxygen and nutrients. This can lead to brain damage, disability, or even death. Stroke is the second leading cause of death worldwide, and it is a major cause of disability.

Early diagnosis and treatment of stroke are crucial for improving patient outcomes. However, accurately predicting the risk of stroke can be challenging. Traditional risk assessment methods often rely on a limited set of factors, such as age, family history, and medical history. This can lead to misidentification of individuals at high risk of stroke.

Machine learning, a field of artificial intelligence, offers a promising approach to improving stroke prediction. Machine learning algorithms can analyze large datasets of patient data to identify patterns and relationships that may not be apparent to traditional methods. This can lead to the development of more accurate and personalized stroke prediction models.

The development of a heart stroke predictor using machine learning techniques has the potential to significantly improve the prevention and management of stroke. By identifying individuals at high risk of stroke, we can provide them with timely interventions, such as lifestyle changes or medication, to reduce their risk of stroke. Additionally, machine learning models can be used to guide treatment decisions for stroke patients, helping to improve patient outcomes.

PROPOSED SYSTEM

The suggested work examines the four classification methods listed above and performs performance analysis to predict heart attacks. This study's goal is to accurately predict if a patient will experience a heart attack. The healthcare provider inputs the data from the patient's health report. The data is incorporated into a model that forecasts the likelihood of suffering a heart attack.

METHODOLOGY

Data Description:

For this investigation, we used the cardiac stroke dataset which is available on the Kaggle website. There are a total of 12 attributes in this collection. The following provides a detailed summary of the characteristics used in the suggested work:

- id: This property refers to a person's ID. Numbers make up the data.
- Age: This quality refers to a person's age. Numbers make up the data.
- Gender: This characteristic refers to a person's gender. Data with categories.
- Hypertension: This characteristic indicates whether this person has hypertension. Numbers make up the data.
- Work type: The person's work environment is represented by this property. Data with categories.
- Residence type: This characteristic shows the individual in the situation. Data with categories.
- Heart disease: This characteristic indicates whether the person has heart disease. Numbers make up the data.
- Average glucose level: This characteristic indicates a person's glucose state at that time. Numbers make up the data.
- BMI: This characteristic refers to a person's body mass index. Numbers make up the data.
- Ever married: This characteristic denotes a person's marital status. Data with categories.
- Smoking Status: This characteristic refers to a person's smoking history. Data with categories.
- Stroke: This characteristic indicates whether a person has ever had a stroke. Numbers make up the data. The choice class and the remaining attributes in this all-attribute stroke are the answer class.

RANDOM FOREST CLASSIFIERS

Random forest is a versatile machine learning algorithm used for classification and regression tasks. It belongs to the ensemble learning method, combining multiple decision trees to create a more robust and accurate model.

Here's how a Random Forest Classifier works:

- Decision Trees: Random forests consist of multiple decision trees. A decision tree is a flowchart-like structure where each internal node represents a test on an attribute, each branch represents the outcome of the test, and each leaf node represents a class label or a numerical value.
- Ensemble Learning: Random forests create an ensemble of decision trees. Each tree is constructed using a random subset of the training data and a random subset of features (random selection of features at each split). This randomness helps in reducing overfitting and increasing diversity among individual trees.
- Voting or Averaging: During classification, the random forest aggregates predictions from individual trees. For example, in classification tasks, each tree "votes" for a class, and the most popular class label among all the trees becomes the final prediction. In regression tasks, the average of all predictions is taken as the final output.

Advantages of Random Forest Classifiers:

- High Accuracy: Random forests typically provide high accuracy in both classification and regression tasks.
- Robustness: They are robust against overfitting because of the randomness introduced in building individual trees.
- Feature Importance: They can measure the relative importance of each feature in the classification.

However, there are considerations:

- Complexity: Random forests can be computationally expensive and might not be as easy to interpret compared to individual decision trees.
- Memory Usage: As an ensemble of multiple trees, they might consume more memory.
- Hyperparameter Tuning: They require tuning of hyperparameters such as the number of trees, depth of trees, etc.

K-NEAREST NEIGHBORS CLASSIFIERS

The k-Nearest Neighbors (k-NN) algorithm is a simple and versatile machine learning algorithm used for both classification and regression tasks.

Here's how a k-Nearest Neighbors classifier works for classification:

- Basic Principle: The algorithm works on the assumption that similar data points belong to the same class. It calculates the similarity (often using distance metrics like Euclidean, Manhattan, etc.) between the input data point and other points in the dataset.
- Nearest Neighbors: The 'k' in k-NN refers to the number of nearest neighbors to consider. For a given data point, the algorithm identifies the 'k' nearest neighbors in the training dataset based on the chosen distance metric.
- Voting for Classification: For classification tasks, once the 'k' nearest neighbors are identified, the algorithm takes a majority vote among these neighbors to assign a class label to the input data point. The class label with the most occurrences among the 'k' neighbors becomes the predicted class for the new data point.

Key features and considerations:

- No Model Training Phase: k-NN doesn't explicitly build a model during the training phase. It memorizes the entire training dataset and uses it for predictions at test time.
- Parameter 'k' Selection: The choice of 'k' significantly influences the performance of the algorithm. A smaller 'k' value can lead to overfitting, while a larger 'k' might cause the model to oversimplify.
- Sensitive to Distance Metrics: The choice of distance metric (Euclidean, Manhattan, etc.) affects the algorithm's performance. It's crucial to choose a suitable distance measure based on the nature of the data.
- Scalability: As the size of the dataset increases, the computational cost of k-NN also increases since it compares the new data point to all training samples.

Advantages of k-NN:

- Simple Implementation: Easy to understand and implement.
- Adaptability to New Data: It can adapt well to new data.
- No Assumptions About Data: k-NN doesn’t assume anything about the underlying data distribution.

Limitations:

- Computational Cost: Prediction time can be slow, especially with large datasets.

- Need for Feature Scaling: It's important to scale features as k-NN is sensitive to the scale of features.
- Memory Usage: It stores the entire training dataset, which could be memory-intensive for large datasets.
- k-NN is commonly used in various fields, especially in recommendation systems, pattern recognition, and some classification tasks where interpretability and simplicity are more critical than computational efficiency.