**Will you Die of Heart Disease?**

Brett Tully, [btully@bellarmine.edu](mailto:btully@bellarmine.edu)

**ABSTRACT**

I wanted to get creative with this project. After learning many different concepts of machine learning, I experimented with numerous different models and boosting techniques in attempt to find the best score even if they were not required. My main model is the Voting Classifier that contains the other 4 classification models inside it.

1. **INTRODUCTION**

The dataset I chose is for predicting heart failure. It is an interesting classification dataset because it is a real-life topic. The models/techniques I will be using are a Voting Classifier containing Logistic Regression, Random Forest Classifier, Support Vector Classifier, and Decision Tree Classifier. I also used Bagging Classifier, Gradient Boosting Classifier, and Ada Boosting Classifier.

**BACKGROUND**

* 1. *Data Set Description*

This dataset can be found on Kaggle.com. I chose this because I found this interesting because I have always had an interest in health care, and as a Data Scientist, working in the healthcare is where I want to eventually end up. Luckily, it came with no null values, so it made my job a bit easier

* 1. *Machine Learning Model*

Logistic Regression is a supervised learning algorithm which is used to solve the classification problems. It classifies the dependent variable usually by either 0 or 1, Yes or no, True or False. It uses the sigmoid function or the logistic function that is represented as **f(x)=Output between the 0 and 1 value, x=input to the function, and e=base of natural logarithm.** Random Forest is a form of ensemble learning that are combined to produce an optimal prediction model. Random Forest is an ensemble of Decision Trees, generally trained via the bagging method (or sometimes pasting), typically with max\_samples.

Support vectors are data points that are closer to the hyperplane and influence the position and orientation of the hyperplane. Using these support vectors, we maximize the margin of the classifier. Deleting the support vectors will change the position of the hyperplane. These are the points that help us build our SVM. Decision Trees is the non-parametric (a type of statistic that does not make any assumptions about the characteristics of the sample) supervised learning approach. Decision trees implement a sequential decision process. Starting from the root node, a feature is evaluated, and one of the two nodes (branches) is selected. Each node in the tree is basically a decision rule. This procedure is repeated until a final leaf is reached, which normally represents the target. Decision trees are also attractive models if we care about interpretability.

Although I had other models, they were not required as I simply was curious about the scores.

1. **EXPLORATORY ANALYSIS**

In this dataset, it contains 299 rows of data, with 13 columns. I had a somewhat normally distributed dataset, but our y variable needed stratification. I did not have any null values to deal with.

**Table 1: Data Types**

|  |  |
| --- | --- |
| *Variable Name* | *Data Type* |
| V1 Age | Float64 |
| V2 Anaemia | Int64 |
| V3 Creatine\_phosphokinase | Int64 |
| V4 Diabetes | Int64 |
| V5 Ejection\_fraction | Int64 |
| V6 High\_blood\_pressure | Int64 |
| V7 Platelets | Float64 |
| V8 Serum\_creatinine | Float64 |
| V9 Serum\_sodium | Int64 |
| V10 Sex | Int64 |
| V11 Smoking | Int64 |
| V12 Time | Int64 |
| V13 Death\_event | Int64 |

1. **METHODS**
   1. *Data Preparation*

The first thing I did was drop the “Time” column. I did this because it did not have much relevance to our data and not useful in helping the classification. I then assigned the X and y variable, then passed the X variable into the standard scaler to normalize our data. After noticing that our dependent variable, “Death\_event”, was not distributed evenly, I decided to stratify this variable.

* 1. *Experimental Design*

Table X: Experiment Parameters

|  |  |
| --- | --- |
| **Experiment Number** | **Parameters** |
| 1 | Switching the solver and multi\_class function for Logistic Regression |
| 2 | Changing Random Forest Classifier criterion and max\_featurues to different parameters |
| 3 | Alternating between ‘hard’ and ‘soft’ voting for the voting classifer |
| 4 | Passing the voting classifier through a bagging technique to get optimal model |
| 5 | For all the models where its applicable, I changed the n\_estimators and learning rates |
|  |  |

* 1. *Tools Used*

For my coding environment, I used Google Colab running Python. For the libraries, I generally put in the same ones for every project or assignment because you never know when it will become useful. I used Pandas, Train Test Split, SVR, SVC, Seaborn, Numpy, StandardScaler, DecisionTreeClassifier, RandomForestClassifier, BaggingClassifier, GradientBoostingClassifier, AdaBoostClassifier, ConfusionMatrixDisplay, and ClassificationReport.

1. **RESULTS**
   1. *Mean square Error and R-Square calculation*

A picture containing calendar

Description automatically generated

Text

Description automatically generated with low confidence

A picture containing text

Description automatically generated

Calendar

Description automatically generated

* 1. *Discussion of Results*

At first, my best model was using the Bagging Classifier that used the voting classifier as model options. I figured that this would end up being my highest scoring model due to it having the most flexibility with different options for models. But I ended up adding additional parameters for the Decision Tree Classifier and SVC that goes into the voting classifier, and that ended up making the voting classifier the winning model. My worst model was the SVC model. It had the lowest score out of all the other models/boosting techniques

* 1. *Problems Encountered*

I had an odd trend of all my model’s overfitting the data. I am not sure exactly why, but it could be from not having enough data. This is something I would want to investigate for future projects.

* 1. *Limitations of Implementation*

I feel like I used the best models that I can. I believe the limitations are on the lack of data points to fix the overfitting. All my train scores were very high, generally 95% and up. Hyperparameter tuning could have been more useful.

* 1. *Improvements/Future Work*

I believe to improve this, I could find a dataset with more data points or learn different techniques to fix the overfitting of data. I liked the models and techniques I used, but there could always be room for improvement by adding more ranges of parameters. I would also like to find a way to implement either a RandomSearchCV or GridSearchCV into a similar project next time using all the models/techniques I used for this project.

1. **CONCLUSION**

I had a lot of fun doing this assignment. My favorite part about coding is that once you grasp more concepts, it allows you to be creative in how you solve a problem, such as inserting the voting classifier as the model to use for the bagging technique. One issue I had about my project that I am not too happy about is how every model overfitted the data for the test. I do not believe this is anything on my end that I did wrong, but I think by acquiring a dataset that is larger, it could potentially fix the issue.

Using models in the way that we have been using them makes me excited to get into the neural network side of machine learning. One thing I learned about myself as a coder through this project is that I much prefer hyper parameter tuning over standard given parameters. I personally think using hyper parameter tuning helps to bring out the best of each model, regardless of how processor heavy they can be. As I mentioned at the beginning, health care is a field that I am quite interested in going into one day in terms of the data science world, so picking a dataset that has relevance to your personal interest is key.

**REFERENCES**

Kaggle.com

Sklearn.com