Max Trotter

## Choosing Methodology

### Justification

SEMMA has been chosen as the data mining methodology I will use for this coursework, SEMMA has been chosen because it places more priority on the processes that occur after sourcing the data and deciding objectives, as I will not be able to access the greater context of the data beyond the given dataset during this coursework. Due to this a methodology that focuses almost solely on the processing of the data is ideal. Other methodologies such as CRISP-DM and OSEMN place emphasis in their earlier stages on business understanding, but in this case the objective is already clear: to produce predictors of risk in cardiovascular patients.

### What is SEMMA?

SEMMA consists of 5 stages:

#### Sample

Selecting subsets of the data to use for construction of models and identifying variables of importance.

#### Explore

Perform univariate and multivariate analysis to understand how certain variables affect each other and to find gaps in the data with the aid of data visualisation.

#### Modify

Based on findings in the explore stage, before modelling, modify the data as necessary to produce a more indicative model.

#### Model

Apply data mining techniques to modified dataset/s to produce a prediction model.

#### Assess

Evaluate the created model for accuracy and reliability and estimate its overall efficacy.

# Using SEMMA

## Stage 1 – Sample

The dataset I must work with is a set of patient records, there are 1519 records before any data cleaning has taken place. Before the data could be used for further processing it had to be cleaned: When I attempted to create label dummies there were 5 entries with risk labels that were empty or invalid. They were separated from the dataset as without having a risk label the entry cannot be trained upon for a risk predictor, these entries could however be used for finding correlations between the attributes.

When I tried to create indication dummies, I found there was an inconsistency in the labelling of Asymptomatic Stenosis patients with some being logged down with ‘ASx’ and others with ‘Asx’, the ‘Asx’ entries were converted to ‘ASx’ for data consistency.

A picture containing text

Description automatically generated

There was also an empty string value in the Contra column, which needed to be converted to a float to make the data usable for analysis/modelling.



## Stage 2 – Explore

The first analysis I performed was plotting confusion matrices of the different indicators that hospitalised the patient resulting in them becoming a data entry and plotting matrices of the other conditions the patients had when they were hospitalised. These matrices showed me who had an indicator or condition and was not at risk, and those who were. Here is an example of the code I used to produce one of the matrices and the resulting matrix:

Text

Description automatically generated

Chart

Description automatically generated

From these matrices, I acquired this data which can inform us of the correlation of the various indicators/conditions and risk.

|  |  |  |  |
| --- | --- | --- | --- |
| Indicator/Condition (I/C) | Number of valid entries | Risk rate | Percentage of entries |
| Atrial Fibrillation (I) | 493 | 41.2% | 32.6% |
| Asymptomatic Stenosis (I) | 217 | 47.9% | 14.4% |
| Cardiovascular Arrest (I) | 413 | 32.4% | 27.3% |
| Transient Ischemic Attack (I) | 389 | 16.5% | 25.7% |
| Diabetes (C) | 76 | 92.1% | 5% |
| IHD (C) | 716 | 45.9% | 47.4% |
| Hypertension (C) | 729 | 52.3% | 48.2% |
| Arrhythmia (C) | 328 | 97% | 21.7% |
| History (C) | 23 | 34.8% | 1.5% |

I also found the average risk rate for all patients, there were 507 entries with risk, 1008 without resulting in an average risk percentage of 33.5%, very close to a third. From this we can see that Diabetes and Arrhythmia are almost certain indicators of risk, and that Transient Ischemic Attack is considerably less risky indicator to be hospitalised for, and that asymptomatic stenosis is the highest risk indicator to be hospitalised for.

I then created box plots of the percentage of both types of Cerebral Ischemic Lesions,

Ipsilateral Cerebral Ischemic Lesions: Chart, box and whisker chart

Description automatically generated

Contralateral Cerebral Ischemic Lesions:

Chart, box and whisker chart

Description automatically generated

These show a high correlation between percentage of lesions and risk rate, with contralateral lesions being the more effective indicator of the two, with 50% of those who were at risk (excluding outliers) having above ~75% contralateral lesions. 75% of those who were not at risk had below ~60% contralateral lesions. This displays the percentage of contralateral lesions as an extremely strong indicator of risk.

## Stage 3 – Modify

In order to make the data into the best format for modelling, the hospitalisation indicators needed to be converted into dummies so the model could work with binary values rather than a set of varying labels. This data then needed to be combined with the dummies for the conditions each patient had and the percentage of types of cerebral ischemic lesions:

Text

Description automatically generated

Before the cerebral ischemic lesions could be added to the data frame that would be used for modelling, they needed to have their NaN values replaced, I chose to replace them with the mean as the data is quite evenly distributed, and I chose to replace rather than remove because I did not want to remove whole entries with otherwise useful data that can help with the model.

Text

Description automatically generated

I also removed entries with cerebral ischemic lesion percentages that were outliers, in this case 3 standard deviations from the mean, to minimise the effect of extreme or potentially mis recorded entries on the model.

Text

Description automatically generated

## Stage 4 – Model

I used 3 different modelling techniques, Decision Trees, Neural networking and Random Forest, I will be using various settings on each to find which one can provide the optimal prediction model.

### Decision Tree

For making the Decision Tree Model I utilised sklearn’s tree model which makes use of the CART algorithm:

Text

Description automatically generated

I plotted the structure of the tree and a confusion matrix to see how well it can predict, when I allowed it to reach a maximum depth of 20 (a higher depth than it needs to make a tree with 0 Gini outputs), the structure of the decision tree looked like this:

Diagram, engineering drawing

Description automatically generated

It is too large to determine the individual decisions on this document however we can see its depth caps out at 10. The first split it decides to make every time is whether the patient had arrythmia, followed by their percentage of contralateral ischemic lesions, these two factors are deemed very significant.

### Neural network

A picture containing diagram

Description automatically generated

For my neural network I used sklearn’s MLPClassifier, which can use several different methods to generate the weightings of the network: ‘lbfgs’, which is a quasi-newton method, ‘sgd’, a Stochastic gradient descent or ‘adam’, which is a different stochastic gradient based method optimised for larger datasets. I will be testing all of them to see which provides the most accurate predictions.

### Random Forest

Text

Description automatically generated

For my random forest I will be using sklearn’s RandomForestClassifier, settings that will be tested are the number of trees in the forest (n\_estimators), the depth of each tree (max\_depth) and the criterion for a split (criterion).

## Stage 5 – Assess

For testing each setting of each model for efficacy, I ran them 5 times with a shuffled 70% train 30% test split of the data and used the mean of the results to display the metrics of their predictions.

### Decision Tree

Firstly, I tested the decision tree with a limitless tree depth and the results were:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| TP | FP | TN | FN | Accuracy | Precision |
| 148 | 3 | 301 | 3 | 98.352 | 98.444 |

A very promising result, there was little variation in the accuracy of the predictor upon shuffling the train test data which shows consistency. Accuracy varied at most 1.2%. The accuracy and precision scores were about the same, very high but not perfect.

Next I reduced the max depth of the tree to 5 to see how that would change results:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| TP | FP | TN | FN | Accuracy | Precision |
| 138 | 6 | 297 | 14 | 95.6 | 96 |

Reducing the depth the tree reached by half reduced the accuracy and precision by around 2.5%, the performance gains from making such a switch weren’t considerable enough to make using a lower depth tree advisable.

### Neural network

First I tested the MLPClassifier with a sgd solver, a maximum iterations of 500 and a hidden layer size of 20:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| TP | FP | TN | FN | Accuracy | Precision |
| 105 | 39 | 269 | 42 | 80.59 | 74.14 |

The accuracy results were significantly lower than the decision tree results however this is a low amount of iterations and a small hidden layer size so hopefully upon raising these settings the accuracy and precision will greatly increase.

Next, I upped the maximum iterations to 5000 and the hidden layer size to 200:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| TP | FP | TN | FN | Accuracy | Precision |
| 125 | 12 | 292 | 27 | 91.5 | 91.71 |

First thing to note is the processing time for these results was far larger than for any of the previous, about 10x as long. There was also very often a disparity between the accuracy and the precision during despite them averaging out closely. It sometimes excelled at identifying positives or negatives but rarely at the same time.

I then upped the maximum iterations to 20000 and the hidden layer size to 500 to see if there were any more efficacy gains to be made:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| TP | FP | TN | FN | Accuracy | Precision |
| 109 | 12 | 301 | 33 | 90.1 | 90.08 |

Adding extra layers and iterations did not bring performance up any higher, and accuracy and precision so far remain below the decision tree which also runs far quicker.

Next, I changed the solver to ‘lbfgs’:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| TP | FP | TN | FN | Accuracy | Precision |
| 146 | 3 | 304 | 2 | 98.856 | 97.83 |

This took incredibly long to make predictions (~5 minutes), however it makes notably better predictions than the gradient descent solver that was in use before, with results competitive with the decision tree model.

### Random Forest

First, I tested the random forest with 100 estimators:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| TP | FP | TN | FN | Accuracy | Precision |
| 148 | 1.6 | 302 | 3 | 98.98 | 98.924 |

Very strong results with the best accuracy and precision scores of all models so far, random forest seems very well suited to handling data that contains lots of binary variables. (The dummies for indicators and conditions).

With the processing speed also being fast, next I increased the number of estimators to 500 to see if the model can improve at all:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| TP | FP | TN | FN | Accuracy | Precision |
| 146 | 1.6 | 305 | 2.4 | 99.12 | 98.918 |

There was a small accuracy gain but this can be attributed to the random nature of the training and testing, there likely isn’t a noticeable improvement here because adding extra decision trees to the forest is redundant as most of the variables are binary, with only 2 scalar variables, which means 100 trees is already enough to make accurate predictions, as a node in a decision tree for any of the binary variables can only be a statement checking if they have the indicator/condition or not.

### Conclusion

The best predictive model of those tested was the random forest which had the highest accuracy and precision. I believe this to be the case due to the high number of binary variables in the data once it has been converted to dummies. Although there was little disparity between accuracy and precision with any of the tested models, it is worth noting the importance of precision in models for medical purposes: if someone is incorrectly predicted as not at risk, they could be put in great danger, while if they are incorrectly identified as at risk there may be some inefficient use of medical resources in response to it but it does not place health at risk. Models with higher precision should be favoured for a role like this.

## Conclusion on Methodology

I found SEMMA to be a suitable methodology for this project, however I found the need to move between stages on occasion during the process, usually between the modification and sampling stages, as in order to explore the data using visualisation techniques, removing entries with missing values and then removing outliers was helpful, which can be considered part of the modify stage. The sample stage wasn’t the most useful for this project, but this is because most of the dataset was already useful data for modelling, in a context where the dataset needed to be selected from a large data warehouse or from raw unorganised databases, this stage would become much more important. Overall, however, the process was clear and mostly linear under SEMMA and achieved good results.