Optimizing Predictions of Brain Stroke Using Machine Learning

Venkata Sravan Telu, Vinay Padimi and Devarani Devi Ningombam

Department of Computer Science and Engineering,
GITAM Institute of Technology, GITAM University, Visakhapatnam, Andhra Pradesh, 530045
telusravan@gmail.com, vinaypadimi@gmail.com, devaraninin@gmail.com

***** Correspondence: devaraninin@gmail.com

Abstract

Stroke, also known as a brain attack, happens when the blood vessels are blocked by something or when the blood supply to the brain stops. In any of these cases, the brain becomes damaged or dies. Our brain controls every action in our body, like how many hormones are produced and released, breathing, memory, and everything. If the flow of blood to the brain gets occluded, then the cells in the brain start to die within a moment due to the lack of oxygen. This eventually causes strokes. Stroke is one of the most common causes for death globally. According to the World Health Organization (WHO), stroke is responsible for 11% of global deaths. So, in this paper, we propose a novel machine learning model with supervised learning techniques that can predict whether a person is likely to get a stroke or not by taking medical inputs such as medical risk factors which can cause strokes like smoking status, heart disease, glucose value, and hypertension. This paper compares various state-of-the-art machine learning algorithms, such as the Support Vector Machine (SVM), random forest, KNN algorithms, etc. Our simulation results show that the proposed scheme increases accuracy significantly (94.6%) and improves system performance.

Keywords: Machine Learning, Brain Stroke, KNIME, Random Forest, Support Vector Machine, Recall, Precision.

1. Introduction

Stroke, also known as a cerebrovascular accident, occurs due to many physical and medical factors. Brain stroke is one of the common problems that may lead to severe effects and may result in death. The leading causes of brain stroke are when blood cannot reach the brain and the brain's functionality stops working. As the brain is an essential organ in the body that does all the tasks, the person must be safe from brain stroke.

The American government has declared that around 1,40,000 Americans die due to strokes. The survival rate after a stroke is 60%. With the emergence of machine learning (ML), we can

create a model that can predict a brain stroke at its early stage. So, it is imperative to create a novel ML model that can optimize the performance of brain stroke prediction. An ML model for predicting stroke using the machine learning technique is presented in [1]. In this paper, the authors proposed the model by under-sampling the majority class in the target variable, stroke, and it was reduced up to 498. The main difficulty in their work is that in ML, a record size of 498 may not result in the optimal solution, but rather in near-optimal or sub-optimal solutions. This methodology motivates us to think about another technique that can improvise learning performance. Improvising the learning rate is proposed in [2]. The authors analyzed the data imbalance in the target variable and oversampled the minority class. The main issue in this paper is outliers. They trained the models with outliers.

The main contributions to the paper are listed as follows:

1. To solve the issues mentioned earlier, we propose techniques for removing outliers and oversampling the minority class in our work.
2. Moreover, we propose a novel technique to increase the ML model's accuracy and compare various algorithms, such as the Naïve Byes algorithm, the Random-forest algorithm, the Support-vector machine (SVM), the K-nearest neighbor (KNN), etc.
3. We have observed that after performing the above mentioned techniques, we have achieved a better Receiver Operative Characteristic (ROC) curve than [1] and better accuracy than [2].

The overview of the paper is listed as follows: Section 2 presents the related work, Section 3 presents the proposed technique, performance analysis is present in Section 4, and paper is concluded in Section 5.

2. Related Work

In [3], an imbalanced data set, under-sampling the majority class to the minority class is a common practice. This paper demonstrates that combination of under-sampling the majority class and over-sampling the minority increases the Area Under Curve (AUC) in the ROC more than only under-sampling the majority class. The authors in the paper [4] discussed the cross-validation method, which is one of the most commonly used data resampling methods to examine the generalization of the model's prediction. Tuning model parameters can be examined by cross-validation. General types of cross-validation and their data sampling methods are discussed.

Another existing work [5] conducted training of 15 medical datasets with different algorithms. Among the different algorithms applied, naïve Bayes proves to be effective. The author stressed using the Naïve Bayes algorithm, which is good at predicting medical problems. Moreover, in [6], the authors proposed evaluating supervised learning methods such as decision tree and naïve Bayes using the KNIME platform. Furthermore, they evaluated the models with precision, recall, and F-measure metrics. The authors in the paper [7] assessed six machine learning algorithms on real-world medical diagnostics data sets. This paper emphasizes the use of AUC in ROC and overall accuracy to evaluate the model better. The paper also discussed that an increase in test observations and AUC decreased standard error.

3. Proposed Technique

3.1 Architecture and Framework

This section will discuss the conceptual view of the entire research. We have considered the KNIME simulator [8]. KNIME, Konstanz Information Miner, is a free and open-source data

analytics, reporting, and integration platform. This platform allows users to run machine learning models in the same way that traditional methods such as Python or R do. The dataset is imported from [9]. The data is imported into KNIME and then preprocessed with appropriate techniques. After data preprocessing, the data is trained with machine learning algorithms.

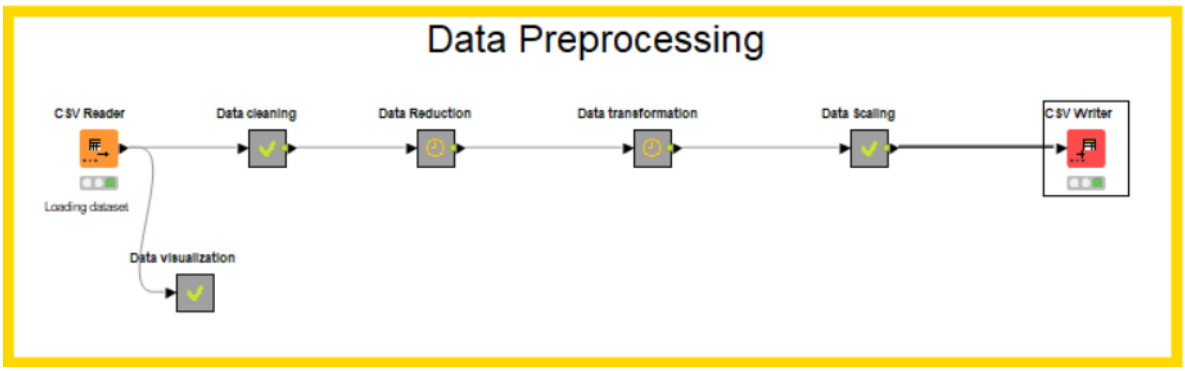

Fig. 1 Represents overview of data preprocessing

3.1.1 Data Preprocessing

A model gets good accuracy with good quantity and quality of data. Preprocessing the data is essential for good accuracy. Duplicate records can cause overfitting issues. So, we have made sure that there are no duplicate records in our dataset. We have visualized all attributes of the data and found that there are missing values in the 'bmi' attribute. 'bmi' means body mass index. The missing values are replaced with the mean. Establishing a relationship between two variables is vital for any machine learning model. So, we calculated the correlation between all attributes, set a threshold correlation of 0.5, and filtered the attributes. In this manner, 'ever_married' is filtered due to less correlation. 'ever_married' is an attribute that tells whether that person is married or not. Attribute 'id' has been removed as it is not significant for model learning.

Removing outliers in the model can make the model perform better with good accuracy. Consequently, the outliers in the data have been removed to ensure good model performance. After removing the outliers, the occurrence of 1 and 0 has been reduced to 112 from 249 and 3814 from 4861. This process can reduce noise and, thus improve the performance of our model. Another issue while analyzing the large dataset is the data imbalance. The problem with data imbalance is that one of the variables in the target attribute will have fewer records. With fewer records, the metrics of the variables like recall and precision will be affected, which could affect the performance of the model. In order to solve this problem, we have oversampled the occurrence of 1 to match the occurrence of 0.

All machine learning models take inputs in numeric format only. If we have categorical data, then change it to numerical data. In our dataset, the attributes "gender", "work_type", "Residence_type", and "smoking status" are changed from categorical to numerical data by ordinal encoding. Normalization is one of the processes in data preprocessing that increases the performance of the model. Normalization brings all the attributes of different ranges to a fixed range without changing the difference in the ranges of values. We have normalized every column with Z-score normalization. This normalization normalizes the data values so that the mean is

equal to 0 and the standard deviation to 1. Eventually, we enhanced the quality of the data needed for training the model by doing all the necessary tasks. The classification algorithms used in our research are listed as below:

a) Logistic Regression
b) Decision Tree
c) Random forest
d) KNN
e) Naïve bayes
f) SVM

3.1.2 Training Phase

Training a model is the heart of machine learning. In our paper, we used supervised learning methods to train the model. In supervised learning, machine learning algorithms train the model. In our work, as we need to predict whether the person will affected by a stroke or not, it indicates that we are trying to find a discrete value (a finite number). To predict a discrete value in supervised learning, we use classification algorithms.

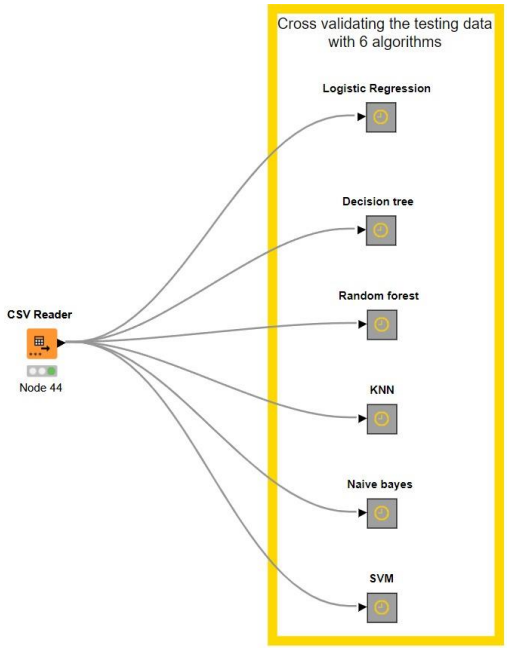

Fig 2 Represents cross validation stage

For training, we have partitioned the dataset into an 80:20 ratio. As we have mentioned six algorithms, we need six models to build. We have performed cross-validation for every model. When the model is subjected to cross-validation, we can approximate how the model responds to unseen data before testing it on 20% of the testing data. We have executed k-fold cross validation because k-fold guarantees that every record in training and testing data has an opportunity to impact the model. In k-fold cross-validation, the dataset is divided into k folds, and the model is trained with (k-1) folds and will be tested with the kth fold. This process continues for "n" iterations, where n is the user assigned number. We have performed cross-validation on each

model to check its accuracy. We have set a threshold of at least 60% overall accuracy. If we keep the threshold value at less than 60%, the model might produce a sub-optimal solution. If the model is underperforming, we need to change the data preprocessing. Even after modifying the data preprocessing, if we get poor accuracy, that algorithm is not suitable for our dataset.

a) Logistic Regression

In logistic regression, it shows the linear relationship between the independent variable (X-axis) and the dependent variable (Y-axis). Based on the given data points, we try to plot a line that models the points the best. This line is also called thea best fit straight line. We have performed k-fold cross validation on the training dataset with 10 iterations. The accuracy of the model is 78.5%.

b) Decision Tree

In decision tree algorithm, decisions are based on conditions on any of the features. Based on the decisions, it forms a tree structure. We have performed k-fold cross validation on the training dataset with 10 iterations. The accuracy of the model is 91%.

c) Random Forest

In random forest algorithm , it builds multiple decision trees and merges them together to get better performance. We have performed k-fold cross validation on the training dataset with 10 iterations. The accuracy of the model is 94.3%.

d) KNN

In KNN algorithm, we need a set value k (k nearest neighbours to a data point which needs to be classified). It calculates the Euclid distance and forms a class with the k nearest data points. In this class, the data points classifiers into the category that has more occurrences. We have performed k-fold cross validation on the training dataset with 10 iterations. The accuracy of the model is 90.6%.

e) Naïve Bayes

The Naive Bayes classifier works on the principle of conditional probability. We have performed k-fold cross validation on the training dataset with 10 iterations. The accuracy of the model is 76.2%.

f) Support Vector Machine (SVM)

SVM works by mapping data to a high-dimensional feature space so that data points can be categorized. A separator between the categories is found, which is called a hyperplane. Following this, characteristics of new data can be used to predict the group to which a new record should belong. We have performed k-fold cross validation on the training dataset with 10 iterations. The accuracy of the model is 62.3%. In the training phase, all the models exceeded the threshold of 60% overall accuracy. So, we are considering all the models for testing.

3.1.3 Testing Phase

An ML model is subjected to testing to check its performance on a model on testing dataset. As discussed earlier, we partitioned the data set into a training set and a testing set in an 80:20 ratio. Now we are going to test 20% of the testing data.

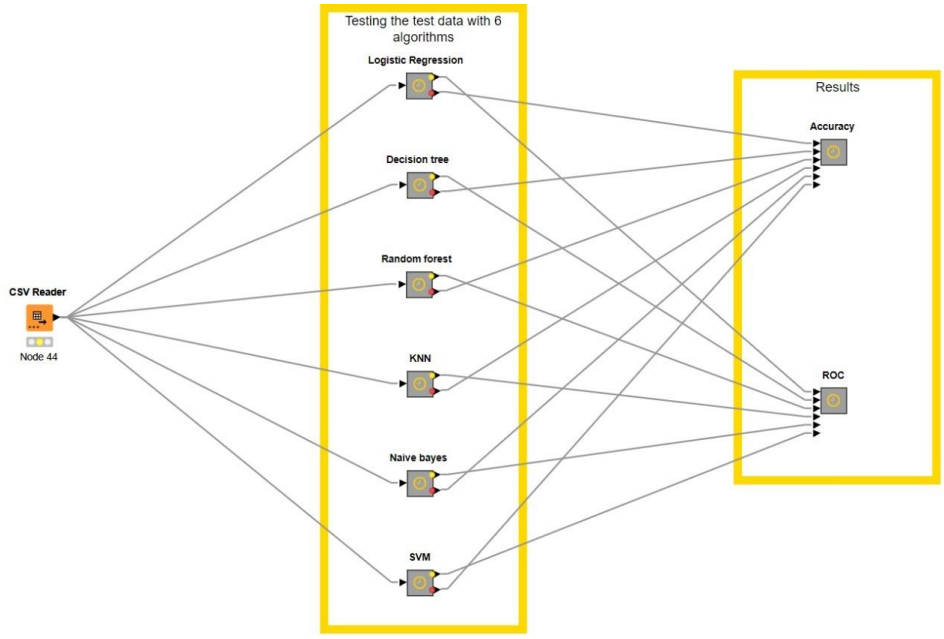

Fig 3 Represnts testing phase

a) Logistic Regression

In logistic regression, we have tested on 20% of the testing data. After testing, the model has performed an overall performance of 77.7%. In ROC, the model achieved an AUC of 0.86.

b) Decision Tree

In decision tree, we have tested on 20% of the testing data. After testing, the model has performed an overall performance of 77.7%. In ROC, the model achieved an AUC of 0.913.

c) Random Forest

In random forest, we have tested on 20% of the testing data. After testing, the model has performed an overall performance of 94.4%. In ROC, the model achieved an AUC of 0.992.

d) KNN

In KNN, we have tested on 20% of the testing data. After testing, the model has performed an overall performance of 93.3%. In ROC, the model achieved an AUC of 0.971.

e) Naïve Bayes

In naïve bayes, we have tested on 20% of the testing data. After testing, the model has performed an overall performance of 80.3%. In ROC, the model achieved an AUC of 0.86.

f) Support Vector Machine (SVM)

In SVM, we have tested on 20% of the testing data. After testing, the model has performed an overall performance of 94.6%. In ROC, the model achieved an AUC of 0.99.

3.1.4 Overall Phase

The dataset contains 201 missing values of the "bmi" attribute, which are replaced by the mean of "bmi". We have observed that the data is imbalanced between the two target variable "stroke" classes. Before proceeding any further, we checked if any duplicate records were there. We have examined any correlation between attributes and have eliminated the "ever_married" attribute by taking 0.5 as the threshold and removing the "id" attribute as it is of no use. The outliers of all the attributes have been removed. As we discussed before, all supervised learning algorithms need numerical inputs. We transformed all the categorical data into numeric data by ordinal encoding. We have over-sampled the minority class, 1, to equal the observations of the majority class 0, and resulted in a record size of 7628 X 10. Each class resulted in 3814 records. The data has normalized with Z-score normalization. With this, data preprocessing has finished.

We have split the data into training and testing data with an 80:20 ratio. Then we performed cross-validation with training data on all the models with k-fold cross-validation with a k value equal to 10. If the model has not performed with more than 60% overall accuracy, then model is to be discarded as below 60% overall accuracy can result in sub-optimal performance. After testing the validation set, every model has reached more than the 60% overall accuracy. So, we are considering all the models for testing. All the models have been evaluated with test data. After testing with the test dataset, all the models performed with more than 60% accuracy. We have generated an ROC curve for all the models. The overall architecture is shown in Fig. 2 and the corresponding pseudo code is given in Algorithm 1.

4. Performance Analysis

We have analysed various state-of-the-art algorithms like logistic regression, decision tree, random forest, naïve bayes, KNN, and support vector machine. We have taken accuracy and Receiver Operating Characteristic (ROC) curve has the performance measure. The greatera the value accuracy and AUC in ROC, the better the performance of the model. We have also measured recall, precision, F-score for each model. The formulas of recall, precision, F-score, specificity, nd sensitivity are below [10]:

Algorithm 1: Pseudo Code of the Proposed Technique
Step 1: Loading the dataset into KNIME
Step 2: Analysis of the dataset thoroughly by looking the available attributes
Step 3: Preprocessing of the dataset

Step 4: Split the dataset into training and testing data into 80:20 ratio respectively
Step 5: Cross validate the training dataset with k-fold cross validation with all the algorithms
Step 6: Keeping the threshold value as 0.6. If overall accuracy is less than 60%, discard the model
Step 7: Test the models with testing data
Step 8: Select the model which has highest overall accuracy

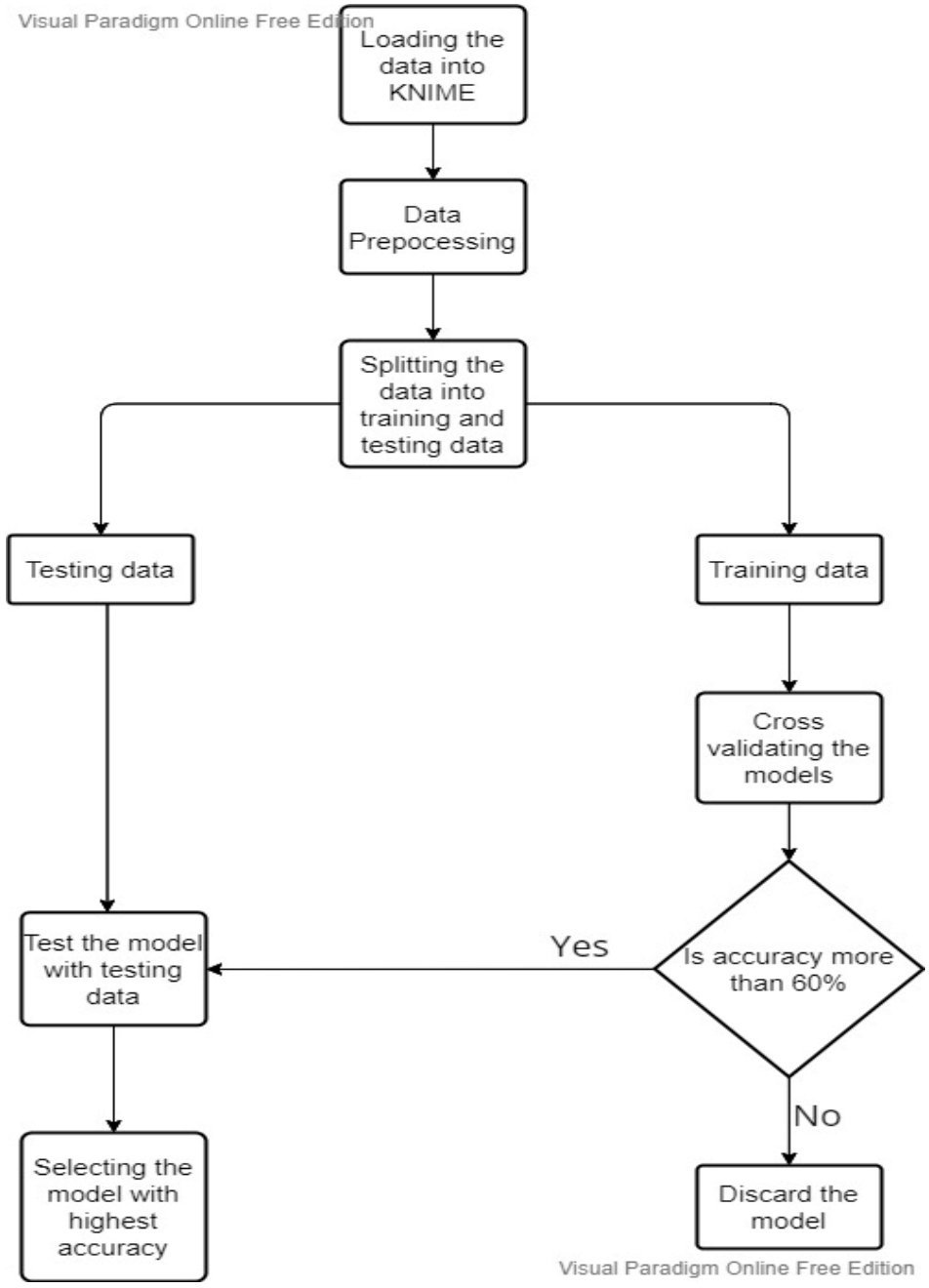

Fig 4 Represents the overall system methodologies.

4.1 Precision: It is the measure that tells how many positive observations are correct from the confusion matrix and is calculated as,

$$Precision = \frac{True\ Positive}{(True\ Positive+False\ Postive)} \quad (1)$$

where true positive tells the proportion of positive classes it has classified correctly and false positive tells the proportion of positive class it has classified wrongly.

4.2 Recall: It the measure that tells how much actual positive observations are correct from the confusion matrix and computed as,

$$Recall = \frac{True\ Positive}{(True\ Positive+False\ Negative)} \quad (2)$$

where false negative tells the proportion of negative classes it has classified wrongly.

4.3 F-score: It is the weighted mean of recall and precision and calculated as.

$$F\_score = \frac{2(Recall \times Precision)}{(Recall+Precision)} \quad (3)$$

4.4 Accuracy: It is the measure that tells total correct predictions from the model and calculated as,

$$Accuracy = \frac{True\ Positive+True\ Negative}{(True\ Positive+True\ Negativ+False\ Positive+False\ Negative)} \quad (4)$$

where true negative tells the proportion of negative class it has classified correctly.

The accuracies of the algorithm are represented in table as below:

Algorithm	**Overall Accuracy**
Logistic regression	78%
Decision tree	91.2%
Random forest	94.4%
KNN	93.3%
Naïve bayes	80.3%
SVM	94.6%

Table 1 Represents overall accuracies.

The AUC of all algorithms are represented in ROC as below:

Algorithm	**Area Under curve**
Logistic regression	0.86
Decision tree	0.913
Random forest	0.992
KNN	0.971
Naïve bayes	0.86
SVM	0.99

Table 2 Represents AUC performance.

Precision of all algorithms are presented in table as below:

Algorithm	**1**	**0**
Logistic regression	0.76	0.805
Decision tree	0.9	0.926
Random forest	0.914	0.977
KNN	0.886	0.928
Naïve bayes	0.754	0.875
SVM	0.996	0.905

Table 3 Represents precision of class 0 and 1 of all algorithms.

Recall of all algorithms are presented in table as below:

Algorithm	**1**	**0**
Logistic regression	0.82	0.74
Decision tree	0.928	0.896
Random forest	0.979	0.908
KNN	0.993	0.872
Naïve bayes	0.899	0.706
SVM	0.895	0.996

Table 4 Represents recall of class 0 and 1 of all algorithms.

F-Score of all algorithms are presented in table as below

Algorithm	**1**	**0**
Logistic regression	0.789	0.771
Decision tree	0.914	0.911
Random forest	0.946	0.942
KNN	0.936	0.928
Naïve bayes	0.82	0.782
SVM	0.943	0.948

Table 5 Represents F-score of class 0 and 1 of all algorithms.

Finally, the overall performce result is presented in Fig. 3. Dataset used in the paper and additional results are presented in **Appendix 1(A) and 1(B),** respectively.

5. Conclusions

In this work, we have implemented a novel ML model to predict the stroke rate. We have analysed various state-of-the-art algorithms such as, Naïve algorithm, Random-forest algorithm, K-nearest, Decision tree, Support Vector Machine, Logistic regression. From the analysis, we have concluded that the SVM algorithm provides the best accuracy and highest AUC among all the considered models, with 94.6% accuracy and 99% AUC.

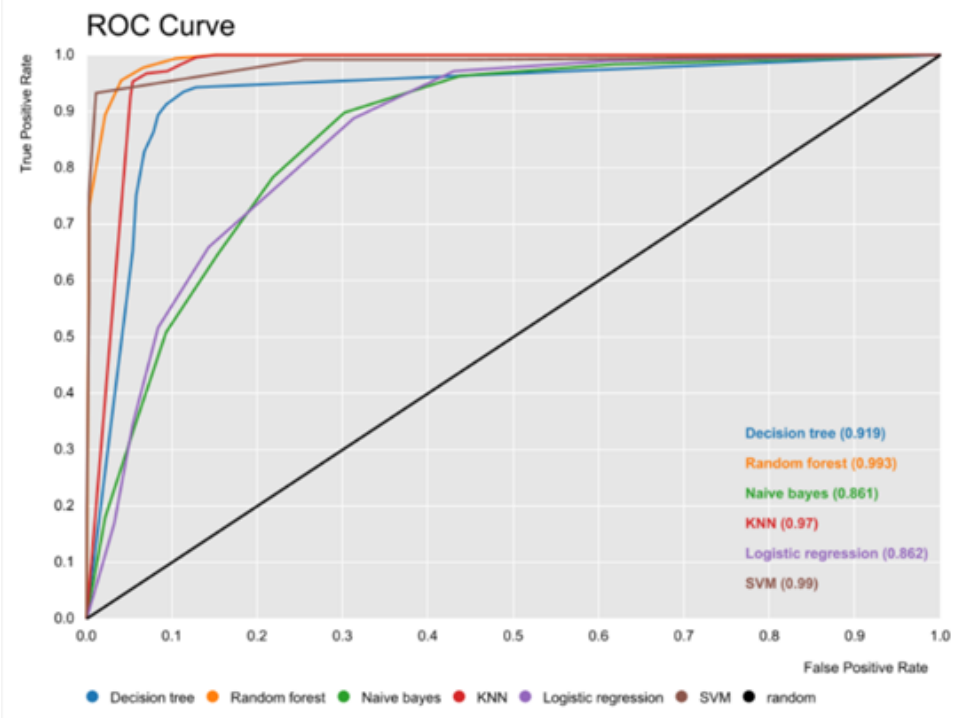

Fig. 5 Represents ROC curve for all the algorithms.

Conflicts of Interest: "The authors declare no conflict of interest."

References

[1] K. Akash, H. Shashank, S. .S, and T. A.M, "Prediction of Stroke Using Machine Learning," Jun. 2020.

[2]. A. Kshirsagar, H. Goyal, S. Loya, and A. Khade, "Brain Stroke Prediction Portal Using Machine Learning," vol. 07, no. 03, p. 7, 2021.

[3] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," J. Artif. Intell. Res., vol. 16, pp. 321–357, Jun. 2002, doi: 10.1613/jair.953.

[4] D. Berrar, "Cross-Validation," 2018. doi: 10.1016/B978-0-12-809633-8.20349-X.

[5]"Medical Data Classification with Naive Bayes Approach." https://scialert.net/abstract/?doi=itj.2012.1166.1174 (accessed Apr. 14, 2022).

[6] "Evaluating the Performance of Supervised Classification Models: Decision Tree and Naïve Bayes Using KNIME - VIT University." https://research.vit.ac.in/publication/evaluating-the-performance-of-supervised (accessed Apr. 14, 2022).

[7] A. P. Bradley, "The use of the area under the ROC curve in the evaluation of machine learning algorithms," Pattern Recognit., vol. 30, no. 7, pp. 1145–1159, Jul. 1997, doi: 10.1016/S0031-3203(96)00142-2.

[8] https://www.knime.com/

[9] https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset

[10]https://towardsdatascience.com/accuracy-recall-precision-f-score-specificity-which-to-optimize-on-867d3f11124.

APPENDIX 1:

A. Datasets: The dataset has been taken from Kaggle.

id	gender	age	hypert...	heart_...	ever_m...	work_t...	Reside...	avg_gl...	bmi	smoking_...	stroke
9046	Male	67	0	1	Yes	Private	Urban	228.69	36.6	formerly smoked	1
51676	Female	61	0	0	Yes	Self-employed	Rural	202.21	?	never smoked	1
31112	Male	80	0	1	Yes	Private	Rural	105.92	32.5	never smoked	1
60182	Female	49	0	0	Yes	Private	Urban	171.23	34.4	smokes	1
1665	Female	79	1	0	Yes	Self-employed	Rural	174.12	24	never smoked	1
56669	Male	81	0	0	Yes	Private	Urban	186.21	29	formerly smoked	1
53882	Male	74	1	1	Yes	Private	Rural	70.09	27.4	never smoked	1

Table 6 Represents sample of dataset

Target variable “stroke” has data imbalance

stroke	Count (stroke)
0	4861
1	249

Table 7 Represents occurrence of stroke

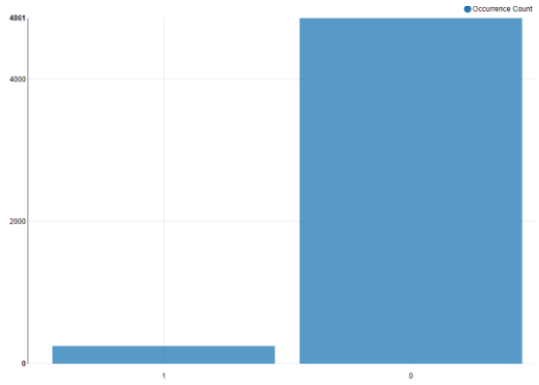

Fig. 6 Represents bar chart of occurrence of stroke

Data preprocessing includes data cleaning, data reduction, data transformation, data scaling. It is a standardised practice to keep correlation threshold to 0.5.

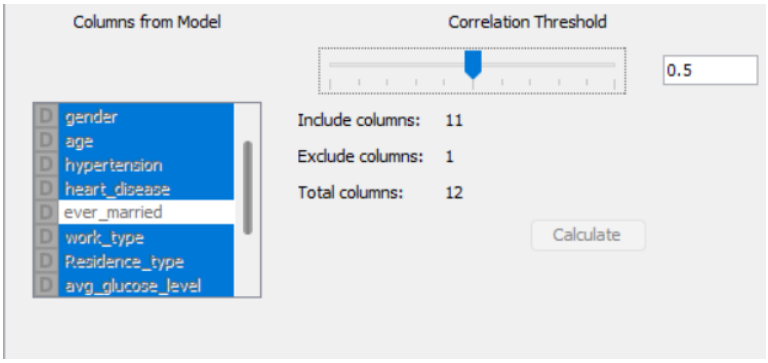

Fig. 7 Represents correlation threshold

Over sampling the stroke variable of class 1.

stroke	Count (...
1	3814
0	3814

Table 8 Represents occurrence of stroke after over sampling

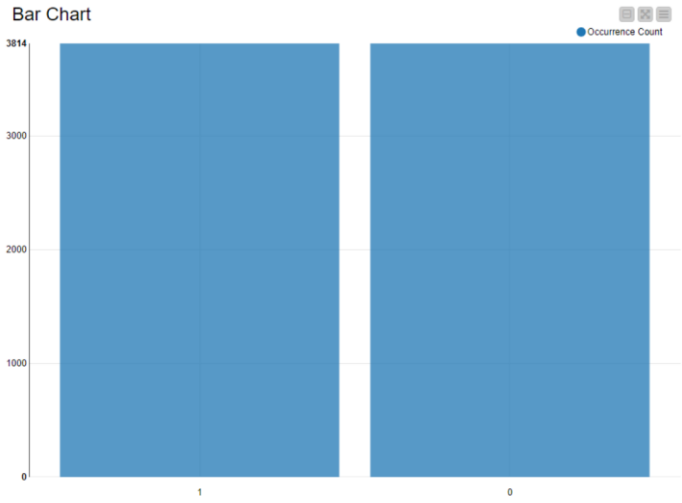

Fig. 8 Represents bar chart of stroke attribute after over sampling

B Additional results

We want to show you the additional results of the work by providing the error rate in each fold of the model in the cross-validation. Error rates of each fold in cross validation in logistic regression and decision tree are presented below.

Row ID	Error in %	Size of Test Set	Error Count
fold 0	18.985	611	116
fold 1	22.586	611	138
fold 2	21.803	610	133
fold 3	22.459	610	137
fold 4	21.475	610	131
fold 5	24.754	610	151
fold 6	19.836	610	121
fold 7	19.344	610	118
fold 8	21.639	610	132
fold 9	21.803	610	133

Table 9 Represents error rates of each fold in logistic regression

Row ID	Error in %	Size of Test Set	Error Count
fold 0	8.511	611	52
fold 1	7.856	611	48
fold 2	9.508	610	58
fold 3	7.377	610	45
fold 4	9.18	610	56
fold 5	9.672	610	59
fold 6	10.328	610	63
fold 7	9.836	610	60
fold 8	8.525	610	52
fold 9	8.852	610	54

Table 10 Represents error rates of each fold in decision tree

Error rates of each fold in cross validation in random forest and KNN are presented as

Row ID	Error in %	Size of Test Set	Error Count
fold 0	4.583	611	28
fold 1	5.728	611	35
fold 2	6.066	610	37
fold 3	6.393	610	39
fold 4	5.574	610	34
fold 5	5.574	610	34
fold 6	5.082	610	31
fold 7	6.721	610	41
fold 8	6.23	610	38
fold 9	4.918	610	30

Table 11 Represents error rates of each fold in random forest

Row ID	Error in %	Size of Test Set	Error Count
fold 0	8.674	611	53
fold 1	12.951	610	79
fold 2	10.328	610	63
fold 3	12.295	610	75
fold 4	10.492	610	64
fold 5	9.656	611	59
fold 6	10.164	610	62
fold 7	8.033	610	49
fold 8	6.393	610	39
fold 9	5.246	610	32

Table 12 Represents error rates of each fold in KNN

Error rates of each fold in cross validation in naïve bayes and SVN are presented as:

Row ID	Error in %	Size of Test Set	Error Count
fold 0	28.805	611	176
fold 1	32.459	610	198
fold 2	32.623	610	199
fold 3	32.787	610	200
fold 4	32.951	610	201
fold 5	13.093	611	80
fold 6	13.607	610	83
fold 7	13.279	610	81
fold 8	13.934	610	85
fold 9	24.262	610	148

Table 13 Represents error rates of each fold in naïve bayes

Row ID	Error in %	Size of Test Set	Error Count
fold 0	0.818	611	5
fold 1	1.803	610	11
fold 2	1.311	610	8
fold 3	1.311	610	8
fold 4	1.148	610	7
fold 5	64.484	611	394
fold 6	76.393	610	466
fold 7	77.541	610	473
fold 8	75.738	610	462
fold 9	76.885	610	469

Table 14 Represents error rates of each fold in SVM.