Project 3

Data Analysis and Machine Learning FSY-STK4155

Maria Beatrice Cattini
Matteo D'Alessandro
Lisa Incollingo
Vanessa Ventura

December 13, 2022

Abstract

This project aims to implement, study and compare the performance of different methods for classification on the Pima Indians Diabetes dataset, making predictions on whether a patient's diabetes diagnosis is positive or negative based on 8 variables.
Firstly we consider tree-based methods: we apply a Decision Tree classifier applying cost-complexity pruning and ensemble methods (Random Forest, AdaBoost), analysing their results for an increasing number of trees in the ensemble. We compare the performance of our algorithms with the corresponding `scikit-learn` methods. We also consider variable selection, making use of the internal variable importance metric given by the Random Forest algorithm.
Subsequently, a code for a Feed-Forward Neural Network with flexible number of hidden layers and nodes and a Logistic Regression model with Stochastic Mini-batch Gradient Descent are implemented. We tune the hyperparameters associated to the models above such as the learning rate, the number of hidden layers, neurons and epochs.
All methods are compared on multiple classification metrics (accuracy, precision, recall, specificity, F-score, AUC) with the pruned Decision Tree obtaining the best accuracy (81.8%) and Logistic Regression showing the best True Positive Rate (recall) of 87.2%.

Contents

1 Introduction

Diabetes mellitus is an incurable chronic illness that develops either when the pancreas is not able to generate sufficient insulin or when the body does not utilize the insulin produced effectively [1]. It is the leading non-communicable disease globally.
Diabetes results from genetic and environmental factors including ethnicity, family history of diabetes, age, excess weight, unhealthy diet, physical inactivity, and smoking. In addition to this, the absence of early detection of diabetes has been known to contribute to the development of other chronic diseases.
Predicting the probability of an individual's risk and susceptibility to a chronic illness like diabetes is a crucial task. Diagnosing chronic illness at an early stage saves on medical costs and reduces the risk of more complicated health problems.
This analysis makes use of the Pima Indian Diabetes dataset [1], a popular benchmark dataset consisting of data on 768 female patients 21 years and older. Pima Indians are a Native American group that lives in Mexico and Arizona, USA [2]: this minority group was discovered to present a high incidence rate of diabetes mellitus. Hence, research around them was thought to be significant.

The aim of this project is to predict whether the patient has diabetes or not based on some of their physical and health characteristics (i.e. BMI, plasma glucose concentration) by implementing and testing different statistical methods.
We firstly consider tree-based methods such as Decision Tree, Random Forest and AdaBoost attempting their implementation and comparing the obtained performance with the one from the corresponding models from `scikit-learn`. Secondly, we apply Logistic Regression and a Feed-Forward Neural Network with a flexible number of layers and neurons and different activation functions.
In order to compare the results of the models considered, we look at different classification metrics: accuracy, precision, recall, specificity, F-score and AUC. In determining which method would be best in analysing our dataset we also look at interpretability of its results: in fact, the ability to explain the reasonings behind a machine learning prediction is particularly crucial in the healthcare field, where mistakes could be fatal [4].

Our work is structured in three main parts: in Section 2 we discuss the methods that will later be applied and analysed in Section 3. In the last Section 4 we will present our final comparison results.
Our code can be consulted at the following GitHub link: `https://github.com/matteodales/FYS-STK4155_Applied_Data_Analysis_and_Machine_Learning/tree/main/project3`.

[1] `https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database`

2 Methods

2.1 Tree-based methods

Tree-based methods partition the feature space into a set of regions, and then assign a single predicted value to each one. Each time a region is divided into two, until a stopping criterion applies. Finding the best partition in terms of loss function is generally computationally infeasible, so the algorithm used for developing trees is usually *greedy*: at each step the algorithm selects the splitting variable and point that minimizes the loss function for that specific step.

Tree-based methods are conceptually simple and highly interpretable, yet powerful, and can be applied both to regression and to classification problems.

When considering a classification problem, the prediction are made as follows: let

$$\hat{p}_{mk} = \frac{1}{N_m} \sum_{x_i \in R_m} I\left(y_i = k\right),$$

be the proportion of class k observations in node m, where R_m the region defined by the node, N_m is the number of observations in R_m, and I is the indicator function. We classify the observations in that node to the majority class $\hat{c}_m = \arg\max_k \hat{p}_{mk}$. Different measures $Q_m(T)$ of node impurity include the following:

- Misclassification error: $\frac{1}{N_m} \sum_{i \in R_m} I\left(y_i \neq k(m)\right) = 1 - \hat{p}_{mk(m)}$.
- Gini index: $\sum_{k \neq k'} \hat{p}_{mk} \hat{p}_{mk'} = \sum_{k=1}^{K} \hat{p}_{mk}\left(1 - \hat{p}_{mk}\right)$.
- Cross-entropy or deviance: $-\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}$.

Cross-entropy and the Gini index are usually preferred, being more sensitive to changes in the node probabilities than the misclassification rate. In our analysis we use the Gini Index.

2.1.1 Cost-complexity pruning

Tree size is a tuning parameter governing the model's complexity, and the optimal tree size should be chosen from the data. The preferred strategy is to grow a large tree T_0, stopping the splitting process only when some minimum node size is reached. This large tree is then pruned using cost-complexity pruning.
We define a subtree $T \subset T_0$ to be any tree that can be obtained by pruning T_0, that is, collapsing any number of its internal (non-terminal) nodes. We index terminal nodes by m, with node m representing region R_m. Let $|T|$ denote the number of terminal nodes in T.

Consider a tree $T \subset T_0$ computed by pruning T_0 and define $|T|$ as the number of terminal nodes in T. Then, the cost complexity criterion is:

$$C_\alpha(T) = \sum_{m=1}^{|T|} N_m Q_m(T) + \alpha |T|$$

The idea is to find, for each α, the subtree $T_\alpha \subseteq T_0$ to minimize $C_\alpha(T)$. The tuning parameter $\alpha \geq 0$ governs the tradeoff between tree size and its goodness of fit to the data. Large values of α result in smaller trees T_α, and conversely for smaller values of α. As the notation suggests, with $\alpha = 0$ the solution is the full tree T_0. To find T_α we use weakest link pruning: we successively collapse the internal node that produces the smallest per-node increase in $\sum_m N_m Q_m(T)$, and continue until we produce the single-node tree.

2.1.2 Random Forest

Random Forest is an ensemble method. The idea behind it is to reduce the high variance associated to large decision trees by building a large collection of de-correlated trees, and averaging their results. Random forest improves over bagging methods by not only reducing correlation by training trees on different bootstrap samples but also by considering just a subset of variables at each split.

Random Forest algorithm

```
i. For $b = 1$ to $B$
    1. Draw a bootstrap sample $Z^*$ of size $N$ from the training data.
    2. Grow a random-forest tree $T_b$ to the bootstrapped data, by recursively re-
       peating the following steps for each terminal node of the tree, until the
       minimum node size $n_{\min}$ is reached.
        (a) Select $m$ variables at random from the $p$ variables.
        (b) Pick the best variable/split-point among the $m$.
        (c) Split the node into two daughter nodes.
ii. Output the ensemble of trees $\{T_b\}_1^B$.
```

The number m of variables considered at each split regulates the model's complexity and its ability to reduce correlation. For $m = n$ the model is equivalent to bagging. The suggested default value for m is $\lfloor \sqrt{p} \rfloor$ for classification and $\lfloor p/3 \rfloor$ for regression. However, as argued by Hastie et al.[2], it could still be beneficial to tune this parameter in order to obtain the best results.

[2] T. Hastie et al., 2001, [3] p. 592

2.1.3 Consensus and Probability

In the case of a classification problem, two different methods to aggregate the results of a random forest ensemble are possible:

- Consensus: $\hat{G}(x) = \text{argmax}_k\, q_k(x), k \in \{1, \ldots, K\}$, where $q_k(x)$ is the proportion of trees voting for the category k.
- Probability: $\hat{G}(x) = \text{argmax}_k\, B^{-1} \sum_{b=1}^{B} p_k^{[b]}(x)$, $k \in \{1, \ldots, K\}$, where $p_k^{[b]}(x)$ is the probability assigned by the b-th tree to category k;

In our analysis we make use of probability aggregation.

2.1.4 Variable importance

Variable importance plots can be constructed for random forests. At each split in each tree, the importance measure attributed to the splitting variable is the improvement in loss obtained through the split. This quantity is accumulated over all the trees in the forest separately for each variable.

For each variable ℓ and each tree T, we can compute the importance:

$$\mathcal{I}_\ell^2(T) = \sum_{t=1}^{J-1} \hat{\iota}_\ell^2 \mathbf{1}(v(t) = \ell)$$

where:

- the measure is computed for each internal node t, $t = 1, \ldots, J-1$;
- $v(t)$ is the variable selected for the partition in two regions;

- $\hat{\iota}_\ell^2$ is the estimated improvement in loss due to the split.

The extension to a random forest is:

$$\mathcal{I}_\ell^2 = \frac{1}{B} \sum_{b=1}^{B} \mathcal{I}_\ell^2\left(T_b\right).$$

This measure is much more reliable, due to the stabilizing effect of averaging; the importance values are scaled to sum to 1.

2.1.5 AdaBoost

Boosting is one of the most powerful learning methods of the last decades. It was introduced by Freund and Schapire (1997) and designed for classification problems, and applied to regression later. The purpose of boosting is to sequentially apply "weak" learners to repeatedly modified versions of the data, thereby producing a sequence of classifiers $G_m(x), m = 1, 2, ..., M$. The predictions from all of them are combined through a weighted majority vote to produce a final prediction. AdaBoost is an example of a boosting method, which for a two class problem, with output variable coded as $Y \in \{-1, 1\}$ and a vector of predictor variables X, is obtained through the following algorithm.

AdaBoost Algorithm

```
1. Initialize the observation weights $w_i = 1/N$, $i = 1, 2, \ldots, N$.
2. For $m = 1$ to $M$ :
    (a) Fit a classifier $G_m(x)$ to the training data using weights $w_i$.
    (b) Compute the weighted error
        $$\mathrm{err}_m = \frac{\sum_{i=1}^{N} w_i I\left(y_i \neq G_m\left(x_i\right)\right)}{\sum_{i=1}^{N} w_i}.$$
    (c) Compute the voting weights $\alpha_m = \log\left(\left(1 - \mathrm{err}_m\right) / \mathrm{err}_m\right)$.
    (d) Set $w_i \leftarrow w_i \cdot \exp\left[\alpha_m \cdot I\left(y_i \neq G_m\left(x_i\right)\right)\right], i = 1, 2, \ldots, N$.
3. Output $G(x) = \mathrm{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$.
```

Here $\alpha_1, \alpha_2, ..., \alpha_M$, computed by the boosting algorithm, weight the contribution of each respective $G_m(x)$: their effect is to give higher influence to the more accurate classifiers in the sequence.
In the first step the weights are set to $w_i = 1/N$, then for each successive iteration $m = 1, 3, ..., M$ the observation weights are individually modified and the classification algorithm is reapplied to the weighted observations. At step m, those observations that were misclassified by the classifier $G_{m-1}(x)$ have their weights increased, whereas the weights are decreased for those that were classifies correctly. Thus as iterations proceed, observations that are difficult to classify correctly receive ever-increasing influence. Each successive classifier is thereby forced to concentrate on those training observations that are missed by previous ones in the sequence.

In our implementation the $G(x)$ classifier are represented by stumps, decision trees of depth 1.

2.2 Logistic Regression

Logistic regression is used for classification problems to estimate the probability that a given datapoint $x = (x_1, \ldots, x_N)$ leads to outcome y_i for $i = 1, \ldots, k$, where k is the number of possible labels for the outcome classes. We assume now that we have two classes with y_i either 0 or 1. By using the sigmoid function, we can write the probabilities of each of the two outcomes as:

$$p(y_i = 1|x_i, \beta) = \frac{e^{\beta^T x_i}}{1 + e^{\beta^T x_i}}$$
$$p(y_i = 0|x_i, \beta) = 1 - p(y_i = 1|x_i, \beta)$$

where β is the vector of the parameters of the model which we want to estimate. To simplify the notation we will call $p(y_i = 1|x_i, \beta) = p_1(x_i, \beta)$ and $p(y_i = 0|x_i, \beta) = p_0(x_i, \beta)$.

Furthermore, our results were obtained introducing a l_2 regularization term λ. The cost function we look to minimize is:

$$C(\beta) = -\sum_{i=1}^{n} \left[y_i \beta^T x_i - \log(1 + e^{\beta^T x_i}) \right] + \lambda ||\beta||_2^2$$

In order to obtain the optimal vector of parameters β , we will use the mini-batch stochastic gradient descent algorithm [3].

2.3 Feed-Forward Neural Network

Neural Networks (NNs) are computational learning systems based on a collection of connected units or nodes called neurons, which reproduce the activity of neurons in the brain.
This project focuses on a specific NN model called *Multilayer Perceptron* (MLP): all nodes of a layer will be connected to every node in the subsequent layer, making it a fully connected network.
Each connection in the network is associated to a weight and each node has an intrinsic bias parameter and an activation function which returns the output of the node. Hence, the output node y is produced by the activation function f:

$$y = f(\sum_{i=1}^{n} w_i x_i + b_i) = f(z)$$

where w_i are the weight, b_i are the biases and x_i are the inputs which correspond to the outputs of the nodes in the previous layer.
Figure 1 [4] allows us to visualize the process:

[3] For a deeper explanation look at section 2.4 of [8]

[4] M. Ledum, 2017 [6]

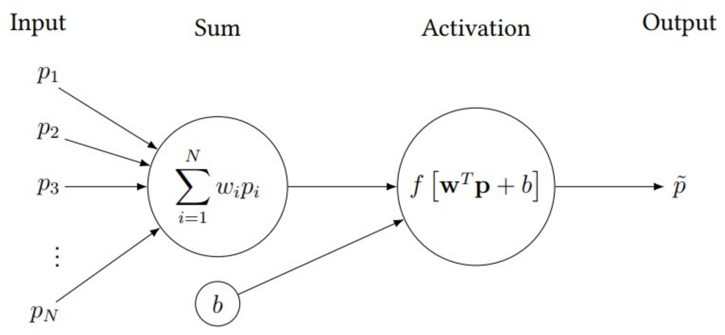

Figure 1: Feed-forward algorithm: the input values $p_i,\ i = 1, ..., N$ from the first layer multiplied by corresponding weights $w_i,\ i = 1, ..., N$ and summed. Then the activation function f is applied to $\mathbf{w^T p} + b$, in which b is a vector of dimension $N \times 1$. The output goes on to become input for neurons in the next layer.

By repeating the feed forward and back propagation algorithms over a certain number of epochs, we hope to move in the coefficient space towards a minimum for the cost function. In our case, to reduce the computational cost of the process we will use mini-batch SGD [5].

[5]For a deeper explanation look at section 2.3.1 of [8]

3 Discussion and results

3.1 Pima Indians Diabetes Dataset

The dataset we have considered in this analysis is been vastly used in classification settings since its publication in 1988. This dataset is originally from the US National Institute of Diabetes and Digestive and Kidney Diseases and its objective is to diagnostically predict whether or not a patient has diabetes, based on certain diagnostic measurements. The data is selected from a larger diabetes database by imposing several constraints: in particular, all patients considered are females at least 21 years old of Pima Indian heritage.
The dataset contains 8 numerical attributes which are presented in Table 1 and a binary outcome variable, which is equal to 1 for diabetic patients and 0 otherwise.

Table 1: Summary of the Pima Indians Database dataset variables.

Feature	Description	Data type
Preg	Number of times pregnant	Integer
Glucose	Plasma glucose concentration at 2 Hours (GTIT)	Continuous
Press	Diastolic Blood Pressure (mm Hg)	Continuous
Skin	Triceps skin fold thickness (mm)	Continuous
Insulin	2-Hour Serum insulin (lh/ml)	Continuous
BMI	Body mass index [weight in kg/(Height in m)]	Continuous
DPF	Diabetes pedigree function	Continuous
Age	Age (years)	Integer
Outcome	Binary value indicating non-diabetic /diabetic	Factor

The dataset contains information about 768 patients and has no null values and no missing values. However, as noted in 5, some attributes present inconsistent values: glucose concentration (Gluc), blood pressure (BP), skin fold thickness (Skin), insulin and BMI, contain zero values which are not within the normal range and are therefore inaccurate. Since the dataset is not very large, we prefer avoiding removing rows unnecessarily: for this reason, the zero values in the cited columns have been replaced by the column mean. The resulting distributions for all variables are presented in Figure 2.

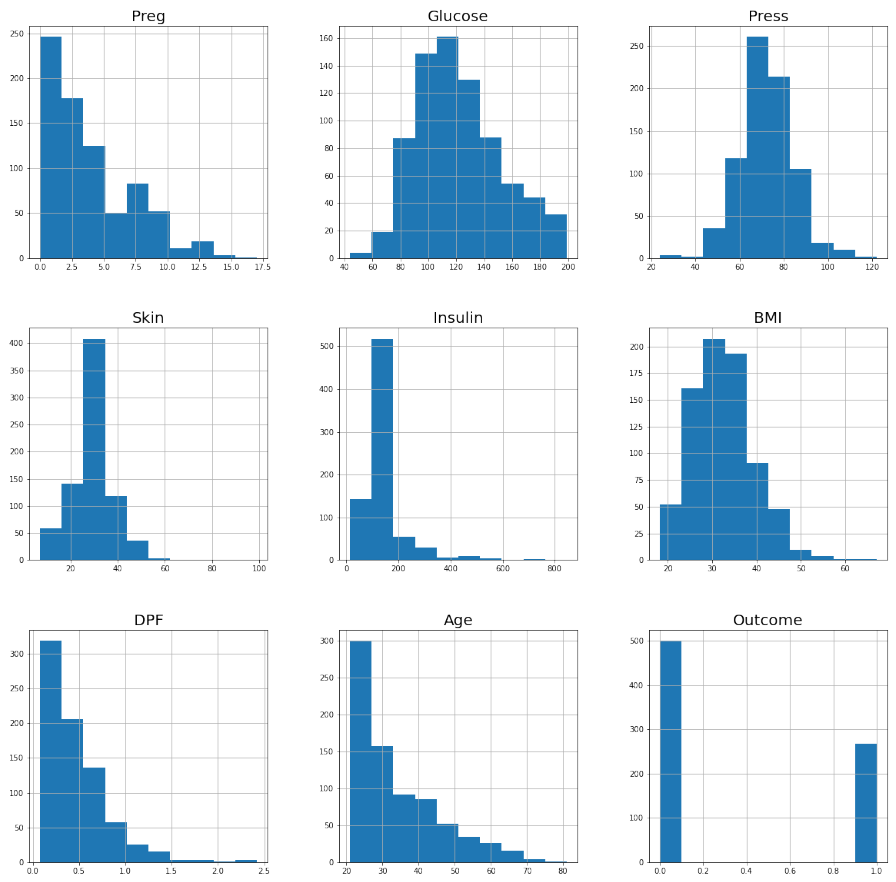

Figure 2: Histograms for the 8 predictors in the Pima Indians Diabetes dataset and the Outcome variable after data cleaning.

It is interesting to notice the imbalance between the two classes considered in the dataset: almost double the cases belong to the 0 class rather than the 1. This characteristic of our data will be taken into account in the final comparison of the methods performance.

We can also look at the correlation values between the predictors and the outcome, presented in Figure 3. As we can see, the variables most correlated with the outcome are Glucose and BMI. The other most correlated pairs between regressors are BMI and Skin Thickness and Age and Pregnancies.

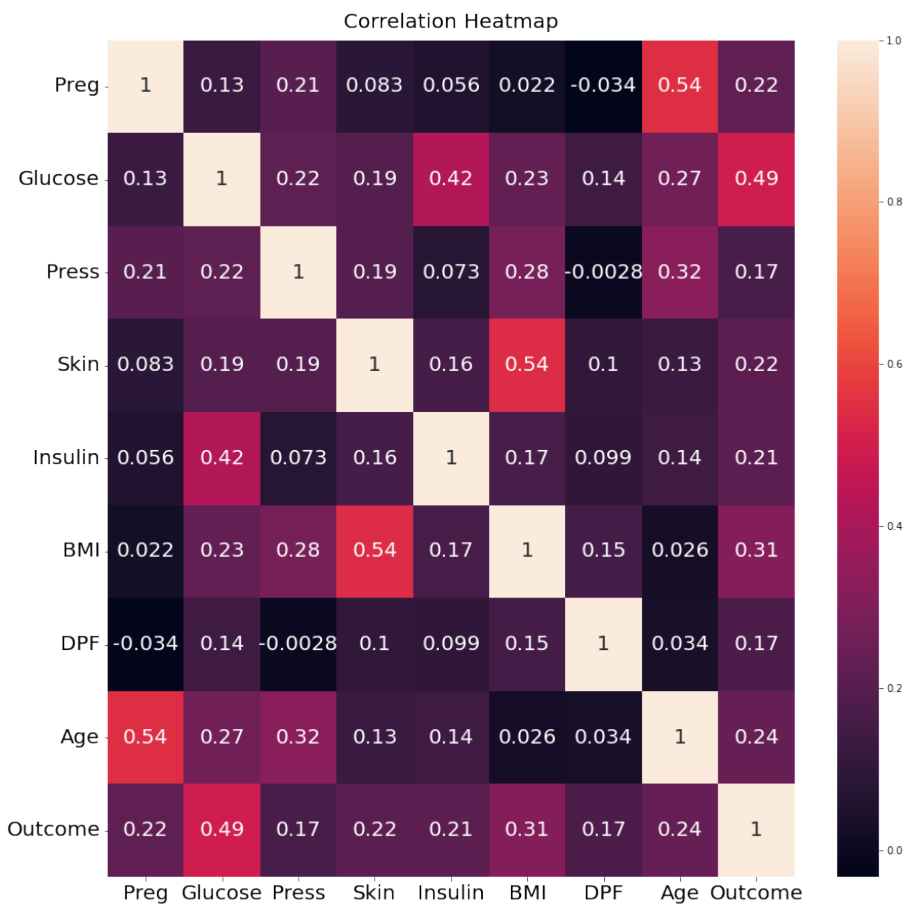

Figure 3: Correlation matrix for the 8 predictors in the Pima Indians Diabetes dataset and the Outcome variable.

3.2 Implementation

To deepen our understanding of the methods used in the analysis, we attempted to make our own implementation of them.

For the tree-based methods, the implementation was done in an object-oriented manner. The base class `Node` represents a single node of a classification tree: if the node is not a leaf, it contains the feature we are splitting on and its corresponding threshold, together with references to its left and right child nodes. If it is a leaf, it only contains the predicted value for that region of the feature space. The main class `DecisionTreeClass` implements a full decision tree for classification: through the implemented methods, the tree can be trained up to a certain depth, using both Gini index or cross-entropy as loss functions, and predict values for the test data.

Finally, additional classes `RandomForest` and `AdaBoost` were created as an implementation of the respective statistical methods. Table 2 reports the accuracy results for the tree-based methods that we implemented compared with the `scikit-learn` methods: as shown, the performances are similar, which supports the functionality of the methods we implemented.

However, the following analysis is performed using `scikit-learn` methods. This was done for computational time reasons: despite obtaining good results in terms of accuracy, our implementation, probably given the recursive nature of its methods, took a substantial time to be trained.

Table 2: Accuracy comparison for methods and our implementation.

	`scikit-learn`	Our implementation
Decision Tree Classifier	0.714	0.701
Random Forest	0.812	0.805
AdaBoost	0.779	0.786

The Feed-Forward Neural Network implementation was done in an object-oriented manner, creating both a `Layer` class and a `Model` class to be able to train the Neural Network through feed-forward and back-propagation functions.

Finally, Logistic Regression with mini-batch Stochastic Gradient Descent was implemented.

Both of these methods were already tested and used in the previous project for this course: the related code can be found in the previously referenced GitHub.

3.3 Classification metrics

The methods presented were compared on the basis of multiple metrics associated to classification problems, computed through the values in the confusion matrix, which contains information on the actual and predicted outcome classes on the testing set, as shown in Figure 4.

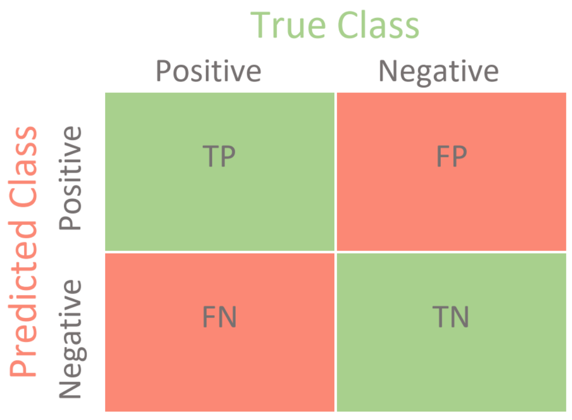

Figure 4: Example of confusion matrix.

- Accuracy refers to the percentage of all samples that have been predicted correctly. It is the ratio of the sum of true positives and true negatives to the total number of predictions made.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

- Precision refers to the percentage of all samples that have been correctly predicted as true among all those which were predicted as true, even if they were false.

$$Precision = \frac{TP}{TP + FP}$$

- Recall refers to the percentage of all samples that have been correctly predicted as true among all those which were predicted true as well as those predicted false but were true.

$$Recall = \frac{TP}{TP + FN}$$

- Specificity refers to the percentage of all samples that have been correctly predicted as false among all those which were false even if predicted incorrectly.

$$Specificity = \frac{TN}{TN + FP}$$

- The standard F-score is an indicator of a binary classification model's accuracy, calculated by the weighted average of the precision and sensitivity.

$$F - score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

3.3.1 ROC curve

The receiver operating characteristics (ROC) curve and the resulting area under the curve (AUC) represent the degree of separability of binary classes. The ROC curve is created by plotting *Recall* against $1-$ *Accuracy* at various threshold settings. AUC measures the entire two dimensional area underneath the ROC curve from (0.0) to (1.1) and has a maximum value of 1, with an uninformative classifier giving a value of 0.5.

3.4 Tree-based methods

The first method we considered in the classification analysis is the Decision Tree classifier. One of the main tuning parameters for this method is the depth of the final tree, which regulates the model complexity.

The regulation of the depth was done by developing a full tree and subsequently performing cost-complexity pruning. The results are shown in Figure 5. For $\alpha = 0$ we consider the full tree, which overfits on the training data reaching accuracy 1, but clearly loses in generalization ability. The best test accuracy is reached for $\alpha = 0.006$. The resulting pruned tree decision path is represented in Figure 6.

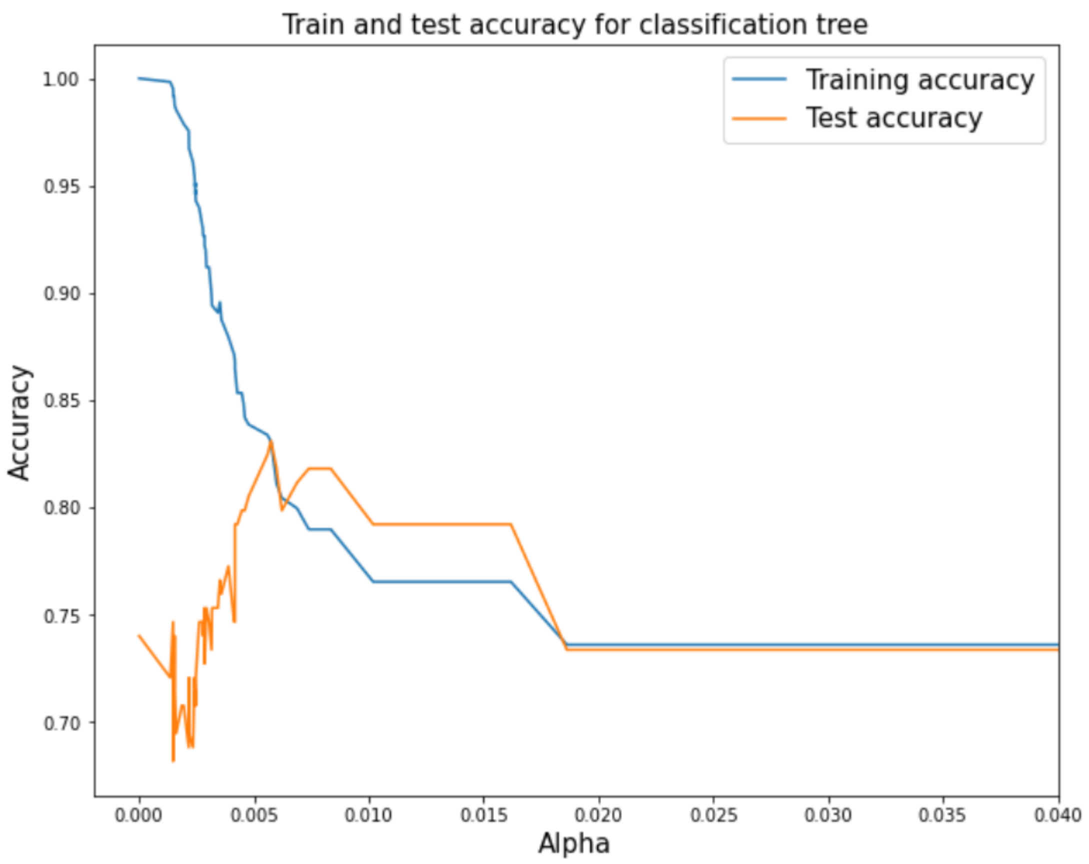

Figure 5: Accuracy on the training and test sets for different values of the cost-complexity parameter α for the Decision Tree classifier.

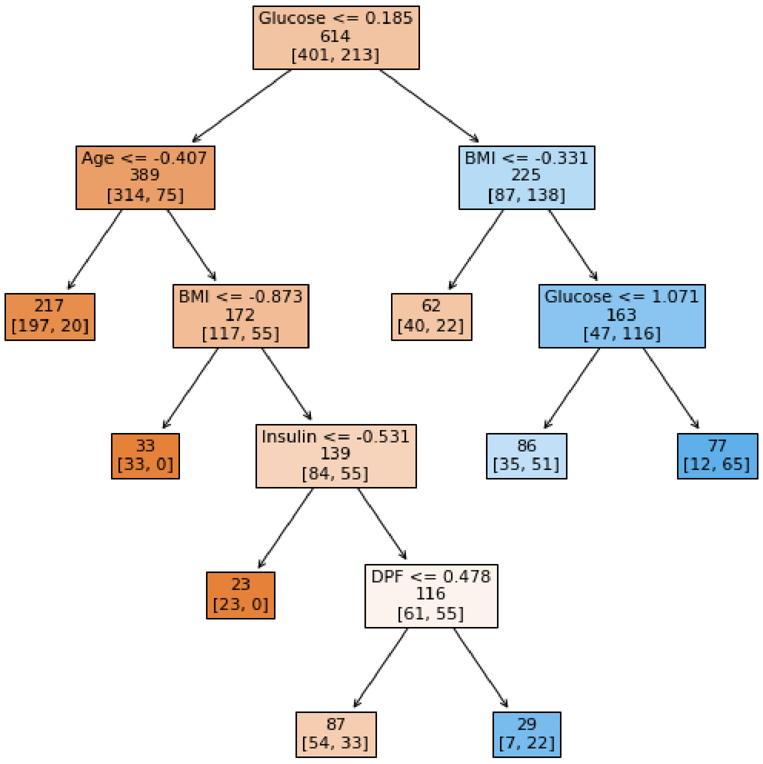

Figure 6: Decision path for the selected decision tree. The coloring visually represents the majority class in each node.

Secondly we analyse the data with the Random Forest method. One of the most crucial parameters for this method is the number of predictors considered by the trees at each split: considering an higher parameter m, results in higher model complexity and causes more correlation between the trees in the ensemble. For example, choosing $m = n$, with $n = 8$ the total number of predictors, results in selecting every variable in our model at each split, introducing no randomization: the model is therefore equivalent to bagging.

We go on to tune m. As shown in Figure 7, the best performance is obtained for $m = 2$: this corresponds to the recommended value for classification $\lfloor\sqrt{8}\rfloor = 2$.

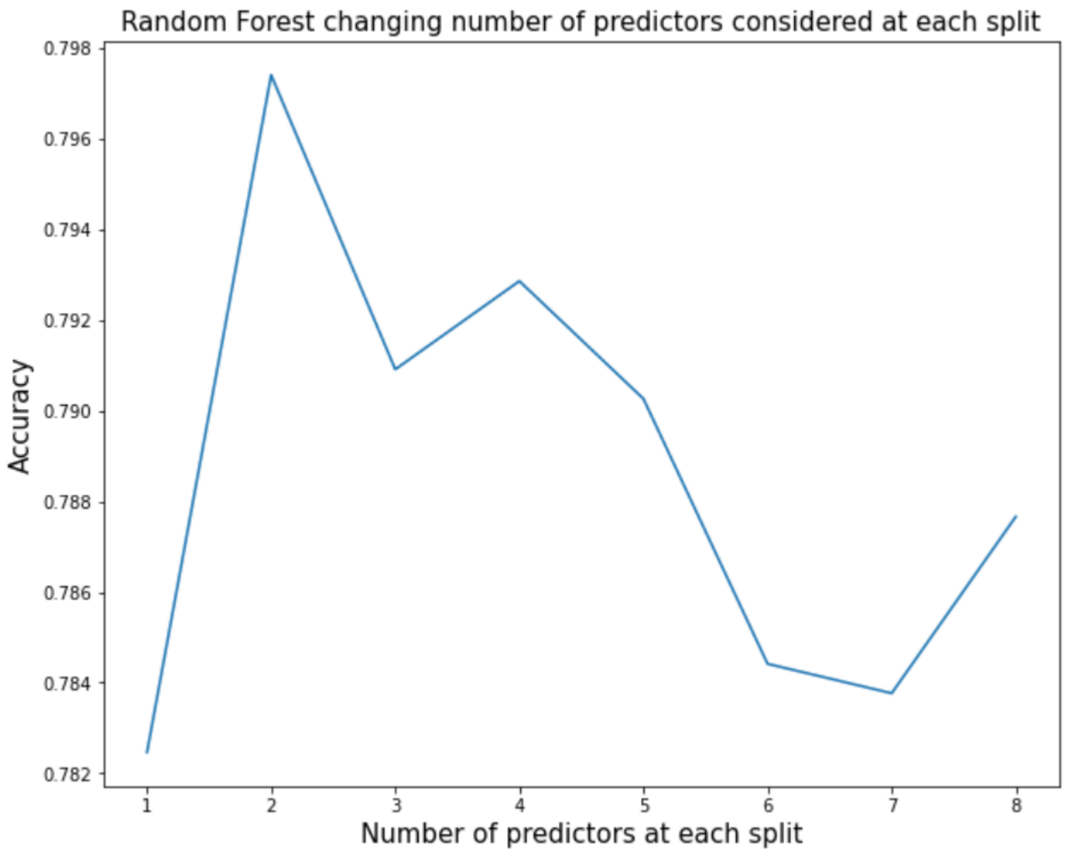

Figure 7: Test accuracy for Random Forest with changing number of predictors considered at each split.

Finally we applied the AdaBoost algorithm to our data. The results shown in Figure 8 highlight how the ensemble methods considered provide a better performance in terms of test accuracy then a single unpruned tree.

Random Forest and Adaboost present similar results up to about 200 trees in the ensemble, but accuracy for AdaBoost is reduced when increasing the number of trees further. This could be due to overfitting: as proved by Leo Breiman when firstly introducing the method [7], random forests do not suffer from overfitting when changing the number of trees considered, while when dealing with AdaBoost it is usually beneficial to introduce some form of early stopping to prevent the method to fit the training data to closely and lose generality.

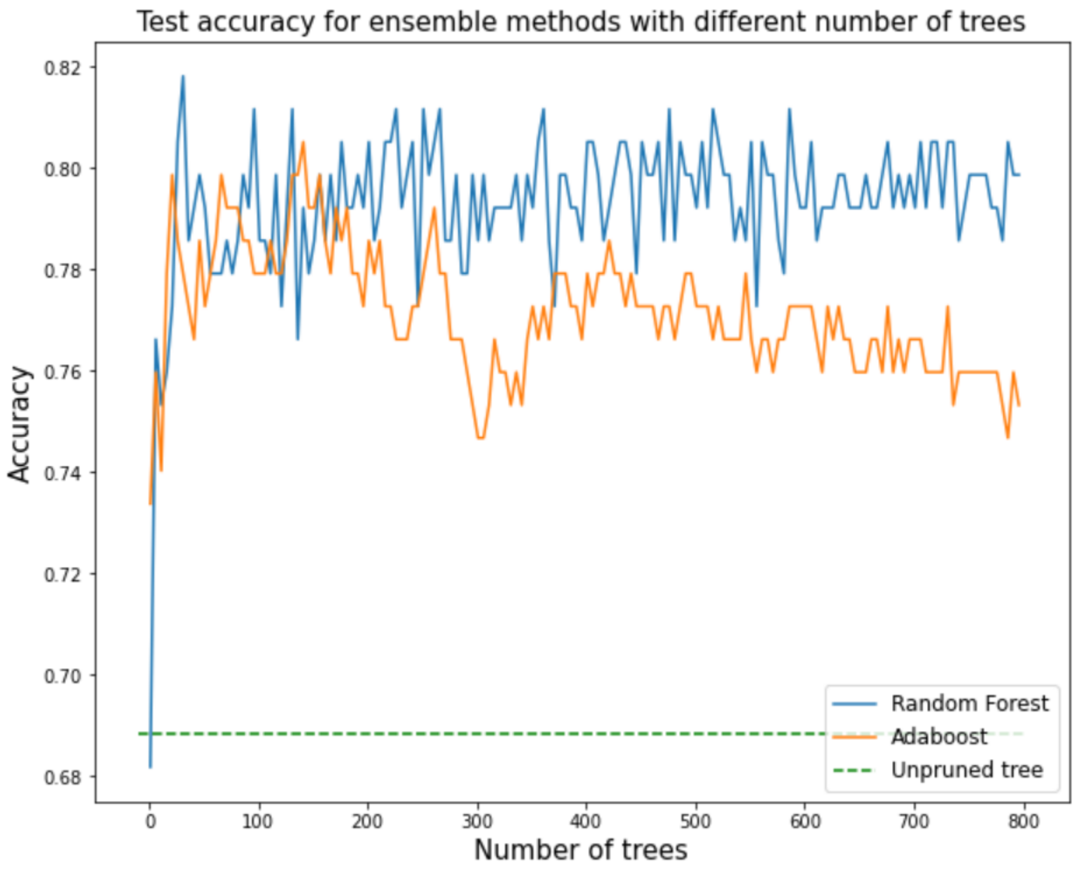

Figure 8: Comparing ensemble methods accuracy for increasing number of trees. The green dotted line shows the result for a single unpruned tree.

Figure 9 shows the ROC curves for the three classification methods presented so far. The curves are relatively similar and give similar values of AUC (Area Under Curve).

Table 3 shows the comparison on different metrics for the three classification methods presented so far. We found that the pruned decision tree presents the best result for all metrics: given the enhanced interpretability and portability that this method allows, it is probably the best choice for the classification of our data so far.

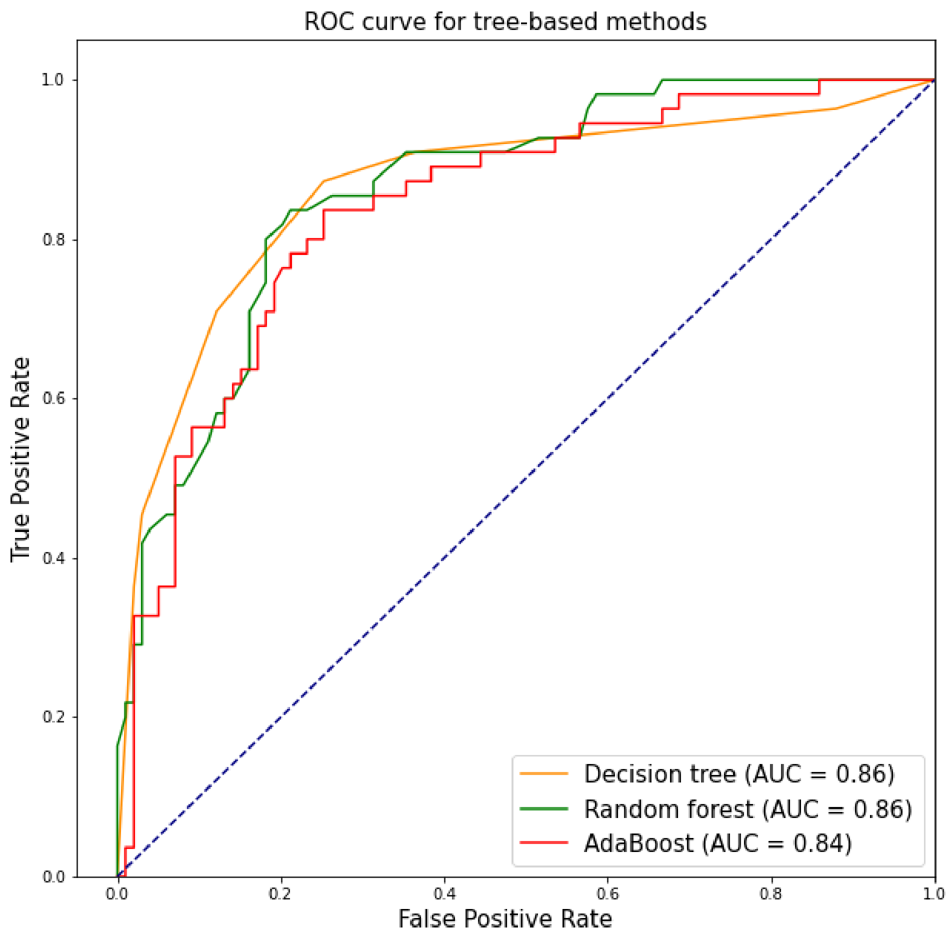

Figure 9: ROC curves plot comparison for Decision Tree, Random Forest and AdaBoost.

Table 3: Metrics comparison for tree-based methods.

	Accuracy	Precision	Recall	Specificity	F-Score	AUC
Decision Tree	0.818	0.765	0.709	0.878	0.736	0.864
Random Forest	0.766	0.711	0.581	0.868	0.640	0.863
AdaBoost	0.772	0.685	0.672	0.828	0.678	0.839

3.4.1 Variable importance analysis

In this section, we analyse a possible variation on our model through variable selection.

As shown in 2.1.4 Random Forest can be used naturally to obtain a measure of variable importance. Figure 10 shows this analysis: as we also noticed when considering correlation between variables and outcome, Glucose and BMI are clearly the two crucial variables in determining the classification.

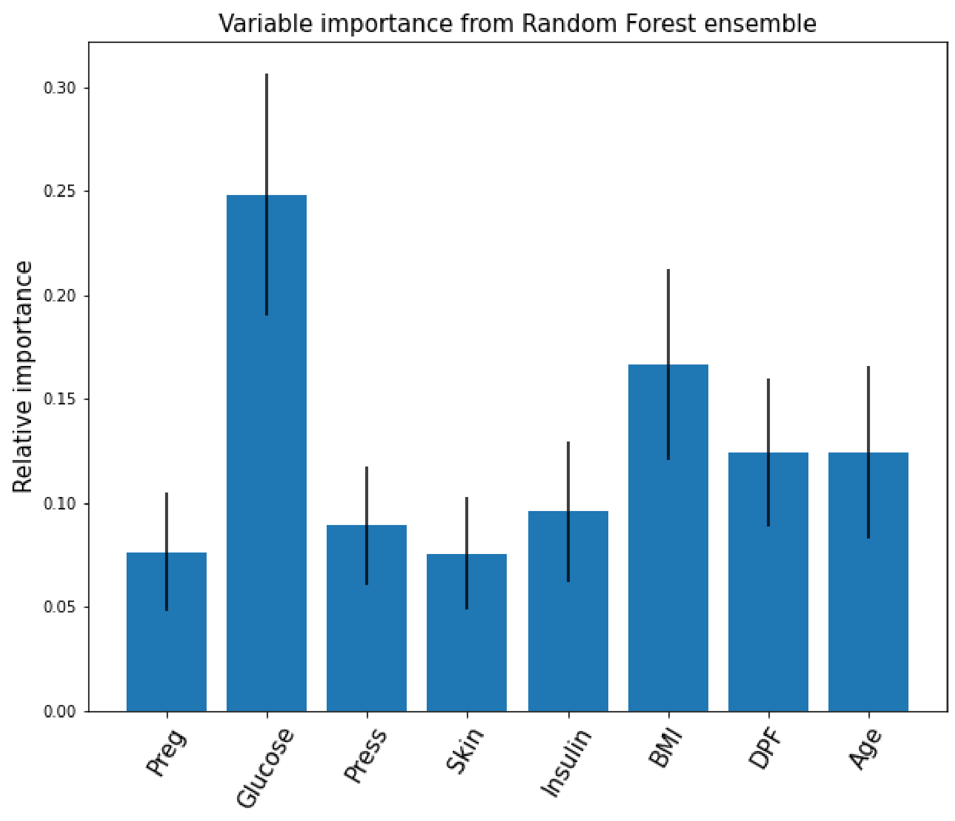

Figure 10: Relative importance plot of the 8 explanatory variables from a Random Forest ensemble.

Given this results, we go on to consider a model restricted to Glucose and BMI variables. As shown in Table 4, the performances in terms of accuracy for the full and restricted model are relatively similar. It is usually preferable to choose a simpler model over a denser one: however, given that our dataset only contains 8 variables and is still highly interpretable, we conclude that the best choice is still the full model.

Table 4: Accuracy comparison for full and restricted model on tree-based methods.

	All variables	2 variables
Decision Tree	0.818	0.792
Random Forest	0.766	0.753
AdaBoost	0.772	0.768

3.5 Logistic Regression and Neural Networks

We now continue our analysis considering Logistic Regression for classification. In particular, the method implemented determines Logistic Regression coefficients through Stochastic Gradient Descent with 200 epochs and mini-batches of size 50.

The parameters for the method are the learning rate μ and the coefficient for the l_2 penalty term λ. We tune these parameters through an accuracy grid-search, shown in Figure 11. The best results are obtained for $\mu = 0.001$ and $\lambda = 1$.

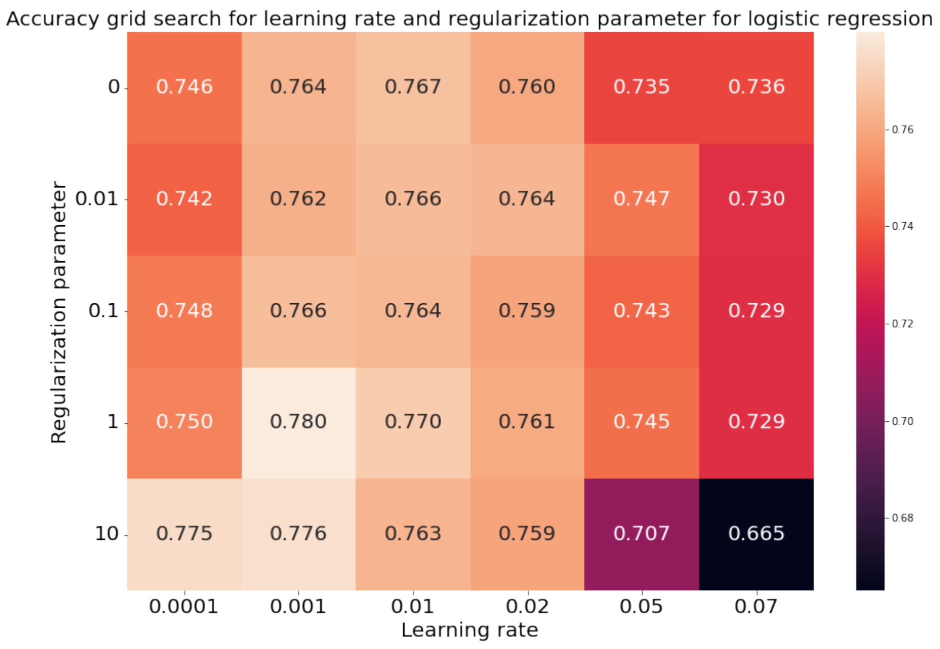

Figure 11: Grid search for the optimal accuracy as function of the learning rate and regularization parameter.

Finally we try fitting our data by implementing a Feed Forward Neural Network with a flexible number of layers, nodes per layer and different activation functions. Since we are considering two classes, the output layer for our network will always contain a single neuron with a sigmoid activation function: the output will lie between 0 and 1 and will be interpreted as the probability of the datapoint belonging to class 1. The cost function we considered is this case is the cross-entropy.

The results of tuning the parameters for the method, considering varying number of neurons and hidden layers, as well as different activation functions is shown in the following figures (12, 13, 14, 15).

The final chosen model has two hidden layers with 20 neurons each and sigmoid activation function with learning rate $\eta = 0.02$ and without introducing any regularization.

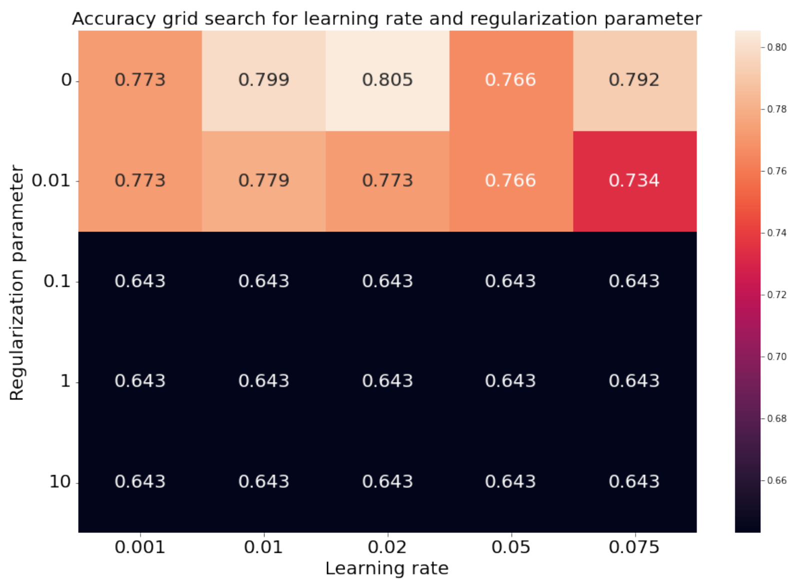

Figure 12: Grid search for the optimal accuracy as function of the learning rate and regularization parameter. The chosen parameters are $\eta = 0.02$ and $\lambda = 0$.

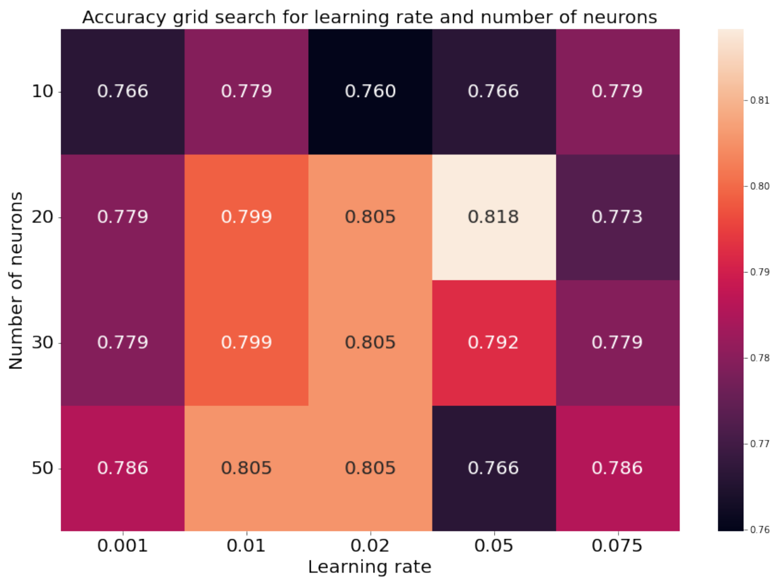

Figure 13: Grid search for the optimal accuracy as function of the learning rate and number of neurons in a single layer.

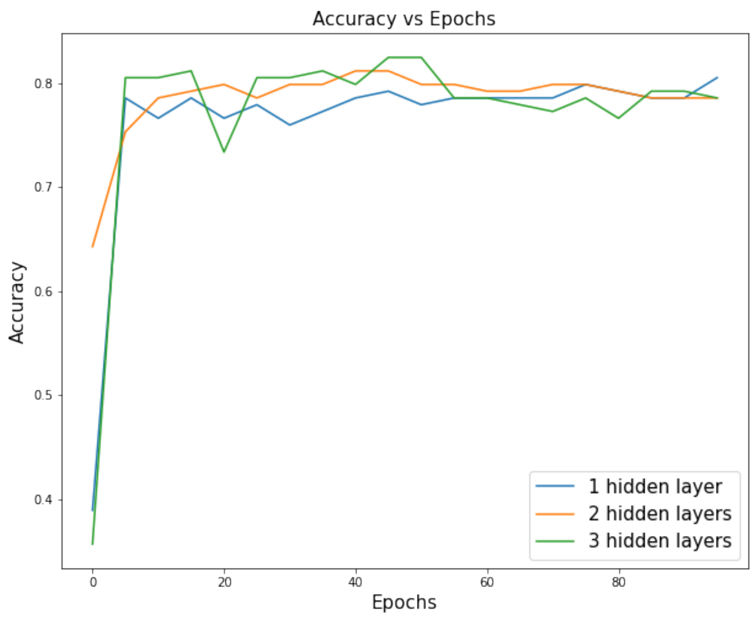

Figure 14: Grid search for the optimal accuracy changing the number of layers in the network with 20 neurons each. We choose a network with 2 layers since it gives the most consistent results.

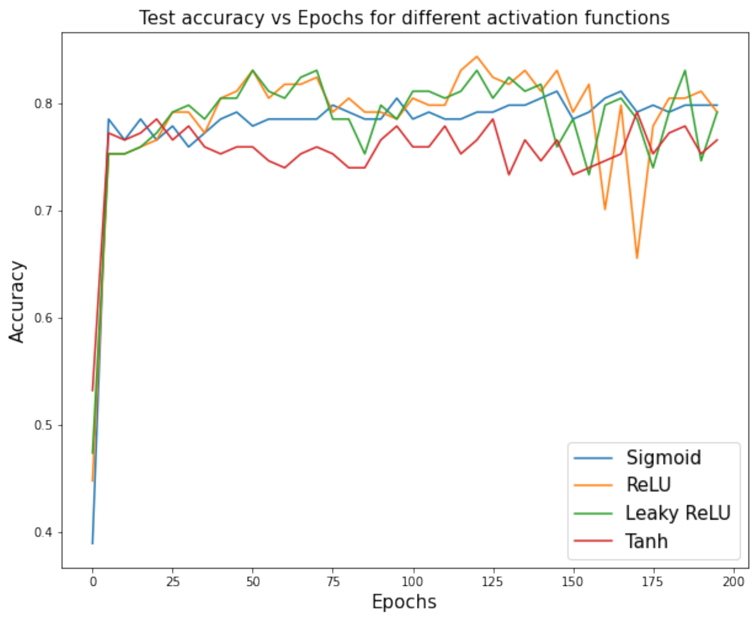

Figure 15: Plot for the optimal accuracy with a 2-layer network changing activation function. Even though ReLU and Leaky ReLU obtain some of the best results for certain number of epoch, we choose Sigmoid which appears more consistent.

4 Conclusion

The aim of this project was to implement, study and compare the performance of different statistical methods for classification on the Pima Indians Diabetes dataset. The methods were trained to predict whether a subject's diabetes mellitus diagnosis is positive according to 8 given attributes.

We started by attempting the implementation of Decision Tree, Random Forest and AdaBoost classifiers and testing their performance compared to the corresponding `scikit-learn` methods. We also made use of previously implemented codes for a Feed-Forward Neural Network and logistic regression with mini-batch SGD.

Through our analysis we also considered the possibility of restricting the number of variables in the model, choosing the ones that were deemed most crucial in the variable importance analysis performed with Random Forest. This showed that considering just the Glucose and BMI variables only slightly impacts the model's performance: given that the full model only contains 8 variables and is still highly comprehensible in its results, we concluded that using the full model is still preferable.

Table 5 shows the comparison of all methods tested on various classification metrics: as we can see, the single pruned decision tree classifier outperforms all other methods when considering accuracy or AUC, while Logistic Regression is the best choice for recall and F-Score. Finally, the Neural Network model presents the highest values for precision and specificity. Given its simplicity and high interpretability, it could be argued that the Decision Tree is the best choice to study our dataset. The best result for accuracy we could obtain was 81.8%: although already high, future research could be dedicated to expanding the data collection around diabetes to identify other important variables not yet considered, in order to improve the predictive capability of statistical models.

It is also interesting to note the high imbalance in the results of almost all methods between recall and specificity. This is probably due to the imbalance between the number of datapoints in the two classes considered (diabetes positive or negative): most methods predict the 0 class for most datapoints, resulting in high specificity (correctly identifying the negative cases) but lower recall (correctly identifying the positive cases). This is not true for Logistic Regression, which presents a very high recall score: this can also be seen when looking at the models confusion matrices in Figure 16, observing that Logistic Regression predicts more cases as belonging to class 1 then all other methods.
In the case of medical screenings, recall is a crucial metric: we prefer predicting false positives, which could be later excluded with additional medical testing, than false negatives. Given this observation, it could be argued that Logistic Regression is the best choice for our analysis.

Decision Tree

		Predicted	
		0	1
Actual	0	87	12
	1	16	39

AdaBoost

		Predicted	
		0	1
Actual	0	82	17
	1	18	37

Random Forest

		Predicted	
		0	1
Actual	0	86	13
	1	23	32

Logistic Regression

		Predicted	
		0	1
Actual	0	74	25
	1	7	48

Neural Network

		Predicted	
		0	1
Actual	0	89	10
	1	20	35

Figure 16: Confusion matrices on test data prediction for all classification methods considered.

Table 5: Comparison of all methods tested on different classification metrics. The highest value in each column is in bold.

	Accuracy	Precision	Recall	Specificity	F-Score	AUC
Decision Tree	**0.818**	0.765	0.709	0.878	0.736	**0.864**
Random Forest	0.766	0.711	0.581	0.868	0.640	0.863
AdaBoost	0.772	0.685	0.672	0.828	0.678	0.839
Logistic Regression	0.792	0.657	**0.872**	0.747	**0.750**	0.863
Neural Network	0.805	**0.777**	0.636	**0.899**	0.700	0.862

References

[1] World Health Organistion (2016) Global Report on Diabetes. Introduction, https://www.who.int/publications/i/item/9789241565257, accessed December 13th 2022

[2] L.O. Schulz, P.H. Bennett, E. Ravussin, J.R. Kidd, K.K. Kidd, J. Esparza, Valencia ME (2006) Effects of traditional and western environments on prevalence of type 2 diabetes in Pima Indians in Mexico and the US. Diabetes Care 29(8):1866–1871

[3] T. Hastie, R. Tibshirani, and J. Friedman, The Elements of statistical learning, Springer, New York, USA, (2001), Jan 13th 2017, pp. 305-310, pp. 338-339, pp. 587-594

[4] V. Chang, J. Bailey, Q.A. Xu, et al., Pima Indians diabetes mellitus classification based on machine learning (ML) algorithms. Neural Computing and Applications (2022).

[5] Zia UA, Khan N (2017) Predicting diabetes in medical datasets using machine learning techniques. Int J Sci Eng Res 5(2):257–267

[6] M. Ledum, A Computational Environment for Multiscale Modelling, chapter 7. MA thesis, Universitetet i Oslo, https://www.duo.uio.no/handle/10852/61196

[7] Breiman, L. Random Forests. Machine Learning p. 45, pp. 5–32 (2001). https://doi.org/10.1023/A:1010933404324, accessed December 13th 2022

[8] M. B. Cattini, M. D'Alessandro, L. Incollingo, V. Ventura, Project 2, Applied data analysis and machine learning, 2022 https://github.com/matteodales/FYS-STK4155_Applied_Data_Analysis_and_Machine_Learning/tree/main/project2