Stroke Prediction using Machine Learning Algorithms

David Kristensen and **Natnael Mulat**
{dakristensen,namulat}@davidson.edu
Davidson College
Davidson, NC 28035
U.S.A.

Abstract

Identifying and classifying stroke is a useful process that machine learning can facilitate and benefit the health care industry to save thousands of lives. In this research, we attempted to create a classification model that could accurately predict if a patient will have a stroke given several medical information about their health. We tuned and evaluated five types of classification models: a logistic regression, adaboost classifer, random forest classifier, support vector classifier, and multilayer perceptron neural network. These classifiers achieved test set F1 scores of 0.8852, 0.7457, 0.5147, 0.9333 and 0.8695, respectively on experiment where we used all the data retrieved from NIH.

1 Introduction

Someone in the United States has a stroke every 40 seconds. Every 4 minutes, someone dies of stroke. Every year, more than 795,000 people in the United States have a stroke. About 610,000 of these are first or new strokes (CDC 2021). For that reason, being able to predict who is more likely to have a stroke, will be extremely helpful in order to help vulnerable people avoid strokes. Our goal is to create a model that predict if someone will have stroke given relevant features for any given person with a highest possible precision. In order to develop these models, we used a dataset from Cardiovascular Health Study (NIH 2011), which has profiles for 6800 individuals and their respective medical information such as age, gender, BMI, diabetes, smoker, and 64 other features.

The accurate prediction of a disease outcome is one of the most interesting and challenging tasks for physicians. As a result, machine learning methods have become a popular tool for medical researchers. These techniques can discover and identify patterns in complex datasets which allows for the accurate predictions of future potential stroke patients.

This problem has been studied by Aditya Khosla Dept. of Computer Science Stanford University using similar dataset also from CHS (Khosla et al. 2010). In their work, they mainly focused on different feature selections to classify the potential stroke patients using support vector classification (SVC), MisClassification Rate (MCR) and Cox proportional hazards model (Cox). They found that their conservative mean (CM) feature selection outperformed another study's 16 features. They also found that the MCR generally outperformed the other models, making their best model MCR + CM feature selction yielding an AUC score of 0.777.

Similar to Khosla's work, we developed supervised machine learning models to predict potential stroke patients using the same features. However, rather than focusing mainly on the feature selection aspect we focused on different preprocessing techniques mentioned in next paragraph and creating different models that we thought would have a better chance at handling the higher number of features.

The rest of this paper will show the various data preprocessing techniques that we employed, the process of tuning model hyperparameters, and the evaluation of several different classification models to determine which model classifies the potential stroke patients most accurately.

2 Data Preparation

The dataset consisted of 6800 samples each described by 69 features. In order to prepare the data for developing models, we first shuffled it in order to remove any potential patterns existing in the data and then split it into 80% training set and 20% test set. Then we performed min-max scaling so that it converges faster when ran models.

The first thing we noticed about our data was the large imbalance in the stroke values, as far fewer individuals have had a stroke than people who have not. This is illustrated in figure 1 where you can see only 321 out of our 6479 cases or 4.9% have had stroke. This we found potentially problematic as our models will have a greater incentive just to predict no stroke rather than making a valid estimate.

In the first experiment, we ran the different models without changing the train set, but instead by tuning the hyper parameter of the model. Then, we ran two more experiments where we remove 25% and 50% of the negative cases from the training set in order to reduce the class imbalance. After removing some of the negative cases from the training set, we ran the models.

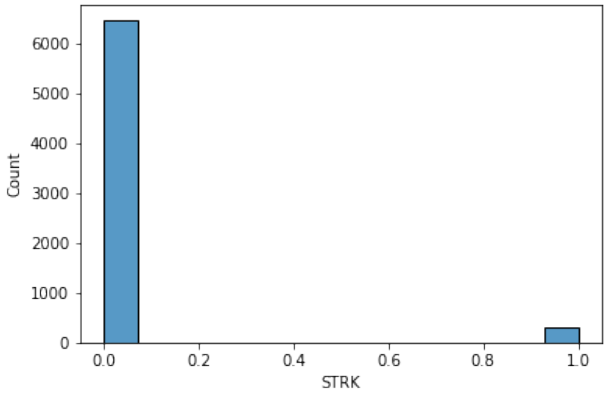

Figure 1: Bar graph showing the number of negative and positive samples in the dataset

For our last experiment, we tried using a different technique called Synthetic Minority Oversampling Technique(SMOTE). Essentially, SMOTE synthesises new minority instances between existing minority instances. Specifically, the SMOTE algorithm first sets the minority class set A for each $x \in A$. The k-nearest neighbours of x are then obtained by calculating the Euclidean distance between x and every other sample in set A. The sampling rate N is set according to the imbalance proportion. For each $x \in A$, N examples are randomly selected from its k-nearest neighbours and make up set A_1(Chawala et al. 2002). For each example, $x_k \in A_1$ where $k = 1, 2, 3 \ldots N$, we generate new examples using the formula below:

$$x^{'} = x + (\alpha|x - x_k|)$$

where α represents the random number between 0 and 1. By using SMOTE, the classes are balanced, and there is no loss of information. And according to Chawala et al, SMOTE alleviates over-fitting caused by random oversampling as synthetic examples are generated rather than replication of instances.

3 Experiments

To develop an optimal classification model, we ran five different experiments by varying the training set and applying different classification models. In each experiment, we used grid search with 5-fold cross validation to tune the hyper-parameter of each models. Overall, we ran logistic regression(LR), random forest classifier(RFC), support vector classifier(SVC), adaptive boosting classifier(ADAC), and multilayer perceptron neural network(MLP).

For our research, since increasing the true positive rate (TP), and decreasing false negative rate (FN) is imperative, we decided to use a modified version of F1-score that gives more weight to recall than precision in the calculation of F1-score. The full definition of the F-measure is given as follows(Sasaki 2007).

$$F_\beta = \frac{(\beta^2 + 1)P * R}{\beta^2 P + R}$$

β is a parameter that controls a balance between precision, P, and recall R. When $\beta = 1$, $F1$ comes to be equivalent to the harmonic mean of P and R. If $\beta > 1$, F becomes more recall-oriented and if $\beta < 1$, it becomes more precision-oriented. The 5-fold cross validation grid searches attempted to find the hyper-parameters that yielded the highest $F1_\beta$ scores for each model and experiment with $\beta = 2$ to give more emphasis to recall.

The following experiments focused on tuning the hyper-parameters of each of these models with grid searches and comparing the efficacy of each model.

Experiment 1

In the first experiment, we used the training set as given without any attempt to balance the data. In this experiment, we ran LR, RFC, SVC, ADAC, and MLP. The hyper-parameters of each model were tuned using a grid search over different sets of values (see Section 7). Each grid search used $F1_\beta$ scores to compare models with β set to 2. Additionally, the grid searches used 5-fold cross-validation to find an average measure of performance in each model that was fitted. The mean $F1_\beta$ scores were measured on the cross-validation folds.

In LR, we tuned the class weight since we have a severe imbalance between the classes of the data. In addition, we also tuned the regularization strength of $l2$ regularization term to account for over-fitting. Similarly, we tuned the class weight in RFC in addition to the number of estimators and the criterion. The SVC models were tuned over five different hyper-parameters: kernel, C, degree, gamma, and coef0. The degree is only for the polynomial kernel of SVC, while gamma, and coef0 and the inverses regularization strength, C, are the terms in the specific kernels used in hyper parameter tuning.

For ADAC, we tuned both the number of estimators and learning rate. Finally, for MLP, we tuned the hidden layer sizes, which is the ith element represents the number of neurons in the ith hidden layer, activation function for the layers, solver for weight optimization, alpha value for $l1$ regularization parameter, and the learning rate schedule for weight updates. These grid searches attempted to find the hyperparameters that yielded the highest $F1_\beta$ scores for each model.

Experiment 2

On experiment 2, we decreased the majority class from the training set without touching the test set to fairly balance out the imbalance of the class. In this experiment, we took out 25% of majority class from the training set. As experiment 1, we ran LR, RFC, SVC, ADAC, and MLP models with the specified parameters in Section 7.

Experiment 3

In this experiment, we took out 50% of majority training set and similar to experiment 1 and experiment 2, we ran LR, RFC, SVC, ADAC, and MLP models with the specified parameters in Section 7.

Experiment 4

In experiment 4, we used SMOTE algorithm to oversampled the minority class. With SMOTE, both categories have equal amount of records. More specifically, the minority class has been increased to the total number of majority class. Since in this case, the classes appear to be equal, we removed the class weight when we ran LR and RFC, but besides that we ran all the models developed in experiment 1, 2, and 3 with specified hyper parameters in section 7. Note that, we used SMOTE on the training set only instead of the test so that we can test result in a real data as opposed to a replicated one.

Experiment 5

In the last experiment, we compared the $F1_\beta$ scores of each tuned model and the individual $F1_\beta$ scores for each class, as well as how our models compared to Khoslas'model in terms of AUC score. The $F1_\beta$ and AUC score were evaluated on the test set. By calculating the $F1_\beta$ scores, we were able to determine which model performed the best. Then, we analyzed whether the models were overfit and by how much in terms of $F1_\beta$ score differences between the train and test sets.

4 Results

Results of Experiment 1, 2, and 3

By running different experiments and with the help of 5-fold grid-search, we got the best $F1_\beta$ score for each model in experiment 1 where we used the original train-test split data, experiment 2 where we removed 25% of the majority class in the train set, and experiment 3 where we removed 50% of the majority class in training set. The results of these experiment is summarized in Figure 2.

The $F1_\beta$ score of LR model in experiment 1,2, and 3, slightly decreased as we removed some of the majority instances from the training set. The score decreased from 0.904 in experiment 1 to 0.8992 in experiment 2, and finally experiment 3 has a $F1_\beta$ score of 0.8964. This result seem to indicate that by removing some of instances, there is a loss of information that can help the LR model predict better. Interestingly, the best set of parameter for LR in experiment 1 and 2 are the same with regularization term set to $10,000$, and with class weight of $1:4$, where positive cases of *stroke* label are given weight of 4 and negative cases of *stroke* are given a weight of 1. But in experiment 3, since the class imbalance is better than that of experiment 1 and 2, grid search determined that a weight of $1:1$ with regularization term set to 10000 had the best result. Figure 3 summarizes the relationship between class weight and the regularization term in LR. In Figure 2, the regularization term

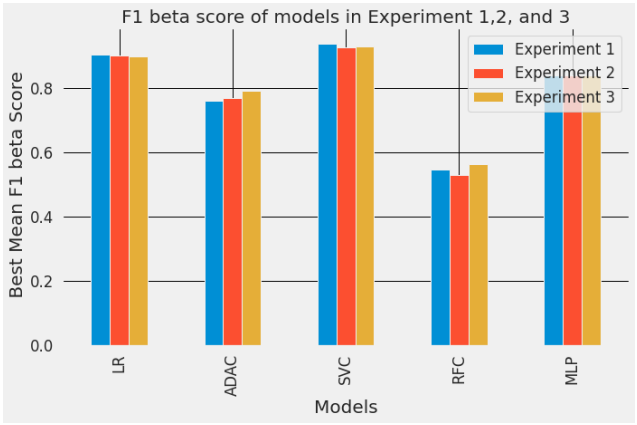

Figure 2: Bar graph summarizing the results of experiment 1, 2, and 3 using $F1_\beta$score. 5 models: LR, ADAC, SVC, RFC, and MLP for each experiment.

plateaued at $10,000$ for experiment 1 and 2, while for experiment 3, it takes longer to to reach that plateau and lower regularization term yield lower $F1_\beta$ score, this maybe due to the smaller size of the training sample in experiment 3, the more prone the model is to overfitting, thus a larger regularization term is required to overcome that overfit. Another observation Figure 2 is that large class weight like $1:50$ seem to have lower results for a smaller regularization terms and take larger regularization term to reach a plateau.

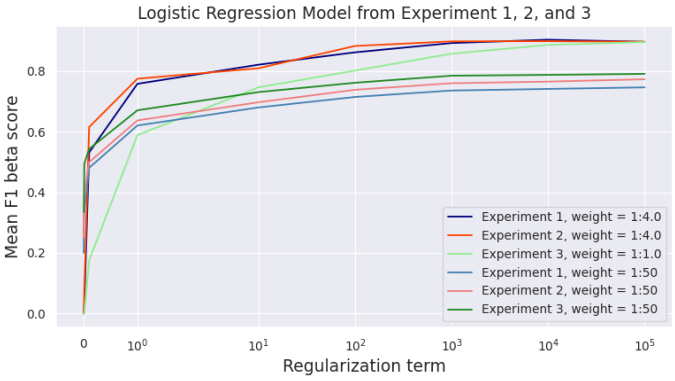

Figure 3: Line chart showing how different regularization terms and class weight give different $F1\beta$ scores in Experiment 1,2, and 3 of LR models.

In Figure 2, ADAC models seem to have a better $F1_\beta$ score as majority instance of positive cases of *stroke* are removed. In experiment 1, grid search got the best result with parameters of learning rate set to a high 1.5 and 1000 number of estimators. These set of parameter yield $F1_\beta$ score of 0.7589. In experiment 2, the $F1_\beta$ score increased to 0.768 with parameter of 1.5 learning rate and 2500 number of estimators. In experiment 3, we see the same trend where the best set of parameter is with learning rate of 1.5 and the number of estimators set to 3000. Interestingly, these models seem to have better $F1_\beta$ score as the learning rate is at 1.5,

which is really high considering the small step size usually chosen for convergence.

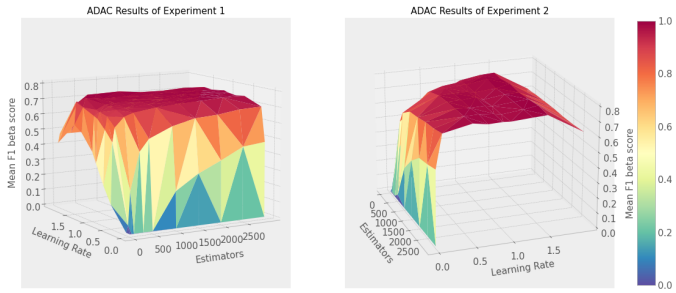

Figure 4: ADAC result of Experiment 1 and 2 summarized with number of estimators, learning rate with mean $F1_\beta$ score

The best $F1_\beta$ score out of all the models that were ran in each experiment is SVC with linear kernel parameter. Experiment 1, with parameters set to linear kernel, with class weight of 1 : 2, where positive cases of *stroke* are given weight of 2 and negative cases of *stroke* are given a weight of 1 yield the highest $F1_\beta$ of 0.9376. Further, Coef0 and Gamma were set to .00001. This score slightly decreases in experiment 2 and 3, but the change is not as significant that experiment 2 and 3 have fairly the same results. As it can be seen from the heat-map on Figure 5, linear kernel seem to suit this dataset better than the other kernels ran in grid search.

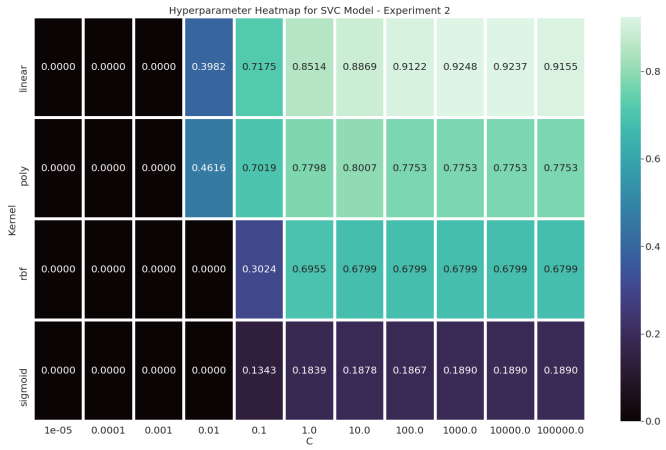

Figure 5: Heat map of $F1_\beta$ score of each Kernel and regularization term used in grid search. Each mean $F1_\beta$ score is from 5-fold cross-validation

In addition to LR, ADAC, and SVC, we also ran RFC, but $F1_\beta$ score we got by running grid search was significantly lower than the other models we ran. This maybe due to the fact that the target variable is linearly separable, and does not require complex models to achieve a good score(Kirasich, Smith, and Sadler 2018). Experiment 3 has the highest RFC $F1_\beta$ score of 0.5627 with parameters class weight set to 1 : 5 with criterion of entropy, and with number of estimators set to 100. These best set of parameter of RFC is almost the same as Experiment 1's best set of parameter, except that the class weight is set to 1 : 1 instead of 1 : 5. In Experiment 2 the best $F1_\beta$ score has the exact same set of parameters as the best RFC model in Experiment 1, but with a slightly lower $F1_\beta$ score of 0.5282. Figure 6 shows the results of Experiment 1, and the plateau at 100 estimators with different class weight tuning.

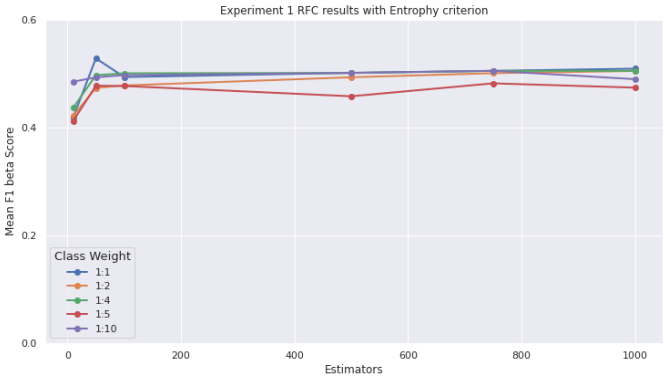

Figure 6: Line graph of class weight results with number of estimators evaluated with $F1_\beta$ score for experiment 1

The grid search for MLP model in Experiment 1 determined activation of logistic with alpha value of $1e{-}05$ with 3 hidden layers each with 50, 75, and 100 neurons using adaptive learning rate and adam for solver achieved $F1_\beta$ score of 0.8383. In Experiment 2, we see a very slight increase in $F1_\beta$ score to 0.8399 with activation of tanh, alpha value of 0.01, 3 hidden layers each with 50, 75, and 100 neurons using adaptive learning rate, and adam solver. In Experiment 3, we got the same set of parameters with $F1_\beta$ score of 0.8387.

Results of Experiment 4

The results of Experiment 4 using SMOTE is better in term of $F1_\beta$ than that of other other experiments we ran on the previous section. The results are summarized on Table 1.

	LR	ADAC	SVC	RFC	MLP
Mean $F1_\beta$	0.9803	0.9786	0.9996	0.9944	0.9995

Table 1 : Mean $F1_\beta$ of Experiment 4 best models ran using LR, ADAC, SVC, RFC, and MLP

The mean $F1_\beta$ of each model is almost equal, with SVC having a marginally better result than the other models, which is similar to the previous experiments. In LR and RFC, we decided to remove the class weight since the classes are balanced. The regularization term in LR seem to plateau at 100, which is significantly lower than the other experiments, this is seen on Figure 7.

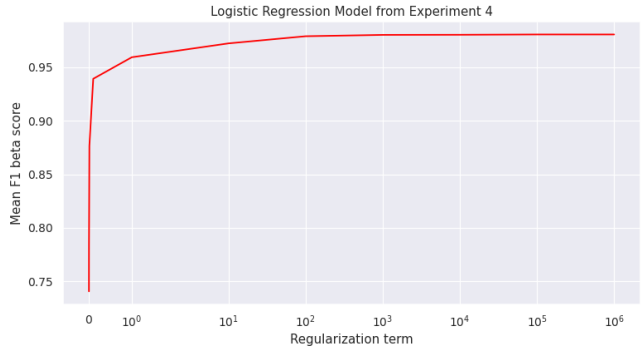

Figure 7: Line graph showing how different regularization term affect the mean $F1\_beta$ scores for experiment 4

In ADAC, as we saw in previous experiments, the best models found using grid search had a high learning rate of 1.5, but in experiment 4, the best model has a relatively low learning rate of 0.1. This maybe due to the nature of the data, as there is more information to learn from, and so a learning rate that is high might not get good result since it will miss out in learning relevant information from the data. But in the previous case since majority of the target labels were negative *stroke* label, the model can afford to skip some information to get to an optimal model. The number of estimators is set to 1000 in the best model, which is the same number of estimators as experiment 1. The results of the grid search for ADAC is summarized on Figure 8.

Figure 8: Line graph showing how different regularization term affect the mean $F1_\beta$ scores for experiment 4

In SVC, we noticed that the kernel changed for the best model from *linear* in experiment 1,2, and 3 to rbf in experiment 4. Further, the regularization term seem to plateau at 10 producing the same $F1_\beta$ score as the regularization term increases while holding the kernel at rbf. No class weight was added to these models.

In RFC, the best set of parameters included a criterion of *gini* with number of estimators set to 1000. From the previous experiments, the number of estimators seem to have increased since we have a relatively bigger training set. From the best set of parameters in the previous experiment for MLP, the only paramter that had a different value was the activation, which instead of *logistic* is set to *relu*.

Results of Experiment 5

After fitting each model with their tuned hyperparameters on the training set, the $F1_\beta$ scores for each model were calculated on the test set. Each score of experiments model is summarized in the table below.

Experiments	1	2	3	4
LR	0.8852	0.8852	0.8803	0.8598
ADAC	0.7457	0.7885	0.7755	0.7095
SVC	0.9333	0.918	0.9333	0.677
RFC	0.5147	0.5474	0.5956	0.4814
MLP	0.8695	0.8415	0.7966	0.7627

Table 2 : Test $F1_\beta$ of Experiments using LR, ADAC, SVC, RFC, and MLP

In addition, comparing our results with Aditya Khosla, we found that experiment 1,2, and 3 had an area under a ROC curve of 0.94 - 0.95 for the best model in each experiment, SVC. Khosla's models of SVC has a ROC curver of 0.747. Our RFC model in experiment 1,2, 3, and 4 are comparable to the area under a ROC curve of Khosal's with 0.7295, 0.7459, 0.7704, and 0.7131 respectively. Overall except for RFC, all the models developed in this paper outperform that of Khosla's in terms of area under a ROC curve.

Experiments	1	2	3	4
LR	0.8852	0.8852	0.8688	0.8852
ADAC	0.7213	0.7704	0.7704	0.7049
SVC	0.918	0.918	0.918	0.6393
RFC	0.459	0.4918	0.5409	0.4262
MLP	0.8524	0.836	0.7704	0.7377

Table 2 : Recall score of Experiments using LR, ADAC, SVC, RFC, and MLP

From Table 2, we can see that experiments 1,2, and 3 had an $F1_\beta$ score that very similar to the results we got in the mean cross validation $F1_\beta$ score. However, comparing our results for experiment 4, we find that the mean cross validation $F1_\beta$ score is much higher than the test $F1_\beta$ score as it can be seen from Table 1 and 2. One possible reason why that is case can be that since supervised learning relies on the fact that training and test data follow the same distribution. By using SMOTE, we have changed the distribution of the training set as such the models perform well in training data but does not on test data. Another plausible reason is that using SMOTE overfits the models, giving us a much lower result on the test set. Figure 9 shows comparison between the mean cross validation $F1_\beta$ score and the test $F1_\beta$ score. Interestingly, as it can be seen in Figure 9 experiment 4 has

the least difference between the $F1_\beta$ score of the cross validations and the $F1_\beta$ score of the test set.

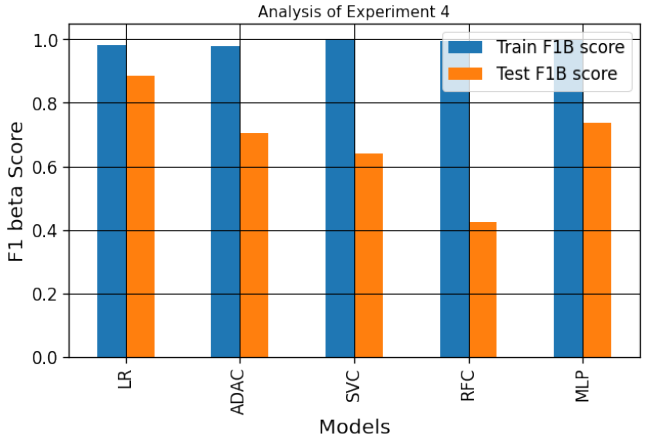

Figure 9: Analysing the $F1_\beta$ scores for mean cross validation and test set score in experiment 4

Table 3 summarizes the recall score of each model in each experiments. The best recall score in experiment 1 and 3 with SVC models. Given that our focus was to build a model that can maximize the true positive while having a good precision rate to predict stroke, a recall score of 0.918 is really good considering the very few cases of positive stroke out of the many negative cases. Similar to the $F1_\beta$ score, LR model also has the least variance of recall score across all experiments. One trend that can be noticed is the improvement of recall score from experiment 1 to experiment 3, where the recall score improves across all models except in SVC model where it is stable across experiments.

5 Broader Impacts

The use of machine learning within the medical field has grown tremendously. There has been multiple papers written on the application of machine learning algorithms for both cardiovascular disease and prediction of strokes. Predictive algorithms are important, especially for patients in terms of lifestyle and quality-of-life decisions. Further such models possess the ability to predict strokes without doing extensive research on what exactly might cause a stroke and yet still get just as accurate results.

However, this reliance comes with some drawbacks. One main issue is the fact that our model is only giving a binary predictive results can both be a positive and a negative. Having a simple model to tell which individuals to watch out for can be very beneficial, especially when working with large numbers, however one area that our specific model is lacking, is defining which features are the main risk factors. Since we only have made classification models we are not identifying which features carry the largest weight when creating the model, which has been done an other papers (Khosla et al. 2010). Here the main contributing features would be the risk factors potential stroke patients would want to avoid.

Another issue of deploying this model in real life is the inability of the model to generalize. In the study used for all our models (NIH 2011), all the participants were from the US. This is a common issue when creating models that the data is not representative for all areas. Since ours only is from the US, even though collected from four different communities, it is not to be said whether or not it will have the same accuracy in other parts of the world as race and culture may play a large role in stroke.

6 Conclusions

We attempted to classify stroke data belonging to patients into positive cases of stroke and negative cases of stroke types using five different classification models in different experiments. These models were the LR, ADAC, SVC, RFC and MLP. We found that the SVC was the model that yielded the highest $F1_\beta$ scores on held-out data in experiment 1,2, and 3. Experiment 4 have different $F1_\beta$ score and recall score. We theorize that this is due to the usage of SMOTE that has changed the distribution of the training set as such the models perform well in training data but does not on test data. Further, we noticed a trend of under sampling the majority class gives us a better recall score except in the model of SVC, where it is stable.

Further Work

Our paper primarily focused on undersampling and use of SMOTE method in order to develop LR, ADAC, SVC, RFC, and MLP models to solve classification of stroke using imbalanced set of data. However, another possible avenue for classifying this data could be through undersampling of 75% of the traing data as opposed to just 25% and 50% . As we have seen the recall score comparably improves as the percentage of undersampling increase, but the precision also drops. But given that getting the prediction of stroke is much more important, the trade off might be worth it. In addition, one can also try these methods and models using different set of data as the distribution might be different with perhaps a better results.

References

CDC. 2021. Stroke facts. `https://www.cdc.gov/stroke/facts.htm`.

Chawala, N.; Bowyer, K.; Hall, L.; and Kegelmeyer, P. 2002. Smote: Synthtic minority over-sampling technique. `https://arxiv.org/pdf/1106.1813.pdf`.

Khosla, A.; Cao, Y.; Lin, C. C.-Y.; Chiu, H.-K.; Hu, J.; and Lee, H. 2010. An integrated machine learning approach to stroke prediction. `file:///C:/Users/david/Downloads/kdd2010-3.pdf`.

Kirasich, K.; Smith, T.; and Sadler, B. 2018. Bias in AI: What it is, Types Examples, How Tools to fix it. *SMU Data Science Review* 1(9).

NIH. 2011. Cardiovascular health study. https://biolincc.nhlbi.nih.gov/studies/chs/.

Sasaki, Y. 2007. The truth of the f-measure. https://www.cs.odu.edu/~mukka/cs795sum10dm/Lecturenotes/Day3/F-measure-YS-26Oct07.pdf.

7 Appendix

The following shows the values searched over in the grid searches mentioned in Section 3 experiments.

- LR
  - class_weight:[{0:0.005,1:1.0},{0:1.0,1:1.0},{0:1.0,1:2.0}, {0:1.0,1:4.0},{0:1.0,1:5}, {0:1.0,1:10}, {0:1.0,1:50}]
  - C:[.00001, .0001, .001, .01, .1, 1, 10, 100, 1000, 10000, 100000]
- ADAC
  - Number of Estimators: [10, 50, 100, 300, 500, 1000, 1500, 2000, 2500, 3000]
  - Learning Rate: [0.0001, .001, .01, .1, 0.5, 0.7, 0.9, 1 ,1.3,1.5, 2]
- SVC
  - Kernel: ['linear', 'poly', 'rbf', 'sigmoid']
  - C: [.00001, .0001, .001, .01, .1, 1, 10, 100, 1000, 10000, 100000]
  - Degree: [2,3,4]
  - Gamma: [.00001, .0001, .001, .01, .1, 1, 10]
  - Coef0: [.00001, .0001, .001, .01, .1, 1, 10]
  - class_weight:[{0:0.005,1:1.0},{0:1.0,1:1.0},{0:1.0,1:2.0}, {0:1.0,1:4.0},{0:1.0,1:5}, {0:1.0,1:10}, {0:1.0,1:50}]
- RFC
  - Number of Estimators: [10, 50, 100, 500, 750, 1000]
  - Criterion: ['gini', 'entropy']
  - class_weight:[{0:0.005,1:1.0},{0:1.0,1:1.0},{0:1.0,1:2.0}, {0:1.0,1:4.0},{0:1.0,1:5}, {0:1.0,1:10}, {0:1.0,1:50}]
- MLP
  - Hidden Layer Sizes:[(10,30,10),(20,),(100,), (50,100,), (50,75,100,)]
  - Activation: ['tanh', 'relu', 'logistic']
  - Solver: ['sgd', 'adam', 'lbfgs']
  - Alpha: [.00001, .0001, .001, .01, .1, 1, 10]
  - Learning Rate: ['constant','adaptive']