Predicting Stroke Risk With Classification Models

Kayla Wright, Ethan Schmitt, Lauren Taylor

Abstract

We compared the accuracy of two classification models (logistic regression and decision tree), on predicting stroke risk in patients. For the logistic regression model we obtained approximately 89% percent test accuracy. For the decision tree we found a recall of 95%, an accuracy of 56%, a precision score of 10%, and a F1 score of 19%. Our conclusion was that a decision tree and logistic regression were able to obtain reasonable accuracy and recall. However, a larger dataset with more diverse predictors are needed such as: MRI results, EEG data, and a more accurate smoking history (such as years smoked and packs per day).

Introduction

A stroke is a medical condition where the supply of blood to some part of the brain is cut off or when an artery in the brain suffers a rupture [1]. Developing a systematic way to determine a person's predisposition to this issue is frequently sought after. There have been many papers released on the topic trying all different kinds of methods but what makes this problem well suited to a machine learning solution is the wide availability of data.

Some algorithms used in the past are as simple as logistic regression [2], all the way up to a full deep learning model [3]. Using these various models, they were able to quantify the relationship between various inputs to each other and to the probability of each patient having a stroke. The study done by Tazin et al. focused on applying different machine learning techniques on the same dataset [4] that we ended up using in our own study. They looked at using a logistic regression, both decision tree and random forest classification, and voting classifier. We decided in our own study to focus on and compare a logistic regression and decision tree classification model, the details of which will be further discussed in the methods section. The models they generated had accuracies listed below in figure 1 for reference.

Random forest	96
Decision tree	94
Voting classifier	91
Logistic regression	79

Figure 1: Previous Paper Results [2]

Most of these models are fairly accurate but in the case of a stroke predicting model we do not only want to minimize error, we want to specifically

remove as many false negatives as possible since false positives have much less severe consequences than incorrectly telling a patient they are not at risk for a stroke. We decided to make this a focus for tuning our models in our study.

The dataset we chose included variables like age, gender, and bmi but also common risk factors of a stroke like hypertension and smoking which both appear in the chart below which is a graph of the most common risk factors associated with a stroke.

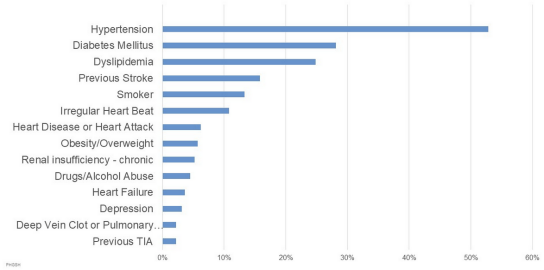


Figure 2: Stroke Risk Factors [5]

These factors are typically linked due to the effects they have on blood flow. Due to this, we could reasonably expect to see high correlation between hypertension and smoking and the output of whether or not the person had a stroke.

Methods

For our study we chose the Stroke Prediction Dataset on Kaggle by fedesoriano [4]. We chose this set because it has very clean input and output data that can make it much easier to generate prediction models. We chose to create our models in python because of the easy usability available with the packages like Sklearn and Pandas which allow us to work with the data and create our models with relative ease.

The dataset was fairly clean, however it did have some issues. An example is that the output is quite imbalanced in favor of no stroke. This means that we had many more examples of inputs that resulted in patients who were unlikely to have a stroke rather than inputs that yielded a patient likely to have a stroke. The figure below shows just how significant the differential is, where 0 is patients who did not have a stroke and 1 is patients who did.

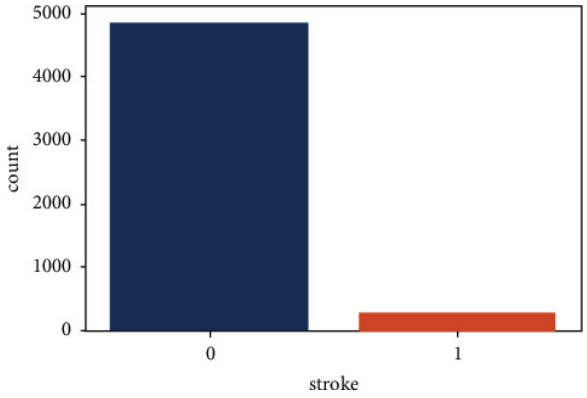


Figure 3: Distribution of Output [2]

We solved this issue using the SMOTE algorithm, the details of which are discussed a bit later on.

Another issue was that some of the metrics we expected to have high correlation to the output actually ended up not being significant in the logistic regression model according to the p-values. This fix is rather simple, those variables that had high p-values, which surprisingly were most notably

hypertension and heart disease, can be removed from the model as they only lead to overfitting.

The last issue we encountered was that a couple inputs were missing a value for BMI. After observing the distribution of the BMI we determined the best course of action would be to impute the median for these values.In addition to this, we decided to remove the ID column as it is completely uncorrelated to the output.

Decision Tree

The decision tree needed to not be overfit, since that is a large concern with such a model. This model was split on a 70/30 train/test to give enough learning and testing for the tree. First, we attempted a bare decision tree with only the Gini Index as an impurity measure. The following predictors were used:

- Age
- Gender
- Hypertension
- Heart disease
- Ever married
- Work type
- Average glucose level
- BMI

After looking at the metrics, a second decision tree was used with GridSearchCV to optimize recall (with the same predictors). Optimizing recall score helps the model reduce false negatives, which tell patients they will not have a stroke when they actually do. This can be very costly to patient lives and this study would like to avoid false negatives as much as possible. The class weights were also adjusted for GridSearchCV due to an imbalanced dataset of stroke 5% and no stroke 95%, class weights were: class_weight = {0:0.05, 1:0.95}. GridSearchCV came up with the following parameters:

- criterion = “entropy”
- max_depth = 5
- min_impurity_decrease = 0.01
- random_state = 1
- splitter = “random”

Next, all scores (recall, accuracy, F1, and precision) were carefully considered in order to understand why the model is performing in such a way given the dataset and parameters. This will be cross referenced with the logistic regression model that we make as well.

Logistic Regression

For our logistic regression we used the Python package Statsmodels for training the regression. For splitting training and testing data, preprocessing, and performance metrics we imported functions from Sklearn.

In order to account for the imbalanced data set, we used and algorithm known as Synthetic Minority Oversampling Technique (SMOTE). The algorithm works by oversampling the minority class by creating new synthetic data points based on existing instances. While this has the advantage of addressing imbalance data it does have

the drawback of higher false positive and/or negative rates [5].

To implement this in our work we used the package Imblearn's SMOTE algorithm.

We chose the predictors that differed between stroke/no stroke groups:

- Gender
- Residence type
- Ever married
- Gender
- Average glucose level
- Age
- Smoking Status

We set aside 20% percent of the data to be used for model testing

Results

Decision Tree Results

The first decision tree's results are:

```
Training performance:
   Accuracy  Recall  Precision  F1 Score
0       1.0     1.0        1.0       1.0
```

Figure 4: First decision tree training results

```
Testing performance:
   Accuracy    Recall  Precision  F1 Score
0  0.910633  0.120482   0.135135  0.127389
```

Figure 5: First decision tree testing results

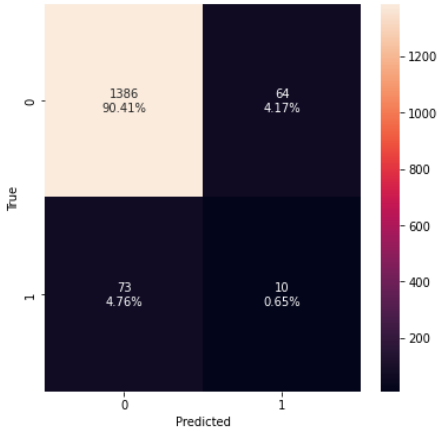


Figure 6: First decision tree confusion matrix for test set

This model is clearly overfit: since all F1, recall, precision, and accuracy scores are 1 for the training set and much lower for the testing set. Decision trees are a model that can easily be overfit due to no limit on the branches they can make. They can easily overlearn the training data and reflect that in the test set, such as in this first model shown below.

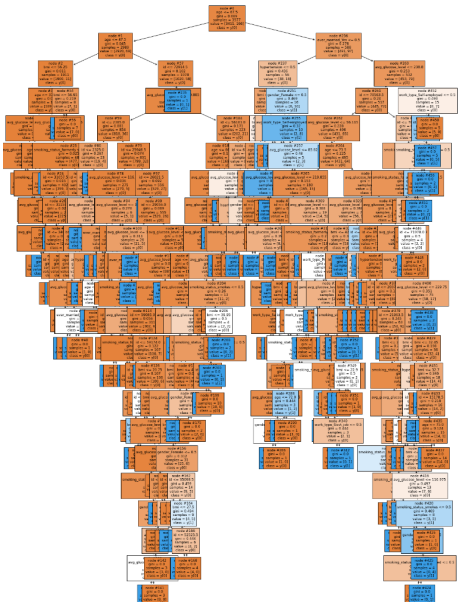


Figure 7: Overfit first decision tree (please see attached notebook for a better view)

Figure 7 is clearly overfit to the training set due to the sheer amount of branches. This causes the tree to overlearn the noise inside of the training set and perform less optimally on the test set.

The second GridSearchCV decision tree's results are:

```
Training performance:
   Accuracy    Recall  Precision  F1 Score
0  0.533687  0.951807   0.086909  0.159274
```

Figure 8: Training performance on second decision tree

```
Testing performance:
   Accuracy    Recall  Precision  F1 Score
0  0.563601  0.951807   0.106183  0.191052
```

Figure 9: Testing performance on second decision tree

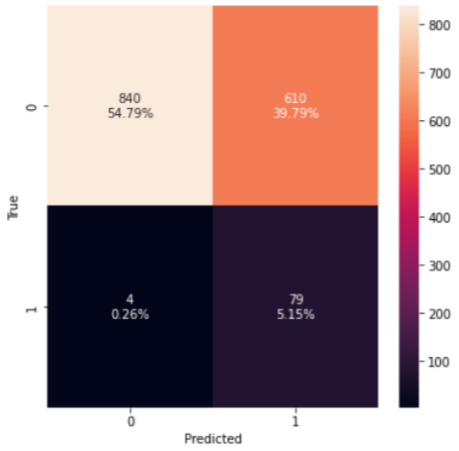


Figure 10: Second decision tree confusion matrix for test set

This model is clearly not overfit due to the fact that in all scores, the testing set outperforms the training set, even with just a small amount of data. The low recall score is what we aimed for and it came to about 95% recall. As seen below, the tree is much less complex and gives better results.

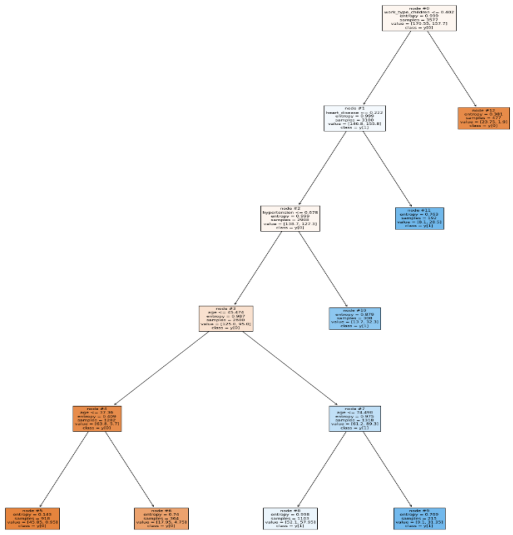


Figure 11: Second GridSearchCV decision tree (please see attached notebook for a better view)

Logistic Regression Results

Our results from the logistic regression are as follows:

Logit Regression Results

Dep. Variable:	stroke	No. Observations:	7518
Model:	Logit	Df Residuals:	7509
Method:	MLE	Df Model:	8
Date:	Fri, 02 Dec 2022	Pseudo R-squ.:	0.6421
Time:	10:32:55	Log-Likelihood:	-1864.9
converged:	True	LL-Null:	-5211.0
Covariance Type:	nonrobust	LLR p-value:	0.000

	coef	std err	z	P>|z|	[0.025	0.975]
x1	-0.5292	0.043	-12.227	0.000	-0.614	-0.444
x2	2.0459	0.070	29.255	0.000	1.909	2.183
x3	-0.3070	0.053	-5.744	0.000	-0.412	-0.202
x4	-0.5013	0.042	-11.859	0.000	-0.584	-0.418
x5	0.2719	0.040	6.770	0.000	0.193	0.351
x6	-2.4643	0.086	-28.819	0.000	-2.632	-2.297
x7	-1.7327	0.055	-31.700	0.000	-1.840	-1.626
x8	-2.1925	0.068	-32.037	0.000	-2.327	-2.058
x9	-1.4938	0.051	-29.528	0.000	-1.593	-1.395

Figure 12: Logistic Regression Results

As illustrated from the p-values, all of the chosen predictors were of significance.

The three predictors that had the most weight in the model were

age(positive weight), smoking status unknown (negative weight), and smoking status, never smokes (negative weight).

Age and never smoked make immediate sense as those are risks and protective factors respectively. One mystery is smoking status unknown. In order to get to the bottom of this we would need to look closer at the group of unknown smoking status, and see if they vary in a systematic way.

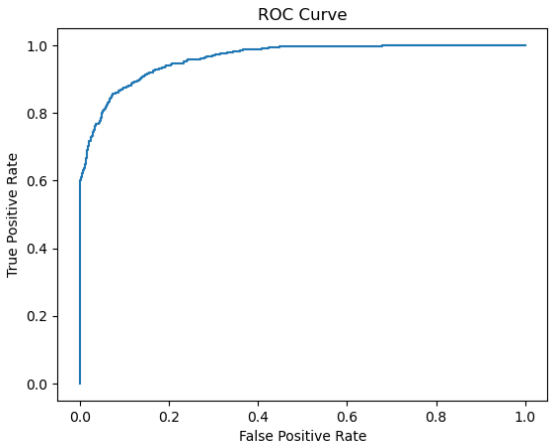
The training accuracy of the model was approximately 88%. Test accuracy was very similar at 89%. This indicates that the model was unlikely to be overfit.

Below is our confusion matrix:

		Predicted	
		Positive	Negative
Actual	Positive	856	103
	Negative	99	822

From this matrix we can calculate that the false positive and false negative rates are both around 10.7%. These rates might be driven by the nature of the SMOTE algorithm.

The ROC curve corresponding to the model is as follows:



Visually, looking at the graph we can see that the model has good predictive accuracy. We can have a relatively high true positive rate without an excessively high false positive rate. Calculating the AUC score, which was 0.96 confirms the predictive power of this model.

Discussion/Conclusion

For our models, we decided that false negatives should be considered more problematic than false positives. In a clinical setting this could be used to assess an individual's risk of developing a stroke.

Our logistic regression model seemed to be most refined for accuracy, as shown by the 89% accuracy opposed to the about 56% accuracy that our decision tree had. This was a conscious effort, however, because GridSearchCV was able to help refine parameters that we needed to lower recall as much as possible. Getting the tree to have such a low recall required a lot of tuning. We ultimately decided that we needed to

have a model optimized for potentially life-threatening false negatives.

If the model determines that the person is at risk for stroke the doctor can suggest some interventions (eg. blood sugar management, smoking cessation), that can help lower their risk.

If the interventions are the aforementioned lifestyle changes there can be less harm and possibly even benefits if one implements those interventions in the absence of high risk of stroke.

However, if the model falsely categorizes them as low risk there could be harm done to the patient if they fail to implement changes.

Overall there is high utility in being able to predict which individuals are at risk for disease. If we are able to accurately make such predictions it can inform preventative treatments and interventions that can be implemented in clinical settings.

In order to refine the model new predictors could be introduced. Some particular classes of predictors could be MRI data, daily blood sugar measurements, daily blood pressure measurements, smoking severity (packs per day and for how long). Such predictors can refine the model in terms of it having more data to rely on for predictions. Doctors can use this amalgamation of data in the future to predict and prevent strokes.

A detrimental limitation to this study was the length of the dataset, with only 5110 entries, the models were not able to learn the intricacies of real world data in a way that it could. Overall, the models performed well, and are a great stepping stone for what the future can hold.

References

[1] About Stroke. Retrieved December 5, 2022, from https://www.cdc.gov/stroke/about.htm

[2] Tazin, T., Alam, M. N., Dola, N. N., Bari, M. S., Bourouis, S., & Monirujjaman Khan, M. (2021). Stroke Disease Detection and Prediction Using Robust Learning Approaches. *Journal of healthcare engineering*, *2021*, 7633381. https://doi.org/10.1155/2021/7633381

[3] Early Stroke Prediction Methods for Prevention of Strokes. (2022, April 11). Retrieved December 5, 2022, from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9017592/

[4] Stroke Prediction Dataset. (2020). Retrieved December 5, 2022, from https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset

[5] Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-sampling Technique. Journal of Artificial Intelligence Research, 16(16), 321–357. https://doi.org/10.1613/jair.953