[[1]](#footnote-0)

Applying Machine Learning Models to Stroke Classification

(March 2021)

Benedikt A. Graf (105652212), Eleanor Jackson (405651051), and Tianyu Sun (105657318).

***Abstract*—In determining the effects of risk factors on whether an individual is more likely to have a stroke, our project focuses on five models that attempt to determine the effects of gender, marital status, occupation, habitual smoking, and other factors that may increase stroke risk. Within this report we detail the issues and assumptions we made given the dataset, our approach to preprocessing the data, as well as our comparative results and conclusions to the models used.**

# Introduction

According to the World Health Organization (WHO), strokes are the second most common cause of death in the world [1]. In this project we examine a dataset of 5110 observations, of which 249 experienced a stroke event. It is our goal to correctly classify as many of the stroke outcomes as possible using different machine learning models. The development of a robust stroke classification algorithm could have many applications for clinical use including the screening of patients. We decided to compare five different models: Logistic Regression, Decision Trees, Random Forest, Support Vector Machines (SVM), and Forward Neural Networks (FNN).

# Literature Review

According to [Allen](https://pubmed.ncbi.nlm.nih.gov/?term=Allen+CL&cauthor_id=18706004) and [Bayraktutan](https://pubmed.ncbi.nlm.nih.gov/?term=Bayraktutan+U&cauthor_id=18706004) (2008), approximately 80% of stroke events are due to “modifiable risk factors,” which are risk factors that can be modified through lifestyle choices [2]. Moskowitz, Lo, and Iadecola (2010) review several studies to identify the major modifiable risk factors for stroke [3]. They list several “causal factors” that we have information on, including hypertension, diabetes, cigarette smoking, and heart disease. In addition, the authors also cite a “probable factors” that we have information on: the body mass index and psychosocial stress, the latter of which we may be capturing through the marital status (see Chin et al. 2017) and work type (see Johnson et al. 2005) variables [4][5].

There have been previous efforts to predict stroke events using machine learning techniques. Khosla et al. (2010) compare traditional statistical models to a proposed “automatic feature selection algorithm” and find that their algorithm outperforms the standard Cox proportional hazards model in predicting stroke events [6]. In a similar vein, we compare the performance of several different machine learning models for stroke prediction.

# Data Description

The data were obtained from Kaggle [7].

## Dependant Variable and Imbalance

Our dependent variable is a binary (0,1) variable that denotes whether or not the individual had a stroke. During our analysis, we noticed that there were many more people that did not have a stroke (0) than those who did (1) in the dataset. If we were to naively fit our models to the baseline dataset, this imbalance could lead to possible model specification and interpretation errors. Specifically, our models would observe that there are very few cases of strokes and would not accurately learn the effects of each feature on the probability of having a stroke. These naive models would undoubtedly have a high accuracy as they would predict mostly 0’s because the training set did not teach the algorithms how to correctly identify strokes. Later in the report, we demonstrate this observation by using a base model. The accuracy of the naive models were high, but had very low recall and precision rates. As a result, we identified an approach to adjust our dataset in order for our models to correctly learn to predict strokes.

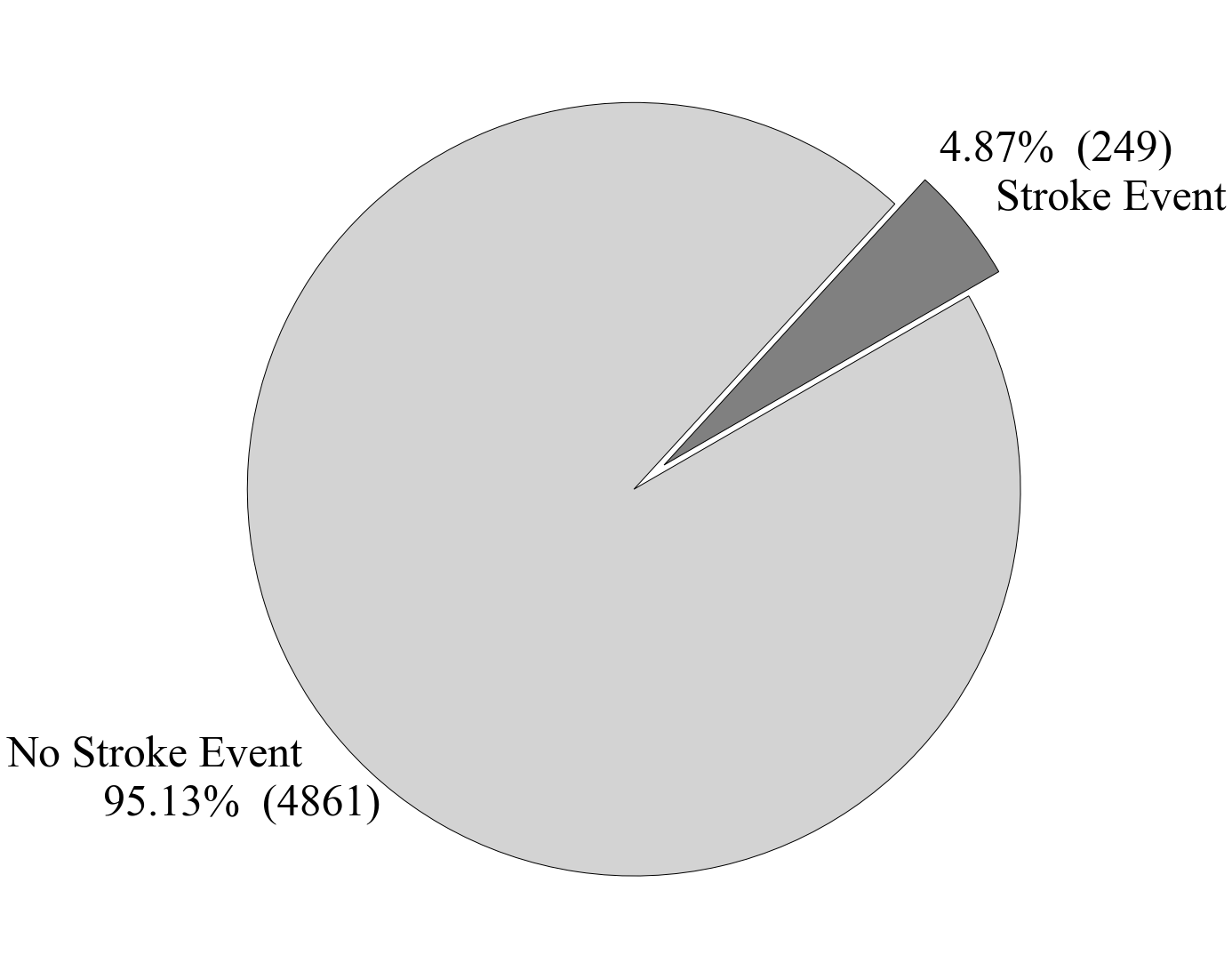


Fig. 2: The pie chart shows the imbalanced initial data set with 95.1% of the dependent variable being non-stroke observations and only about 5% being stroke or positive classifications.

## Independent Variables

The dataset includes 12 independent variables including information on gender, age, hypertension, heart disease, marital status, work type, location (urban or rural), average glucose level in the blood, the body mass index (BMI), and smoking status. During the early stages of our project, we first cleaned up the data and ensured that all the explanatory feature variables were formatted correctly. This entailed properly defining the indicator variables and seeing how many continuous variables we were working with. Within these variables we saw that occupation and smoking status had multiple classifications and so we split these variables up further: we added indicator variables for occupation (“never worked,” “government job,” “private,” “self-employed,” or “children”) and smoking status (“formerly smoked,” “never smoked,” “smokes,” or “unknown”).

## Missing Value Imputation

Originally, we removed 201 rows containing missing values for the BMI column, however, we realized that 40 of these missing values correspond to people who had strokes. Therefore, we imputed the missing values using the k-nearest neighbors (kNN) algorithm. To find an appropriate value for k, we artificially created missing values in the data and then tested how different values of k affect the absolute error. We found that the absolute error decreased rapidly until k=20 and then slowly levelled off. Following Nadkarni (2016) we chose the square root of N (70) as the optimal k [8].

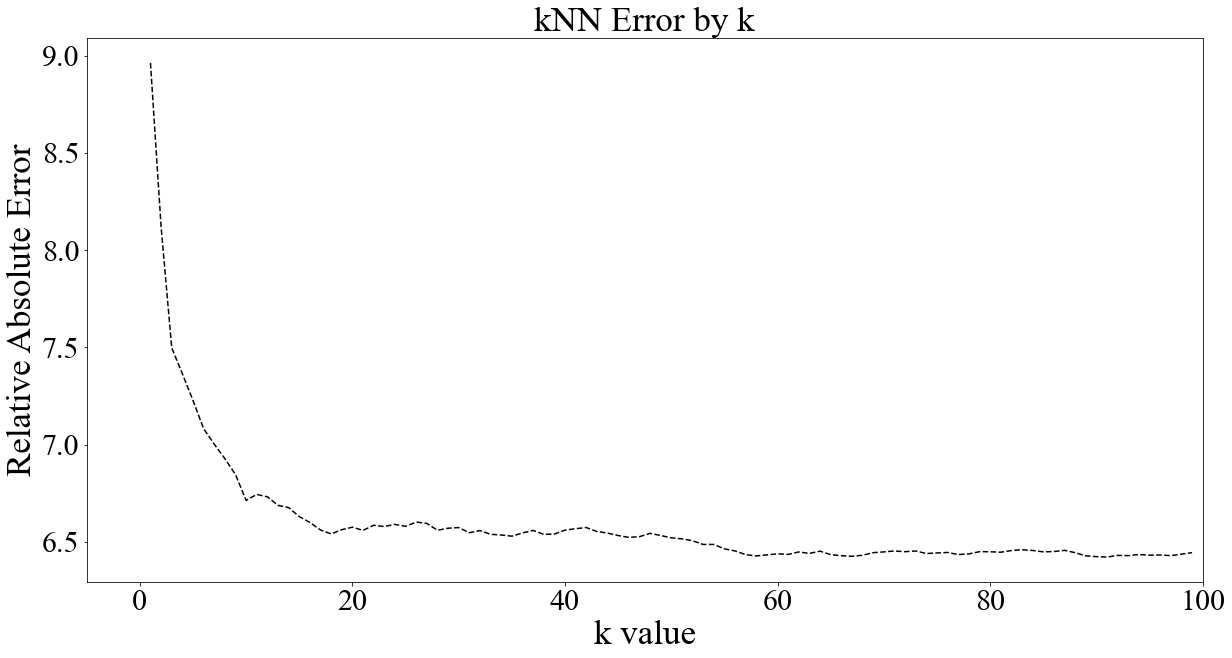


Fig.1: The Relative Absolute Error (Absolute Error/Number of Missing Values Imputed) observed in our simulation over different values of k.

# Our Approach

Before applying different models to the data, we split the data into training, validation, and testing subsets using a random state to ensure that we were all working with the same data. We normalized the data using Scikit-learn’s MinMaxScaler function, but later tested our models without performing any data normalization to capture the effect of scaling our data [9].

Initially, we applied the baseline Scikit-learn model (without any hyperparameter specifications) to our data. However, in order to overcome the problem introduced by the imbalanced class split, we decided to use oversampling and undersampling techniques to adjust the ratio of observations with strokes and non-strokes in order to offer our machine learning models more nuanced data to learn from. For undersampling we used the NearMiss technique and for oversampling we used the Synthetic Minority Oversampling Technique (SMOTE), for which we relied on Python’s Imbalanced-learn package [10].

Each model was run with the original, undersampled, and oversampled datasets. In order to evaluate the effectiveness of each model we reported the accuracy, precision, recall, F1 score, and AUC score. We reported the accuracy, but it was not our main decision criterion as an “untrained” model could achieve a 95% accuracy by assigning all observations to the non-stroke class. We decided to focus on the F1 and AUC scores as the primary information criteria for comparing models. The F1 score is a weighted combination of recall and precision metrics and the AUC score demonstrates how well a model did at varying decision stumps.

## Undersampling Approach

By decreasing the number of observations that did not have strokes we shifted more weight to those observations that contained strokes. We implemented this using NearMiss (version 1) or NearMiss-1 which preserves observations from the majority class that have the smallest average distance to the closest neighbors from the minority class and drops other observations. In our hyperparameter selection, we tested different neighbor values for the NearMiss algorithm. Here too, we relied on the Imbalanced-learn package to implement the NearMiss algorithm [10].

## Oversampling Approach

For oversampling, our reasoning was the opposite of our undersampling approach. In our efforts to increase the number of observations that had strokes, we implemented the SMOTE to artificially inject more observations with strokes as well as their corresponding feature matrices into the dataset. SMOTE creates a regression line from the minority class and draws from this line to add more observations to the data. In our models and code, by varying the weight in our SMOTE, we changed the ratio of stroke observations to non-stroke observations. In doing so, with the help of a for loop, we were able to find an optimal version of our model with the coinciding best weight for oversampling our data. Our aim was to increase the number of observations in order to give the models we used a way to correlate the small differences in our feature variables and more accurately predict when someone would be likely to have a stroke given our explanation variables.

# Models Used

Our team decided to use five different models to find the best fit for our data. The models we decided to use are as follows: Logistic Regression, Decision Tree, Random Forest, SVM, and FNN. For all of these models we relied on Scikit-learn for the coding implementation [9]. By using these five models, we hoped to capture as many nuances within the dataset as possible. From these models, we hoped to find differences in our information criteria, and draw cost benefit analyses of using one model over another. In the following sections we describe how each model functions and the specific way we used them for our dataset.

## Logistic Regression

Logistic Regression is a standard classification model. However, it can lead to errors in generalization. It outputs the probability of an event or certain class appearing based on a predefined threshold between 0 and 1. The model has many attributes that can be specified in the function call. There were many parameters that could have been looped through however due to time and computational efficiency we limited it to two parameters. For this project, we decided to run a for loop to optimize C, which is the inverse of regularization strength. If C is smaller it indicates that there is stronger regulzarization occurring in the model. Additionally, we also use solver as a parameter in the Logistic Regression model, which is the algorithm that the model uses to optimize the problem. Solver can take on five forms: Newton-cg, lbfgs, liblinear, sag, saga.

## Decision Tree

Decisions trees are a basic, yet powerful machine learning algorithm. In essence, a decision tree is a set of rules that can be learned from data and used to predict an unknown value. To “grow” a Decision Tree, first, all data points are collected in a single node. Next, the “best” binary decision rule is specified and used to split the data into two groups. The tree “grows” by repeating the last step, until the data is split into both outcome classes.

We incorporated several hyperparameters into a nested for loop to fit the Decision Tree to our data. Among them were the splitter (best and random) and criterion (gini and entropy), the former of which is the strategy used to split the data at each iteration and the latter of which evaluates the split. In addition, we attempted to optimize the depth of the tree, the minimum number of samples required to be at a node, and the maximum number of features to be considered for each split.

## Random Forest

The Random Forest algorithm is an extension of the basic Decision Tree. Here, an ensemble model is built by having multiple Decision Trees vote on the data point to be classified. The idea behind such ensemble models is that multiple “weak” classifiers can be merged into one “strong” classifier. In addition to the Decision Tree hyperparameters mentioned above, here, we also optimized the number of estimators (number of Decision Trees) to be used in the model.

## SVM

A Support Vector Machine is similar to Logistic Regression with one key difference in that SVM uses a hyperplane to differentiate among the classification types, unlike Logistic regression which uses a threshold. Unfortunately a Logit model does not look for the best margin. Therefore we decided to also implement SVM as this separated the classes thereby reducing the risk of error. We ran a loop to find the optimal kernel and optimal C value. The possible kernels were linear, poly, and rbf. C represents the regularization parameter. It is important to note that the power of the regularization is inversely proportional to C.

## FNN

Forward Neural Networks are powerful algorithms that allow the user to create a network architecture that will take in some amount of inputs from the feature matrix and densely map and identify the varying weights and effects these variables may contribute to our output through a number of hidden layers and activations functions. The most important part of a Neural Network is the architecture. In the scope of this course, we first created three “random” FNN models just to see how different models perform. Our independent variable is binary so ideally the best Network architecture would take in the dependent variables for each observation and produce a binary output for our binary stroke variable. Upon creating, compiling, fitting, and testing our initial models on the baseline data set we found that they performed very poorly. Exhibiting very low accuracy and predominantly most returned 0’s as the predictions, at this point it was safe to assume that a coin toss may have been better than these Neural Networks. This is a result of our data set being imbalanced. As a result we decided to implement SMOTE and NearMiss to correct for this. By using the FNN model with the best accuracy out of the three initial FNN models made, we varied the SMOTE weights, NearMiss neighbors as well as the keras optimization, loss function, and metrics with a nested for loop. In doing so we iterated over each combination to find the best permutation of our FNN that gave us the highest F1 score. This process was done for both the oversampled and undersampled data.

# Results

Using 5 different models and many variations, we narrowed down the best models to be: oversampled Logistic Regression with both normalized and unnormalized data, undersampled SVM without normalization, and oversampled FNN without normalization according to the F1 and AUC scores.

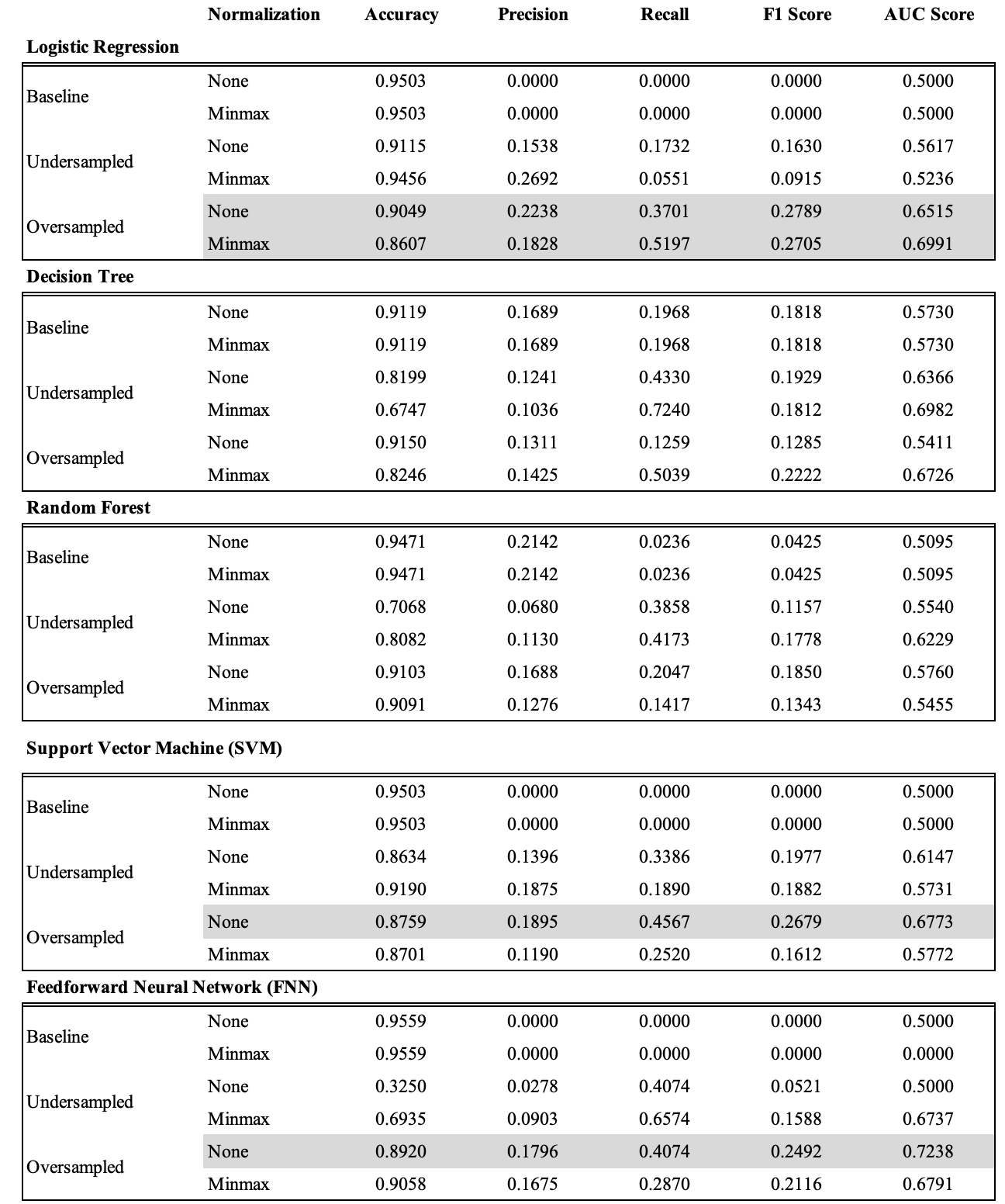
The oversampled Logit without normalization had F1 and AUC scores of 0.2789 and 0.6515, respectively. The oversampled Logit with normalization had F1 and AUC scores of 0.2705 and 0.6991, respectively. The undersampled SVM without normalization had F1 and AUC scores of 0.1977 and 0.6147, respectively. The oversampled SVM without normalization had F1 and AUC scores of 0.2493 and 0.7239, respectively. Please see the Appendix below for a complete tabulation of our results. The “best” models which we described above are highlighted in grey.

We did not expect the Logistic Regression Model to perform so well due to the model being fairly basic. However. we believe an explanation as to why the Logistic Regression model performed well is because it demonstrates the parsimony principle that a relatively less complex model can have the best results.

# Conclusion

Looking at the results of our models, we see trade-offs between the evaluation metrics we described earlier. Whereby some models reflected higher recall or precision rates, others fared better in regards to F1 and AUC score. These variations can be attributed to the innate differences in the prediction, learning, and optimization of the models. The Logistic Regression had a higher F1 score relative to our SVM and FNN models which are better equipped to deal with changing decision stumps. We also saw that for some models, the unscaled data set performed much better than the scaled data set. This is most likely the case due to our original data set containing a large amount of binary variables and similarly scaled continuous variables, which may have decreased the effects of the scaling. Due to these opportunity costs, it is hard to definitively say which model performed best. In future work, a new ensemble model could be specified using the bagging method by aggregating the above models and voting based on either an equal weighting or other voting system to cumulatively decide whether an individual has a stroke or not. This is something that future Machine Learning group projects could focus on.

Appendix

Table 1.: Summary of Results, 

selected models highlighted in grey.

References

1. World Health Organization (WHO). “The top 10 causes of death.” https://www.who.int/news-room/fact-sheets/detail/the-top-10-causes-of-death (accessed March 3, 2021).
2. C.L. [Allen](https://pubmed.ncbi.nlm.nih.gov/?term=Allen+CL&cauthor_id=18706004), and [U.](https://pubmed.ncbi.nlm.nih.gov/?term=Bayraktutan+U&cauthor_id=18706004) Bayraktutan. (2008, May). Risk factors for ischaemic stroke. *Int J Stroke*. [Online]. 3(2), pp. 105-16. Available: https://pubmed.ncbi.nlm.nih.gov/18706004/
3. M.A. Moskowitz, E. H. Lo, and C. Iadecola. (2010, July). The science of stroke: mechanisms in search of treatments. *Neuron*. [Online]. 67(2), pp. 181–98. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2957363/
4. B. Chin, M.L.M. Murphy, D. Janicki-Deverts, and S. Cohen. (2017, April). Marital status as a predictor of diurnal salivary cortisol levels and slopes in a community sample of healthy adults. *Psychoneuroendocrinology*. [Online]. 78, pp. 68-75. Available: https://www.sciencedirect.com/science/article/abs/pii/S0306453016308058
5. S. Johnson et al. (2005). The experience of work-related stress across occupations. *Journal of Managerial Psychology.* [Online]. *20(2),* pp.178-87. Available: https://www.lancaster.ac.uk/staff/taylorpj/papers/JMP2005.pdf
6. A. Khosla et al. (2010, July). An Integrated Machine Learning Approach to Stroke Prediction. Presented at   
   ACM SIGKDD. [Online]. Available: https://people.csail.mit.edu/khosla/papers/kdd2010.pdf
7. Stroke Prediction Dataset, Kaggle, Jan, 2021. [Online]. Available: https://www.kaggle.com/fedesoriano/stroke-prediction-dataset
8. P. Nadkarni, “Core Technologies: Data Mining and ‘Big Data’,” in [*Clinical Research Computing*](https://www.sciencedirect.com/science/book/9780128031308) *- A Practitioner's Handbook*. Cambridge, US: Academic Press, 2016, ch. 10, pp. 187-204. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128031308000105
9. F. Pedregosa et al. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*. [Online]. 12(85), pp.2825-30. Available: https://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html
10. G. Lemaître, F. Nogueira, and C.K. Aridas (2017. Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning. *Journal of Machine Learning Research*. [Online]. 18(17),pp. 1-5. Available: http://jmlr.org/papers/v18/16-365

1. This paper constitutes the final project of UCLA’s Machine Learning Course (ECON 425) and was submitted on March 12, 2021.

   B. A. Graf is a graduate student at UCLA (e-mail: bagraf@g.ucla.edu).

   E. Jackson is a fellow graduate student working on this project (e-mail: elliejackson@g.ucla.edu).

   T. Sun is the third member of this group and is also a graduate student affiliated with UCLA’s MQE program (e-mail: tonysun9810@gmail.com) [↑](#footnote-ref-0)