Brain Stroke Detection using Machine Learning Techniques

A PROJECT REPORT

Submitted by

Mohit Karelia
(202211080)
&
Jay Joshi
(202211079)

Under the Guidance of

Professor Tapas Kumar Maiti

Dhirubhai Ambani Institute of Information and Communication Technology

December 2022

DECLARATION

We hereby declare that this project, 'Brain Stroke Detection using Machine Learning Techniques' is submitted by us to Professor Tapas Kumar Maiti and this is not submitted to any other University or college for the award of a Degree or certificate.

Date- 20/12/2022

ABSTRACT

Brain Stroke is the 2nd leading cause of death globally, responsible for approximately 11% of total deaths. The adverse consequences of stroke have led to global interest and work for improving the management and diagnosis of stroke. A stroke, sometimes called a brain attack, happens in one of two ways:

• Ischemic stroke—when the blood supply to the brain is blocked

• Haemorrhagic stroke—when a blood vessel in the brain bursts

In either case, parts of the brain become damaged. A stroke can cause lasting brain damage, long-term disability, or even death. Our goal is to detect the chances of stroke by use of advanced machine learning algorithms and deploy it for use in real world. This can save lives of many people if the stroke is identified in the early stage.

Table of content

List of Abbreviations

EDA- Exploratory Data Analysis

SMOTE-Synthetic Minority Oversampling Technique

BMI- Body Mass Index

KNN-K Nearest Neighbours

Chapter 1: Introduction

A brain stroke occurs when a blood vessel in the brain ruptures and bleeds or when there's a blockage in the blood supply to the brain. The break or blockage prevents blood and oxygen from reaching the brain's tissues. Detecting the chances of a stroke on time can help save one's life. Stroke symptoms can include

- Paralysis
- Numbness or weakness in the arm, face and leg, especially on one side of the body
- Trouble speaking or understanding others
- Slurred speech
- Confusion, disorientation, or lack of responsiveness
- Sudden behavioural changes, primarily increased agitation
- Vision problems include trouble seeing in one or both eyes with vision blackened or blurred or double vision
- Trouble walking
- Loss of balance or coordination
- Dizziness
- Severe, sudden headache with an unknown cause
- Seizures
- Nausea or vomiting

Suppose we have sufficient information and data about patients suffering from brain stroke. In that case, we can apply machine learning techniques and models that learn from given data to predict the chances of having a stroke very accurately.

The Brain stroke detection dataset available on Kaggle consists of ten features that help to determine the chances of a brain stroke. Using this dataset, we will try to create accurate machine-learning models that can predict the chance of having a brain stroke based on the given features.

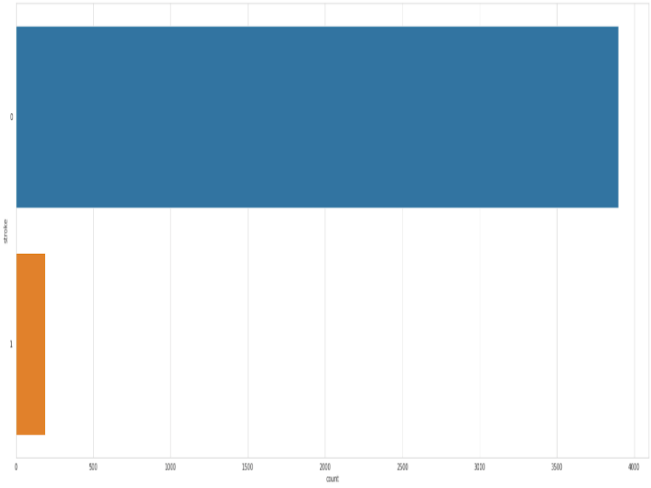
The dataset contains 5110 observations and twelve attributes, out of which Id is removed and stroke, which is the label of the class, these two attributes are not used in training the model.

The ten features that help determine the chances of stroke are

- Gender
- Age
- Hypertension
- Heart Disease
- Marriage status
- Work Type
- Residence Type,
- Average Glucose Level
- BMI
- Smoking Status.

We will use the first nine features of the before-mentioned and not use Smoking Status as there was no correlation between smoking status and chances of a brain stroke.

CHAPTER 2: CHALLENGES IN DATA



(Figure 2.1) Labels of our data

From the above figure (Figure 2.1), We can see that the occurrence of class 0 (No stroke) is much more than the occurrence of class 1(Stroke). This leads to an imbalance in the dataset. This means that when a machine learning algorithm is training on the data, it will be biased more towards class 0 rather than balancing between both classes. In other words, the model is trained to identify class 0, which is much more occurring in the dataset than class 1. This data imbalance causes the model to generalise poorly and give inaccurate results when the outcome should be class 1.

We need to deal with this imbalance to generalise our model better. For this purpose, we use the SMOTE. This method creates synthetic data of the class in the minority to balance the data.

SMOTE uses the K Nearest Neighbours algorithm to synthetically generate data points similar to the original data points of the minority class.

This helps create sufficient data to deal with class imbalance helping achieve an accurate machine-learning algorithm for stroke prediction.

CHAPTER 3: SUMMARY OF EDA

EDA helps understand what a model learns from data. Here is a quick summary of what observations were made in EDA.

- Age, Hypertension, Average Glucose Level, Hypertension and Heart Disease have moderate to high positive correlation coefficients.

- There is a 3.96% chance of having a stroke without hypertension. There is a 13.25% chance of having a stroke when there is hypertension. Thus, having hypertension increases the chance of having a stroke by 9.29%

- There is a 4.187% chance of having a stroke when there is no heart disease. There is 17.02% chance of having a stroke when there is heart disease. Thus having heart disease increases the chance of having a stroke by 12.832%.

- There is a 1.65% chance of having a stroke when the person is single. There is a 6.56% chance of having a stroke when a person is married. Thus marriage increases the chance of having a stroke by 4.91%.

- 5.09% of people having private work type have a stroke. 7.93% of people having self-employed work type have a stroke. 0.29 % of people having children have a stroke. No records of people who have never worked and had brain strokes.

- The proportion of stroke among the Gender was almost similar,(43% in male and 57% in female).

CHAPTER 4: DATA PRE-PROCESSING

Machine learning models take only numerical values as input. Our data has columns which have values that are not numerical and thus need to be converted into equivalent numerical values such that the data stays the same. For this, we use label encoding that maps the values in the columns to a number in alphabetic order.

We have chosen to omit to scale in the pre-processing step. The reason for this will is explained in the model selection chapter.

Next, we split the data into train and test sets in the ratio 80:20. The split is random, meaning that data points are randomly selected and shuffled for train and test sets.
This helps us select a model that best fits our data minimising overfitting.

CHAPTER 5: MODEL SELECTION

Every model fits the data differently. We need to check which model gives the best result for our given data.

We will use the following machine learning algorithms on our dataset:

1. Logistic Regression
2. K-Nearest Neighbours
3. Decision Tree Classifier
4. Random Forest Classifier
5. AdaBoost Classifier
6. XGBoost Classifier.

To evaluate the performance of each of these algorithms, we will use the following evaluation metrics:

1. Precision Score
2. Recall Score
3. Accuracy Score
4. Cross-Validation Score
5. Confusion Matrix.

We must note that we derived precision, recall and accuracy scores from the confusion matrix.

Whilst training the data, we have observed that model complexity increases to a certain extent. Thus, models like random forest and xgboost perform better than simpler models like logistic regression.
Here is the model performance for each model on the given evaluation metrics

1. Logistic regression
Precision Score: 0.7898
Recall Score: 0.811
Accuracy Train: 0.7977
Accuracy Test: 0.9363
Cross Val Score Train: 0.7977
Cross Val Score Test: 0.9363462458153993

2. K Neighbours Classifier

Precision Score: 0.8163878163878164
Recall Score: 0.9761477301872274
Accuracy Train: 0.8783021287509618
Accuracy Test: 0.9326805385556916
Cross Val Score Train: 0.8783020220757523
Cross Val Score Test: 0.9326936872309901

3. Decision Tree Classifier:

Precision Score: 0.8343263061411549
Recall Score: 0.9338291869710182
Accuracy Train: 0.874198512439087
Accuracy Test: 0.9130966952264382
Cross Val Score Train:0.8738138558892792
Cross Val Score Test: 0.9070002391200382

4. Random Forest Classifier:

Precision Score: 0.9174041297935103
Recall Score: 0.9571685047448064
Accuracy Train: 0.9354962810977173
Accuracy Test: 0.9388004895960832
Cross Val Score Train:0.9372932864980068
Cross Val Score Test:0.9388032042085128

5. AdaBoostClassifier:

Precision Score: 0.8712696004046535
Recall Score: 0.8835598871505514
Accuracy Score Train: 0.8765067966145166
Accuracy Score Test: 0.9192166462668299
Cross Val Score Train:0.8765089263396088
Cross Val Score Test: 0.9192192730750837

6. XGBoost:

Precision Score: 0.9577972207925888
Recall Score: 0.9546037445498846
Accuracy Train: 0.9562708386765837
Accuracy Test: 0.9143206854345165
Cross Val Score Train:0.9562736314480799
Cross Val Score Test: 0.91431731229077

CHAPTER 6: MODEL DEVELOPMENT

From the results of the previous section, two models:
Random Forest Classifier and XGBoost, give great results on both training and test sets. As XGBoost is more complex and harder to deploy, we chose Random Forest Classifier as our base classifier for the project. As we stated earlier that we did not scale the data the reason is that tree based classification models have negative sensitivity to scaling. That is the model performance worsens when data is scaled thus we chose to omit scaling.We had train the model on the entire dataset using these tuned parameters:

" RandomForestClassifier(n_estimators = 200,n_jobs = -1,min_samples_split = 2,min_samples_leaf = 2)"

Once the model is trained, we save the trained model using pythons inbuilt Pickle library.

- What is pickle?

  Python pickle module is used for serializing and de-serializing a Python object structure. Any object in Python can be pickled so that it can be saved on disk. What pickle does is that it "serializes" the object first before writing it to file.

- Why do we need to pickle model?

  Pickle is a useful Python tool that allows you to save your ML models, to minimize lengthy re-training and allow you to share, commit, and re-load pre-trained ML models. Most data scientists working in ML will use Pickle or Joblib to save their ML model for future use.

CHAPTER 7: Deployment

Once the model is trained and saved, the next step is to deploy our model. For deployment, we have decided to deploy our model on the web using python's Flask Framework.

We created an HTML form to take input values from the user process and fit the data into our model to get predictions.

Algorithm for Deployment:

1. Train Random Forest Classifier on dataset.
2. Save the trained model.
3. Create a web application using Flask.
4. Create a form for user to enter data.
5. Collect suitable data and fit it into the model for prediction.
6. Predict the outcome and show the results.

CHAPTER 8: CONCLUSION AND DISCUSSIONS

GITHUB LINK:

https://github.com/Mohit501/Programing_lab_project

CONFRONTATION

FUTURE SCOPE

Due to unavoidable circumstances we could not deploy our website on the internet due to server errors. Thus for future score we would like to deploy our project live on the internet.

REFERENCES:

- https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset
- https://towardsdatascience.com/what-is-feature-engineering-importance-tools-and-techniques-for-machine-learning-2080b0269f10
- https://machinelearningmastery.com/feature-selection-with-real-and-categorical-data
- https://machinelearningmastery.com/a-gentle-introduction-to-model-selection-for-machine-learning/
- https://www.analyticsvidhya.com/blog/2020/04/how-to-deploy-machine-learning-model-flask/