Stroke Prediction Through Machine Learning

Andrei Gonçalves
andrei.massaini@hotmail.com
ICEI - PUC-MG
Belo Horizonte, Minas Gerais, Brazil

Arthur Kazuo
afuzikawa@sga.pucminas.br
ICEI - PUC-MG
Belo Horizonte, Minas Gerais, Brazil

João Vítor Belchior
jbelchior@sga.pucminas.br
ICEI - PUC-MG
Belo Horizonte, Minas Gerais, Brazil

Matheus Marcolino
m atheus.marcolino.1320642@sga.pucminas.br
ICEI - PUC-MG
Belo Horizonte, Minas Gerais, Brazil

Vinicius Souza
vfsouza@sga.pucminas.br
ICEI - PUC-MG
Belo Horizonte, Minas Gerais, Brazil

Abstract

This article aims to analyze the "Stroke Prediction Dataset" dataset [1], with the aim of identifying relevant characteristics for the prediction of strokes. To do this, we will use a methodology based on Decision Tree and Random Forest. All algorithms were implemented via Python, using mainly the Pandas, SKLearn and MatPlotLib libraries.

*Keywords:*Datasets, AVC, Machine Learning, Decision Tree, Random forest

ACM Reference Format:
Andrei Gonçalves, Artur Kazuo, João Vítor Belchior, Matheus Marcolino, and Vinicius Souza. 2023. Stroke Prediction Using Machine Learning. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA,6pages.https://doi.org/none

1. Introduction

Stroke (Cerebral Vascular Accident) happens when a blood vessel ruptures, leading to a cerebral hemorrhage, or when the blood flow to the region is interrupted. It is responsible for the highest death rate among diseases in Brazil, with around 307 fatalities per day, and the number of victims increases even more when we consider the people who survived with sequelae[two]. Regarding sequelae, around 70% suffer a stroke and do not return to work and 50% of them need help from other people to carry out daily activities[3]. To mitigate this possible damage, it is essential that the diagnosis is made as quickly as possible.

Given the severity of the disease, it is important to be able to predict which profiles and characteristics are most related to the victims, so that it is possible to prevent and treat possible comorbidities before they occur. Furthermore, even if it is not possible to predict, these analyzes can also facilitate a diagnosis during rescue, thus reducing the chances of deaths.

Conference'17, July 2017, Washington, DC, USA
2023. ACM ISBN 978-1-4503-XXXX-X/18/06. . .$15.00
https://doi.org/none

2 Database Description

The database used was theStroke Prediction Dataset [1], which contains information about demographic, health and lifestyle factors of patients who have undergone a medical screening. The dataset consists of 5111 instances, each of which represents a patient.

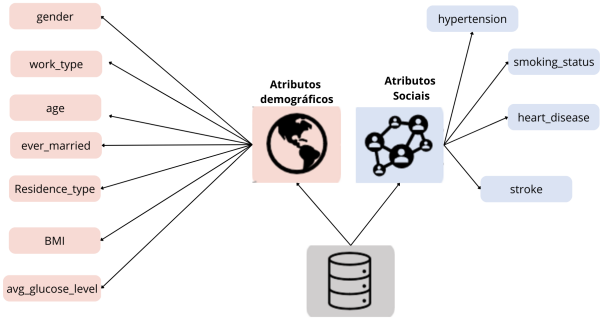

Figure 1.Attributes contained in the dataset

1.Gender:patient's gender (male or female) was coded using the label encoder.
two.Age:patient's age in years
3.Hypertension:Indicates whether the patient has hypertension (1 for yes, 0 for no)
4.Heart disease:Indicates whether the patient has heart disease (1 for yes, 0 for no).
5.Ever married:Indicates whether the patient has ever been married or not (yes or no), it was coded using the label encoder.
6.Work type:Patient's work type (child, Private, Self-employed, No work), was dummified through OneHotEncoder.
7.Residence type:The patient's type of residence (urban or rural) was coded using the label encoder.
8.Avg glucose level:Average blood glucose level in mg/dL
9.BMI:Patient body mass index
10.Smoking status:Patient smoking status (never smoker, ex-smoker, smoker, or unknown) was dummified using OneHotEncoder.

11.stroke:Indicates whether the patient has suffered a stroke (1 for yes, 0 for no)

The attributes of the original dataset are divided into three types of variables:

1.Categorical variables:gender, ever married, work type, residence type and smoking status. These variables describe a patient characteristic that cannot be quantified in terms of quantity, but rather in terms of categories.

two.Continuous numeric variables:age, avg glucose level and bmi. These variables are quantitative and represent numerical measurements that vary continuously.

3.Binary variables:hypertension, heart disease and stroke. These variables are binary, that is, they take on only two possible values: 0 or 1, indicating the presence or absence of a certain medical condition.

3 Pre-Processing

3.1 Missing Values

The first stage of pre-processing consisted of identifying and treating missing values in the dataset. In total, we found 201 instances that had missing values in the 'bmi' column, as seen in the image below:

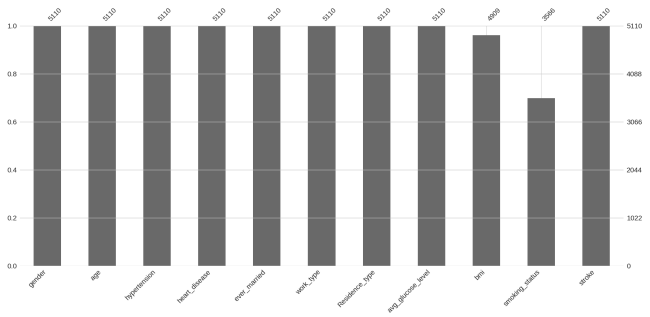

Figure 2.Proportion of missing data per column:

To deal with missing values, we use the Simple Imputer algorithm from the scikit-learn (sklearn) library. We chose the imputation strategy that consists of calculating the average of the non-missing values of the 'bmi' column in relation to the other instances of the data set.

```
fromsklearn.imputeimportSimpleImputer
imputer = SimpleImputer(strategy='mean')
imputer. fit_transform
df [['bmi']] = imputer. fit_transform (df [['bmi']])
```

3.2 Variable coding:

After this step, we proceed to coding the variables. As there was no attribute that expressed an order, per se, we applied the Label Encoder to the entire data set:

```
fromsklearnimportpreprocessing
label_encoder=preprocessing.LabelEncoder()
df=df.apply(label_encoder.fit_transform)
```

3.3 Analyzing Correlations:

After coding the variables, it became feasible to analyze the potential correlations between the input attributes and the class attribute. To perform this analysis, we generate a correlation matrix and establish a threshold or *threshold* of 0.6 in relation to the class attribute.

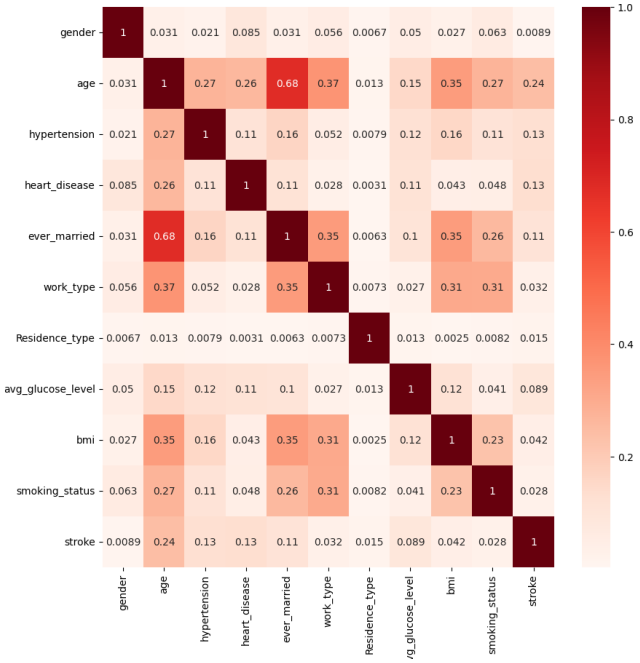

Figure 3.Correlation Matrix

By analyzing the correlation matrix, we came to the conclusion that there is no significant correlation between the input attributes and the class attribute. Therefore, it was not necessary to remove any column from the dataset.

3.4 Balancing

The last stage of pre-processing consisted of treating the evident imbalance in the database, as can be seen in Figure 4 below:

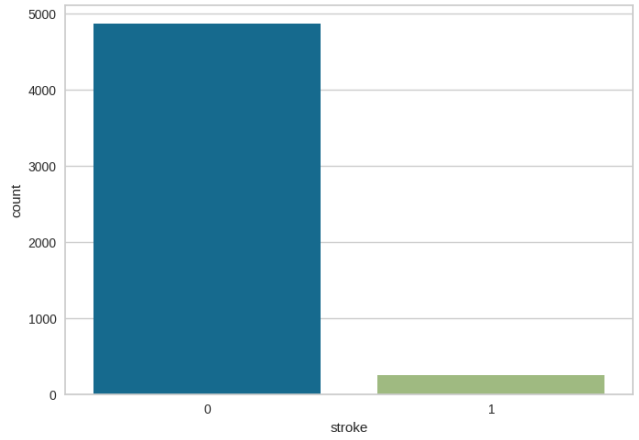

Figure 4.Original distribution

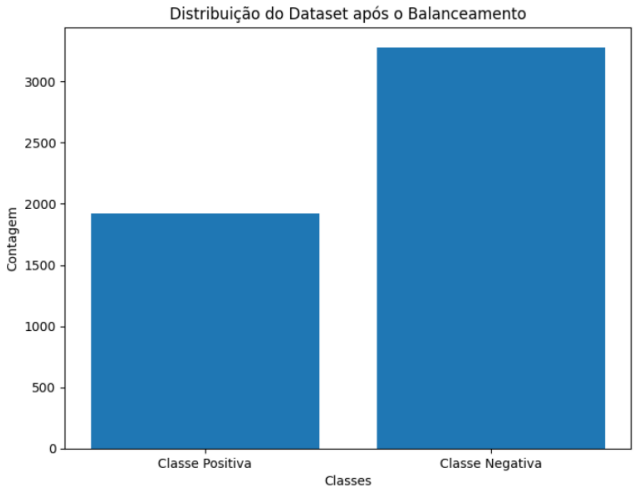

Figure 5.Distribution after balancing.

The technique used to treat the current imbalance was the *SMOTE Tomek*. This technique combines the approaches of *SMOTE (Synthetic Minority Over-sampling Technique)* It is *Tomek Links* to handle class imbalance in datasets.

O *Smote* is used to generate synthetic instances of the minority class, while Tomek Links identifies and removes instances of both classes that are close and considered ambiguous.

The combination of these techniques seeks to create a balanced training set, while improving separability between classes. This approach can be effective in improving the performance of machine learning models in class imbalance problems.

The hyperparameter used in balancing was the "sampling strategy", which is used to define the desired proportion between the minority and majority classes after applying the *Smote Tomek*. It indicates what balance between classes you want to achieve. In our case, we stipulate the value as 0.6, that is, we want to increase the minority class so that it represents 60 percent of the majority class.

Balancing:

```
fromimblearn.MatchimportSMOTETomek os
=SMOTETomek(sampling_strategy=0.6)
```

After balancing, the proportion of positive and negative classes was as follows:

3.5 Division of the test and training set:

The following tables demonstrate how the data set was divided between testing and training:

Dataset	Positive Classes	Negative Classes
Training	174	3403
Test	75	1458

Table 1.Splitting the dataset before balancing.

Training Set	Number of Instances
Positive	1914
Negatives	3276

Table 2.Training set split after balancing.

It is worth noting that balancing was applied only to the training set, both during the cross-evaluation and in the individual evaluation of the models. This way, the test set remained "intact", in order to prevent data leakage. Ensuring that the model is tested and validated only with unknown instances, so to speak.

4 Learning Algorithms used:

4.1 Decision Tree:

Decision Tree is a supervised learning algorithm that builds a prediction model in the form of a tree structure. The tree is composed of internal nodes that represent tests on attributes and leaf nodes that represent classes or output values. The central idea is to divide

the dataset based on selected attributes, such that each split maximizes the homogeneity of the resulting data.

The tree construction process involves selecting the best attribute to divide the data set at each node, using criteria such as information gain or Gini index. This split is repeated recursively on each subset until stopping conditions are reached, such as reaching a minimum number of samples at a node or when there are no more attributes available to split.

Decision trees are attractive for their interpretability and ease of explaining decisions made. However, they can be prone to overfitting if not properly controlled.

4.2 Random Forest:

Random Forest is an ensemble learning technique based on Decision Trees. It creates a collection of individual decision trees, where each tree is built on a random subset of the training data set

The main idea behind Random Forest is to combine predictions from multiple trees to obtain a more accurate and robust final prediction. During training, for each tree, the data subset is randomly sampled with replacement (known as bootstrap sampling). Furthermore, at each node split, only a random subset of attributes is considered to limit the correlation between trees.

During the prediction phase, each tree in the forest produces a prediction and the most frequent class or average of the predictions is chosen as the forest's final prediction.

Random Forests have several advantages, such as being less susceptible to overfitting compared to individual trees, being able to handle large, high-dimensional datasets, and providing an estimate of the importance of attributes to the problem.

5 Testing methodology

To evaluate the performance of both models mentioned above, we stipulate two types of tests:

5.1 Types of assessment:

Individual assessment:We carry out domod training elo using the training data from the original dataset in a balanced way. We then use the validation set to make the prediction and evaluate its performance.

Cross Validation:We performed cross-validation of the model using the K-fold method, where the data set is divided into K folds. Each fold goes through a balancing, training, validation process individual as well as the *Grid Search*, which seeks the best parameters in all *folds*. We calculate the performance metrics for each fold separately and, in addition, we average the metrics across all folds. At the end of the cross-validation, we select the best trained model and use it to predict the instances.

6 Individual assessment:

For the individual assessment, the following hyperparameters were considered in the research:

- *criteria: gini, Entropy*
- *max depth: None, 2, 4, 6, 8, 10*
- *max features: None, sqrt, log2, 0.2, 0.4, 0.6, 0.8*

The respective hyperparameters and metrics obtained were:

Table 3.Best Parameters in individual assessment

Model	Parameter	Value
Random Forest	*criterion*	*gini*
Random Forest	*max_depth*	*None*
Random Forest	*max_features*	*0.6*
Decision Tree	*criterion*	*entropy*
Decision Tree	*max_depth*	*None*
Decision Tree	*max_features*	*None*

Table 4.Metrics obtained individual assessment

Algorithm	Class	Accuracy	Recall	F-score	ROC Area
Random Forest	0	0.96	0.94	0.95	0.20
Random Forest	1	0.17	0.24	0.20	0.79
Decision Tree	0	0.97	0.91	0.93	0.40
Decision Tree	1	0.17	0.37	0.23	0.59

7 Cross Validation

Next, we will see the metrics obtained when cross-validating both models:

7.1 K-Folds

7.1.1 Random forest.In the table below, the hyperparameters used in each of the Random forest folds are available.

Fold	Criterion	Max_Depth	Max_Features
1	*Entropy*	None	0.4
two	*Entropy*	None	0.2
3	*Entropy*	None	0.4
4	Gini	None	Sqrt
5	Gini	None	0.6

Additionally, we collect the average F1 Score, Precision and Recall for the upper and lower classes, as shown in the following table:

Metric	Upper Class	Lower Class
F1 Average Score	0.95	0.18
Average Recall	0.93	0.24
Average Accuracy	0.96	0.15

7.1.2 Decision Tree.The table below shows the hyperparameters used in each fold of the Decision Tree:

Fold	Criterion	Max_Depth	Max_Features
1	Gini	None	log2
two	*Entropy*	None	sqrt
3	*Entropy*	None	0.8
4	Gini	None	log2
5	Gini	None	log2

We also collect the average F1 Score, Precision and Recall for the upper and lower classes, as shown in the following table:

Metric	Upper Class	Lower Class
F1 Average Score	0.94	0.16
Average Recall	0.93	0.2
Average Accuracy	0.96	0.13

7.2 Comparison

After obtaining the results of the Random Forest and Decision Tree models through the cross-validation process with k-folds, we proceeded to compare the performance metrics and the AUC curve (Area Under the Curve).

For a visual analysis of the models' performance, we plot the AUC curves. The AUC graph allows you to evaluate the models' classification capacity at different cutoff points, considering the rates of true positives and false positives. Based on this graph, we can compare the relative performance of the models in terms of their discriminative ability.

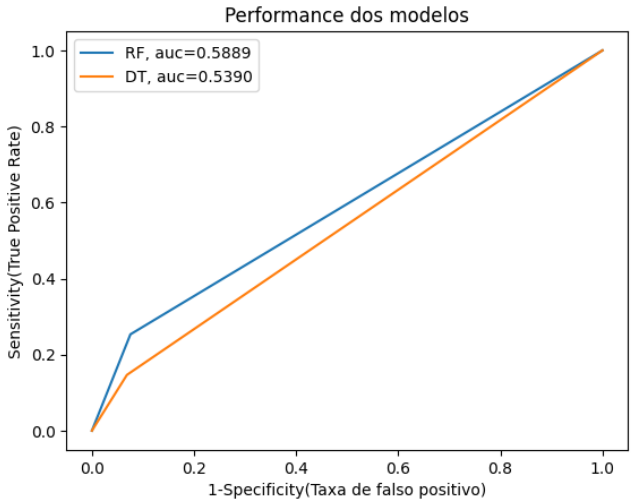

Figure 6.AUC Chart

When comparing the results of Random Forest and Decision Tree in the AUC graph, we observed that Random Forest presented an AUC curve closer to 1 during most of the test, but more similar to Decision Tree at the end. Therefore, we concluded that there was no significant difference between the two.

We also made comparisons based on Recall, Precision and F1-Score metrics between the models, as can be seen below:

Metric	Decision Tree	Random forest
F1 Average Score	0.175	0.125
Average Recall	0.15	0.25
Average Accuracy	0.11	0.15

7.2.1 T Test.The T test was performed to compare the results of the Random Forest and Decision Tree models. The value obtained for the T-value was -1.25, and the critical value provided was 1.96. When comparing these values, we concluded that there is no statistically significant difference between the means of the compared groups. This suggests that the models have similar performances in relation to the evaluated metrics.

8 Conclusions

Based on previous data, we concluded that the database used was not of sufficient quality to obtain satisfactory results in the classification and prediction of stroke cases. This conclusion was reached through analysis of the models' performance, which did not show significant improvements even after applying different treatments. Furthermore, we attribute these problems to the lack of a significant number of samples, given the extreme imbalance of the database. These factors combined

indicate that the results obtained are not reliable and limit the effectiveness of the models in the task of classifying stroke cases.

Although the average results and the T Test did not indicate such large differences between the models, when analyzing each execution individually, the Random forest presented better results.

9 Code and Database

For more details, the source code is available at https://colab.research.google.com/drive/1psTB1H1Y7YLGq4B_HB8n_mZkQOjzsBKA?usp=sharing. Database available at https://www.kaggle.com/datasets/fedesoriano/stroke-predictiondataset

References

[1] F. Soriano, "Stroke prediction dataset," Jan 2021. [Online]. Available: https://www.kaggle.com/datasets/fedesoriano/stroke-predictiondataset

[2] M. da Saúde, "Acidente vascular cerebro," 2023. [Online]. Available: https://www.gov.br/saude/pt-br/assuntos/saude-de-aaz/a/avc

[3] Pfizer, "What is a stroke, what types are there, how to prevent and treat it," 2019. [Online]. Available:https://www.pfizer.com.br/noticias/ultimas-noticias/o-que-eacidente-vascular-cerebral-AVC-tipos-prevencao-tratamento