Data Mining and Decision Systems  
600092  
Assigned Coursework Report

Student ID: 561438  
Date: 18 November 2019

## Due Date: 12 December 2019

**Report must be within 8 page maximum. Strict page limits will be enforced. Any extra pages will be ignored and no marks awarded for any work on these. Exclusions to this limit are the front page, the references section, and any appendices. Please keep to the given section headings and format; subsections are permitted.**

# Methodology

Cross-industry standard process for data mining, also known as CRISP-DM, is a methodology commonly used by data scientists when analysing big amounts of data.

This methodology has 6 steps: Business Understanding, Data Understanding, Data Preparation, Modelling, Evaluation and Deployment. Each one of these steps help gathering the right information for a specific purpose. During the first 4 steps, it is quite common to backtrack between them in order to understand the data and be able to take right action in order to get the right results for the models.

Business Understanding and Data Understanding allow the data scientist to understand the final objectives and the business perspective. Once having gathered the right raw data it is time to understand and get familiarized with it.

During data preparation, data scientists study the data more in depth and find any anomalies that the data contains. These anomalies can be any incorrect values that do not fit in the data, such as Nan values, misspelt values or characters that do not correlate with the rest of the values in the data. It is also helpful to rename certain labels to relate them to the data set they are given or even drop certain information if it does not contribute towards the data modelling step.

Next step is Data Modelling. During this step it is common to revisit some of the previous steps in order to adapt the data to make the required models. Some models such as decision trees and confusion matrixes require the variables in the data to be numeric or binary, for which its necessary to go back one step (to Data Preparation) to change some data types into binary.

Once finished with the Data Modelling, it is now time for the Evaluation. Now we need to look at the model see if their predictions are similar and if every aspect of the business has been considered during the whole process.

Scrum Agile is another methodology used in software development which is helpful at reducing time from the beginning of the development to the final deployment in the real world. When compared to the CRISP-DM methodology, Agile can be more flexible as it does not have a specific development order, where as CRISP-DM is much stricter to its steps in order to achieve the outcome. On the other hand, when using the CRISP-DM methodology, it is possible to step back from first 4 phases if needed, where as it is not as easy when following an Agile methodology.

**Business understanding**

The data we are going to analyse is about medical records. This data comes from 1520 patients from the real world. There are 8 columns representing different diseases that patients may or may not have and one column representing whether the patient is on risk or no risk. The Random column represents each of the records for every visit of the patient. The Id column represents each patient which can be repeated. This is because a patient can revisit a clinic and have one more record with new symptoms which will be then treated as a new data entry. This can be helpful as this will show whether the new record would make an impact on the risk.

Once we understand the data and we know what each column represents, we can proceed to the next step, which is data cleaning.

**Data Preparation**

In order to start creating models, it is crucial to make sure there are no invalid entries in the data set to avoid inaccurate results when making models.

First of all, we need to read the .csv file and make sure that any entry that only contains a space is read as a ‘Nan’ value. Then we need to make sure that the original document does not get affected by the changes we are going to make during the data cleaning process. For this, it is best to make a deep copy. This will create a parallel copy in memory, so when we make changes, it does not affect the original document.

Now that we have the data loaded, we will start with cleaning the data. We first need to find the unique values for each column to identify invalid entries, duplicate entries and even spelling mistakes. Next step is to count how many columns have ‘Nan’ values and how many we of them we have before even cleaning the data, which so far, we have 18 ‘Nan’.

When we identify all the unique values from different columns, we also find two ‘Unknown’ entries in the ‘label’ column. Since this entry is equivalent a ‘Nan’ entry, it is better to replace it to a ‘Nan’ value just as we replaced the white space value. Another issue with the ‘label’ column is the name. The name has no relation to the data, for which is better to rename it to ‘Risk’, since the values that the column contains are ‘Risk’ and ‘NoRisk’.

It is also worth noticing that there is a spelling mistake in the ‘Indication’ column. This column contains 5 unique values (excluding the ‘Nan’ values) for which ‘ASx’ and ‘Asx’ are repeated. Since the values are the same but entered in different capitals, it is best to opt for a replacement to all capital ‘ASX’ just as the rest of the values in the ‘Indication’ column.

After replacing all the invalid values to ‘Nan’ values, it is now time to make a decision about what to do with records that do not contain a value. For this, we need to compare the valid records with the invalid records containing the ‘Nan’ and calculate how much it will impact in the models. Records containing ‘Nan’ values are equivalent to 1.3% ***(Appendix, Section 1.0)*** of the whole record, for which it is better to drop those records as it will not make a big impact on the models.

Another issue with the data is that the ‘Random’ column is not unique. The ‘Random’ column was supposed to represent a unique record number for each record. If we count the amount of unique values in the ‘Random’ column, we get 84.4% unique and 19.6% of duplicate values. After knowing that there are about 20% repeated values, this defeats the purpose of the entire column. The ‘Id’ column has a similar purpose, which represents every patient’s unique id. However, having an identifier for each record and for each patient does not influence in the result when making the models. Since these columns do not help in getting a result, it is better to drop them.

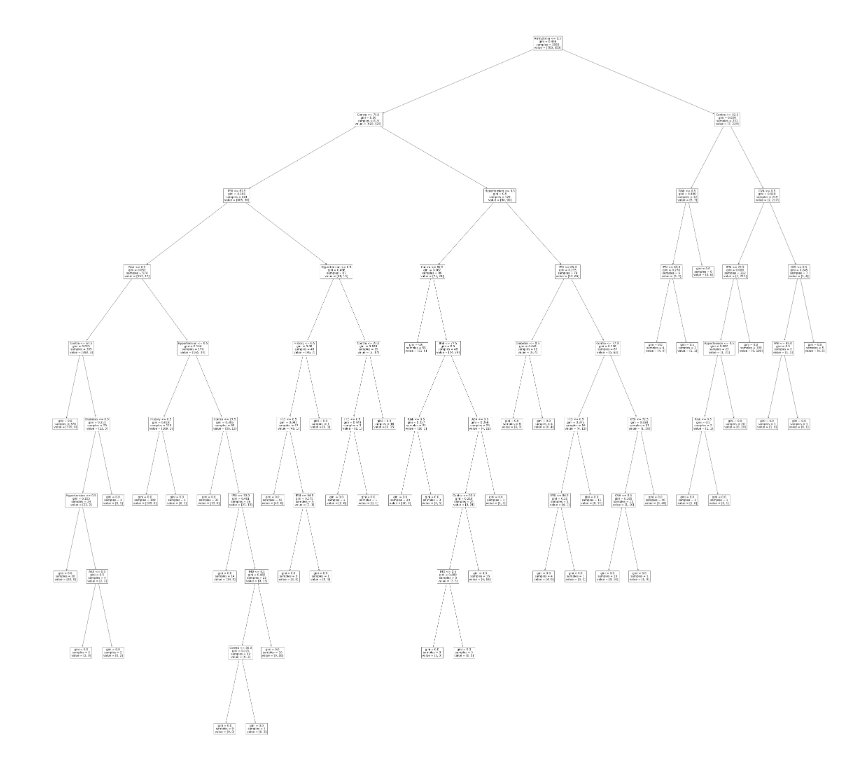
Now that we have all the records with ‘Nan’ values, the ‘Random’ column which contains repeated values and the ‘Id’ column which serve no purpose, we will proceed to drop them all and also replace the name of the ‘label’ column to ‘Risk’ to understand better the data included in that column.

**Modelling**

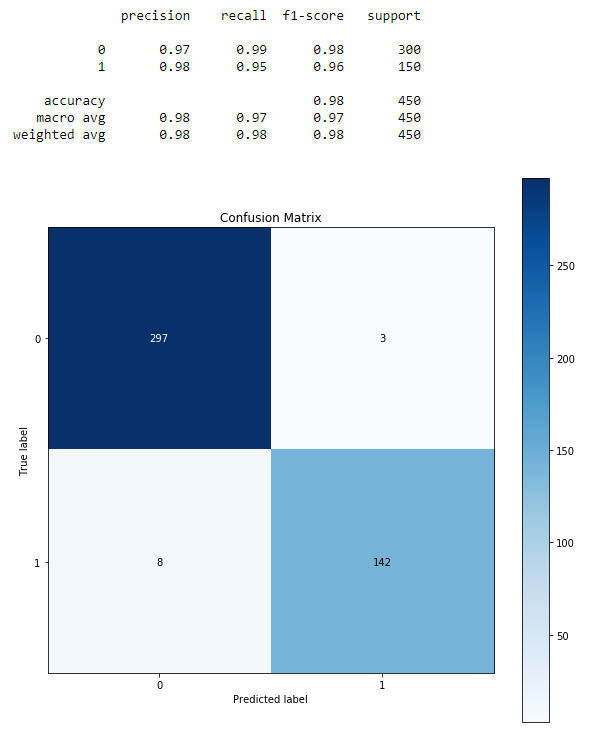
When using models, it is crucial to make the all the columns into binary. This means that we need to go back to the Data Preparation stage. In order to make these into binary, columns that contain ‘No’ and ‘Yes’ will be replaced with 0 and 1. These columns are: Diabetes, IHD, Hypertension, Arrhythmia, History and Risk. In case of the Indication column, because it has 4 unique values, it cannot be replaced with 0 and 1 only, for which we will split the column into 4 columns: A-F, ASX, CVA and TIA.

# Results

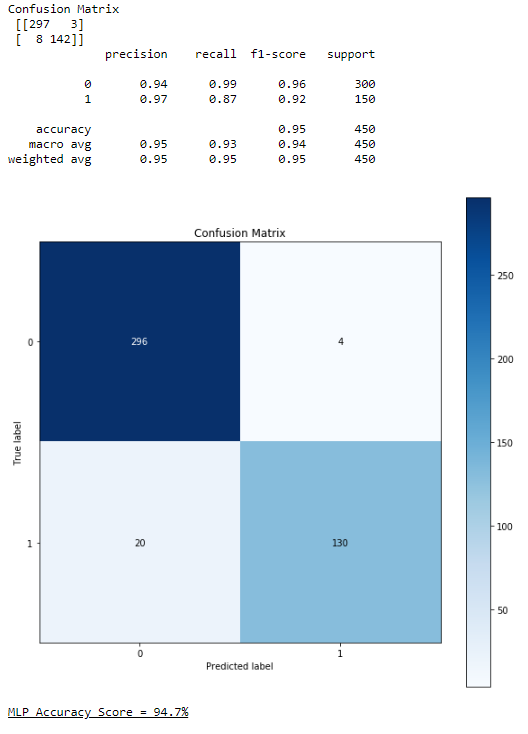
***Figure 1. Decision Tree (Decision Tree [Modelling] - Jupyter Notebook)***

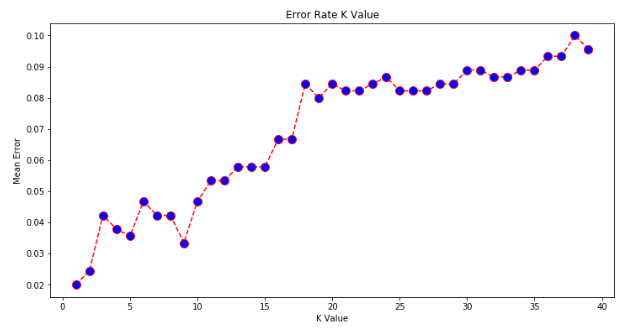
******

***Figure 2. Confusion Matrix based on the Decision Tree results ([Confusion Matrix based on the Decision Tree] - Jupyter Notebook)***

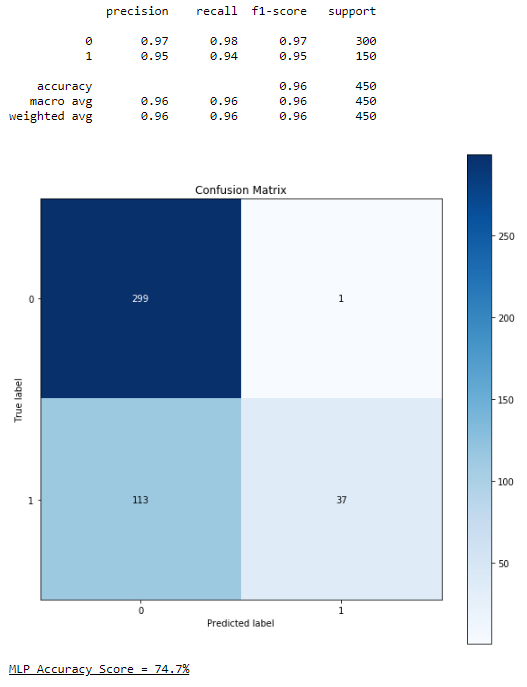
******

***Figure 3. Multi Layer Perceptron Confusion Matrix (Multi Layer Perceptron Confusion Matrix [Modelling] – Jupyter Notebook)***

******

***Figure 4. K-Nearest Neighbours Algorithm (KNN) ([K-Nearest Neighbours Algorithm] - Jupyter Notebook) ***

***Figure 5. K-Nearest Neighbours Algorithm Confusion Matrix (K-Nearest Neighbours Algorithm Confusion Matrix [Modelling] – Jupyter Notebook)***

******

# Evaluation & Discussion

A decision tree was chosen as it is easy to understand and is good for visualisation, but when it comes to handling big amounts of data it can be hard to see as it tents to overfit. The decision tree is supposed to evaluate every single possibility and conclude whether a patient is on risk or not. Due to the fact that the tree is unreadable, we cannot know what conclusion by looking directly at the tree ***(Figure 1)***.

This is why confusion matrices are useful and often used when making decision trees ***(Figure 2)***. This confusion matrix displays all the different possibilities combinations of results. These results can be: False-Negative, False-Positive, True-Positive and True-Positive. If a patient is predicted at risk but they are actually not, this would be displayed as a True-Negative. A False-Positive is when a patient is predicted not to be at risk and they are actually not at risk. True-Positive it is when a patient is predicted to be at risk and the patient is actually at risk. But, the riskiest prediction is when a patient is predicted to be not at risk but they are actually at risk. This last prediction can be dangerous as the patient will not be treated on time.

There are 2 more matrices that are calculated using different models. One of the matrices is based on the Multi Layer Perceptron, which has an accuracy of 94.7% **(Figure 3)**. The last model used is the KNN algorithm, which has an accuracy of 74.7% **(Figure 5)**.

As shown in the confusion matrices, it seems like the decision tree has a better accuracy compared to the rest of the models.

Compare accuracies and the false negatives for those confusion matrices

# References

Any references used throughout the report should be included here in Hull Harvard Style. If no references used, remove this section.

<https://www.sv-europe.com/crisp-dm-methodology/>

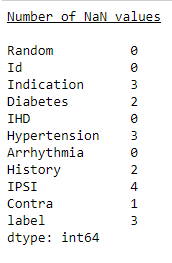
<https://www.datasciencecentral.com/profiles/blogs/crisp-dm-a-standard-methodology-to-ensure-a-good-outcome>

<https://www.springboard.com/blog/beginners-guide-neural-network-in-python-scikit-learn-0-18/>

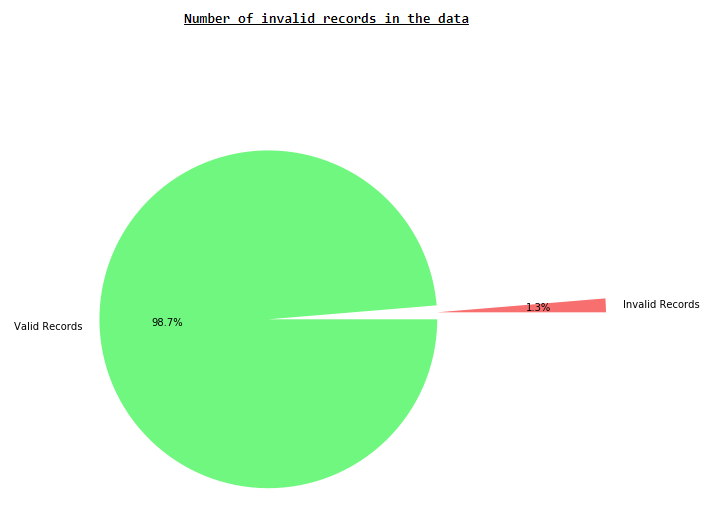
<https://stackabuse.com/k-nearest-neighbors-algorithm-in-python-and-scikit-learn/>

# Appendix

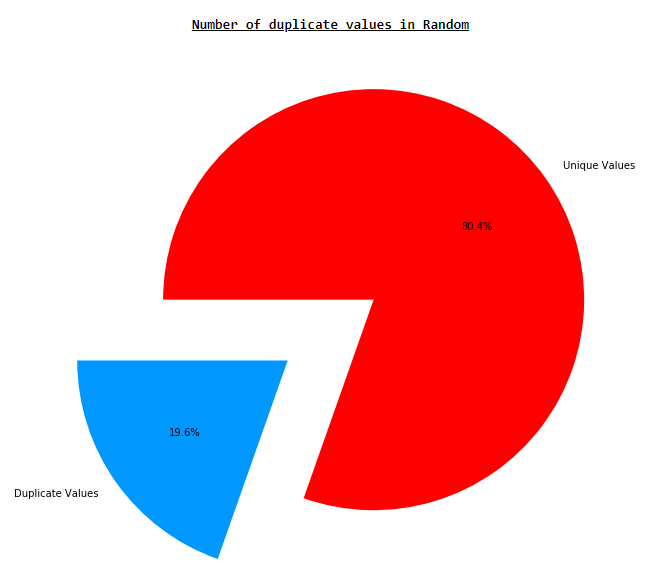
***1.0 - Table with number of ‘Nan’ values in each column before Data Cleaning (Number of Nan values before Data Cleaning [Data Understanding & Data Cleaning - Jupyter Notebook])***

******

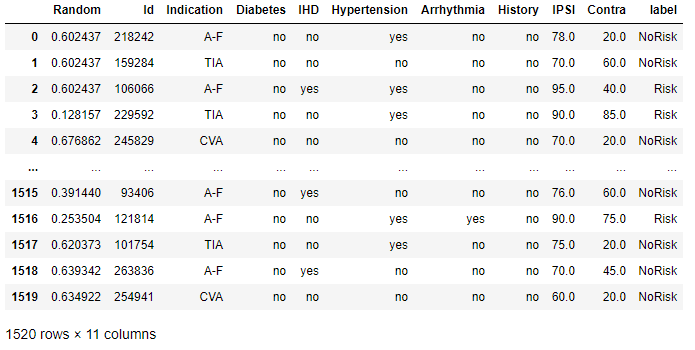
***1.1 - Pie chart representing Valid Records with Invalid Records (Number of invalid records in the data [Modelling - Jupyter Notebook])***



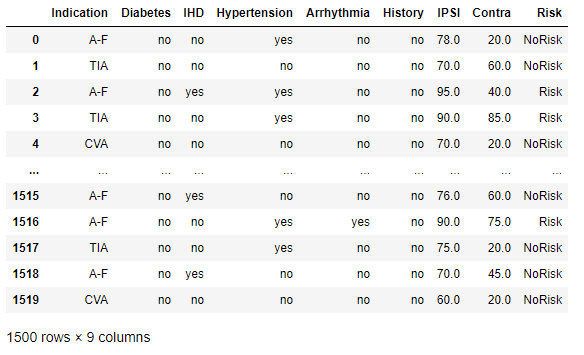
***1.2 - Pie chart representing duplicate values in Random Column ([Number of duplicate values in Random - Jupyter Notebook])***



***1.3 - Table with raw data (Raw data [Data Understanding & Data Cleaning - Jupyter Notebook])***

******

***1.4 - Table after Data Cleaning (Table after Data Cleaning [Dropping Records and Columns - Jupyter Notebook])***



***1.5 - Table after making all variables into binary (Complete Binary Replacement [Replacing Boolean values to Binary Values - Jupyter Notebook])***

