International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Brain Stroke Prediction Using Machine Learning

Puranjay Savar Mattas[a]

[a]*ORCID ID: https://orcid.org/0000-0002-5314-5647, #129/2 Sharda University, New Delhi 110030, India*

DOI: *https://doi.org/10.55248/gengpi.2022.31211*

A B S T R A C T

Brain Stroke is considered as the second most common cause of death. We use a set of electronic health records (EHRs) of the patients (43,400 patients) to train our stacked machine learning model to train, test and predict with an accuracy whether the input data points towards a stroke or not. This will help the patient as well as the medical professionals to examine and streamline their work making the process efficient. This study helped us get a result score which was promising enough to continue the research on an extensive level with a professional approach taken here in this paper.

Keywords: Stroke, machine Learning, Prediction, Classifiers

1. Introduction

A stroke is nothing but an interruption of the blood supply to any part of the brain. If blood stops for longer than a few seconds and the brain cannot get blood and oxygen, brain cells can die and their abilities controlled by that area of the brain are lost. A stroke can cause lasting brain damage, long-term disability, or even death (*About Stroke | Cdc.gov*, 2022).

Stroke is regarded as the second biggest killer (Virani et al., 2020). The World Health Organization (WHO) estimates that 17.7 million people died from cardiovascular illnesses in 2017, with 6.7 million of those deaths attributable to stroke (Shikany et al., 2020). Stroke mortality and prevalence are still rising (Mourguet et al., 2019). Every year, people are informed about stroke and its prevention on World Stroke Day (Lindsay et al., 2019). To avoid stroke's negative effects, early detection and prevention have become crucial.

Due to the increase in technical breakthroughs over time, the field of medical sciences has seen enormous advancements. The availability of affordable wearable devices has been a major benefit of the Internet of Things (IoT) (Gold, 2018) (Ma et al., 2017) (Mettler, 2016), making it simpler to collect data relevant to healthcare. Using various data-mining techniques, a large amount of raw medical data is gathered from these devices in order to identify meaningful patterns. Additionally, the collected insights are applied to decision-making in the healthcare industry and have demonstrated to be a cost-cutting component (Yadav et al., 2017). In recent years, the field of medical sciences has formed a promising sector for machine learning (ML). EHRs can be used with ML models to accurately estimate each patient's risk of suffering a stroke (Chen, 2017).

Machine learning is a subsection of artificial intelligence in which computer systems are programmed to learn on their own. Machine learning techniques can be used to extract the data's hidden complicated patterns. The healthcare sector can produce, store, and analyze vast amounts of heterogeneous data from CT scans, MRIs, fMRI, PET, SPECT, DTI, and DOT, among other technologies. Other sources of data that hospitals are in charge of include hospital administration records, patient medical records, exam results, and data produced by the devices. Manually processing this kind of multidimensional data is a difficult undertaking that runs the risk of lowering forecast accuracy, which will directly impact the patient's quality of life. Therefore, in order to derive accurate and useful information, the world needed a proper smart data management and analysis method. By offering the pertinent answers in less time and with higher accuracy, machine learning may play a crucial role in the modern healthcare business. It also gives systematic and algorithmic approach tools for data administration, analysis and interpretation (K & Sarathambekai, 2021) (Sasubilli & Kumar, 2020).

The features/attributes chosen from the data determine how predictively accurate the models are. Numerous elements of the patients' conditions are contained, gathered, and archived in their EHRs. All of the EHR's gathered features, though, might help with stroke identification. Only significant characteristics related to the result should be chosen for the prediction model in order to increase the prediction performance of the machine learning

* *Corresponding author.*
E-mail address: puranjaysavarmattas@gmail.com

models and decrease the machine training time (Amin et al., 2019). In order to extract the significant and pertinent elements for forecasting the incidence of strokes, a number of data mining techniques have also been proposed (Esfahani & Ghazanfari, 2017; Kolukisa, 2018). (Ratajczak, 2019). Lack of adequate systematic guidance for feature selection when creating prediction models, which is essential for model performance, is the main obstacle to current research on stroke risk assessment (Gupta & Sedamkar, 2019). We will only be able to effectively identify stroke once we have extracted the key risk variables that are most strongly linked to its onset.

The goal of this study is to develop a prediction model that is accurate enough to be able to recognize and warn of a stroke based on information provided by the patient or by medical personnel. Furthermore, if the model predicts a "true" outcome that improves efficiency, this study will assist the healthcare sector in doing in-depth analysis.

2. Methodology

The internal structure of the model that we are building for this research will be explained in this section of the paper. Here, all approaches and their selection criteria will be covered in full disclosure.

2.1. Dataset discovery

Our study, which uses the dataset of McKinsey & Company's healthcare hackathon, employed the Electronic Health Record (EHR) under its management as the dataset (McKinsey Analytics, 2018). The dataset is available for free download from a dataset repository. The collection includes 43,400 patient records with 12 common attributes. 10 of the 11 attributes are input values that serve as the model's operating features. The last record determines whether the parameters taken together will cause a stroke or not, serving as the model's output value. The input attributes are as follows (in order of input):

1. Gender: Male/Female
2. Age: 55 (Example Value Here)
3. Hypertension: 1(True)/0(False)
4. Heart Disease: 1(True)/0(False)
5. Ever Married: 1(True)/0(False)
6. Average Glucose Level: 95.12 (Example Value Here)
7. Body Mass Index (BMI): 18 (Example Value Here)
8. Work Type: Govt/Never-Worked/Private/Self-Employed/Children
9. Residence Type: (0) Rural/ (1) Urban
10. Smoking Status: Formerly Smoked/Never Smoked/Smokes

The sum of all the aforementioned parameters should enable the model to correctly determine whether or not a stroke is likely to occur in the user.We must eliminate all null values and values that could affect the accuracy of the model's prediction of the outcome from the 43,400 data records that have been given to us. To achieve this, we must identify the crucially flawed data and preprocess all the data before submitting it to the model.The type of data fed into the model and the training techniques employed both affect how accurate it is. As this model can be used by healthcare experts, this research will employ a stack model methodology to ensure that accuracy is not dependent on one method.

2.2. Stacking Model

Stacking, which consists of two-layer estimators, is a method of assembling classification or regression models. All of the baseline models that are used to forecast the results on the test datasets are contained in the first layer. The second layer is made up of a Meta-Classifier or Regressor that creates new predictions by using all of the predictions from baseline models as input (Fig. 1) (Stacking in Machine Learning, 2021).

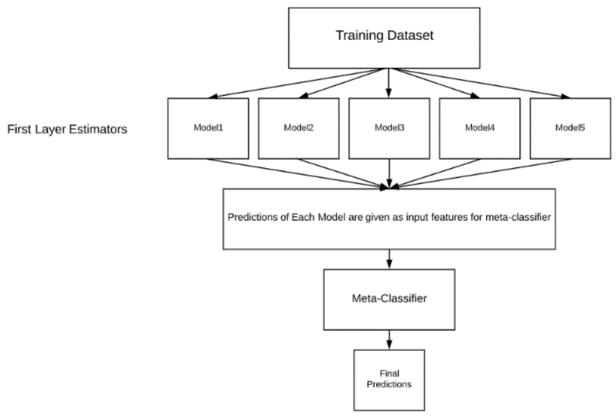


Fig. 1 -stacking model illustrative working

They can increase the accuracy that is currently demonstrated by individual models. As different algorithms capture distinct trends in training data, we can obtain the majority of stacked models by selecting a variety of algorithms for the first layer of the architecture. By combining both models, we can obtain more precise and superior outcomes.

The two most well-known ensemble modelling techniques are bagging and boosting. Bagging enables the averaging of several comparable models with significant volatility in order to reduce variance. Boosting creates numerous incremental models in order to reduce bias while minimizing variation.

A different paradigm is called stacking (also known as stacked generalization). Exploring the space of several models for the same problem is the goal of stacking. The concept is that you can approach a learning problem with several sorts of models that can learn a portion of the problem but not the entire problem space. As a result, you may create a variety of learners and use them to create intermediate predictions, one for each learnt model. Then you include a new model that picks up on the same target as the intermediate forecasts.

The name of the last model comes from the claim that it is layered on top of the others. As a result, we may enhance our total performance, and we frequently produce a model that is superior to each individual intermediate model (*Stacking in Machine Learning*, 2019).

Stacking works in the following steps:

1. The programmer splits the training data into K-folds just like K-fold cross-validation.
2. A base model is fitted on the K-1 parts and predictions are made for the Kth part.
3. The programmer does this for each part of the training data.
4. The base model is then fitted on the whole train data set to calculate its performance on the test set.
5. The programmer repeats the last 3 steps for other base models.
6. Predictions from the train set are used as features for the second level model.
7. Second level model is used to make a prediction on the test set.

This paper chose five base models and one as the meta classifier. The models are chosen because of their relation to the kind of dataset we are working on and the desired result we require from them. The models which we incorporate in this singular model are divided in two parts, i.e., The Estimator List and The Meta-Classifier.

2.2.1. The estimator list:

This comprises a list of five models that were used to stack them one on top of the other and provide The Meta-Classifier with five results in order to increase the model's prediction accuracy. The following five models were used to construct this estimator list:

2.2.1.1. KNeighborsClassifier

A non-generalizing or instance-based learning method is KNeighborsClassifier. Instead of attempting to build a broad internal model, it only saves examples of things like the training data. A query point is assigned to the data class that contains the most representatives among its nearest neighbours after classification is calculated using a simple majority vote of each point's nearest neighbours. The most used method is K-neighbors classification in KNeighborsClassifier. The best value to choose depends heavily on the data; typically, a bigger value reduces the impact of noise. The fundamental nearest neighbors classification employs uniform weights, meaning that a query point's value is determined by the simple majority vote of its closest neighbors. In some cases, it is preferable to weigh the neighbors so that those who are closest to you contribute more to the match. The weights keyword can be used to achieve this. Each neighbor is given a uniform weight when using the default setting of weights = "uniform". When you specify weights = "distance," you'll get results that are inversely proportional to your distance to the query point. The weights can also be computed using a user-defined function of the distance (Fig. 2) (*1.6. Nearest Neighbors — Scikit-Learn 1.1.2 Documentation*, n.d.).

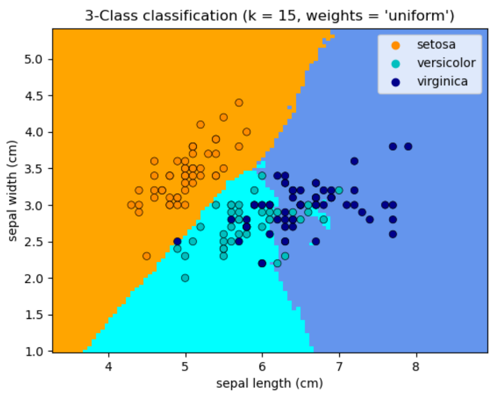


Fig. 2 -KNeighborsClassifier working example

2.2.1.2. Support Vector Classifier (SVC)

Support vector machines (SVMs) are a group of supervised learning techniques for classifying data, performing regression analysis, and identifying outliers. Support vector machines' benefits include:

- Efficient in high-dimensional environments.
- In conditions that involve more dimensions than samples, it is still functional.
- It is also memory efficient because it only uses a portion of the training points (known as support vectors) in the decision function.
- Different Kernel functions can be given for the decision function, making it versatile. There are common kernels available, but you can also define your own kernels

Scikit-support Learn's vector machines accept any scipy.sparse sample vector as well as dense (numpy.ndarray, which may be converted to that using numpy.asarray) and sparse (numpy.ndarray) sample vectors as input. An SVM must, however, have been fitted to such data in order to be used to make accurate predictions for sparse data. Use scipy.sparse.csr matrix (sparse) or C-ordered numpy.ndarray (dense) with dtype=float64 for the best performance. On a dataset, the classes SVC, NuSVC, and LinearSVC may do multi-class and binary classification (Fig. 3).

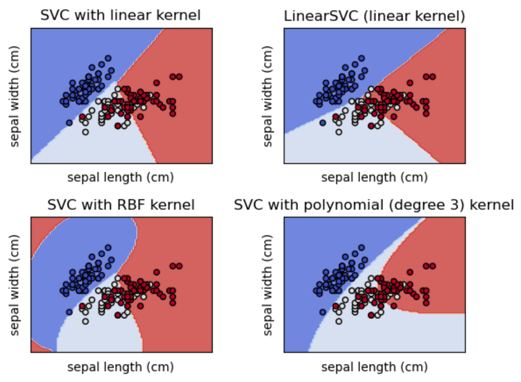


Fig. 3 -SVC working withdifferent kernels

Despite accepting somewhat different sets of parameters and having various mathematical formulations, SVC and NuSVC are similar algorithms. For the situation of a linear kernel, LinearSVC is an additional (faster) application of Support Vector Classification. Keep in mind that because the kernel is presumed to be linear, LinearSVC doesn't always accept the kernel as a parameter. Additionally, it lacks some SVC and NuSVC properties, such as "support_". SVC, NuSVC, and LinearSVC, like other classifiers, require two arrays as inputs: an array X of shape (n samples, n features) containing the training samples, and an array y with class labels (strings or integers), of shape (n samples). The model then has the ability to forecast new values after being fitted. The support vectors, a component of the training data, are what SVMs use to make decisions. The attributes "support vectors_," "support," and "n support" list some of these support vectors' features. The "one-versus-one" method is used by SVC and NuSVC for multi-class categorization. Each classifier analyses data from two classes, for a total of n classes * (n classes - 1) / 2 classifiers. The decision function shape option enables monotonic transformation of "one-versus-one" classifier results into a "one-vs-rest" decision function of shape (n samples, n classes) to give a consistent interface with several other classifiers. The "one-vs-the-rest" multi-class technique is implemented by LinearSVC, which trains n classes models (*1.4. Support Vector Machines — Scikit-Learn 1.1.2 Documentation*, n.d.).

2.2.1.3. Decision Tree Classifier

They are a non-parametric supervised learning approach that is used for regression and classification. The objective is to learn straightforward decision rules derived from the data features in order to build a model that predicts the value of a target variable. A piecewise constant approximation of a tree can be thought of.Decision-tree advantages include:

- One can visualise trees.
- Little data preparation is necessary. Data normalisation, the creation of dummy variables, and the elimination of blank values are frequently necessary for other procedures.
- As more data points are utilised to train the tree, the cost of using it to forecast data increases exponentially.
- Capable of working with both categorical and numerical data. Categorical variables are not currently supported by the Scikit-Learn implementation. Other methods are typically focused on analysing datasets with just one kind of variable.
- Able to manage issues with several outputs.
- Using the white box model. Boolean logic makes it simple to explain a condition if it can be observed in a model for a given situation. Results may be more challenging to interpret in a black box model, such as an artificial neural network.
- It is possible to use statistical tests to verify a model. This enables the model's dependability to be taken into account.
- Performs effectively even though the underlying model based on which the data were created slightly violates some of its basic assumptions.

For instance, in the example below (Fig. 4), decision trees learn from data to approximate a sine curve with a set of if-then-else decision rules. The deeper the tree, the more complex the decision rules and the fitter the model.

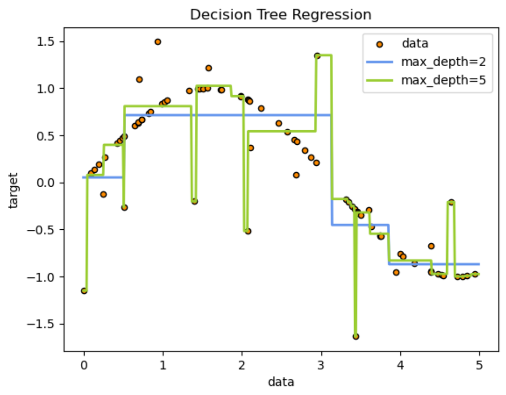


Fig. 4 -example illustration of decision tree

A class that can do multi-class classifications on a database is called a DecisionTreeClassifier.

The DecisionTreeClassifier, like other classifiers, accepts two arrays as input: an array X, sparse or dense, of shape (n samples, n features), storing the training samples, and an array Y, of integer values, form (n samples,), carrying the class labels for the training sets. The model can then be utilized to forecast the class of samples after being fitted. The classifier will forecast the class with the shortest index among those classes if several classes have the same and highest likelihood. Instead of producing a specific class, it is possible to forecast the likelihood of each class, which is the proportion of training samples for the class in a leaf.

Binary classification, where the labels are [-1, 1], and multiclass classification, where the labels are [0,..., K-1], are both possible using DecisionTreeClassifier (1.10. Decision Trees — Scikit-Learn 1.1.2 Documentation, n.d.).

2.2.1.4. Random Forest Classifier

A random forest is a meta estimator that employs averaging to increase predicted accuracy and reduce overfitting after fitting numerous decision tree classifiers to distinct dataset subsamples. If bootstrap=True (the default), the size of the sub-sample is determined by the max sample's argument; otherwise, each tree is constructed using the entire dataset. Each tree in a random forest ensemble is constructed using a sample from the training set that was drawn with replacement (also known as a bootstrap sample) (Sklearn.ensemble.RandomForestClassifier — Scikit-Learn 1.1.2 Documentation, n.d.).

Additionally, the optimal split is determined from either all input features or a randomly selected subset of size max features when dividing each node throughout the creation of a tree.

These two randomness sources serve to lessen the variance of the forest estimator. Individual decision trees do, in fact, frequently show substantial variance and a propensity for overfitting. Decision trees with partially dissociated prediction errors are produced by injecting randomization into forests. Some mistakes can be eliminated by averaging their predictions. By merging several trees, random forests reduce variation, sometimes at the expense of a modest increase in bias (1.11. Ensemble Methods — Scikit-Learn 1.1.2 Documentation, n.d.). In practice, the variance reduction is frequently large, producing an overall superior model.

Instead of allowing each classifier to vote for a single class, like in the original publication (1.11. Ensemble Methods — Scikit-Learn 1.1.2 Documentation, n.d.), the scikit-learn solution combines classifiers by averaging their probabilistic predictions.

2.2.1.5. Multi-Layer Perceptron Classifier

Adataset is used to train the multi-layer perceptron (MLP), a supervised learning technique, to learn the function $f(): RmRo$, where m is the number of input dimensions and o is the number of output dimensions. It is possible to learn a nonlinear function approximator for either classification or regression given a set of features $X = x1, x2, \dots, xm$ and a target y. There may be one or more non-linear layers, known as hidden layers, between the input layer and the output layer, which distinguishes it from logistic regression. A single hidden layer MLP with scalar output is shown in Figure 5.

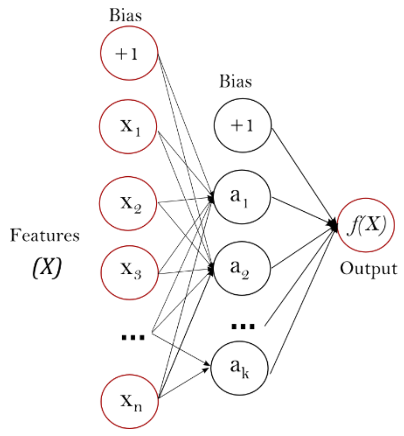


Fig. 5 -hidden layer MLP with scalar output

The input layer, which is the topmost layer, is made up of a group of neurons called$xi|x1, x2, \ldots, xm$ that reflect the input attributes. The values from the preceding layer are transformed by each neuron in the hidden layer using a weighted linear summation ($w1x1 + w2x2 + \cdots + wmxm$), followed by a nonlinear activation function ($g():RRg$), which is similar to the hyperbolic tan function. The values from the final hidden layer are sent to the output layer, where they are converted into output values.

The public attributes are included in the module. coefficients and intercepts. Coefs_ is a list of weight matrices, where the weights between layer I and layer i+1 are represented by the weight matrix at index i. The list of bias vectors intercepts_ contains the bias values that were added to layer i+1 in the form of the vector at index i.

The advantages of Multi-layer Perceptron are:

- Being able to learn non-linear models
- ability to use partial_fit to learn models online and in real-time.

A multi-layer perceptron (MLP) algorithm that trains via backpropagation is implemented by Class MLPClassifier.

MLP trains on two arrays: array Y of size (n samples,), which contains the target values (class labels) for the training samples, and array X of size (n samples, n features), which includes the training data expressed as floating point feature vectors. The model can predict labels for new data after fitting (training). The training data can be used by MLP to fit a non-linear model. The weight matrices that make up the model parameters are contained in the file clf.coefs_. Only the Cross-Entropy loss function, which enables probability estimations by using the predict proba method, is currently supported by MLPClassifier.

Backpropagation is used in MLP training. To be more exact, it trains using a variation of gradient descent, with backpropagation used to determine the gradients. It minimizes the Cross-Entropy loss function for classification, producing a vector of probability estimates P(y|x) for each sample x. Using Softmax as the output function, MLPClassifier provides multi-class classification. The model also supports multi-label categorization, which allows samples to belong to many classes. The raw output is processed using the logistic function for each class. Rounding is to 1 for numbers greater than or equal to 0.5, otherwise to 0. The indices with a value of 1 represent the sample's assigned classes for a predicted output (1.17. Neural Network Models (Supervised) — Scikit-Learn 1.1.2 Documentation, n.d.).

2.2.2. The meta-classifier:

All of the models utilized in the estimator list above offer a base prediction in accordance with the methods described under each heading above. In order to determine the final prediction from the predictions provided in the previous stage, we finally use the meta-classifier. These forecasts are all provided to the meta-logistic classifier's regression model. Despite its name, logistic regression is a linear model for classification as opposed to regression. In the literature, logit regression, maximum-entropy classification (MaxEnt), and the log-linear classifier are also used to refer to logistic regression. In this model, a logistic function is used to simulate the probabilities describing the potential outcomes of a single experiment. With optional $\ell1$, $\ell2$ or Elastic-Net regularization (Regularization is applied by default, which is typical in machine learning but not in statistics), this implementation can fit binary, One-vs-Rest, or multinomial logistic regression. The enhancement of numerical stability is another benefit of regularization. Without regularization, C is set to a relatively high value) (Neal, n.d.).

2.3. Data preprocessing

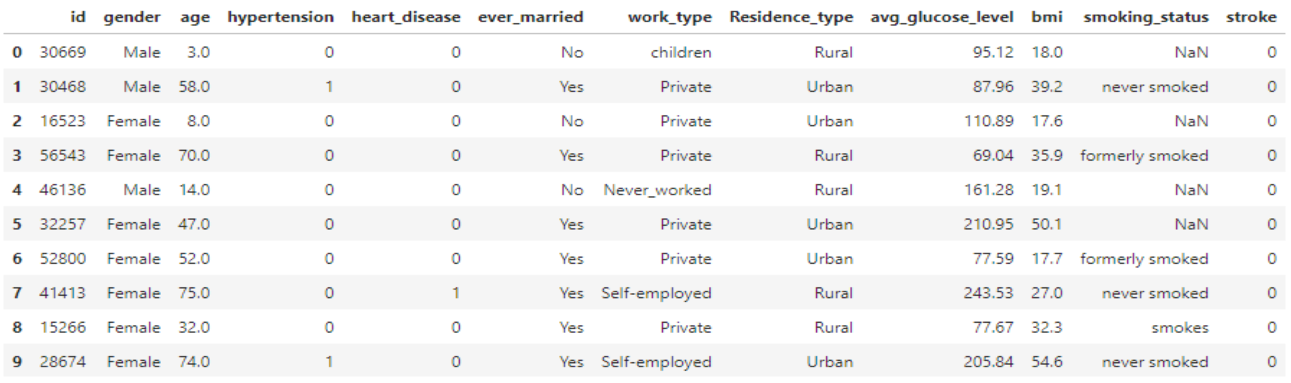
The dataset comprising 43,400 records of patients with varying levels of inputs need to be first preprocessed in such a manner that when the dataset is fed to the model it does not provide us with uneven, irregular or incorrect predictions. To do this we took use of various functions to first understand the data then preprocess it accordingly.

2.3.1. Data Exploration

We used a number of functions in the pandas library for python (Pandas Library, n.d.) which helped us derive critical information to get ready for preprocessing the data. Let us look at the df.info function in pandas which helped us get information like Not-Null count and the datatype of each column. Then we used the df.describe function to find out the count, mean, standard deviation, minimum, 25%, 50%, 75% and maximum.

3. Result

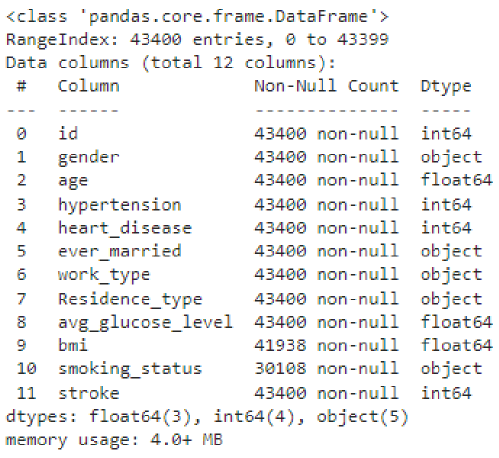
The data provided to us by the source was not fully ready to be loaded in the classifier, therefore it was in need to be preprocessed as explained in the methodology section of this paper. A sample of the data with factors (excluding id number) is shown in Figure 6 below using the ‘pandas.head()’ function in the ‘pandas’ library.



	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	30669	Male	3.0	0	0	No	children	Rural	95.12	18.0	NaN	0
1	30468	Male	58.0	1	0	Yes	Private	Urban	87.96	39.2	never smoked	0
2	16523	Female	8.0	0	0	No	Private	Urban	110.89	17.6	NaN	0
3	56543	Female	70.0	0	0	Yes	Private	Rural	69.04	35.9	formerly smoked	0
4	46136	Male	14.0	0	0	No	Never_worked	Rural	161.28	19.1	NaN	0
5	32257	Female	47.0	0	0	Yes	Private	Urban	210.95	50.1	NaN	0
6	52800	Female	52.0	0	0	Yes	Private	Urban	77.59	17.7	formerly smoked	0
7	41413	Female	75.0	0	1	Yes	Self-employed	Rural	243.53	27.0	never smoked	0
8	15266	Female	32.0	0	0	Yes	Private	Rural	77.67	32.3	smokes	0
9	28674	Female	74.0	1	0	Yes	Self-employed	Urban	205.84	54.6	never smoked	0

Fig. 6 -Dataframe output of uncleandata

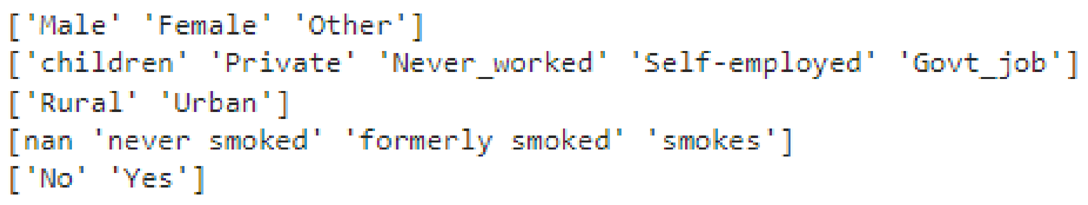
We then further wanted insight on the data and we used the ‘pandas.info()’ function of the ‘pandas’ library to get meaningful information on the data frame which is shown in use below (Fig. 7).



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 43400 entries, 0 to 43399
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   id                 43400 non-null  int64
 1   gender             43400 non-null  object
 2   age                43400 non-null  float64
 3   hypertension       43400 non-null  int64
 4   heart_disease      43400 non-null  int64
 5   ever_married       43400 non-null  object
 6   work_type          43400 non-null  object
 7   Residence_type     43400 non-null  object
 8   avg_glucose_level  43400 non-null  float64
 9   bmi                41938 non-null  float64
 10  smoking_status     30108 non-null  object
 11  stroke             43400 non-null  int64
dtypes: float64(3), int64(4), object(5)
memory usage: 4.0+ MB
```

Fig. 7 -Dataframe information with column division

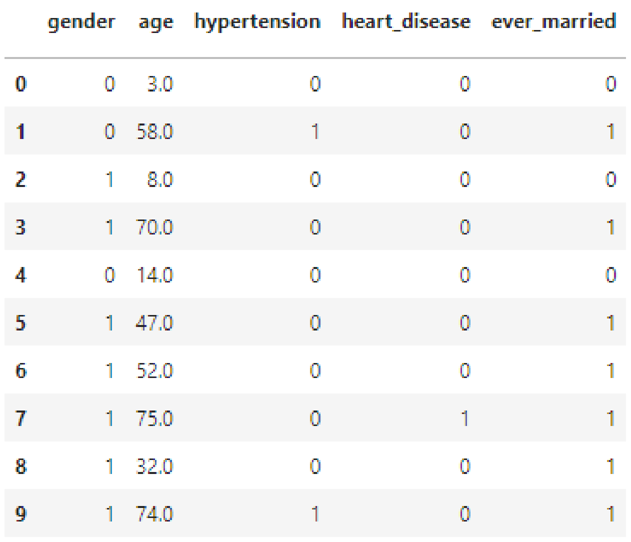
Here, if we look at Figure 7 in the ‘Dtype’ column which represents the ‘Data Type’ of the values inside each column of the dataframe. This column gives us a clue here that not all the data is in the same data type which as we know will be difficult to load in the classifier. The columns ‘gender’, ‘ever_married’, ‘work_type’, ‘residence_type’ and ‘smoking_status’ all are in an ‘object’ data type which the classifier cannot understand. To get a look at what these columns hold inside we use a function ‘pandas[].unique()’ to find out the different values they hold which we can see in Figure 8 below.



```
['Male' 'Female' 'Other']
['children' 'Private' 'Never_worked' 'Self-employed' 'Govt_job']
['Rural' 'Urban']
[nan 'never smoked' 'formerly smoked' 'smokes']
['No' 'Yes']
```

Fig. 8 -Dataframe columns with different data types

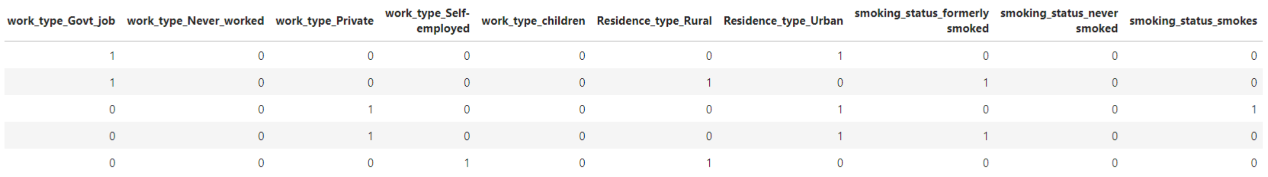
We can see in Figure 8 that columns ‘gender’ and ‘ever_married’ can be converted into binary values using simple code who’s is shown in Figure 9 below.



	gender	age	hypertension	heart_disease	ever_married
0	0	3.0	0	0	0
1	0	58.0	1	0	1
2	1	8.0	0	0	0
3	1	70.0	0	0	1
4	0	14.0	0	0	0
5	1	47.0	0	0	1
6	1	52.0	0	0	1
7	1	75.0	0	1	1
8	1	32.0	0	0	1
9	1	74.0	1	0	1

Fig. 9 -Dataframe columns changed to binary values

Changing the values of those two columns was simple but the rest of the three columns need to be first divided into different columns where only one can be true for each column in each category. This can be seen in Figure 10 below where all the three categories are divided into their respective dummy columns.



work_type_Govt_job	work_type_Never_worked	work_type_Private	work_type_Self-employed	work_type_children	Residence_type_Rural	Residence_type_Urban	smoking_status_formerly smoked	smoking_status_never smoked	smoking_status_smokes
1	0	0	0	0	0	1	0	0	0
1	0	0	0	0	1	0	1	0	0
0	0	1	0	0	0	1	0	0	1
0	0	1	0	0	0	1	1	0	0
0	0	0	1	0	1	0	0	0	0

Fig. 10 -Dataframe columns divided in dummy columns

After all of this only the last step was left i.e., to check for any null values that might be created or left out by mistake. This is done using some built in functions of the ‘pandas’ library.

In this study we also studied the data extensively to provide the model with data that is best in this use case, with that in mind we made visualized charts from the data that provide a better understanding of the data. The visualization of the dataset is divided below with their respective titles.

3.1. The proportion of stroke among gender

Gender has a minimal impact on the model, as shown in Figure 10 below, where the male gender dominates with 55.5% of the demographic compared to the female gender's 44.5%. This small difference has led us to assume that gender has a very small influence in what causes strokes.

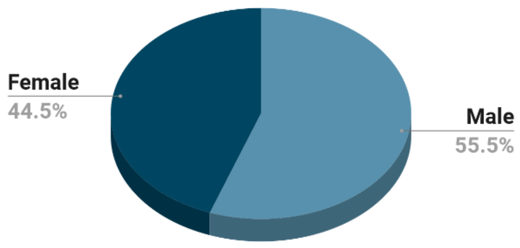


Fig. 11 -stroke among genders

3.2. The proportion of stroke based on marital status

We are shocked to see that the unmarried participants have a percentage of 10.4% while a sizable portion is of the married participants, i.e., 89.6%, which leads us to believe that marital status of a person has to be taken into account while checking for a stroke possibility. One's marital status shows up to be significant enough to play a role in what causes a stroke. Figure 11 below shows the chart that represents all of this.

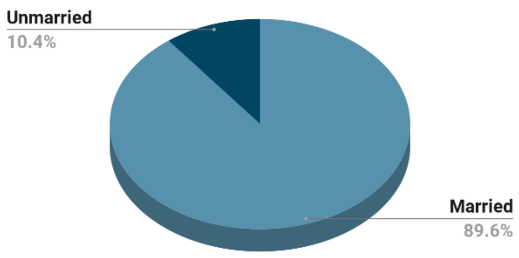


Fig. 12 -stroke based on marital status

3.3. The proportion of stroke based on work type

We can determine the level of work pressure you are experiencing based on your job status. The plot below (Fig. 12) excellently illustrates this since it shows Private jobs with a 55.7% stroke in our dataframe. Other occupations, such as government employment, are at 12%, while self-employment is at 32.2%. Children there are only at 0.2% because of a wide range of factors. By taking into account job status as a component that points us in the proper direction, all of this data aids us in making a prediction.

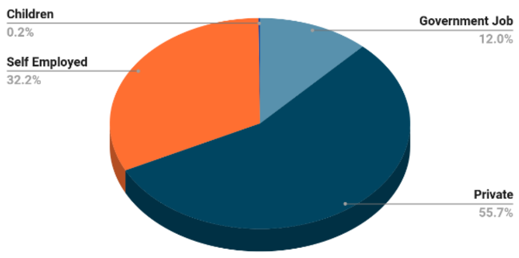


Fig. 13 -stroke based on work type

3.4. The proportion of stroke based on residence type

Looking at whether a participant lives in an urban or rural setting might affect his or her chances of having a stroke, but this was shown to be incorrect when we saw the chart (Fig. 13), which shows that 51% of participants in urban settings and 49% of participants in rural settings experienced strokes. This demonstrates that the participant's place of residence has no bearing on the likelihood that a stroke will occur.

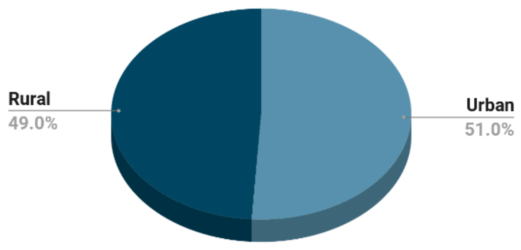


Fig. 14 -stroke based on residence type

3.5. The proportion of stroke based on smoking status

Looking at whether a participant lives in an urban or rural setting might affect his or her chances of having a stroke, but this was shown to be incorrect when we saw the chart (Fig. 13), which shows that 51% of participants in urban settings and 49% of participants in rural settings experienced strokes. This demonstrates that the participant's place of residence has no bearing on the likelihood that a stroke will occThis graph (Fig. 14), which suggests that 46.7% of people who have never smoked are discovered to have had a stroke, exhibits a highly peculiar tendency that cannot be ignored. Participants with a 32.8% stroke rate who had previously smoked. Just 20.4% of individuals who smoke have experienced a stroke. Due to the large amount of unidentified data available in that column, this data does not fully represent the situation. Use of the information in this column with foresight.

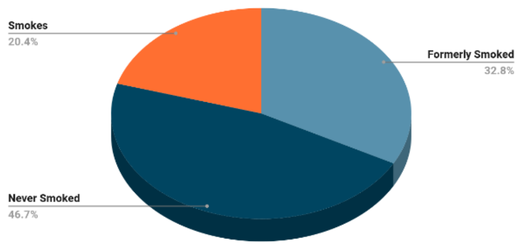


Fig. 15 -stroke based on smoking status

4. Conclusion & future agendas

This research concludes that all the factors we analyzed have a significant enough contribution to our outcomes. Therefore, all the data was used and imported after the preprocessing and the approach we took yielded us an accuracy score of 98.48% for training set and an accuracy score of 98.49% for the testing set of our data. This accuracy score was devised using the sklearn.metrics function. We also used many test cases to determine the accuracy of our model. Creating these test cases was an automated loop of different arrays combined in one single input array with random values within the test and training range so that all values even if random are correct and under the measurements in the data frame. We did more than 10,000 runs of this model and it was consistent every single time as expected from it. Few of the test results with inputs are given below in Table 1 (Note that the factors here are not disclosed due to confidentiality).

Table 1 - test results offew cases.

S. No.	Factor 1	Factor 2	Factor 3	Factor 4	Factor 5	Factor 6	Factor 7	Factor 8	Factor 9	Factor 10	Result
1	1	40	0	0	0	100	35	P	U	N	0
2	0	35	1	0	1	80	45	G	U	F	0
3	1	20	0	0	0	80	27	C	U	S	0
4	1	20	0	0	0	80	22	C	U	S	0
5	1	20	1	0	0	80	26	C	U	S	0
6	**1**	**45**	**0**	**1**	**1**	**50**	**30**	**S**	**R**	**S**	**1**

The table above (Table 1) shows us that a slight variation in the input field can change the whole outcome of whether a person is likely to get a stroke or not. As we can see in Case 4 to 6 only a slight variance has changed it to predict a stroke in the last case of patient number 6 (random id is given with reference to Table 1 to protect confidentiality).

The model created in this research is available to modify or use for personal/commercial purposes under the MIT license, but under limitations. The future scope of this research would be to implement it in the smart wearable devices as those already collect a lot of data and can help a user get a timely signal if his/her health is pointing towards a stroke scenario. To find the source code of the machine learning model please visit this link: https://github.com/psavarmattas/Stroke-Prediction-ML-Model.git .

Acknowledgements

Throughout my research journey, I met a lot of people who assisted me in achieving this life ambition. As a result, I should show my gratitude to those who have unconditionally supported, aided, and empowered me in all situations. First and foremost, I want to thank my parents for their love and support throughout my life. Thank you both for giving me the courage to reach for the stars and pursue my ambitions. Thank you for your patience and encouragement over my innumerable crises. Your friendship enriches my life. I can't remember everyone's names, but you're always on my mind.This day would not have been possible without them. This research is merely the start of my journey.

References

About Stroke | cdc.gov. (2022, May 4). Centers for Disease Control and Prevention. Retrieved October 21, 2022, from https://www.cdc.gov/stroke/about.htm

Amin, M. S., Chiam, Y. K., & Varathan, K. D. (2019, March 1). Identification of significant features and data mining techniques in predicting heart disease. Semantic Scholar. Retrieved October 21, 2022, from https://www.semanticscholar.org/paper/Identification-of-significant-features-and-data-in-Amin-Chiam/a26b80d1f9de273b6fbc7318c4ae1626b404890d

Chen, M. (2017). Disease Prediction by Machine Learning Over Big Data From Healthcare Communities, 5, 8869-8879. IEEE Access. 10.1109/ACCESS.2017.2694446

Esfahani, H. A., & Ghazanfari, M. (2017). Cardiovascular disease detection using a new ensemble classifier, 1011-1014. IEEE 4th International Conference on Knowledge-Based Engineering and Innovation (KBEI). 10.1109/KBEI.2017.8324946

Gold, S. (2018, December 5). Clinical Concept Value Sets and Interoperability in Health Data Analytics. NCBI. Retrieved October 21, 2022, from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6371254/

Gupta, S., & Sedamkar, R. R. (2019, March 1). Apply Machine Learning for Healthcare to enhance performance and identify informative features. Semantic Scholar. Retrieved October 21, 2022, from https://www.semanticscholar.org/paper/Apply-Machine-Learning-for-Healthcare-to-enhance-Gupta-Sedamkar/0928ffcf21fa80cbfab1fd63890cba3b214971dd

K, S., & Sarathambekai, S. (2021). Machine Learning based Prediction Model for Health Care Sector - A Survey, 1-7. IEEE Access. 10.1109/i-PACT52855.2021.9696646

Kolukisa, B. (2018). Evaluation of Classification Algorithms, Linear Discriminant Analysis and a New Hybrid Feature Selection Methodology for the Diagnosis of Coronary Artery Disease. Abdullah Gül Üniversitesi. Retrieved October 21, 2022, from http://www.agu.edu.tr/userfiles/Fuarlar/GSES/EvaluationOfClassificationAlgori.pdf

Lindsay, M. P., Norrving, B., Sacco, R. L., Brainin, M., Hacke, W., Martins, S., Pandian, J., & Feigin, V. (2019). World Stroke Organization (WSO): Global Stroke Fact Sheet 2019. PubMed. Retrieved October 17, 2022, from https://pubmed.ncbi.nlm.nih.gov/31658892/

Ma, Y., Yang, J., & Wang, Y. (2017). Big Health Application System based on Health Internet of Things and Big Data, 5, 7885-7897. IEEE Access. 10.1109/ACCESS.2016.2638449

McKinsey Analytics. (2018, April 14). McKinsey Analytics Online Hackathon - Healthcare Analytics. DataHack. Retrieved October 22, 2022, from https://datahack.analyticsvidhya.com/contest/mckinsey-analytics-online-hackathon/#ProblemStatement

Mettler, M. (2016). Blockchain technology in healthcare: The revolution starts here, 1-3. IEEE 18th International Conference on e-Health Networking, Applications and Services (Healthcom). 10.1109/HealthCom.2016.7749510

Mourguet, M., Chauveau, D., Faguer, S., Ruidavets, J. B., Béjot, Y., Ribes, D., Huart, A., Alric, L., Balardy, L., Astudillo, L., Adoue, D., Sailler, L., & Pugnet, G. (2019). Increased ischemic stroke, acute coronary artery disease and mortality in patients with granulomatosis with polyangiitis and microscopic polyangiitis. PubMed. Retrieved October 17, 2022, from https://pubmed.ncbi.nlm.nih.gov/30236485/

Neal, R. M. (n.d.). 1.1. Linear Models — scikit-learn 1.1.2 documentation. Scikit-learn. Retrieved October 23, 2022, from https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

1.4. Support Vector Machines — scikit-learn 1.1.2 documentation. (n.d.). Scikit-learn. Retrieved October 22, 2022, from https://scikit-learn.org/stable/modules/svm.html#svm-classification

1.11. Ensemble methods — scikit-learn 1.1.2 documentation. (n.d.). Scikit-learn. Retrieved October 22, 2022, from https://scikit-learn.org/stable/modules/ensemble.html#b2001

1.11. Ensemble methods — scikit-learn 1.1.2 documentation. (n.d.). Scikit-learn. Retrieved October 22, 2022, from https://scikit-learn.org/stable/modules/ensemble.html#forests-of-randomized-trees

1.17. Neural network models (supervised) — scikit-learn 1.1.2 documentation. (n.d.). Scikit-learn. Retrieved October 22, 2022, from https://scikit-learn.org/stable/modules/neural_networks_supervised.html#multi-layer-perceptron

1.10. Decision Trees — scikit-learn 1.1.2 documentation. (n.d.). Scikit-learn. Retrieved October 22, 2022, from https://scikit-learn.org/stable/modules/tree.html#tree

1.6. Nearest Neighbors — scikit-learn 1.1.2 documentation. (n.d.). Scikit-learn. Retrieved October 22, 2022, from https://scikit-learn.org/stable/modules/neighbors.html#classification

Pandas Library. (n.d.). pandas - Python Data Analysis Library. Retrieved November 2, 2022, from https://pandas.pydata.org/

Ratajczak, R. (2019, May 7). Automatic Land Cover Reconstruction From Historical Aerial Images: An Evaluation of Features Extraction and Classification Algorithms. Archive ouverte HAL. Retrieved October 21, 2022, from https://hal.archives-ouvertes.fr/hal-02003932/document

Sasubilli, G., & Kumar, A. (2020). Machine Learning and Big Data Implementation on Health Care data, 859-864. 4th International Conference on Intelligent Computing and Control Systems (ICICCS). 10.1109/ICICCS48265.2020.9120906

Shikany, J. M., Safford, M. M., Soroka, O., Newby, P. K., Brown, T. M., Durant, R. W., & Judd, S. E. (2020, March 2). Abstract P520: Associations of Dietary Patterns and Risk of Sudden Cardiac Death in the Reasons for Geographic and Racial Differences in Stroke Study Differ by History of Coronary Heart Disease. AHA Journals. Retrieved October 17, 2022, from https://www.ahajournals.org/doi/abs/10.1161/circ.141.suppl_1.P520

sklearn.ensemble.RandomForestClassifier — scikit-learn 1.1.2 documentation. (n.d.). Scikit-learn. Retrieved October 22, 2022, from https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html#sklearn.ensemble.RandomForestClassifier

Stacking in Machine Learning. (2019, May 20). GeeksforGeeks. Retrieved October 22, 2022, from https://www.geeksforgeeks.org/stacking-in-machine-learning/?ref=rp

Stacking in Machine Learning. (2021, December 21). GeeksforGeeks. Retrieved October 22, 2022, from https://www.geeksforgeeks.org/stacking-in-machine-learning-2/

Virani, S. S., Alonso, A., & Benjamin, E. J. (2020, March 3). Heart Disease and Stroke Statistics-2020 Update: A Report From the American Heart Association. PubMed. Retrieved October 17, 2022, from https://pubmed.ncbi.nlm.nih.gov/31992061/

Yadav, P., Steinbach, M., Kumar, V., & Simon, G. (2017, February 9). [1702.03222] Mining Electronic Health Records: A Survey. arXiv. Retrieved October 21, 2022, from https://arxiv.org/abs/1702.03222.