Heart Disease Prediction with Machine Learning

Buse Felamur  
*Faculty of Engineering, Environment and Computing*  
*Coventry University*Coventry, England  
felamurb@uni.coventry.ac.uk

*Abstract*—In this paper, machine learning algorithms such as Support Vector Machine, Logistic Regression, K-Nearest Neighbor, Naïve Bayes, Decision Tree and Random Forest applied to the heart disease dataset to predict if a patient is ill or not. Performance of these are compared to each other by varied methods like accuracy, confusion matrix, and overall classification report. When preparing the data, necessary methods were used to fix missing and wrong values, deal with class imbalance issues and dimension reduction. All of these were implemented by using Python programming language on Jupyter Notebook.

Keywords—machine learning; support vector machines; logistic regression; k-nearest neighbor; naïve bayes; decision tree; random forest; class imbalance; accuracy; confusion matrix; heart disease; python

# Introduction

There are many dangerous diseases humans loses their life to. One of them being heart diseases. In some countries, for example in the United States, it is number one killer of men and women. Every year 610,000 people die because of this in the United States. [1] That makes 1 in every 4 death. Heart disease, of course, a blanket term for anything that effects people’s hearts in a bad way. There are ways to detect the illness before it is too late. Some symptoms for heart diseases are; chest pain, shortness of breath, dizziness, high cholesterol, high blood pressure, diabetes, irregular heart beat and many more. [2] If any of these symptoms are detected, patients are advised visit an doctor so they can be tested. Tests include but not limited to; blood tests for cholesterol, electrocardiogram, exercise tests and blood pressure tests [3]. Based on these patients can get the medical help they need.

In this century doctors get a lot help from technology. Results came earlier and more accurate. There are more ways to aid doctors than just that. Machine learning is also an advanced field and it is evolving each day. Programs can be trained to think and predict based on the previous data. It can be possible to detect someone’s disease based on the results early and save a life.

In this paper a medical dataset was chosen to test this. Multiple machine learning techniques were used to train the set and find out if condition of someone can be predicted correctly.

# Literature Review

Over the years, main dataset used for many machine learning projects. For example, a group of researchers (Detrano, R., Janosi, A., Steinbrunn, W., Pfisterer, M., Schmid, J., Sandhu, S., Guppy, K., Lee, S., & Froelicher, V. in press) in 1989 used logistic regression and Bayesian method and then obtained the results 82% and 76%. [4] Person who is responsible from the database (Robert Detrano) is also used logistic regression and got 77% as a result. [5] Another researchers (Resul Das, Ibrahim Turkoglu , Abdulkadir Sengur, in press) used another method, namely neural networks, ended up with 86.4% and 89.011% for train and validation sets.

If we take a quick look at page of UCI, we can see that many people used these and more methods like decision trees with the Cleveland dataset.

# Data Set

The dataset was taken from the UCI Machine Learning repository. Raw database contains 76 attributes but all of the published experiments only used 14 of them so far. Main dataset, Cleveland dataset, has 303 instances and 14 useable attributes. There are three more datasets on the website with different number of instances but so far only Cleveland dataset was used for machine learning researchers for experiments. [6] For this project all four of the existing datasets were combined into one big dataset. Therefore, this makes a dataset with 14 useable attributes and 920 instances before all of the data preparation. Combined dataset has missing attributes. Last attribute is the predicted attribute, rest of the attributes are data related to patient. This can be seen on the Table 1 – Dataset Attributes

|  |  |  |
| --- | --- | --- |
| No | Description | Range |
| 1 | Age |  |
| 2 | Sex (0 for female, 1 for male) | 1 or 0 |
| 3 | Chest pain | 1 to 4 |
| 4 | Resting blood pressure (mmHG) |  |
| 5 | Serum cholesterol (mg/dl) |  |
| 6 | Fasting blood sugar (>120 is 1, else is 0) | 1 or 0 |
| 7 | Electrocardiographic | 0 to 2 |
| 8 | Max heart rate |  |
| 9 | Exercise induced angina (1 is yes, 0 is no) | 1 or 0 |
| 10 | ST depression induced by exercise |  |
| 11 | Slope of the peak exercise ST segment | 1 to 3 |
| 12 | Number of major vessels | 0 to 3 |
| 13 | Thalassemia | 3, 6 or 7 |
| 14 | Num (0 not ill, other ill) | 0 to 4 |

Table 1 – Dataset Attributes

# Data Preparation

This part covers the steps of preparing the data. Starting from replacing missing and wrong data, moves to class imbalance, data standardization and then ends with dimension reduction.

## Missing and Wrong Data

Before actually being able to train our data, we need to have a full dataset with no missing data. First step after combining the dataset is looking for missing data and replace them by nan values. This was done by both hand and code.

Second step is to check how many nan values each row have. If more than half of a row is empty then that row is completely removed. That means each row can hold maximum six nan values.

Moving on to data fixing, it started with rounding numbers so fractional part doesn’t affect the data. Nothing was lost. It was rounded so for example code now counts 120 and 120.0 as same numbers because they are both 120.0 now.

Max heart rate, resting blood pressure and serum cholesterol can never be zero so if it is zero then it is converted to nan too.

Thalassemia and Num columns were edited a little before the data filling. In Thalassemia column values can only be 3, 6 or 7. These three were converted to 0, 1 and 2 respectively. Something similar done to Num column. Values 2, 3 and 4 meant ill too so all three were combined under 1. Purpose of this project is predict if a patient is ill or not so nothing of value was lost.

Now it is time to fill all these empty data. Number of missing data can be seen in Figure 1 – Number of missing data. Method for this was making related groups and take their mean. Patients were divided into six subgroups. Young female, young male, middle age female, middle age male, old female and lastly old male. Young ages ranged from 28 to 39. Middle ages ranged from 40 to 52 and old ages are rest of the patients. To determine these age, mean, minimum age and maximum age of all patients were calculated. Missing data were filled according to these groups.

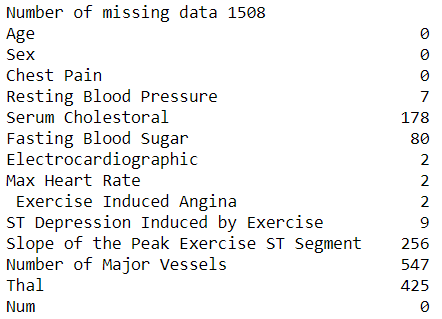


Figure 1 – Number of missing data

## Class Imbalance

Imbalanced data happens when there is unequal representation of classes. In real life datasets there is always a little amount of imbalance. Usually this doesn’t impact gravely but in certain data like medical ones, this can cause problems. If there are more samples of illness is absent overall, model might be more biased towards illness being absent and predict as healthy despite being not. There are multiple ways to fix this. Random under-sampling, random over-sampling and SMOTE. [7]

Random under-sampling basically means code randomly deletes data of overrepresented sample until it reaches a balance. Over-sampling is the exact opposite. It randomly duplicates samples from underrepresented class until data is balanced.

Random over-sampling maybe assigning more weight to data but on the other hand it doesn’t throw away data like under-sampling.

There is also a third option, SMOTE. SMOTE is a statistical way to increase data. SMOTE generates new instances from existing minority cases that is supplied as input.

Newly created instances aren’t only replicas of existing minority cases. First, samples of the feature space are taken for every single class and its nearest neighbors. Creating new examples which merges features of case and the neighbor. Unlike other methods, SMOTE increases the number of features available to every class and makes the samples more general.

For this paper, libraries of python used to implement all three separately. Imbalanced data can be seen at Figure 2.

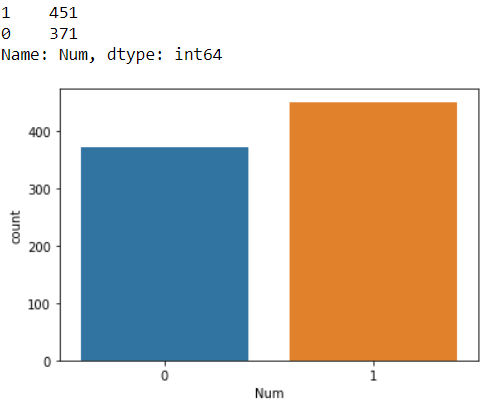


Figure 2 – Class imbalance of Num

## Data Standardization

When comparing measurements which have different units, it is important that it is done with standardizing the features around the center and 0 with a standard deviation of 1. [8] If variables measured at different scales, they cannot contribute equally to the analysis thus creates bias.

For example, there is a variable ranging between 0 to 1000 and there is another between 0 to 1. Larger range will outweigh the shorter one. If this is used without any standardization, analysis will give larger one, larger weight of 1000. To prevent this and evaluate each feature equally data standardization must be used. To standardize a feature, that feature must be subtracted mean from them and then divide all the values by standard deviation.

## Dimension Reduction

Purpose of dimension reduction is increasing the speed and the accuracy. It is done by decreasing the data to be processed without losing anything important. For this paper, PCA (Principal Component Analysis) is used. [9]

PCA finds a new set of dimensions so that all of the dimensions are linearly independent and arranged according to the variance of data along them. It means more variance and data is spread out more.

# Classification Techniques

## Support Vector Machine

Support vector machine (SVM) is a powerful supervised machine learning algorithm. [10]

Support vector machine is a representation of different classes in a hyperplane in multidimensional space. To minimize the error, hyperplane is generated in an iterative manner. Objective is to divide the datasets into classes to find a maximum marginal hyperplane.

SVM has a kernel that transforms an input data space into the required form. It makes kernel take a low dimensional input space and then transforms it into a higher dimensional space. Kernel converts non-separable problems into separable problems by adding more dimensions to it. It makes SVM perform better and more accurately.

## Logistic Regression

Logistic regression is a supervised classification method used for analyzing a dataset in which there exists one or more independent variables that predicts the outcome. There are only two possible outcomes.

This methods objective is to find the best fitting model which defines the relationship among the dichotomous characteristic of interest and a set of independent variables. [10]

## K-Nearest Neighbor

The k-nearest-neighbors algorithm is a lazy learning, supervised, classification algorithm. [10] It stores all instances related to training data points in n-dimensional space and uses them to learn how to label other data points.

When a new data is received, it looks at the data points that closest to that new point, also known as nearest neighbors. After that, neighbors decide, so attribute that is the most popular among the neighbors is the attribute for the new point. This makes k the number of neighbors it looked at.

If it is distance-weighted nearest neighbor algorithm, it weights the k neighbors according to their distance.

## Naïve Bayes

Naïve Bayes algorithms is a classification technique based on applying Bayes’ theorem with a strong assumption that all the predictors are independent to each other. In simple words, the assumption is that the presence of a feature in a class is independent to the presence of any other feature in the same class.

All properties independently add to the probability even if all features depend on each other. Naive Bayes model is easy to build and good with datasets with large data.

## Decision Tree

This model resembles a tree. It develops the tree while dividing the data into smaller subsets continuously.

## Random Forest

This one is made up from multiple decision trees. It creates them based on data samples. These trees each predict the output and after a voting, they pick one of them as true answer. Better than only one tree because of averaging the final output, it reduces chances of over-fitting.

# Experiment and Results

## Support Vector Machine

Using SVC classifier and grid search from python libraries model was trained. [11] Confusion matrix and results can be seen in Figure 3 and Table 2

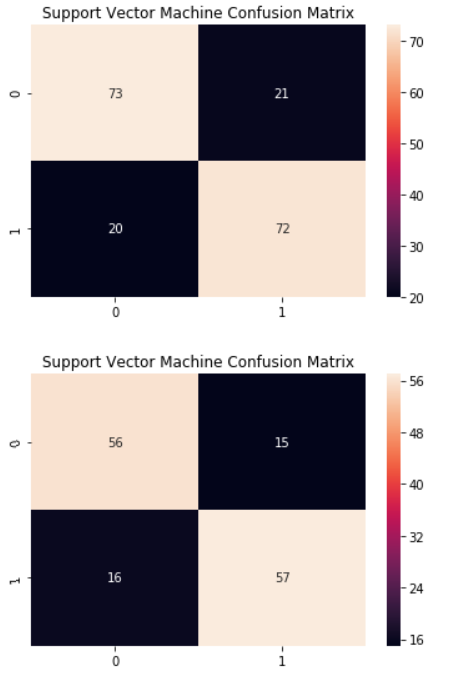


Figure 3 - SVM Confusion Matrix

|  |  |  |  |
| --- | --- | --- | --- |
| C | 10 | 10 | 10 |
| Gamma | 0.001 | 0.001 | 0.001 |
| Kernel | rbf | rbf | rbf |
|  | SMOTE | Over-samp | Under-samp |
| Accuracy | 0.7795 | 0.7795 | 0.7847 |
| Precision | 0.7800 | 0.7800 | 0.7800 |
| F1 | 0.7783 | 0.7783 | 0.7862 |
| Recall | 0.7800 | 0.7800 | 0.7900 |
| Matthews | 0.5591 | 0.5591 | 0.5694 |

Table 2 - SVM Results

First matrix shows the result when it is run with SMOTE and random over-sampling. Both results were exactly same so one picture used for both. Second matrix shows results for random under-sampling.

According to the results under-sampling seems to be doing slightly better but we lose data with that method.

## Logistic Regression

Using logistic regression classifier and grid search from python libraries model was trained. [11] Confusion matrix and results can be seen in Table 3 and Figure 4- Logistic Regression Confusion Matrix

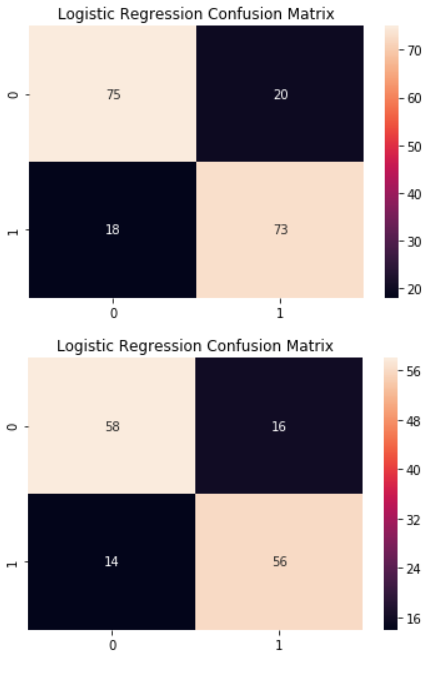


Figure 4- Logistic Regression Confusion Matrix

|  |  |  |  |
| --- | --- | --- | --- |
| C | 0.1 | 0.1 | 0.1 |
| Penalty | L2 | L2 | L2 |
| Random state | 0 | 0 | 0 |
|  | SMOTE | Over-samp | Under-samp |
| Accuracy | 0.7956 | 0.7956 | 0.7916 |
| Precision | 0.7900 | 0.7900 | 0.7800 |
| F1 | 0.7934 | 0.7934 | 0.7887 |
| Recall | 0.8100 | 0.8100 | 0.8100 |
| Matthews | 0.5915 | 0.5915 | 0.5835 |
| ROC | 0.8612 | 0.8612 | 0.8632 |

Table 3 - Logistic Regression Results

SMOTE and over-sampling is exactly same. That’s why the same confusion matrix is used again. Under-sampling is a little worse but nothing important. When compared to SVM, this one is slightly better.

## K-Nearest Neighbor

Using KNN classifier and grid search from python libraries model was trained. [11] Confusion matrix and results can be seen in Figure 5 - KNN Confusion Matrix and Table 4 - KNN Results

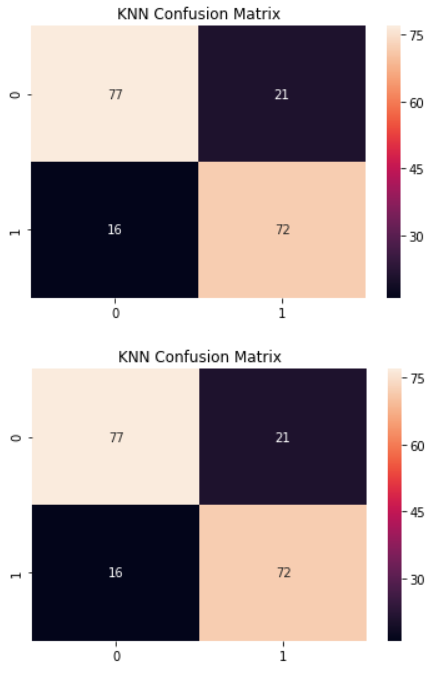


Figure 5 - KNN Confusion Matrix

|  |  |  |  |
| --- | --- | --- | --- |
| n\_jobs | 2 | 2 | 2 |
| N\_neighbors | 12 | 12 | 12 |
|  | SMOTE | Over-samp | Under-samp |
| Accuracy | 0.8010 | 0.8010 | 0.7916 |
| Precision | 0.7900 | 0.7900 | 0.7800 |
| F1 | 0.7955 | 0.7955 | 0.7857 |
| Recall | 0.8300 | 0.8300 | 0.8200 |
| Matthews | 0.6030 | 0.6030 | 0.5842 |
| ROC | 0.8602 | 0.8602 | 0.8622 |

Table 4 - KNN Results

Once again SMOTE and over-sampling is same with under-sampling following closely behind.

Out of the three we tested so far KNN is the best one with accuracy.

## Naïve Bayes

Using Naïve Bayes classifier only from python libraries model was trained. [11] Confusion matrix and results can be seen in Table 5 and Figure 6.

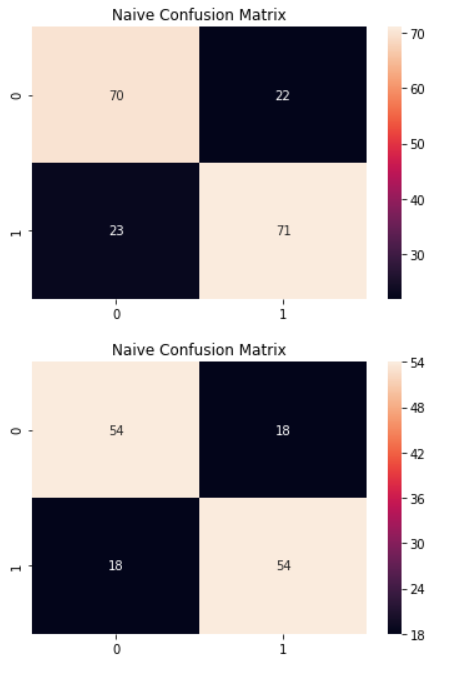


Figure 6- Naive Bayes Confusion Matrix

|  |  |  |  |
| --- | --- | --- | --- |
|  | SMOTE | Over-samp | Under-samp |
| Accuracy | 0.7580 | 0.7580 | 0.7500 |
| Precision | 0.7600 | 0.7600 | 0.7500 |
| F1 | 0.7593 | 0.7593 | 0.7500 |
| Recall | 0.7500 | 0.7500 | 0.7500 |
| Matthews | 0.5161 | 0.5161 | 0.5000 |
| ROC | 0.8602 | 0.8602 | 0.8622 |

Table 5- Naive Bayes Results

SMOTE and over-sampling is same again with under-sampling following closely behind them.

## Decision Tree

Using decision tree classifier and grid search from python libraries model was trained. [11] Confusion matrix and results can be seen in Figure 7 and Table 6- Decision Tree Results.

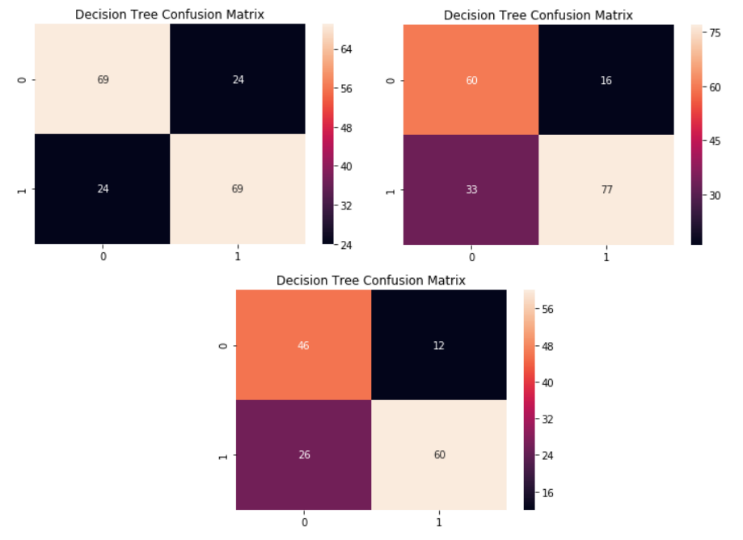


Figure 7- Decision Tree CM

|  |  |  |  |
| --- | --- | --- | --- |
| Max\_depth | 3 | 4 | 4 |
|  | SMOTE | Over-samp | Under-samp |
| Accuracy | 0.7419 | 0.7365 | 0.7361 |
| Precision | 0.7400 | 0.7900 | 0.7900 |
| F1 | 0.7419 | 0.7586 | 0.7594 |
| Recall | 0.7400 | 0.6500 | 0.6400 |
| Matthews | 0.4838 | 0.4812 | 0.4814 |
| ROC | 0.8114 | 0.7931 | 0.8011 |

Table 6- Decision Tree Results

This time SMOTE and over-sampling are not same at all. SMOTE’s accuracy better than both of them. However, f1 score and precision is lower.

## Random Forest

Using random forest classifier and grid search from python libraries model was trained. [11] Confusion matrix and results can be seen in Table 7 and Figure 8

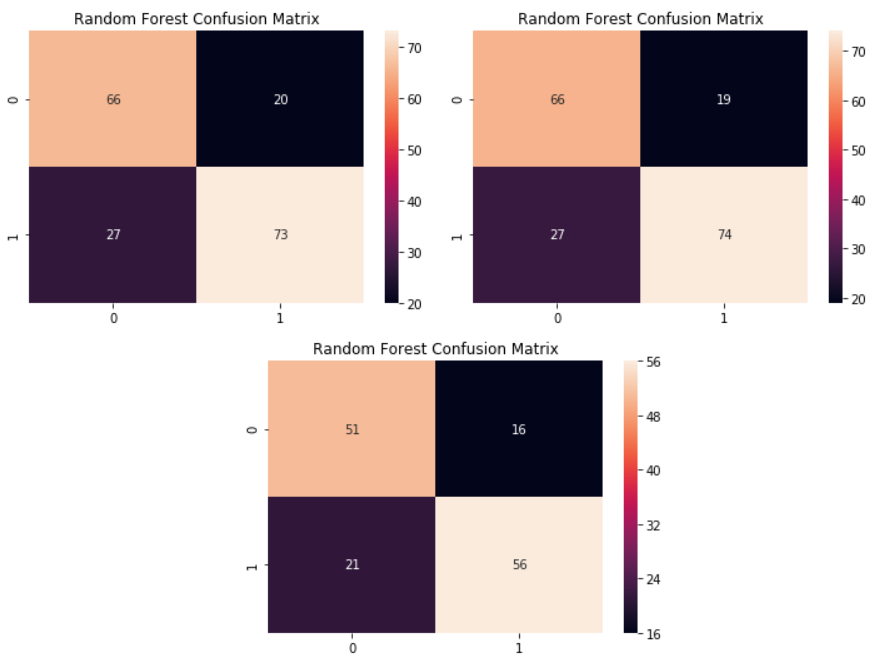


Figure 8 - Random Forest CM

|  |  |  |  |
| --- | --- | --- | --- |
| max\_features | sqrt | auto | auto |
| n\_estimators | 400 | 50 | 50 |
|  | SMOTE | Over-samp | Under-samp |
| Accuracy | 0.7473 | 0.7526 | 0.7430 |
| Precision | 0.7700 | 0.7800 | 0.7600 |
| F1 | 0.7564 | 0.7628 | 0.7516 |
| Recall | 0.7100 | 0.7100 | 0.7100 |
| Matthews | 0.4960 | 0.5072 | 0.4872 |
| ROC | 0.8369 | 0.8269 | 0.8064 |

Table 7- Random Forest Results

Surprisingly over-sampling is doing better than rest of them although there isn’t much difference.

# Conclusion

In the end if we compare every result, it can be seen that using SMOTE, over-sampling or under-sampling does not change the result gravely for this dataset. However SMOTE

seems to be the logical choice if we are looking for every way of increase in accuracy.

Now if we compare the highest accuracy scores of all algorithms, it is clear that there is not a drastic change between the scores. Highest accuracy belongs to KNN with 80% and lowest accuracy is Decision Trees with a score of 74.19%

In the end best way to handle this data is with KNN when n\_jobs is 2 and n\_neigbors is set to 12. Both SMOTE and over-sampling is alright. However, it is worth mentioning Logistic Regression too. When c set to 0.1, penalty to l2 and random state to 0, it ends up with 79.56%. This is very close to KNN’s score.

# Appendix

Link includes dataset and python code

<https://livecoventryac-my.sharepoint.com/personal/felamurb_uni_coventry_ac_uk/_layouts/15/onedrive.aspx?id=%2Fpersonal%2Ffelamurb%5Funi%5Fcoventry%5Fac%5Fuk%2FDocuments%2FBuse%20Felamur%20Project&originalPath=aHR0cHM6Ly9saXZlY292ZW50cnlhYy1teS5zaGFyZXBvaW50LmNvbS86ZjovZy9wZXJzb25hbC9mZWxhbXVyYl91bmlfY292ZW50cnlfYWNfdWsvRWc0TVh4UGNMNEZIb1k5eGUybHFwVUVCX21uazlFX2pIdzJfbzNwWUpFYk8wdz9ydGltZT1razYtR19sVjEwZw>

Dataset and code also available at <https://github.com/felamur/Machine-Learning>

# References

|  |  |
| --- | --- |
| [1] | N. CDC, "Heart Disease Facts," CDC, 28 November 2017. [Online]. Available: https://www.cdc.gov/heartdisease/facts.htm. [Accessed October 2019]. |
| [2] | M. C. Staff, "Heart disease," Mayo Clinic, 22 March 2018. [Online]. Available: https://www.mayoclinic.org/diseases-conditions/heart-disease/symptoms-causes/syc-20353118. [Accessed October 2019]. |
| [3] | NHS, "Coronary heart disease," NHS, 7 April 2017. [Online]. Available: https://www.nhs.uk/conditions/coronary-heart-disease/diagnosis/. [Accessed October 2019]. |
| [4] | R. J. A. S. W. P. M. S. J. S. S. G. K. L. S. &. F. V. Detrano, "International application of a new probability algorithm for the diagnosis of coronary artery disease," *American Journal of Cardiology,* vol. 64, pp. 304--310, 1989. |
| [5] | I. T. A. S. Resul Das, "Effective diagnosis of heart disease through neural networks ensembles," *Expert Systems with Applications,* vol. 36, 2009. |
| [6] | D. W. Aha, "Heart Disease Data Set," UCI, [Online]. Available: https://archive.ics.uci.edu/ml/datasets/Heart+Disease. [Accessed October 2019]. |
| [7] | T. Boyle, "Dealing with Imbalanced Data," [Online]. Available: https://towardsdatascience.com/methods-for-dealing-with-imbalanced-data-5b761be45a18. [Accessed October 2019]. |
| [8] | J. Hale, "Scale, Standardize, or Normalize with Scikit-Learn," [Online]. Available: https://towardsdatascience.com/scale-standardize-or-normalize-with-scikit-learn-6ccc7d176a02. [Accessed October 2019]. |
| [9] | M. Galarnyk, "PCA using Python (scikit-learn)," 5 December 2017. [Online]. Available: https://towardsdatascience.com/pca-using-python-scikit-learn-e653f8989e60. [Accessed October 2019]. |
| [10] | Simplilearn, "Machine Learning Full Course | Learn Machine Learning | Machine Learning Tutorial | Simplilearn," Simplilearn, 29 August 2019. [Online]. Available: https://www.youtube.com/watch?v=9f-GarcDY58. [Accessed October 2019]. |
| [11] | P. e. al, "Scikit-learn: Machine Learning in Python," pp. 2825-2830, 2011. |