

Health risk prediction with imbalanced data

Abstract

Time is an important variable in the treatment of many diseases. Owing the development of data mining techniques, the time to diagnosis can be shortened by using predictive models that identify individuals at risk in the early stages of disease development. In this logic, this paper found that Gradient Boost is powerful to deal with health data that are often imbalanced, with a weighted recall 0.95. Besides, unsupervised clustering has been applied to the same data with moderate results.

1. Introduction

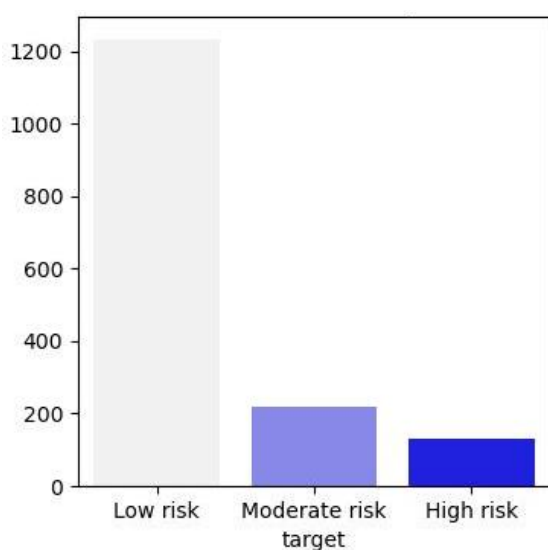
The numerous applications of classification and clustering have sparked interest in machine learning across a wide range of industries, including marketing, finance, and healthcare. Both techniques use automated processes to detect relevant patterns in data and categorize records accordingly. Whether it be figuring out who will be most likely to purchase a product, predicting a borrower's creditworthiness, or early illness detection, classification models offer tools to tackle these challenges. In the health sector, classifying individuals according to their vulnerability to a disease can help to diagnose those most at risk in time to provide effective treatment. However, classification can present many hurdles, as disease-prone individuals tend to be a minority among the patients. In addition, since an individual is at liberty to share certain information, these health data are often incomplete.

Against this background, this project aims at developing a classification model to identify individuals that are prone to a mortal disease known as Vaerosa-19. Furthermore, the increasing amount of unlabelled data available has fostered the development of clustering methods that aim to derive insight from these strategic resources. Thus, this project also offers the important opportunity to compare the performance of supervised and unsupervised machine learning models. This paper is composed of six sections, including this introductory section. The second give a summary of the features that serve as input for the model. The third second is concerned with data pre-processing. The fourth section presents the selection process of the best classifier. The fifth section discuss the findings for unsupervised clustering. Finally, the conclusion gives a summary the findings.

2. Summary of the features and target variable

The dataset for this study included 25 variables, which can be divided into predictors and the variable to be predicted which is the level of vulnerability to Vaerosa-19. The dataset consists of 1584 records, which correspond each to an individual.

Figure 1: Distribution of the target variable



For each record, the level of vulnerability to Vaerosa-19 is given by the variable target which takes the following values: low risk, moderate risk and high risk. As illustrated in Figure 1, the breakdown of this variable shows that there is an unequal distribution between the three categories. More than three quarter of the sample are not significantly vulnerable to the disease, 14% fall in the moderate risk category and 8% are at high risk of developing Vaerosa-19. The 24 attributes in the database are processed with the aim of identifying which combination of variables is more effective in predicting an individual's vulnerability to Vaerosa-19 disease. The subsequent paragraphs present the summary statistics of the potential features.

An exploration of the attributes using graphs and descriptive statistics reveals that the features take on very different values, hence the relevance of applying a standardisation technique to bring them to similar scales. For instance, the attribute x1 ranges from 0 to 2125. As the mean, which is 1053, is slightly higher than the median, which is 1049.5, the distribution is skewed to the right. The standard deviation, which is 616, suggests that the distribution has a high degree of variability and thus a large amount of statistical noise. To contrast, this attribute x4 takes value from 0 to 0.019. The mean of this distribution is 0.003169. However, considering the subset of the person that are at high and moderate risk, we can see that X4 is equal to 0 for the large majority these subgroups.

Figure 3: Distribution of x1

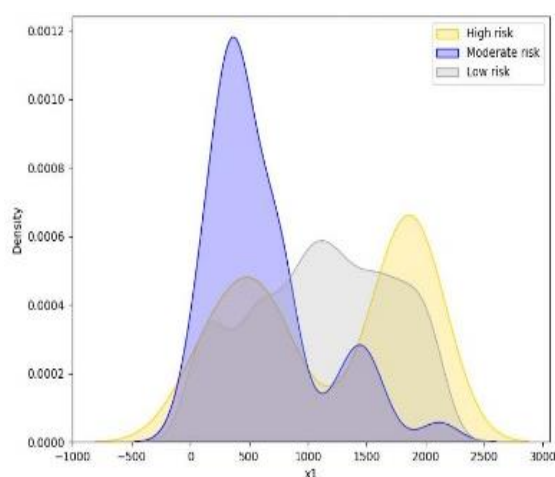
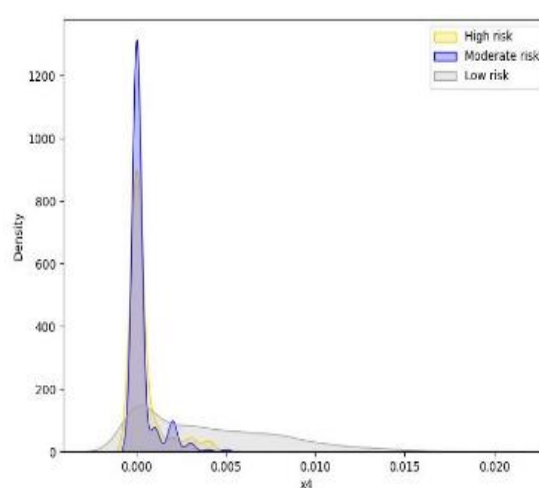


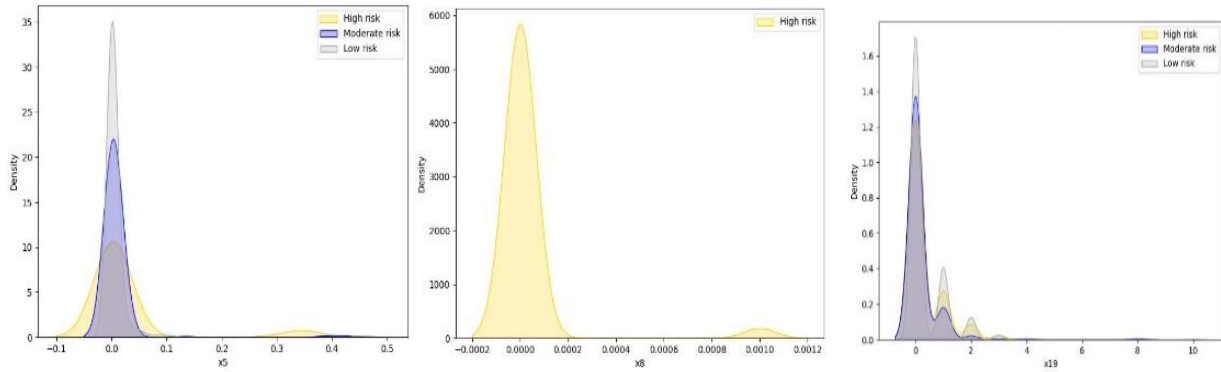
Figure 3: Distribution of x4



The distribution of some attributes, such as x5, x8 and x19, is tightly distributed around their mean. However, from Figure 4, the long tails, that suggest the presence of outliers, can be

observed for the high-risk category for x5 and x8. On top of that, the variable x8 takes the value zero for 1864 records out of 1867 non missing values.

Figure 4: Distribution of x5, x8 and x19



The dataset has two categorical variables X3 and x14. The former refers to the gender of the individual and can take only two values: F for female and M for male. It can be observed that 10.4% of the male are at high risk compared to 5.7% of the female. The other categorical attribute is the blood type of the individual which takes eight values. From the Figure 5, it can be seen that a higher proportion the blood type AB- are at high and moderate risk compared to the other categories. Next come the O- and B- subgroups. From this it is apparent that a Reshus negative is more vulnerable to this disease.

Figure 5: Distribution of sex (x3)

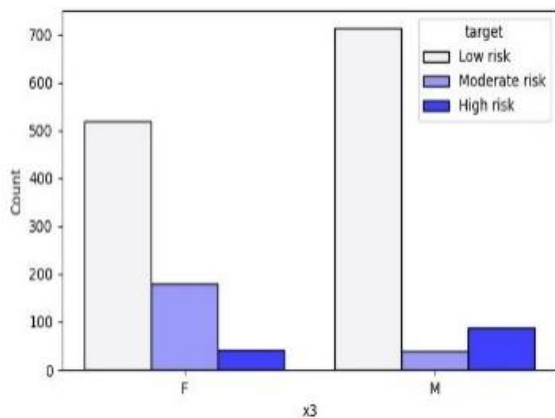
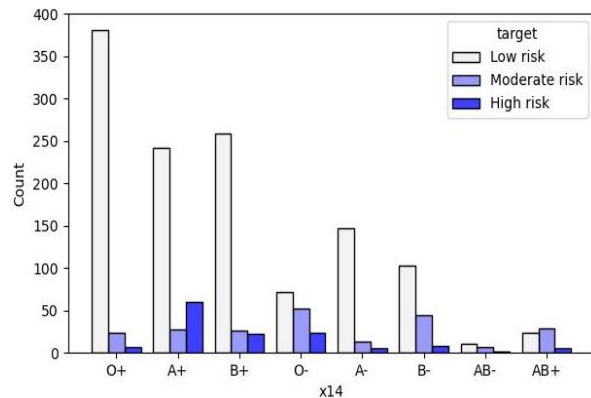


Figure 6: Distribution of blood type (x14)



Regarding the completeness of the dataset, the attributes x5, x8 and x10 have 2.8%, 1.1% and 1.7% missing values respectively. No common pattern between the missing values was identified. Therefore, this paper treats them as missing at completely random. Finally, a correlation matrix reveals that the variables x20, x21 and x22 are very strongly correlated with each other, with coefficients greater than 0.9. These variables are also the ones that present the strongest correlations with the target variable, with coefficients of 0.41, 0.33 and 0.38 respectively for x20, x21 and x22. Then we find x2 with a coefficient of 0.29 and x9 which is negatively correlated to the target variable.


3. Pre-processing

3.1 Encoding

Categorical data encoding converts categorical attributes into numbers so they can be in an appropriate format for modelling. Indeed, most machine learning model are only able to process and derive information from numerical features. Two Sklearn algorithms were considered to encode the two categorical attributes of the dataset. On the one hand, there is OrdinalEncoder that assigns an integer ranging from 1 to n for an attribute with n unique values. During this process the value 1 is given to the category that contains the most observations and so forth. This method offers the advantage of transforming a feature without increasing the number of attributes in the database. However, it assumes the existence of an ordinal relationship between the different categories that would allow them to be classified in respect of one another. On the other hand, OneHotEncoder create a dummy variable for each category while transforming a nominal feature. This method does not assume an inherent order between the categories. However, it increases the number of features by n-1 for a feature with n category. Which is impractical when a variable takes a very large number of unique values.

In this coursework, the categorical attributes sex and blood type, are converted into dummy variables using OneHotEncoder. This process was set up to remove one of the dichotomous variables generated if the features were originally binary. Therefore, for the sex variable (x3) only the dummy variable associated with male sex was retained in the database. The blood type variable (x14), which takes the values: O-, O+, AB-, AB+, B-, B+, A- and A+, was transformed into eight dummy variables. Finally, we use LabelEncoder to convert the label into three integer values, following a categorical data encoding mechanism.

Figure 7: Illustration of One Hot Encoding process



index	x14	index	Blood_C	Blood_O	Blood_A-	Blood_A	Blood_B-	Blood_B	Blood_AB-	Blood_AB
0	O-	0	1	0	0	0	0	0	0	0
1	O+	1	0	1	0	0	0	0	0	0
2	A-	2	0	0	1	0	0	0	0	0
3	A+	3	0	0	0	1	0	0	0	0
4	B-	4	0	0	0	0	1	0	0	0
5	B+	5	0	0	0	0	0	1	0	0
6	AB-	6	0	0	0	0	0	0	1	0
7	AB+	7	0	0	0	0	0	0	0	1

3.2 Handle missing values

Considering that most machine learning models are only able to predict values or label when all the rows of the features vector are complete, handling missing variables becomes a necessary step. To do so, one can drop the rows or columns containing missing values or use imputation techniques to fill the blank. In the Health_train data set, three attribute have about 1.7% to 2.8% missing values, which appear to be missing at completely random and are all encoded as NaN. Since we are working with a small sample, impute missing value appears to be the optimal solution to preserve all available information. While there are many imputation techniques, we only tried four of them and compared the results for the purpose of choosing the most appropriate.

The first method is a univariate imputation algorithm which fills the missing values using the mean of non-missing values of the feature. The second is KNN Imputer which fill each missing value with the mean of k-nearest non-missing values. Thus, we run this algorithm considering the five nearest neighbours and weighing each point by the inverse of their Euclidean distance to the missing record. The third algorithm, Iterative Imputer, predict the missing values for a given feature in a record regressing the non-missing values features on the missing features throughout several iterations that successively considered the incomplete variable as output and the others as input. The last method is Multiple Imputation by Chains Equations (MICE) is an algorithm that combine univariate and multivariate imputation techniques. Throughout the first stage the missing value are filled with the mean of the feature. Then, in each iteration a Random Forest model is used to predict the missing values in a feature, considering the n-1 other attributes as predictors.

Table 1: Table comparing the outcomes of the different imputation techniques

	Original	Transformer Mixin	KNN Imputer	Iterative imputer	MICE
mean	47.094412	47.094412	47.03446	47.065263	47.053030
Standard deviation	17.269621	17.121710	17.22624	17.177095	17.217122
Sample of imputed values					
Row 11	NAN	47.094412	30.790190	45.665557	34.0
Row 32	NAN	47.094412	63.496114	48.300992	64.0
Row 120	NAN	47.094412	18.662686	35.248442	21.0
Row 144	NAN	47.094412	33.751610	29.729799	34.0
Row 161	NAN	47.094412	44.237483	63.979238	47.0

For the rest of the paper, the results obtained using KNNImputer are considered. Since rows with missing values represent only a small fraction of the records, the mean and the average of the imputed features are close to the original distribution, regardless of the method. The values imputed using KNNImputer and MICE are closer than those obtained using the other two approaches. This method is chosen since the imputed values are less likely to be affected by outliers.

3.3 Outliers Detection

The quality of prediction obtained using machine learning models not only depends on the model but also on the quality of the data used to train it. Hence, identifying and treating outliers in a data is important step in data pre-processing. Outliers are datapoints that differs significantly from the underlying pattern observed in a dataset. In the context of this project, three different algorithms were used to detect outliers. Firstly, we generate three clusters with a Density-based spatial clustering of applications with noise (DBSCAN) model and predict whether a datapoint stand out from the rest of the dataset or not. The prediction provided by this model serves as ground truth to evaluate the results obtain with the two other methods. At the end of this stage, the DNSCAN identifies 61 outliers.

Table 2: Outliers detection

Algorithm	Hyperparameters	Number of outliers	Incorrectly identified outliers
-----------	-----------------	--------------------	---------------------------------

Isolation Forest	max_samples=1188, contamination='auto', n_estimators = 1500, max_features =12	52	16
Local Outliers Factor	n_neighbors=12, contamination='auto'	39	20

Secondly, Isolation Forest, and Local Outlier Factor are used to detect outliers. To do so, Isolation Forest performs recursive splits that are randomly generated on attribute values in order to isolate any data point from the rest of the sample. Then, an isolation score is attributed to each datapoint based on the average distance from the tree's root to the branch containing the isolated datapoint (Domingues et al., 2018). As shown in Figure 7, the outliers take value -1 and the normal instances take the value 1. With regard to Local Outliers factor, this algorithm gives an anomaly score to each datapoint that measures the difference in density between an instance and its k-nearest neighbours on a local level. Outliers, or samples whose density is significantly lower than that of their neighbours, can be found by comparing a sample's local density with the local densities of its neighbours (Sklearn, 2007). The larger the score, the larger the circle as shown in Figure 8.

Figure 9: Outliers detected with Isolation Forest

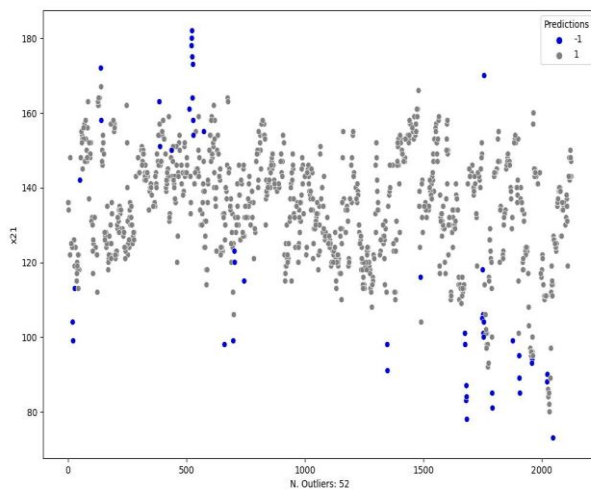
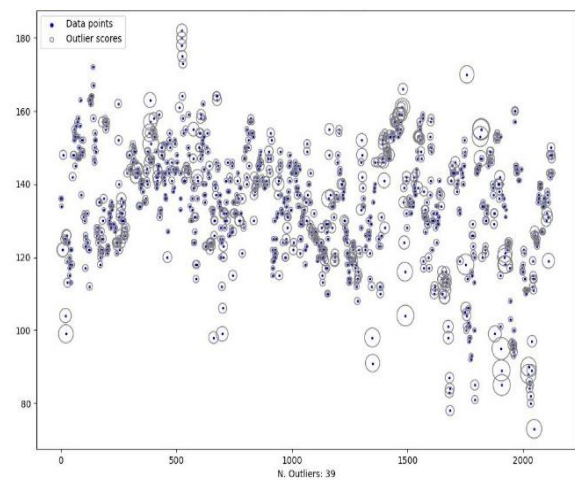


Figure 9: Outliers detected with LOF



Isolation Forest appears to be the most suitable method for this dataset given that the number of outliers identified is more comparable to that provided by DBSCAN. Furthermore, the number of incorrect predictions is smaller than the number of LOF's predictions that do not match the ground truth. These findings are consistent with those of Domingues et al. (2018) who compared Isolation Forest to thirteen outlier detectors and found that this algorithm outperforms all the others on different kinds of datasets.

3.4 Standardization

Many prediction models perform better when the attributes are on comparable scales. Therefore, the data were standardised using the StandardScaler algorithm. For each observation of a numerical variable, the algorithm subtracts the mean and divides it by the standard deviation to obtain a distribution that approximates a standard normal distribution, with a mean of zero and a standard deviation of one. This method is useful but not without limitations as it assumes that each feature follows a Gaussian distribution. Nonetheless, if this condition is not met, the quality of the predictions may be negatively affected. The binary variables obtained as a result of encoding have not been subjected to this transformation process since they only take the values zero and one.

4. Supervised learning

4.1 Class Balancing

Class imbalance is a common issue in medical databases, in which only a small proportion of people suffer from a disease or are vulnerable to one (Rajendran et al., 2020). As noted in section II, a large fraction of the observations in the database fall into the "low risk" category, which is therefore the majority class. The moderate risk and high-risk categories have far fewer observations and can be considered minority classes. When a database presents this characteristic, many machine learning models are unlikely to learn from the minority class and correctly predict the label of the instances belonging to this class. Thus, a model may perform very well in predicting the majority class while the classification rate is very low for the minority class.

To solve the issue of class imbalance, various sampling techniques have been developed, including subsampling, which removes data from the majority class, and oversampling, which adds data to the minority class. In this project, the training dataset has been balanced using an oversampling method which consists of replicating data from the minority class by drawing with replacement until the number of records equals that of the majority class. Considering that the sample size is small, this method seems the most appropriate to preserve all the information available in the training dataset after removing outliers.

4.2 Features selection

Table 3: Mutual information

Features names	Mutual info
x1	0.340028
x11	0.220822
x10	0.188921
x12	0.181485
x4	0.128820
x2	0.125310
x23	0.123513
x21	0.119233
x16	0.108935
x15	0.108810
x22	0.107406
x20	0.106860
x13	0.066939
x17	0.050132
x6	0.044562
x9	0.040170
x24	0.022621
x7	0.020003
x18	0.008461
x19	0.007174
x5	0.004121
x8	0.004026

that bring almost no information gain. Therefore, these four attributes were removed from the training dataset.

Dimensionality reduction is performed to curtail the number of features in the training dataset and thus reduce the complexity of the models selected to predict vulnerability to Vaerosa-19. The selection of the most relevant attributes is made based on the information gain that the inclusion of this feature brings to the model, as measure by Sklearn's mutual info algorithm for classification. The mutual information (MI) is calculated for each combination of an attribute and target variable to assess the dependency between these variables. More precisely, MI measures the reduction in uncertainty for target variable when the value of this attribute is known.

It can be seen from this Table, that x1, x11, x10 and x12 have the potential to increase the predictive power of the models as their mutual information range from 0.34 to 0.18. Then, we also have variables such as x4 and x22 which can modestly contribute to improving classification quality if they are included in the model. Whereas, in the last rows of the table we find x18, x19, x5 and x8

4.3 Presentation of the individual algorithms

Classification is a process that aims to label new data according to the patterns identified by the model in the training dataset. A wide variety of algorithms, with varying degrees of complexity, have been developed to facilitate this supervised learning. Each of these algorithms bases its operation on different set of criteria and follows a specific process to maximise the classification rate. The accuracy obtained at the end of the process partly depends on the data used. Thus, it is good practice to test different models and select the one that performs best for the project in question. Following this principle, we tested individual algorithms such as K Nearest Neighbours, Decision Tree and Naive Bayes and more complex models such as Neural Network. In the subsequent paragraphs, we present a summary description of the models used to predict vulnerability to Vaerosa-19.

K-Nearest Neighbors (KNN)

KNN is a non-parametric classifier that uses the distance between a new data point and k instances already labelled to predict the label of the new instance. The process involves a combination of the labels of these k records, either by a simple majority vote or by or a vote that gives more weight to the closest instances.

Support Vector Machine

Find the hyperplane that best separate the training data in different categories, which mean the one that maximizes the distance between data points that belongs to different categories while keeping an even margin between the separator and the closest point on each side. When it is not possible to create linearly disentangle the subgroups, one could augment the dimensionality of

data before projecting it in another space and then draw the hyperplane. This algorithm uses a kernel transformation function based on the degree of similarity between each pair of datapoint (Asch, 2022).

Naïve Bayes

The Gaussian Naïve Bayes algorithm is based on Bayes's Theorem, which describes the relationship probability that event A knowing event A occurred. In this respect, the algorithm determines the probability that a new instance corresponds to a label, given the value of the predictors. This classifier assumes that every pair of features is conditionally independent given the label associated to a data point in the training dataset.

Stochastic gradient descent

The learning process of Stochastic gradient descent consists of identifying the global minimum of a certain loss function, which measures the extent to which the parameters used during an iteration minimize the differences between predictions and true labels. The learning can be reinforced with penalty classification. One record is randomly selected to calculate the derivative at each iteration, until the algorithm finds the parameters that minimise the errors in the model. This classifier handle multi – class classification by using a “one versus all” scheme (Sklearn, 2007). In other words, for each class, the algorithm processes as if the target variable is binary and all other classes constitute a single label. Thus, it learns to distinguish patterns specific to that class.

Decision Tree

Decision Tree (DT) creates a succession of internal and leaf nodes based on decision rules that maximise the information gain at each step. First, the algorithm identifies the attribute with the highest predictive power to build the root node. Then, the algorithm prioritises the attributes according to their contribution to the quality of the classification in order to create new sub nodes until each branch contains only data belonging to a class or until the specified maximum depth is reached.

Neural Network

Neural networks consist of layers of neurons divided into an input layer, several hidden layers, and a final layer where the final predictions are located. During the learning stage, the probability assigned to each neuron is adjusted in a series of forward and backward propagation until the probability distribution in the output layer becomes more similar to that of the true label.

4.4 Training and results

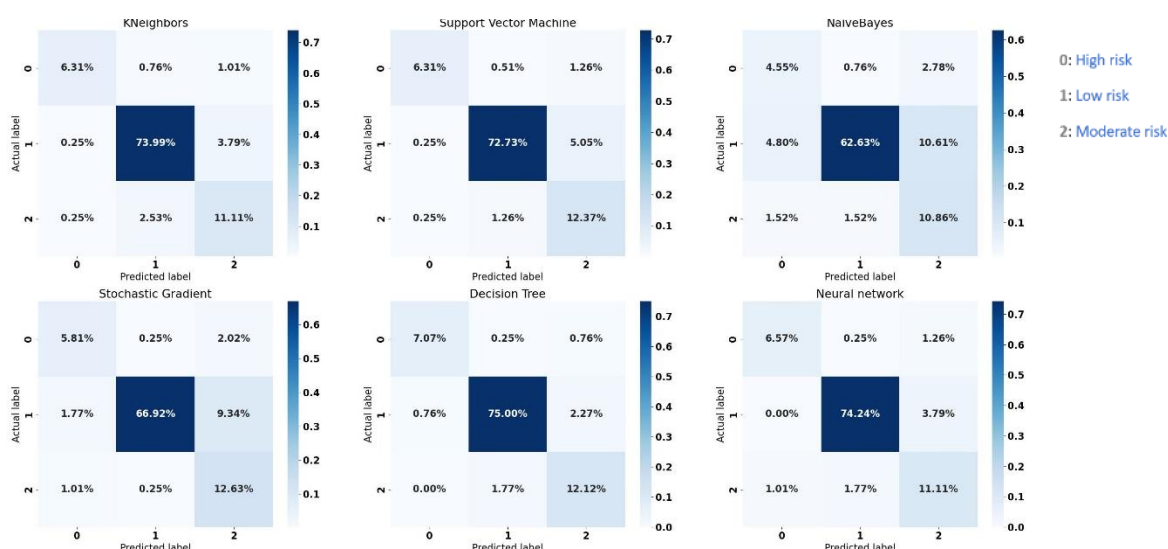
Each of the algorithms presented in the previous section went through a learning stage with the training data set and a prediction stage. Then, model tuning was used to find the hyperparameters that maximise model performance. To do so, different values for each of the given hyperparameters were compared using a grid search. Once the best parameters have been identified and integrated into each model, the classifiers are evaluated according to their comparative performance. The metrics used to assess performance include are balanced accuracy rate, precision, and recall. Precision refers to the ratio of instances correctly labelled for a given category to the total datapoints attributed to this class. Recall gives the proportion of instances whose predicted label corresponds to the class to which they belong. Balanced accuracy rate gives the arithmetic average of proportion of prediction that are correctly made for each class. This metrics account for class imbalanced while assessing the overall performance of the model. The difference between the models is highlighted in the Table below. It can be seen that DT has the highest balanced score, followed by Support Vector Machine. At the bottom of the ranking, we have the Gaussian Naive Bayes model with a rate of 71.6%.

Table 4: Model specification and evaluation metrics

Model	Hyperparameters	Balanced accuracy	Macro precision	Weighted precision	Macro recall	Weighted recall
0	KNeighbors Best leaf_size: 1, Best p: 1, Best n_neighbors: 1	0.843157	0.860618	0.918977	0.843157	0.914141
1	Support Vector Machine kernel=poly, degree=3	0.868066	0.854786	0.928577	0.868066	0.914141
2	NaiveBayes var_smoothing=0.0001519911082952933	0.715636	0.610501	0.849015	0.715636	0.780303
3	Stochastic Gradient Default	0.828482	0.731765	0.902222	0.828482	0.853535
4	Decision Tree class_weight 0: 0.08, 1: 0.79, 2: 0.13, criterion=entropy, max_features=auto, max_depth=14	0.902964	0.892332	0.943935	0.902964	0.941919
5	Neural network Default	0.854652	0.842559	0.925153	0.854652	0.919192

Besides balanced accuracy rate, we can also compare the weighted recall since a good model should reach correctness of fit for the minority classes. Not only a good model should be able to correctly classify a large proportion of all the record, but also rightly allocate a large percentage of individual within each class. Correctly predict high or moderate vulnerability among people that belong to these classes might enable medical stakeholders to save more lives. DT reach a weighted recall of 0.94, which indicates that the model has correctly allocated the records of each classes. As displayed in the confusion matrix below, the high-risk class represents 8% of the test data and the different models label 4.55% to 7.7% as such. Regarding the other minority class, the individuals at moderate risk constitute 14% of the test data and the models correct classify 10.86 to 12.68% in this category. Considering the balanced accuracy and the recall for the high-risk class, Decision Tree appears to be the best model for this project. However, better performance can be achieved by combining the predictions obtained from each model.

Figure 10: Confusion matrix



Ensemble learning

Ensemble learners are models which can use multiple learning algorithms to perform better. Three Ensembles were tested to improve predictive performance for this dataset: Gradient Boosting Classifier, Random Forest and a voting classifier composed of the four best models from the previous section. Both Random Forest and the voting system yielded an accuracy of 93.7%. Gradient Boosting Classifier outperforms all the other algorithms with an accuracy rate of 94.6%, a balanced accuracy of 91%. More importantly, this model only misclassifies four instances of the high-risk class out of thirty-two. The model has precision of 97% for both low and high-risk class. The highest recall is that of the low-risk class (97%), followed by that of the moderate class which is 89%. High risk yielded a recall of 88%, which is mean that the model can identify an individual of that class with a probability of 0.88. Even when the model is trained with imbalanced data, the accuracy rate increases slightly to 94.5%. However, the recall for high-risk class is 0.81 with balanced data while it is 0.78 when resampling is not applied to the train data. Overall, the model performs well with both types of data. However, balancing improves the sensitivity of the model.

The explanation behind the higher performance of Gradient Boosting is mechanism by which this algorithm operates. The Ensemble sequentially applies a basic classifier to repeatedly altered versions of the input in order to generate classifiers whose final class assignment is decided by weighted majority voting. Thus, Boosting Gradient force the base classifier to focus more on data that weren't correctly identified in earlier rounds by giving them more weight. On the one hand we know that many models tend to misclassify minority classes, and on the other hand Gradient boost penalises that mis-classified more often during the training step.

Figure 12: Confusion matrix for Gradient Boosting

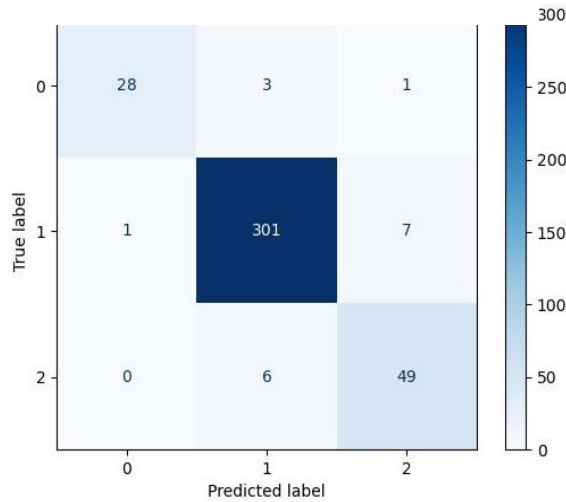
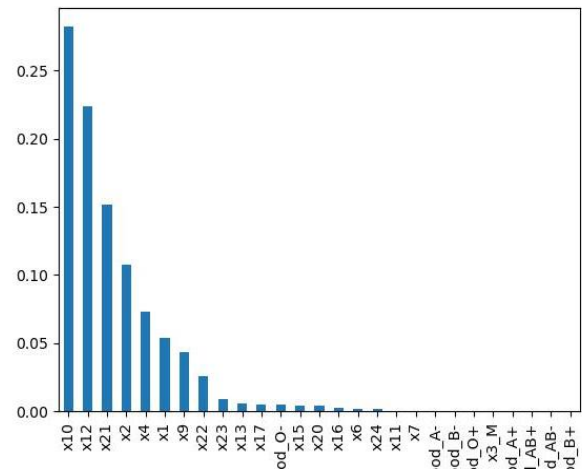


Figure 12: Features importance



The attributes x10, x12, x21 and x2 contribute significantly to the predictive performance of the model. In contrast, the dummy variables generated from the individual's blood type, except for the O- subgroup, appear to have no effect on the model. The same is observed for the attributes x11 and x7.

5. Unsupervised Clustering

5.1 Models

Most data available online are unlabelled and therefore, cannot be trained to predict a label. Other machine learning techniques, such as clustering, are used to group instances that behave similarly. The performance of these algorithms depends on the type of data. For comparative purposes, two unsupervised models were used. On the one hand, Hierarchical clustering uses a bottom-up approach that treats each datapoint as a cluster in the first instance and then clusters the closest clusters as it goes along. On the other hand, K-prototype is widely used for clustering databases containing both quantitative and qualitative data. This algorithm groups the data into K clusters by minimising a cost function (Ji et al., 2012).

5.2 Results

The sub-sample that was used to validate the classification models was used to facilitate the comparative analysis. While looking for the optimal number of K that optimize the loss function for the K-prototype model, we found that three is a local minimum but is not the value that minimises the function over the entire interval. However, this value was used as the database has three labels. Moreover, we found that the hierarchical clustering reached a silhouette score of about 0.30 when the number of clusters is three.

Figure 13: Silhouette score vs number of clusters

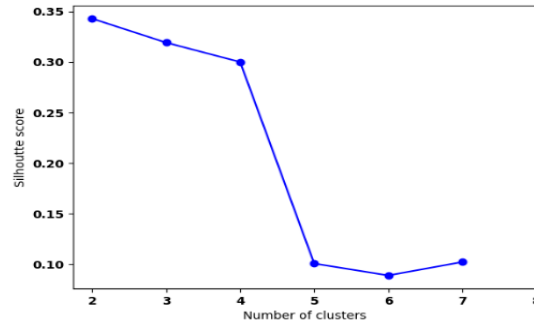
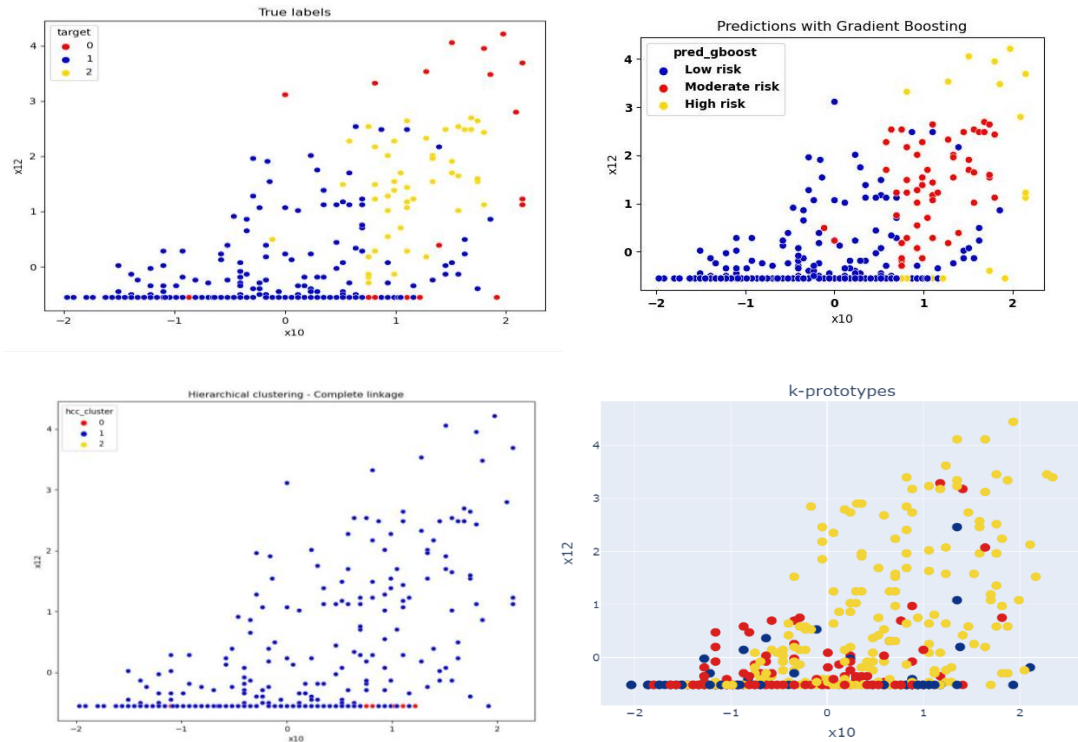


Figure 14 compares the actual distribution of individuals across the three classes, the best classifier's predictions, and the clusters produced by unsupervised models. From the first scatter plot, it can be observed that a 2-dimension representation fails to project three classes that are totally disentangled, even with the true label. Therefore, the other scatter plots are only analysed in comparison to each other.

Figure 14: 2D projection of the clusters



As suggested by the high accuracy rate, the classification obtained with Gradient Boosting is very similar to the true distribution of individuals between classes. The hierarchical clustering model has grouped most low and moderate risk individuals into a single cluster. At the bottom of the graph, a second cluster can be clearly identified which seems to correspond to the high-risk individuals. From the graph it is apparent that K-prototype put the high and moderate risk individuals in the same cluster. However, the model generated two other clusters each consisting mainly of instances of the majority class.

The two models' performance was evaluated using three parameters: homogeneity, completeness and the Rand Index adjusted for chance (ARI). To compute this metrics, we use the true labels of the validation dataset. Prior to that, a ground truth was established with the DBSCAN algorithm. The homogeneity of the classification produced by agglomerative clustering is 0.12 while that of the other model is only 0.09. These figures are much lower than 1 and indicate a very high heterogeneity within each cluster in the sense that they each group together many individuals belonging to different classes. Regarding completeness, the hierarchical model reached a score of 0.26 while the K-prototype displays a score of 0.06. This shows that the former has a greater capacity to group the instances that belong to the same class in the same cluster. The Rand Index adjusted for chance (ARI) which measures the similarity between the clusters, takes value from zero to one. A value close to zero indicates that the clusters are very comparable in the clusters generated using an unsupervised method and the actual clusters. We obtain a score of 0.22 for the hierarchical clustering with complete linkage and a score of 0.03 with K-Prototype.

The metrics would seem to suggest that the hierarchical is more effective at separating the data into clusters whereas the visualization of the clusters generated by the K-Prototype seems more informative. One possible explanation stems from the fact that the data is imbalanced. The first model grouped most instances into a single cluster. Which leads to a better completeness given that the majority of the datapoints belong to the low-risk class. Nevertheless, the separation made by K-Prototype seems to meet the objectives of the project. Indeed, separating low risk individuals from those facing a higher risk would allow medical actors to focus on the cluster that requires preventive or curative care.

6. Conclusion

This paper identity the best model to perform multi-class classification on an imbalanced dataset. The process has been undertaken following some major starting with transforming data into a format that is suitable for machine learning algorithms. Thereafter, we have resampled to correct data imbalance, perform features selection, model training and evaluation. We have found that the Ensemble Gradient Boost outperforms all the other algorithms tested. Furthermore, the data was clustered using Hierarchical clustering and K-Prototype. Despite the lower ARI, le dernier model seems to emulate more closely the behaviour of the original data. Overall, the project's findings suggest that classification is more effective than clustering at identifying subgroups with rare medical conditions. That is because, clustering is less complex and relies mainly on the similarity between the instances to build the clusters. At the opposite, classification models can learn from past data to increase their predictive power.

References

- Asch, M. (2022). Python Data Science Handbook. In *A Toolbox for Digital Twins: From Model-Based to Data-Driven*.
- Domingues, R., Filippone, M., Michiardi, P., & Zouaoui, J. (2018). A comparative evaluation of outlier detection algorithms: Experiments and analyses. *Pattern Recognition*, 74. <https://doi.org/10.1016/j.patcog.2017.09.037>
- Ji, J., Pang, W., Zhou, C., Han, X., & Wang, Z. (2012). Erratum: A fuzzy k-prototype clustering algorithm for mixed numeric and categorical data (Knowledge-Based Systems 30 (2012) (129-135)). In *Knowledge-Based Systems* (Vol. 36). <https://doi.org/10.1016/j.knosys.2012.04.029>
- Rajendran, K., Jayabalan, M., & Thiruchelvam, V. (2020). Predicting breast cancer via supervised machine learning methods on class imbalanced data. *International Journal of Advanced Computer Science and Applications*, 11(8). <https://doi.org/10.14569/IJACSA.2020.0110808>