Data Mining and Decision Systems
600092
Assigned Coursework Report

Student ID: 561438

Date: 18 November 2019

Due Date: 12 December 2019

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 ***(Vorhies, 2016)***.

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 ***(Appendix, 1.0)***. 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 ***(Appendix, 5.0)***.

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 3.0)*** 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 ***(Identifying NaNs and Duplicates [Data Understanding and Cleaning], Jupyter Notebook),*** 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 ***(Appendix, 6.0)***.

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 ***(Appendix, 7.0)***.

Results

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

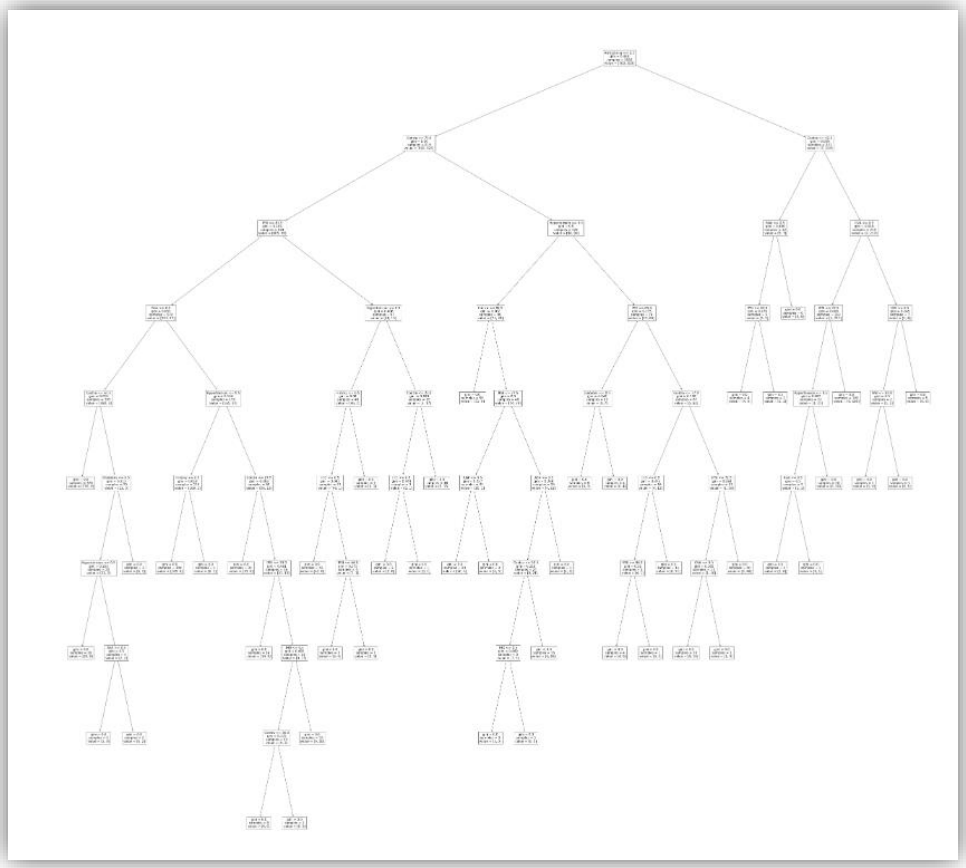
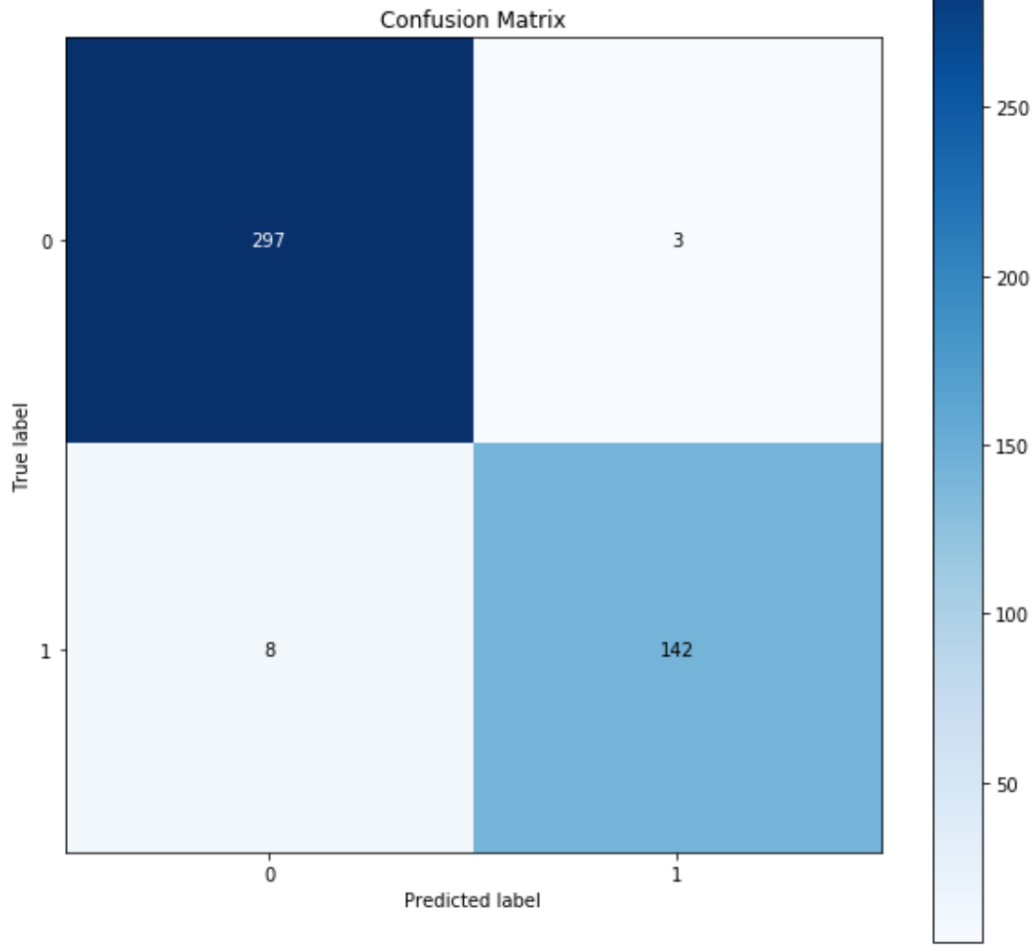


Figure 2. Decision Tree Confusion Matrix ([Confusion Matrix based on Decision Tree] - Jupyter Notebook)

	precision	recall	f1-score	support
0	0.97	0.99	0.98	300
1	0.98	0.95	0.96	150
accuracy			0.98	450
macro avg	0.98	0.97	0.97	450
weighted avg	0.98	0.98	0.98	450



Accuracy Score = 97.56%

Figure 3. Multi-Layer Perceptron Confusion Matrix (Confusion Matrix Based on MLP [Modelling] – Jupyter Notebook)

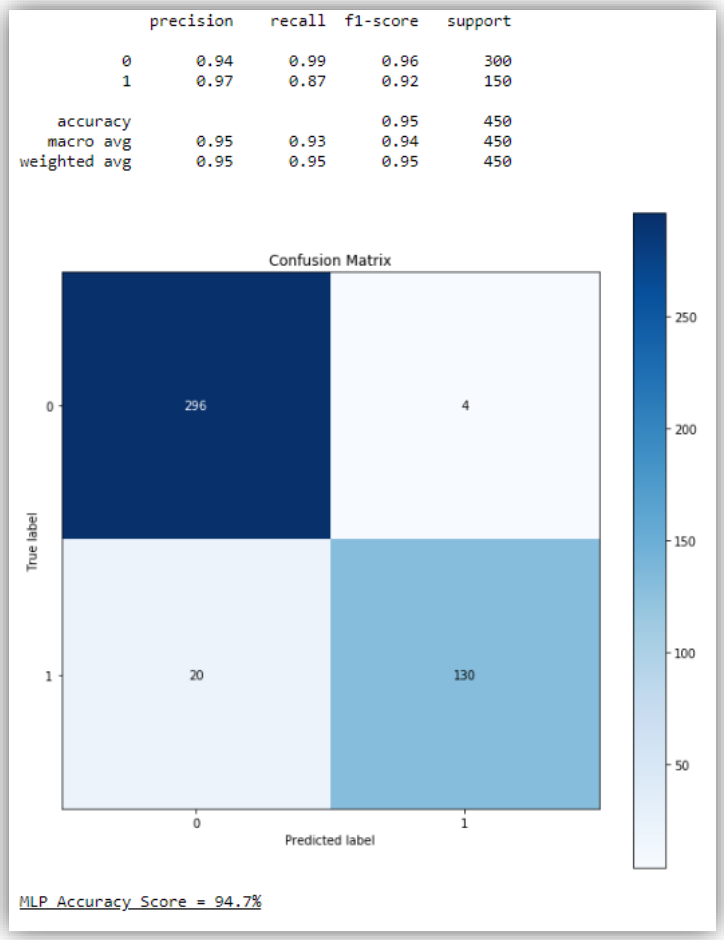


Figure 4. K-Nearest Neighbours Algorithm (KNN) (K-Nearest Neighbours Algorithm [Modelling] - Jupyter Notebook)

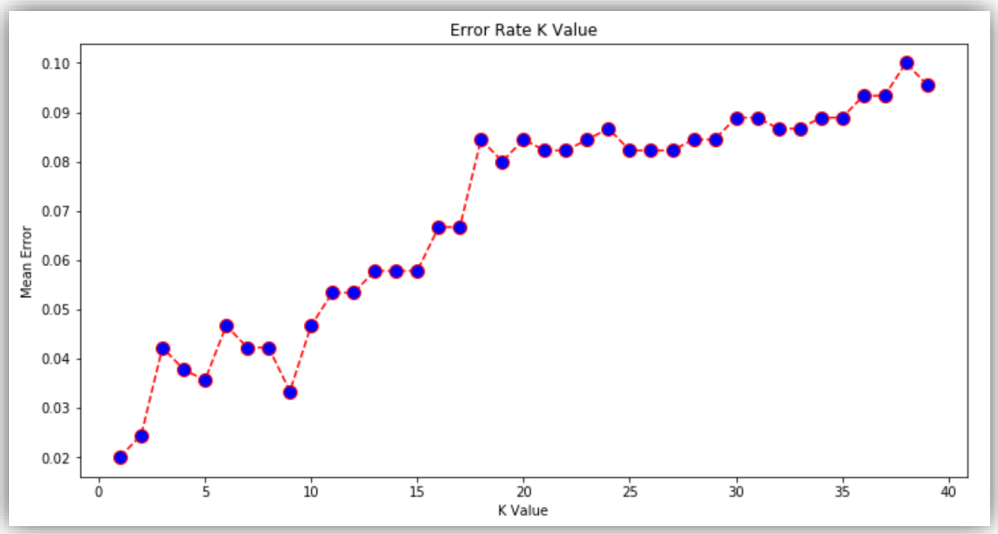


Figure 5. K-Nearest Neighbours Algorithm Confusion Matrix (Confusion Matrix Based on KNN Algorithm [Modelling] – Jupyter Notebook)

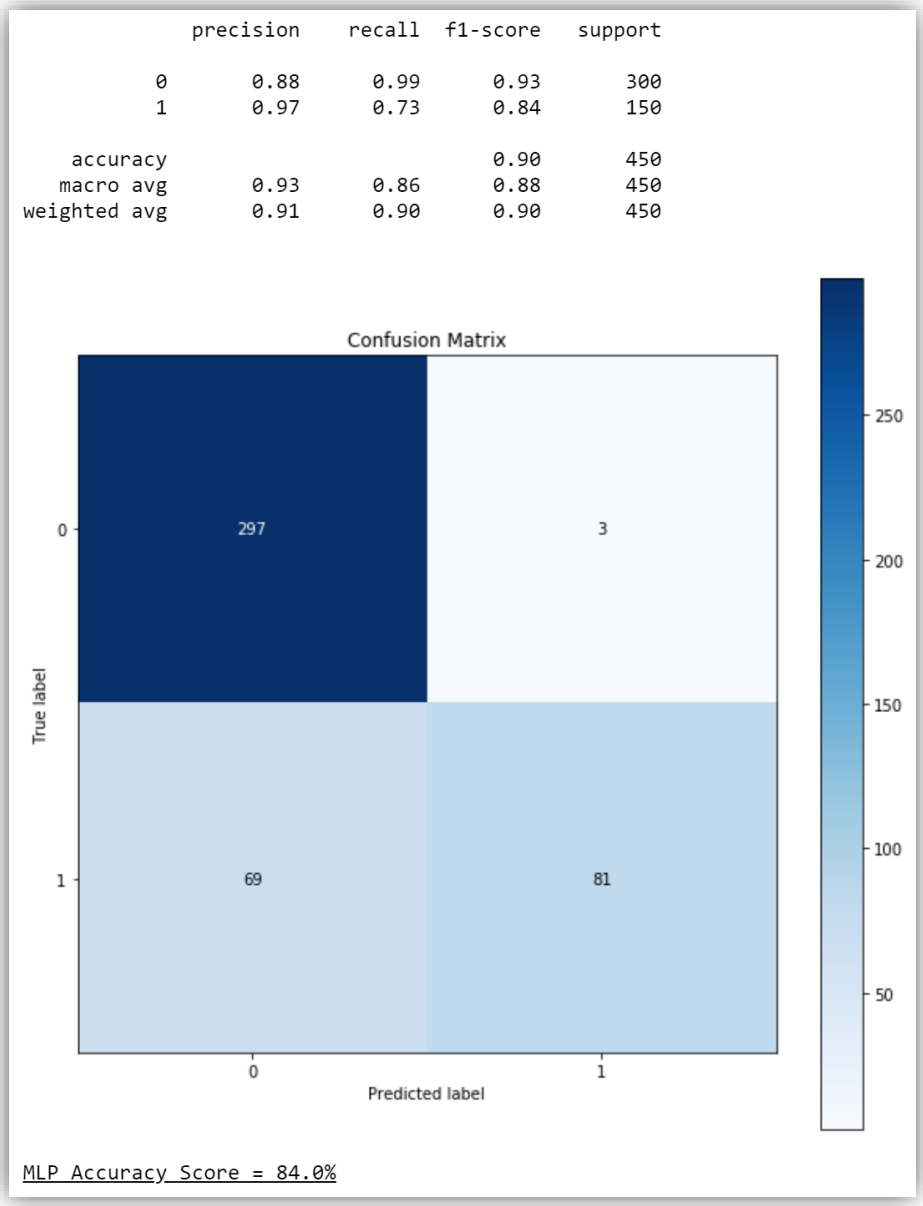


Figure 6. All Confusion Matrices Comparison Table

Model	Accuracy Score	True Negative	True Positive	False Negative	False Positive
Decision Tree	97.56%	297	142	8	3
MLP	94.7%	296	130	20	4
KNN Algorithm	84.0%	297	81	69	3

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. Because the tree is unreadable, we cannot know what conclusion the model has come to by looking directly at the tree ***(Figure 1)***.

Therefore, 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 outcome is dangerous as it predicts something as it will not happen, but, when the time comes, it happens. Which translated to this scenario it would mean that the patient is not at risk when they are at risk. And the end result would be the patient to being treated on time.

Another model used for this is the Multilayer Perceptron. This model has a 94.7% accuracy, which so far it makes this model the 2nd best model.
A Multi-Layer Perceptron (MLP) is artificial neural network which is composed of more than one perceptron. These are composed of an input and output layer. The input layer receives a signal, whereas the output makes a decision/prediction of the input data. And in between those layers, there are an arbitrary number of hidden layers which are the real computational engine of this model. In this situation, when imputing the data, we will predict if a patient is at risk of death or not.

This model's confusion matrix is quite similar to the Decision Tree's confusion matrix. Almost 95% ***(Figure 3)*** of accuracy compared to the 97% ***(Figure 2)*** of accuracy obtained by the Decision Tree. This makes the MLP model to be on the 2nd place right after the Decision Tree. The problem with the MLP model is that it predicts 20 false negatives, whereas on the other hand, the Decision Tree predicts 8 false negatives. As mentioned before, false negatives are the riskiest of them all as it predicts something as not happening but as time goes by, the reality is that it actually happens. Which makes this model to be the least option to be used so far.

The last model used is the K-Nearest Neighbour. It has scored an accuracy of 84% ***(Figure 5)***. Which makes this model the 3rd out of the 3 models. This model works based on minimum distance from the query instance to the training sample to determine the K-Nearest Neighbour.
From the graph ***(Figure 4)*** we can see that higher the "K Value" is, the higher the "Mean Error" becomes.

The confusion matrix for this model predicts 69 false negatives ***(Figure 5)***, which is an extremely high number compared to 8 false negatives predicted by the decision tree ***(Figure***

2) and 20 false negatives predicted by the MLP ***(Figure 3)***. This makes the K-Nearest Neighbour the worst out of the 3 models.
The reason this model performs way different than the other two is because it cannot handle a large amount of data as it becomes difficult for the algorithm to calculate the distance in each dimension ***(Robinson ,2016)***.

After taking all these steps and making all the models, we conclude that the Decision Tree has performed the best out of the 3 models by comparing their highest accuracy (97.56% [Figure 2]). The decision tree comes first with a difference of 2.86% compared to the Multilayer Perceptron, and 13.56% of a difference compared to the K-Nearest Neighbour. What really benefits from this model and its accuracy is the fact that it has the lowest False Negatives (8). This means that the amount of people who are at risk of death but they were predicted not to be at risk is really low ***(Figure 6)***.

It is possible to improve the accuracy of the model. To achieve this, the model will need more training data. By giving more data to the models, it will be able to evaluate more possible outcomes which would allow it to predict the result with more accuracy.
Another way of obtaining a better accuracy is by converting all the erroneous records which contained null values or spaces to actual values. This will be done by imputing them with mean, media and mode. This will bump the percentage of the accuracy.
Last but not least, another way of getting a better accuracy is by using more algorithms to see if other models are able to predict an outcome with a better accuracy score.

References

William, Vorhies. (2016). *CRISP-DM – a Standard Methodology to Ensure a Good Outcome*. [online] Datasciencecentral.com. Available at: https://www.datasciencecentral.com/profiles/blogs/crisp-dm-a-standard-methodology-to-ensure-a-good-outcome [Accessed 6 Jan. 2020].

Robinson, S. (2018). *K-Nearest Neighbors Algorithm in Python and Scikit-Learn*. [online] Stack Abuse. Available at: https://stackabuse.com/k-nearest-neighbors-algorithm-in-python-and-scikit-learn/ [Accessed 6 Jan. 2020].

Appendix

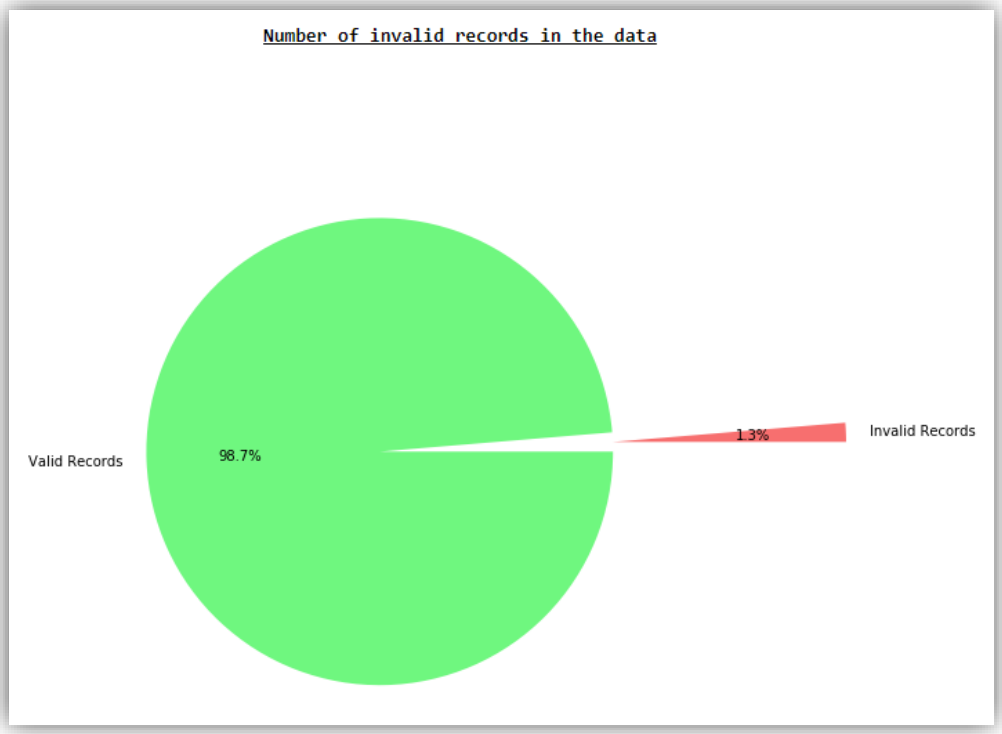
1.0 - Total Unique Values Table (Reading/Importing and Analysing Raw Data [Data Understanding & Data Cleaning])

```
Total Unique Values

Random          1222
Id              1520
Indication         5
Diabetes           2
IHD                2
Hypertension       2
Arrhythmia         2
History            2
IPSI              29
Contra            25
label              3
Name: unique, dtype: object
```

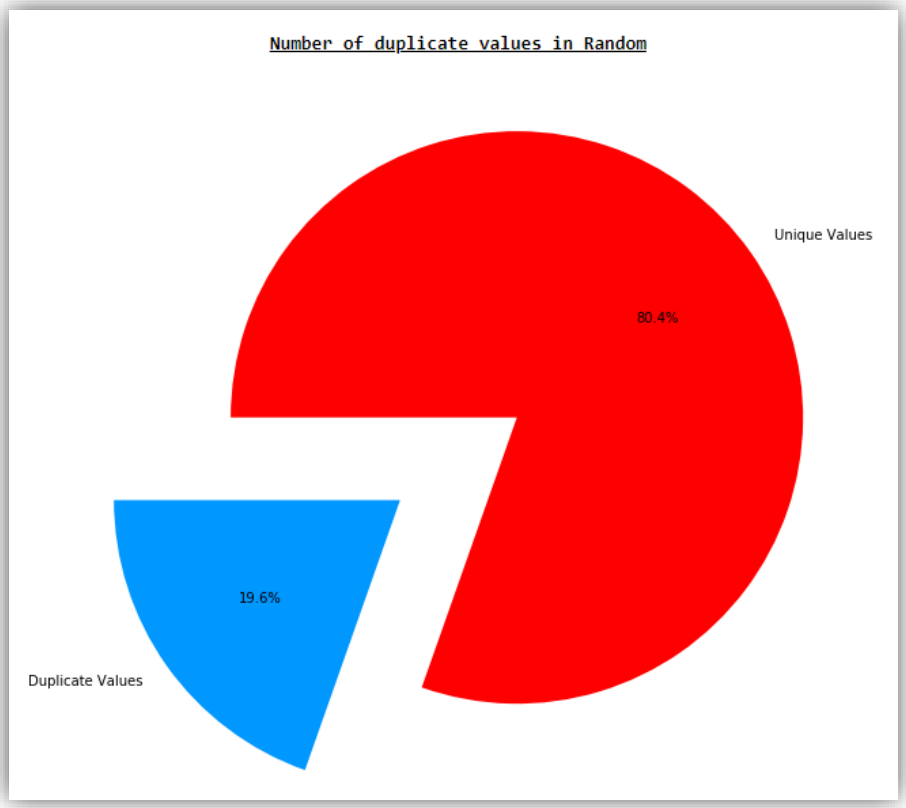
2.0 - Table with number of 'Nan' values in each column before Data Cleaning (Reading/Importing and Analysing Raw Data [Data Understanding & Data Cleaning - Jupyter Notebook])

```
Number of NaN values

Random          0
Id              0
Indication      3
Diabetes        2
IHD             0
Hypertension    3
Arrhythmia      0
History         2
IPSI            4
Contra          1
label           3
dtype: int64
```

3.0 - Pie chart representing Valid Records with Invalid Records (Number of invalid records in the data [Data Understanding & Data Cleaning - Jupyter Notebook])



4.0 - Pie chart representing duplicate values in Random Column (Number of duplicate values in Random [Data Understanding & Data Cleaning - Jupyter Notebook])



5.0 - Table with raw data (Raw data [Data Understanding & Data Cleaning (Reading/Importing and Analysing Raw Data) - Jupyter Notebook])

	Random	Id	Indication	Diabetes	IHD	Hypertension	Arrhythmia	History	IPSI	Contra	label
0	0.602437	218242	A-F	no	no	yes	no	no	78.0	20.0	NoRisk
1	0.602437	159284	TIA	no	no	no	no	no	70.0	60.0	NoRisk
2	0.602437	106066	A-F	no	yes	yes	no	no	95.0	40.0	Risk
3	0.128157	229592	TIA	no	no	yes	no	no	90.0	85.0	Risk
4	0.676862	245829	CVA	no	no	no	no	no	70.0	20.0	NoRisk
...	...	...	...	...	...	...	...	...	...	...	...
1515	0.391440	93406	A-F	no	yes	no	no	no	76.0	60.0	NoRisk
1516	0.253504	121814	A-F	no	no	yes	yes	no	90.0	75.0	Risk
1517	0.620373	101754	TIA	no	no	yes	no	no	75.0	20.0	NoRisk
1518	0.639342	263836	A-F	no	yes	no	no	no	70.0	45.0	NoRisk
1519	0.634922	254941	CVA	no	no	no	no	no	60.0	20.0	NoRisk

1520 rows × 11 columns

6.0 - Table after Data Cleaning (Table after Data Cleaning [Dropping records Containing NaNs - Jupyter Notebook])

	Indication	Diabetes	IHD	Hypertension	Arrhythmia	History	IPSI	Contra	Risk
0	A-F	no	no	yes	no	no	78.0	20.0	NoRisk
1	TIA	no	no	no	no	no	70.0	60.0	NoRisk
2	A-F	no	yes	yes	no	no	95.0	40.0	Risk
3	TIA	no	no	yes	no	no	90.0	85.0	Risk
4	CVA	no	no	no	no	no	70.0	20.0	NoRisk
...	...	...	...	...	...	...	...	...	...
1515	A-F	no	yes	no	no	no	76.0	60.0	NoRisk
1516	A-F	no	no	yes	yes	no	90.0	75.0	Risk
1517	TIA	no	no	yes	no	no	75.0	20.0	NoRisk
1518	A-F	no	yes	no	no	no	70.0	45.0	NoRisk
1519	CVA	no	no	no	no	no	60.0	20.0	NoRisk

1500 rows × 9 columns

7.0 - Table after making all variables into binary (Replacing Boolean Values to Binary [Modelling - Jupyter Notebook])

	Diabetes	IHD	Hypertension	Arrhythmia	History	IPSI	Contra	Risk	A-F	ASX	CVA	TIA
0	0	0	1	0	0	78.0	20.0	0	1	0	0	0
1	0	0	0	0	0	70.0	60.0	0	0	0	0	1
2	0	1	1	0	0	95.0	40.0	1	1	0	0	0
3	0	0	1	0	0	90.0	85.0	1	0	0	0	1
4	0	0	0	0	0	70.0	20.0	0	0	0	1	0
...	...	...	...	...	...	...	...	...	...	...	...	...
1515	0	1	0	0	0	76.0	60.0	0	1	0	0	0
1516	0	0	1	1	0	90.0	75.0	1	1	0	0	0
1517	0	0	1	0	0	75.0	20.0	0	0	0	0	1
1518	0	1	0	0	0	70.0	45.0	0	1	0	0	0
1519	0	0	0	0	0	60.0	20.0	0	0	0	1	0

1500 rows × 12 columns