# Reflection

## Reflection

The aim of this project was to use patient medical data to successfully predict the presence of liver disease. The hepatitis C dataset, from the MCI ML Repository, was discovered quite early on; however, the classification model (and the methods used) evolved considerably over time, when new challenges presented themselves. I devised this project alone, and completed all of the coding in both R and Python. Many resources on the internet were used to help with the research and implementation of the models – these have been included as comments in certain section of code. In the literature of this field, I read how random forests (RF), support vector machines (SVM), and other classifiers had been used to classify/predict medical diseases. This provided a strong rationale for their implementation as models in the project.

After embarking on a literature search, I discovered that the field of medical classification, in machine learning, is a burgeoning (and fast-developing) space. Many of the papers, and some of the packages, used were only created in the last few years. It is a large field, with a lot of literature. A challenge was how to choose which methods to use – there are many promising ones.

Furthermore, none of the papers had examples of code used. Therefore, a lot of research was conducted on how to implement the methods used in other papers. There is a mixture of both R and Python based resources in this field and, ultimately, both languages were used in this project.

The pre-processing, and data imputation, stage took a long time. This is because various methods were used, and it was important to check if their implementation was appropriate for the specific data-types in the data (continuous and categorical). The presence of categorical data was also a challenge, and this was appropriately dealt with during the project.

Due to the solo nature of this project, I decided that using github was not necessary; however, in the future I may consider its use.

The mathematics, and theory, behind the models is discussed in the main project; as well this, there are references to scientific literature with more in-depth explanations of some concepts. Notions from the lecture notes were useful in the implementation of principal component analysis (PCA). PCA is often performed using singular value decomposition (SVD). PCA was implemented in this project, so I shall briefly outline some ideas of PCA below.

### PCA

In the collection of data, one typically finds that measurements – in the form of individual experiments – are collected, as features, and arranged as row vectors. A number, N, of experiments are performed, each measuring 'n' features. The data is then arranged into an $N \times n$ matrix, $\mathbf{X}$. Now, let us assume that each of the columns are centred – this can be done by subtracting the mean; now the means are zero. Typically $N > n$, but this is not always the case.

The $n \times n$ covariance matrix, $\mathbf{S}$, is given by $\mathbf{S} = \mathbf{X}^\top \mathbf{X}/(n-1)$. Because it is symmetric, it can therefore be diagonalised. The diagonalisation is given by $\mathbf{S} = \mathbf{V}\mathbf{D}\mathbf{V}^\top$. Here, $\mathbf{V}$ is the eigen-matrix, where each column of the matrix is given by an eigenvector, $\mathbf{v}_i$. Furthermore, $\mathbf{D}$ is the matrix whose diagonal is given by the eigenvalues, $\lambda_i$, of the eigenvectors in a monotonically decreasing way. We call each of these eigenvectors *principal axes*. *Principal components*, or PCs, are defined to be the projections of the data values onto these axes. These are essentially a bunch of new features that describe the data. The $m$-th PC is the $m$-th column of $\mathbf{XV}$. Note that the coordinates of the $k$-th data value in the new transformed principal component space are determined by the $m$-th row of $\mathbf{XV}$

## The overall method of the project was as follows:

1. Finding the data
2. Exploratory data analysis, PCA, factor analysis of mixed data (FAMD)
3. Imputation of missing values / pre-processing of data – including mean-centring, normalising, and creating dummy variables
4. Over-sampling, and balancing, the data using SMOTE-NC (this was done after an initial evaluation of the models indicated that the data needed to be more balanced)
5. Training models using cross-validation and hyper-parameter optimisation
6. Model Evaluation
7. Conclusion

The performance metrics used in the assessment of the models were accuracy, sensitivity, and specificity – these are discussed at length in the project. These are the most common measures of performance in medical classification models. Nevertheless, there are more measures (such as $ precision = TP/ (TP +FP)$, *F1-score* – the harmonic mean of sensitivity and precision – and others) that could have been used, and certainly should be used, in the future.

## Main challenges, and overcoming them:

1. Imbalanced nature of the data: any classification model requires a broad range of data to be successful – crucially, the minority classes (hepatitis, fibrosis, and cirrhosis) need an adequate number of data values. The absence of this can result in a biased (or simply poor) classifier. Research into this problem led to the implementation of over-sampling, using the method of SMOTE-NC (in python); this is described in the project, and it was shown to be mostly beneficial.
2. As this is an emerging field in data-science, a lot of the resources necessary in the project were divided between R and Python. During the SMOTE-NC stage, it was necessary to use python. This meant having to write the data into a file, reading it in a custom-made python program, and then reading it back into the R markdown file. Although this was fiddly, and an older version of one packages was required in the python code, this challenge was successfully overcome.
3. Another drawback was the inability to use PCA and FAMD for dimension reduction. Intrinsically, using 11 or 12 features in a classifier can lead to poor results due to the presence of noise in the data (vide 'the curse of dimensionality'). It can be hard to determine the best features that contribute the most in disease prediction. 'Shapley values' can be used to rank the most important features for predicting the target category, but the package is not currently available in R, and there was simply not enough time to implement this feature. In the future, this is certainly a method that should be considered – although, it is very nuanced [1].
4. All the data within the dataset are from the same clinical centre – this can lead to certain geographical and socio-economical biases which might skew the predictions. In the future, we should use a greater amount of data from a larger number of locations to try to glean the true nature of the features, and their ability to detect liver disease, inthe wider population.

## Final Remarks

To change (or improve) the model – assuming the data are to remain the same – we could implement the use of Shapley values to rank the importance of features. Then, train a model based on the most important ones. Furthermore, the method of hyper-parameter optimisation used in this project was very basic. In the future, this should be made more robust. A way of doing this could be the use of programs such as 'Optuna.' Additionally, more models should be implemented, such as K-nearest neighbours (K-NN), or neural

networks. Doing this could allow us to find a more optimum classification model for our data. The use of different programming languages, and therefore having to write data and read it again in another program, has made this project less seamless. Nevertheless, instructions in the README.md file, and copious amounts of comments should make this program easy to run on any machine.

Ultimately, the project was fairly successful. The best performing model was the hyper-perameter tuned RF model with SMOTE-NC, with the results (accuracy, sensitivity, specificity) = (97.8%, 0.863, 0.994). The challenges that presented themselves were not ignored, and an attempt was made – often successfully – to overcome them. Many very clear, and illustrative, visual plots were used to explain the data during the EDA and show how the program functions. One such example is the plots showing the tuning of the hyper-parameters. Another example is the scree plot, showing clearly the variance explained by each of the principal components. I have left plenty of comments in an attempt to provide clarity, and lucidity, to the code.

# References

1. Fryer D, Strümke I, Nguyen H (2021) Shapley values for feature selection: The good, the bad, and the axioms