**Heart Disease Prediction**

EDA, Modelling and Evaluating ML models to predict the severity of heart diseases.

Cryptonite Research AI Taskphase

Author:

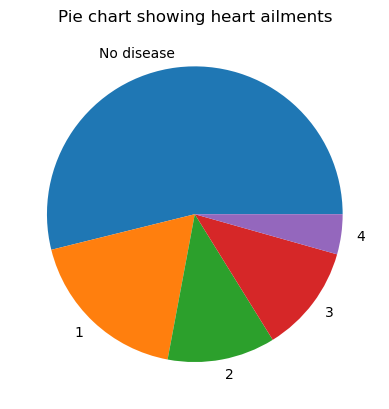
Pratham Shah

9082296993

**Abstract**

Heart disease prediction is an extremely abstract, precise and contemporary exercise. This review will show how the model I submitted fares in modelling their complex and non-standard details, to achieve a maximum accuracy of 82% and an average accuracy of 73.5% .

Predicting the *presence* of heart ailments is indeed much more accurate than classification (86.67%-90%) and the class imbalance and intermingling in the Processed Cleveland dataset makes severity predictions even more heavily skewed and inaccurate.

A diagram of a number of dots

AI-generated content may be incorrect.

**Approach**

The first model predicts output in a multiple regression way. This is unfit for a classification task (68% acc) and discarded. The second model is vanilla logistic regression, which returned a 65% accuracy. Both unfit.

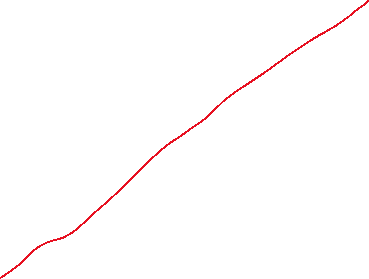
Thus, this model’s approach is hierarchical: 1 model classifies whether a patient has a disease, and another sequentially classifies the severity of the disease – by combining:

Severity levels 2-4: 2 (severe)

Severity level 1: 1 (not severe/mild)

A chart of yellow and purple dots

AI-generated content may be incorrect.



This is how the severity 2D PCA plot looks now – a much more separatable distribution.

**Workflow**

Firstly, make a random state variable (RANDOM\_STATE) that determines what data is fed to the train/test split and

Step 1: Loading the data and making imports

Step 2: OneHotEncoding categorical features and Z-score normalising numerical ones

Step 3: Train-Test split using the said random state

Step 4: Storing 1-4 values into 1 – for classification part one: the PRESENCE of heart disease. 0 – no disease, 1 – disease. We’ll measure severity in part 2

Step 5:

STRATIFIED K FOLD FOR A SET OF 4 ML MODELS – TO DETERMINE WHICH ONES BEST. (Fixed random state variable)

LogisticRegression(max\_iter**=**1000, class\_weight**=**"balanced", random\_state**=**RANDOM\_STATE),

MLPClassifier(max\_iter**=**1000, random\_state**=**RANDOM\_STATE),

SVC(probability**=**True, class\_weight**=**"balanced", random\_state**=**RANDOM\_STATE),

RandomForestClassifier(class\_weight**=**"balanced", random\_state**=**RANDOM\_STATE),

XGBClassifier()

Each of these models receives a cross validated accuracy through the k folds and then, the most accurate model is chosen to predict classification 1.

Most often, Logistic Regression comes out on top – leading on accuracy, precision and F1 score. Here’s why:

Binary classification on a linearly separable decision is logistic regression’s most fundamental and baseline strength.

Logistic regression finds class boundaries reliably and out-of-the-box. Class weight setting to balanced further allows it to handle the class imbalance.

Logistic regression also generalises well with less data due to its linearity and simplicity – unlike say NNs or SVMs.

LR v/s NN (MLPC multi layer perceptron classifier in this case):

LR generalises better than NNs for small linear data. NNs need a lot of tuning and large amounts of data. They are prone to overfitting in small datasets due to restricted information hindering real pattern detection.

LR v/s XGBoost:

XGBoost too can overfit and/or underperform on small data because of its inherent extreme gradient boosting nature. It’s overkill for this dataset and again, works only when fine tuned to an uneccesary extent or when given a large dataset.

LR v/s SVMs:

SVMs fits are very sensitive in nature and need a lot more tuning than logistic regression. Logistic regression can thus beat it out on simple moderately separable structured data right out of the box.

LR v/s Random Forest:

Random Forest was a strong contender for this. It sometimes beat out LR too. RF works well too with class weights balanced and its voting structure leads to more ways to arrive at the best output. But LRs are simpler to interpret, do not need a tree structure and are faster.

Step 6:

For classification 2, we use SMOTE with EEN improvements to upsample the dataset to deal with the class imbalance between

2 and 1 severity levels (2 – 83, 1 - 54). I chose this distribution of 2s and 1s because they were identified as the most separable and balanced on the PCA graph.

Step 7:

Preprocessing for classification 2

Step 8:

Stratified K-Fold cross verification for determing the best model for classification 2.

MLPClassifier(random\_state**=**RANDOM\_STATE, max\_iter**=**1000),

RandomForestClassifier(class\_weight**=**'balanced', random\_state**=**RANDOM\_STATE),

LogisticRegression(class\_weight**=**'balanced', max\_iter**=**1000, random\_state**=**RANDOM\_STATE),

SVC(class\_weight**=**'balanced', random\_state**=**RANDOM\_STATE),

DecisionTreeClassifier(class\_weight**=**'balanced', random\_state**=**RANDOM\_STATE),

Logistic regression outperformed all the other models yet again. This is likely due to the combination of linearity, simplicity out of the box, and a preprocessed but small dataset.

Logistic regression found that this dataset was linearly separable, it worked well out of the box without too much fine tuning, and its nature of training allows it to generalise well on small datasets.

Neural Nets are the best for modelling complex patterns in data, but in this case, the model likely overfit on the small sample size of the training set. Plus, neural nets fail to perform well on small sample sets without advanced tuning and trial/error.

RandomForest classification is a solid competitor to LR in this case. Ensemble models are good at dropping out noise, generalising well and thus preventing overfitting. But, this ensemble model, like most tree based methods, fails to generalise on extrapolated data, since trees are built using splits only from the training range. Also, without extensive tuning, these trees can go too deep and model noise.

SVMs work with decent accuracy, but since they try to find a hard/soft boundary within the data (and the cleveland set does overlap as seen by the PCA plot) they deliver less accuracy.

Decision Trees would likely not perform better than their random forest counterparts, because of the ensemble voting system.

Step 9:

The combined classifiers serve their purposes in a hierarchical manner.

The data from the test set is passed on to classifer 1 first – which returns a list of 0s and 1s as (no diseases, diseases). All the indexes of the 1s in this list of predictions are passed on to the second hierarchical classifier – which parse the respective indexes from xtest and then return one of (1, 2) in place of the original 1. 1 means a mild heart disease, and 2 means a severe heart disease. Note again that in the original set there were 4 severity levels – 1, 2, 3, 4. I have combined 2, 3 and 4 into 2 to prevent too much class imbalance and generate some separability.

Step 10:

Analysing the quality of the predictions.

**Conclusion**

Final model results:

precision recall f1-score support

0 0.85 0.91 0.88 32

1 0.62 0.45 0.53 11

2 0.83 0.88 0.86 17

accuracy 0.82 60

macro avg 0.77 0.75 0.75 60

weighted avg 0.81 0.82 0.81 60

The scores for “1” are indeed low, likely because of the class imbalance due to limited support. Otherwise, 0 and 2 achieve decently high results.