UCI HEART DATA SET ANALYSIS

Setup:

The aim of this project is to address one of the key problems in medical science. As a continuation of assignment stage 1, in stage 2 a deeper aspect of the **Cleveland Heart disease dataset** [1] will be studied and following questions will be answered by means of different data science techniques. The reliability of the findings will be further justified with series of experiments.

Research questions:

a. Compare multiple classification techniques and identify the best classifier to predict effectively if a person is suffering from heart disease or not suffering from heart disease at all

b. Though the Cleveland database is mostly used widely by ML researchers, with availability of multiple data of exact similar structure, it would be interesting to know if the classifier identified in section a, could perform similar task with similar accuracy on other datasets.

For the first question, five different classification techniques i.e., K-Nearest Neighbour, Support Vector Machine, Naïve Bayes, Logistic regression and Random Forest were proposed and implemented. The Cleveland dataset was split into 60% train set, 20% validation set and 20% test set. The validation set was used to tune the hyper parameters. For each hyper parameter, a series of values were tested, both training errors and generalisation errors were plotted, and lastly optimal hyper parameters are chosen. Once final optimal parameters are decided, to test the reliability of these parameters and model performances, 10-fold stratified cross validation was performed, i.e., instead of testing on one set of validation data, the model parameters are tested on 10 different folds. The folds are created ensuring that each fold has the same proportion of observations with the class outcome value. Finally, the optimal parameters are applied on the test dataset to detect final model performance. To understand if any model performs significantly better than another model, several paired-ttest were conducted and the results were presented. With class labels representing whether a person is suffering from heart disease (160 instances: class 0) or not (137 instances: class 1), this data is relatively balanced, but both accuracy and F1-score will still be used to measure the effectiveness.

The figure below (figure 1) gives a quick overview of all the features the *Cleveland dataset*, where any relationship among continuous feature variables, and also the distribution of categorical features could be realised. Since around 1% of the data had missing values, it was discarded from the analysis. It is to be noted that colour coding of these figures is based on classes that represent whether or not a person is suffering from heart disease or not (blue: class 0, orange: class 1).

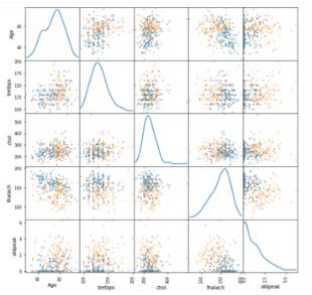


Figure 1: Scatter Plot of continuous features

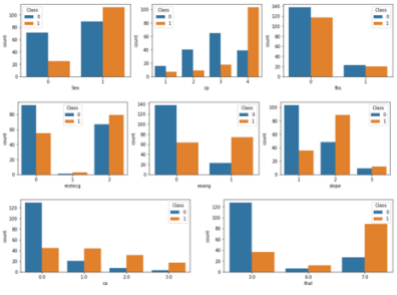


Figure 2: Distribution of categorical features

For the second question, three secondary datasets were also explored, **Hungarian data** (with 294 instances), **Switzerland data** (with 123 instances) and **Long-beach VA data** (with 200 instances) with *exact* same feature sets available. These datasets will be used to assess how the final models trained using **Cleveland data** perform on other similar datasets. Unlike Cleveland data, missing cases proportion in these datasets are very high, so discarding those instances would lead to major information loss. It was decided that mean imputation will be done for variables with more than 50% data available and replacement with 0 for variables with less than 50% of data available.

Approach:

Data Preparation

Once the data structure was comprehended, the next step was to prepare the data for the models. All the continuous features are then *standardized*, which helps to normalise the data within a particular range. Sometimes, it also helps in speeding up the calculations in an algorithm [2]. Also, few of the proposed methods like KNN, SVM etc are distance-based algorithms, which are highly impacted by scale of the features. So even though feature scaling does not guarantee performance improvement of the model, it is ideal to do so.

The dataset in hand has *13 features* variables with *~300 instances*, which makes it reasonable for any modelling exercise and thus rigorous feature engineering such as feature selection, dimension reduction with PCA etc. was not required for this dataset. But it would be very helpful to understand importance of various feature variables in this analysis. To start with, a *correlation matrix* was created (figure 10 in appendix) and no significant correlation was found among features. To further analyse the features a *logistic regression* was run using the primary dataset, and coefficients are used to determine how important a feature variable is. In a logistic regression, *co-efficient determines whether a change in a feature variable*

makes the event more likely or less likely. As demonstrated in figure 3, 'fbs' i.e., fasting blood sugar, is one of the least important features, thus it was decided to drop this variable going forward.

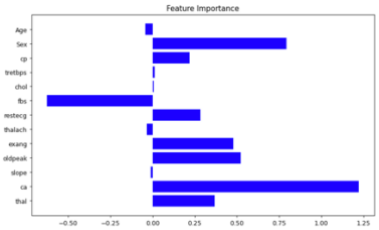


Figure 3: Feature Importance

Classification Algorithms and Implementation

K-Nearest Neighbour (KNN) is one of the widely used supervised machine learning method. It is a non-parametric model which means it does not assume any underlying data distribution and this makes it beneficial with many real-world datasets as most of these data do not follow any mathematical theoretical assumption. The only assumption KNN makes is that 'similar things are closer to each other'. This algorithm is very easy to implement and it does not build any predictive model, thus there is no learning phase. The prediction algorithm starts by choosing a value of 'k' followed by calculating distance of the previously unseen instance from every instance in the training dataset. Different distance metrics are used in this step which include Euclidean distance, Hamming, or Manhattan. Once all the distances are calculated, the distance array was then sorted in ascending order and top k distances were chosen. To decide on the prediction class, a majority vote was conducted and finally the previously unseen instance was assigned to a class that is the most appearing class of the first k observation in the sorted distance array. Even though KNN is a versatile algorithm, it has few drawbacks. To make predictions KNN uses the entire dataset which makes it a lazy learner. This algorithm is distance-based algorithm thus highly affected by scale of the variables. KNN does not perform well with high dimensional data. The data instances need to grow exponentially with increase in dimension for KNN to perform well without overfitting.

Choosing the number of neighbours in KNN is very crucial. A smaller value of k might be influenced by noisy points whereas larger value of k might be under the risk that the neighbourhood may include points from other classes. To select the ideal k a series of value were tested, both training and generalisation errors were plotted, and optimal k was identified to be *9* (figure 11 shown in appendix). Another key factor that needs be considered while running this algorithm, is the distance matrix used. Empirically the *Euclidean distance* measurement is widely considered to have strong performance and is a popular choice. All fine-tuning of the hyper parameters was tested using validation set and once all the optimal parameters are identified those were then applied on the test dataset. This was followed for all the algorithm.

Support Vector Machines is another classification technique that transform points in space to maximise the target outcomes are as clearly separated as possible. A support vector machine (SVM) is also a supervised machine learning model which is essentially designed for a binary classification problem, where the algorithm tends to find a hyperplane in an N-dimensional space (N — the number of features) that distinctly classifies the data points. This classification method can also be used for multi class problem by breaking it down to multiple binary classification cases, which is also called one-vs-one. It is to be noted SVM is limited when the target variables are not reasonably separable. The hyper parameters of interest in this algorithm are 'kernels', and 'C' to improve model performance. SVM algorithms use a set of mathematical functions that are defined as the kernel. The function of kernel is to take data as input and transform it into the required form. These functions can be different types. For example, linear, nonlinear, polynomial, radial basis function (RBF), and sigmoid. In this project after testing different kernels, it was observed that *linear* kernel was producing the least model errors. C is a hyperparameter to control error. Ideally low c value tends to show high training accuracy but with very low C there is a risk of overfitting the model. While testing a series of C values, *10* was identified as the optimal C value. (figure 13-15 in appendix).

Naïve Bayes is a probabilistic approach to machine learning algorithm based on Bayes theorem. Bayes theorem is a mathematical formula that describes how often an event happens given a second event happens(posterior), when the following are known:

1. how often the second event happens given the first event happens (Likelihood)
2. how likely is the first event is on its own (Prior probability)
3. how likely is the second event is on its own (Normaliser)

Formula: $P(A|B) = P(B|A)*P(A)/P(B)$

Naïve bayes models are extremely fast and simple classification and very useful for high dimensional data. The name "naïve" comes from the fact that this approach assumes independence of the feature variables. This assumption may not hold for certain dataset and thus it is advised not to use Naïve Bayes if the feature variables are related. Also, there aren't many tuneable parameters to improve this model.

Logistic Regression is a generalised extension of the linear regression model, allowing for a binary output. The Logistic Regression models the conditional probability of the output as a function of the independent variables, estimating model parameters by their maximum likelihood. This algorithm is popular to implement as it does not require strong assumptions

regarding distributions of classes in feature spaces and is therefore relatively easy to interpret. Furthermore, the determination of the variable's coefficients is particularly beneficial for ascertaining the relative importance of independent variables. This is beneficial for preventing overfitting as well as aiding in understanding. Although Logistic Regression has many benefits it also has some limitations. Notably, although logistic regression is less inclined to overfitting, it can result in overfit with high dimensional datasets. Regularisation remains an option to reduce the impact of overfitting. On the other hand, regularization can be assumed as adding bias if the model suffers from variance but again too much of bias will result in underfitting [3]. Another limitation is the assumption of linearity between the dependent variable and the independent variables.
In this study a wide range of C values were tested using both l1 and l2 regularization parameters and it produced similar and lowest error at C = 1. (figure 18 shown in appendix).

A **random forest** classifier is a non-parametric model that is made up of an ensemble of decision trees and thus makes no assumption about the type of data. The perception behind a decision tree is very simple, it splits the data wherever the most information is gained, then it continues to split the data iteratively until it reaches a point where all classes are separated or until it reaches a given depth [4]. However, it is important that the trees making up the random forest are not correlated, so that it is not constantly creating same trees that only use certain section of the data or, so the results are not governed by only a small number of the features. The algorithm behind a random forest works in a way that will reduce correlation between trees and will maximise the accuracy of each tree. To avoid a correlated tree a bootstrapped sampling of the data was done to build each tree, i.e., the training data will be resampled with replacement to build each individual tree [5]. Whilst this is a good tool to reduce the similarity between trees, there is another trick used. At each split, instead of picking the best feature to split on out of all features, only a subset of features is randomly made available to be split on. This will reduce the similarity between trees, as they will not split on the same feature at every iteration.
There are quite a few hyper parameters that can be tuned in random forest. The number of estimators determines how many trees to be used in the algorithm and after testing several values it was observed 1200 trees would be optimal. Next, focus was on the maximum depth of the trees that the algorithm should allow and again after testing a series of values it was detected that the ideal number to be 5 and final parameter of interest the ideal maximum number of features that the model consider while looking for split and it was realised the number to be 20% of the feature variables. (figure 20-22 shown in appendix).

Results:

In this section all the final models with optimal parameters are evaluated. In below figure (4 and 5) it is shown how the final model accuracy and F1 Score improved from *benchmark models* with *default parameters*.

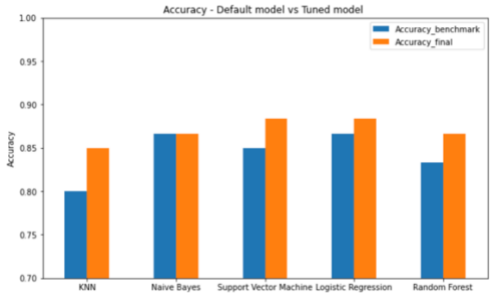


Figure 4: Accuracy comparison of benchmark model vs final model

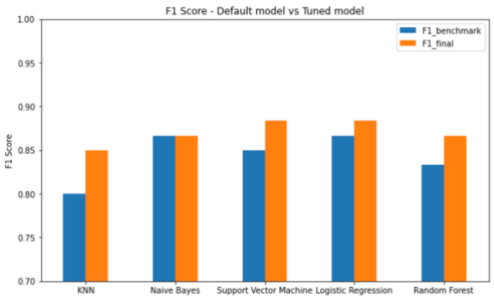


Figure 5: Accuracy comparison of benchmark model vs final model

Each of the model report and confusion matrix provided in Appendix. (refer figures, 12, 16, 17, 19 and 23)

To defend the reliability of the final model parameters a 10-fold stratified cross-validation was conducted (see figure 6). From this experiment following observations were made:

- For *KNN*, the maximum accuracy obtained was 91.6%, minimum accuracy obtained was 65.2% and mean accuracy was 79.2% whereas our final model accuracy was 85%
- For *SVM*, the maximum accuracy obtained was 91.7%, minimum accuracy obtained was 70.8% and mean accuracy was 80.9% whereas our final model accuracy was 88.3%
- For *Naïve Bayes*, the maximum accuracy obtained was 95.8%, minimum accuracy obtained was 66.6% and mean accuracy was 81.8% whereas our final model accuracy was 87%
- For *Logistic regression*, the maximum accuracy obtained was 95.8%, minimum accuracy obtained was 73.9% and mean accuracy was 82.6% whereas our final model accuracy was 88.3%
- For Random Forest, the maximum accuracy obtained was 91.6%, minimum accuracy obtained was 73.9% and mean accuracy was 81.3% whereas our final model accuracy was 87%

For model comparison, ROC curves were also generated and plotted for all the models. A ROC curve visualizes True positive rate and false positive rates. (see figure 7).

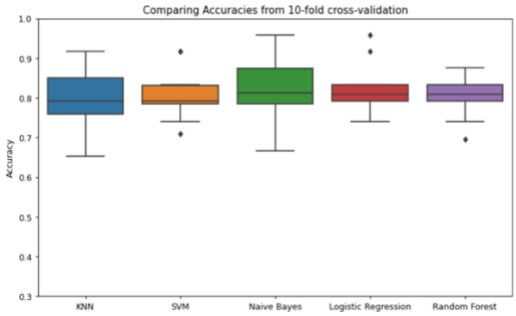


Figure 6: Model comparison with 10-fold stratified cross validation

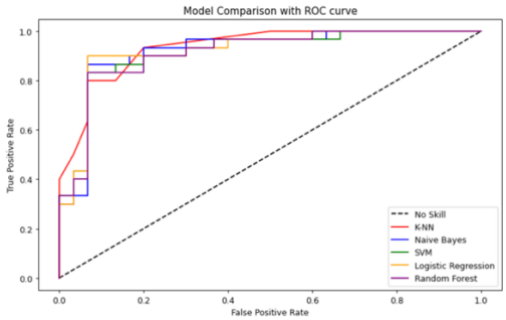


Figure 7: Model comparison with ROC curves

Looking at these figures it is evident that all the classifiers are performing reasonably well. Which can be justified as the Cleveland dataset is relatively small and there weren't sufficient data to train. Without sufficiently sized data sets, it is sometimes difficult to determine if a model is generalizable to previously unseen sets of data [6]. Based on cross-validation result ***Logistic regression classifier*** was chosen to be the best suited model. But it was difficult to interpret if any model is better than another. Even with the ROC curves it cannot be concluded, since each of these curves cross each other and there was no clear victor. To address this, a series of statistical paired t-test were conducted. As per below table (figure 8) it can be observed among all the statistical testing, only KNN vs Logistic regression test produced p-value = 0.03(less than 0.05 threshold), thus the null hypothesis (H0) can be rejected, and it can be argued that *Logistic regression performance is significantly different and improved compared to the 'KNN' performance.*

H0	H1	pvalue	tstat
KNN = Naive Bayes	KNN < Naive Bayes	0.446990	-0.795146
KNN = SVM	KNN < SVM	0.228403	-1.292429
KNN = Logistic Regression	KNN < Logistic Regression	0.038857	-2.416128
KNN = Random Forest	KNN < Random Forest	0.481844	-0.733634
Naive Bayes = SVM	Naive Bayes < SVM	0.714213	0.377970
Naive Bayes = Logistic Regression	Naive Bayes < Logistic Regression	0.749466	-0.329297
Naive Bayes = Random Forest	Naive Bayes < Random Forest	0.861477	0.179558
SVM = Logistic Regression	SVM < Logistic Regression	0.103888	-1.809068
SVM = Random Forest	SVM < Random Forest	0.863434	-0.176992
Logistic Regression Random Forest	Logistic Regression < Random Forest	0.460215	0.771448

Figure 8: Hypothesis testing result

As an extension of this study and with availability of multiple data of *exact* similar structure, it was decided to test all the final trained models on 3 secondary datasets discussed in "Setup" section. Below figure (figure 9) represent the F1-score for all five models tested on the secondary datasets. It is observed that the model performance on Switzerland data followed by Long Beach VA data is very low, which could be justified as these datasets had many missing information which had to be imputed (mean imputation was done for variables with more than 50% data available and replaced with 0 for variables with less than 50% of data available). The classifiers did relatively better job with the Hungarian data. Each of the model report and confusion matrix provided in Appendix. (refer figure 24-38)

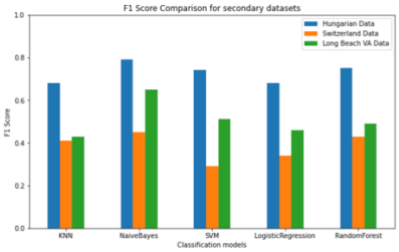


Figure 9: Compare F1-score for secondary datasets

Conclusion:

In this assignment, not only a significant subject in medical science was addressed but also various machine learning techniques were explored. Along with understanding feature importance, in this study, five different classification algorithms were discussed and implemented. The model parameters are fine-tuned using training data (60% of the data) and validation set (20% of the data) and finally optimal parameters are applied on the test dataset (20 % of the data). To further support the reliability of the model 10-fold stratified cross validation was conducted and models are compared. All the models performed reasonably well but Logistic regression was found to be the most suited model for this data. A series of hypothesis testing was also conducted to discover if any model performed better than another and it was established that Logistic Regression performed significantly better than K-nearest neighbour.

In the future, we can extend this study in the following aspects:

- Other classification techniques could be explored
- Since the Cleveland dataset is relatively small, we could use Surrogate Data approach [6].
- Different missing value treatment could be performed on the secondary datasets.

Reference:

[1] Data:
https://archive.ics.uci.edu/ml/datasets/heart+disease
Principal investigator responsible for the data collection at each institution:

- Hungarian Institute of Cardiology. Budapest: Andras Janosi, M.D.
- University Hospital, Zurich, Switzerland: William Steinbrunn, M.D.
- University Hospital, Basel, Switzerland: Matthias Pfisterer, M.D.
- V.A. Medical Center, Long Beach and Cleveland Clinic Foundation: Robert Detrano, M.D., Ph.D.

[2] Python | How and where to apply Feature Scaling by Shaurya Uppal, URL: https://www.geeksforgeeks.org/python-how-and-where-to-apply-feature-scaling/#:~:text=Feature%20Scaling%20or%20Standardization%3A%20It,the%20calculations%20in%20an%20algorithm.

[3] Regularization in Logistic Regression: Better fit and better generalization by Sebastian Raschka, Michigan State university URL: https://www.kdnuggets.com/2016/06/regularization-logistic-regression.html

[4] T. Yiu, \Understanding random forest," 6 2019.

[5] L. Breiman and A. Cutler, \Random forests,"

[6] Overcoming Small Data Limitations in Heart Disease Prediction by Using Surrogate Data by Alfeo Sabay et al.

Appendix:

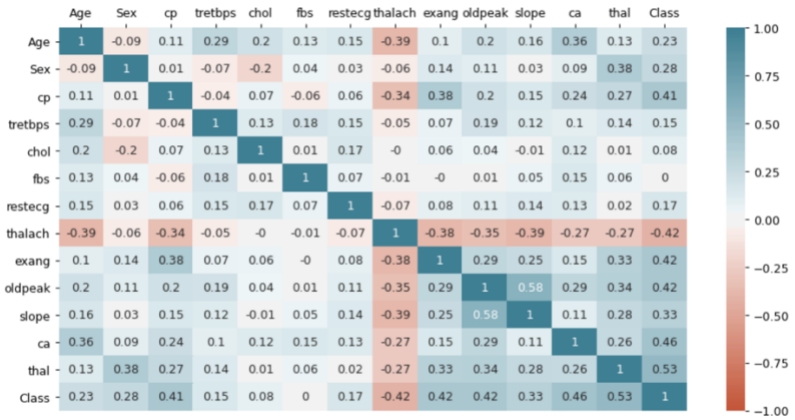
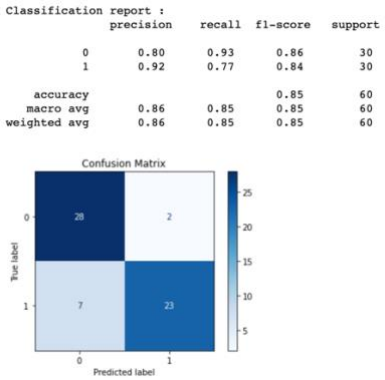


Figure 10: Correlation Matrix

K-Nearest Neighbour

Figure 11: Model Error for different value of k



Classification report :

	precision	recall	f1-score	support
0	0.80	0.93	0.86	30
1	0.92	0.77	0.84	30
accuracy			0.85	60
macro avg	0.86	0.85	0.85	60
weighted avg	0.86	0.85	0.85	60

Figure 12: K-NN final Model Report and Confusion matrix

SVM

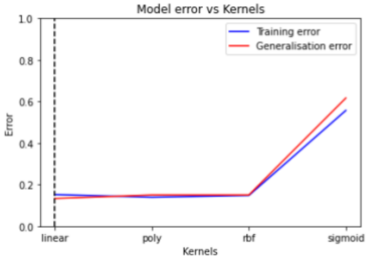


Figure 13: Model Error for different kernel

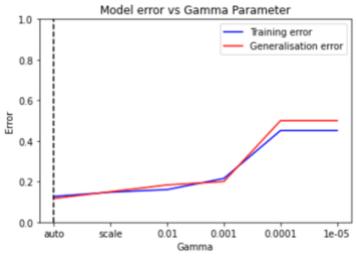


Figure 14: Model Error for different gamma

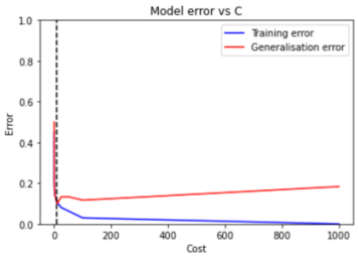


Figure 15: Model Error for different C

Classification report :

	precision	recall	f1-score	support
0	0.85	0.93	0.89	30
1	0.93	0.83	0.88	30
accuracy			0.88	60
macro avg	0.89	0.88	0.88	60
weighted avg	0.89	0.88	0.88	60

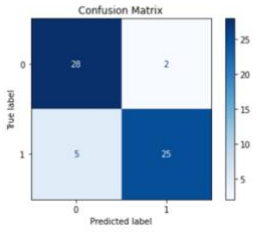


Figure 16: SVM final Model Report and Confusion matrix

Naïve Bayes

Classification report :

	precision	recall	f1-score	support
0	0.84	0.90	0.87	30
1	0.89	0.83	0.86	30
accuracy			0.87	60
macro avg	0.87	0.87	0.87	60
weighted avg	0.87	0.87	0.87	60

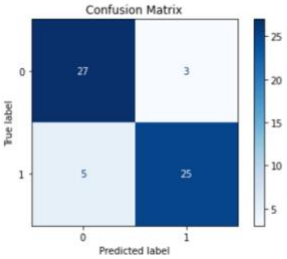


Figure 17: Naïve Bayes final Model Report and Confusion matrix

Logistic Regression

Figure 18: Logistic Regression parameter tuning

Classification report :

	precision	recall	f1-score	support
0	0.85	0.93	0.89	30
1	0.93	0.83	0.88	30
accuracy			0.88	60
macro avg	0.89	0.88	0.88	60
weighted avg	0.89	0.88	0.88	60

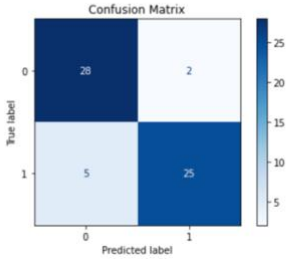


Figure 19: SVM final Model Report and Confusion matrix

Random Forest

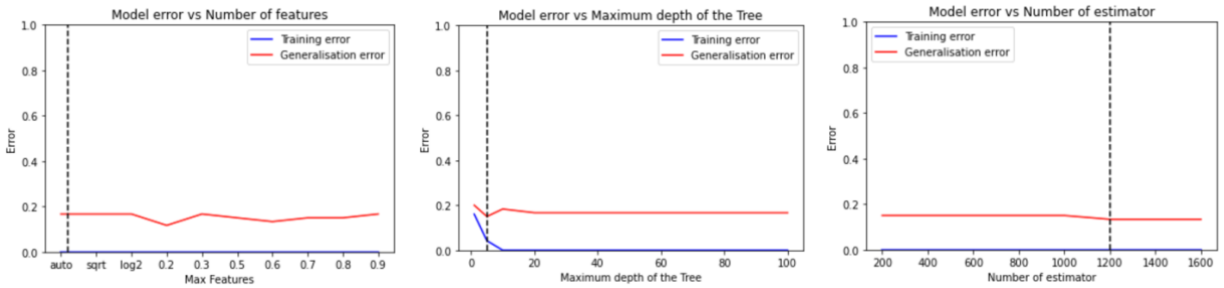


Figure 20: Model Error for different number of features Figure 21: Model Error for different max depth Figure 22: Model Error for different number of estimators

Classification report :

	precision	recall	f1-score	support
0	0.84	0.90	0.87	30
1	0.89	0.83	0.86	30
accuracy			0.87	60
macro avg	0.87	0.87	0.87	60
weighted avg	0.87	0.87	0.87	60

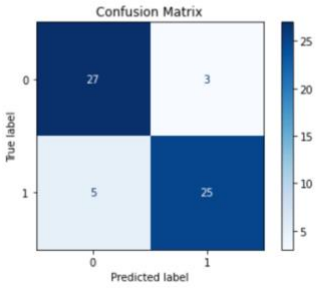


Figure 23: Random Forest final Model Report and Confusion matrix

Classification Report and Confusion matrix of secondary dataset

Classification report :

	precision	recall	f1-score	support
0	0.67	0.98	0.80	181
1	0.83	0.15	0.25	103
accuracy			0.68	284
macro avg	0.75	0.56	0.52	284
weighted avg	0.73	0.68	0.60	284

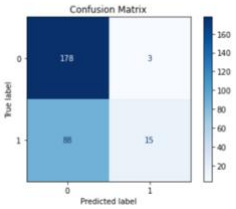


Figure 24: KNN on Hungarian dataset

Classification report :

	precision	recall	f1-score	support
0	0.09	0.88	0.17	8
1	0.98	0.37	0.54	108
accuracy			0.41	116
macro avg	0.53	0.62	0.35	116
weighted avg	0.91	0.41	0.51	116

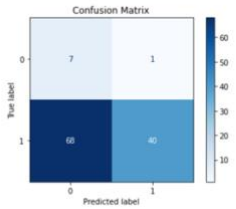


Figure 25: KNN on Switzerland Dataset

Classification report :

	precision	recall	f1-score	support
0	0.28	0.71	0.40	49
1	0.76	0.33	0.46	137
accuracy			0.43	186
macro avg	0.52	0.52	0.43	186
weighted avg	0.63	0.43	0.44	186

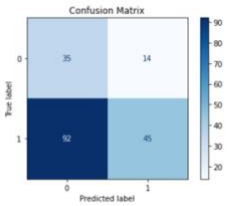


Figure 26: KNN on Long Beach VA dataset

Classification report :

	precision	recall	f1-score	support
0	0.77	0.96	0.85	181
1	0.86	0.50	0.63	103
accuracy			0.79	284
macro avg	0.82	0.73	0.74	284
weighted avg	0.80	0.79	0.77	284

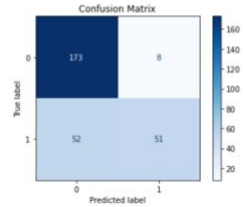


Figure 27: Naïve Bayes on Hungarian dataset

Classification report :

	precision	recall	f1-score	support
0	0.08	0.62	0.14	8
1	0.94	0.44	0.59	108
accuracy			0.45	116
macro avg	0.51	0.53	0.37	116
weighted avg	0.88	0.45	0.56	116

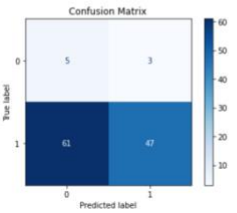


Figure 28: Naïve Bayes on Switzerland Dataset

Classification report :

	precision	recall	f1-score	support
0	0.36	0.43	0.39	49
1	0.78	0.72	0.75	137
accuracy			0.65	186
macro avg	0.57	0.58	0.57	186
weighted avg	0.67	0.65	0.65	186

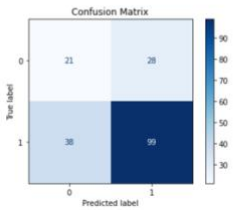


Figure 29: Naïve Bayes on Long Beach VA dataset

Classification report :

	precision	recall	f1-score	support
0	0.71	0.98	0.83	181
1	0.91	0.31	0.46	103
accuracy			0.74	284
macro avg	0.81	0.65	0.65	284
weighted avg	0.79	0.74	0.70	284

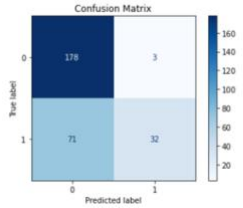


Figure 30: SVM on Hungarian dataset

Classification report :

	precision	recall	f1-score	support
0	0.08	0.88	0.15	8
1	0.96	0.25	0.40	108
accuracy			0.29	116
macro avg	0.52	0.56	0.27	116
weighted avg	0.90	0.29	0.38	116

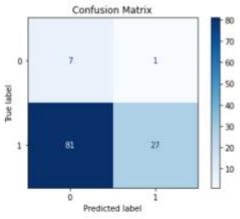


Figure 31: SVM on Switzerland Dataset

Classification report :

	precision	recall	f1-score	support
0	0.28	0.55	0.37	49
1	0.75	0.49	0.59	137
accuracy			0.51	186
macro avg	0.52	0.52	0.48	186
weighted avg	0.63	0.51	0.53	186

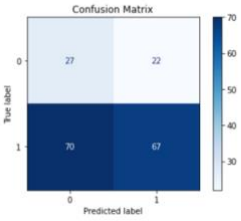


Figure 32: SVM on Long Beach VA dataset

Classification report :

	precision	recall	f1-score	support
0	0.67	0.98	0.79	181
1	0.82	0.14	0.23	103
accuracy			0.68	284
macro avg	0.75	0.56	0.51	284
weighted avg	0.72	0.68	0.59	284

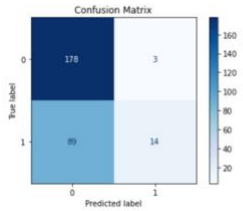


Figure 33: Logistic Regression on Hungarian dataset

Classification report :

	precision	recall	f1-score	support
0	0.09	0.88	0.16	8
1	0.97	0.31	0.46	108
accuracy			0.34	116
macro avg	0.53	0.59	0.31	116
weighted avg	0.91	0.34	0.44	116

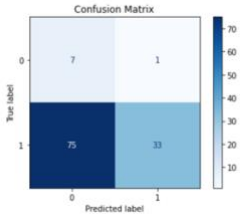


Figure 34: Logistic Regression on Switzerland Dataset

Classification report :

	precision	recall	f1-score	support
0	0.26	0.55	0.35	49
1	0.73	0.43	0.54	137
accuracy			0.46	186
macro avg	0.49	0.49	0.45	186
weighted avg	0.60	0.46	0.49	186

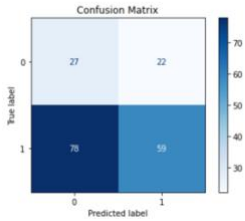


Figure 35: Logistic Regression on Long Beach VA Dataset

Classification report :

	precision	recall	f1-score	support
0	0.78	0.96	0.86	181
1	0.88	0.51	0.65	103
accuracy			0.80	284
macro avg	0.83	0.74	0.75	284
weighted avg	0.82	0.80	0.78	284

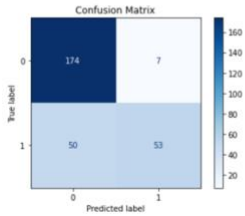


Figure 36: Random Forest on Hungarian dataset

Classification report :

	precision	recall	f1-score	support
0	0.08	0.75	0.15	8
1	0.96	0.40	0.56	108
accuracy			0.42	116
macro avg	0.52	0.57	0.36	116
weighted avg	0.90	0.42	0.53	116

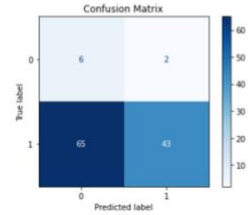


Figure 37: Random Forest on Switzerland Dataset

Classification report :

	precision	recall	f1-score	support
0	0.36	0.80	0.49	49
1	0.87	0.49	0.63	137
accuracy			0.57	186
macro avg	0.61	0.64	0.56	186
weighted avg	0.74	0.57	0.59	186

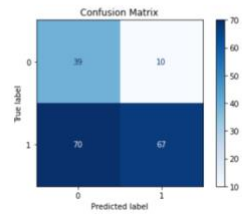


Figure 38: Random Forest on Long Beach VA dataset