Instituto Superior Técnico

Advanced Automation

2023/2024

Project Report
Diabetes Health Indicators Project

Name	**Number**	**Work Done**
Gonçalo Bruno Vitorino de Sousa	100179	Support Vector Machines
Guilherme Pereira Joaquim	100188	Basic and Decision Tree-based Methods
Stefanos Georgios Zafiris	109041	Neural Networks

GitHub repository: `https://github.com/guilhermejoaquim08/AA_Project`

Contents

1 Introduction

Diabetes prevalence has reached alarming levels in recent years, presenting a significant global health challenge. The diagnosis of diabetes is crucial for effective management and the prevention of complications. In this paper, we conduct a comprehensive examination of a predictive model utilizing machine learning (ML) to classify individuals based on their diabetes status across 21 categories. The dataset employed in this study is the *Diabetes Health Indicators Dataset* accessible on Kaggle [1].

This paper provides an in-depth review of the dataset, which contains a wide range of health-related variables. The data is gathered by the government of the United States as part of their "The Behavioral Risk Factor Surveillance System (BRFSS)" a state-based telephone survey. The survey is based on only adults 18 years and older residing in the United States.

The methodology involves the application of various ML algorithms, such as the Decision Tree methods, Support Vector Machines (SVM), and Neural Networks (NN), to identify the effectiveness of each model for diabetes classification. This involves training each algorithm on a subset of the data and then comparing their performance on an unseen test set.

The first section of this report will go over how the data is preprocessed and divided into a training and test set. Following that is a summary of the various ML methods used to categorize diabetes, as well as examples of the results.

2 Data preprocessing

The dataset comprises data from the Behavioral Risk Factor Surveillance System (BRFSS) 2021, a national survey aimed at examining health-related risk factors and chronic conditions among adults in the United States. The dataset specifically focuses on diabetes, a prevalent chronic illness that affects how the body converts food into energy. The data is intended for analyzing risk factors and predicting diabetes status among adults. The dataset is divided into three different kinds of datasets.

We opted for the second dataset where the target variable is binary with $0 =$ No diabetes and $1 =$ Diabetes (also includes pre-diabetes). also, this dataset is already balanced, therefore we need not do any resampling.

The dataset consists of 21 explanatory variables and 1 target variable. The explaining variables are as follows:

- High blood pressure: Binary
- High cholesterol: Binary
- Cholesterol checked in the last 5 years: Binary
- Body mass index: Integer
- Have you smoked at least 100 cigarettes in your entire life: Binary
- Ever had a stroke: Binary
- Physical activity in the past 30 days: Binary
- Consume more than 1 fruit per day: Binary
- Consume more than 1 vegetable per day: Binary
- Heavy drinker: Binary
- Any health care coverage (insurance etc): Binary
- Could not see a doctor in the last 12 months because of the price: Binary
- General health: Integer
- Mental health: Integer

[1] `https://www.kaggle.com/datasets/julnazz/diabetes-health-indicators-dataset?select=diabetes_012_health_indicators_BRFSS2021.csv`

- Physical health: Integer
- Difficulties walking or climbing stairs: Binary
- Sex: Binary
- Age: Integer
- Level of education: Integer
- Income: Integer

When it comes to binary explaining variables, $0 =$ No and $1 =$ Yes. The target variable is also binary.

This dataset needed to be preprocessed before the ML algorithms could be trained. The dataset is divided into explanatory variables (X) and the target variable (Y). The dataset is then split into training and testing sets using the sklearn method `train_test_split`. The division is composed of 80% training and 20% testing.

3 Machine learning methodology and results

3.1 Basic methods

Basic methods offer a solid starting point for understanding and implementing classification in diverse applications. Their simplicity and interpretability make them essential tools for exploring machine learning.

This section presents a comparative analysis of basic classification methods for a diabetes prediction dataset. Initially, the dataset is imported and split into features ('X') and the target variable ('y'), followed by a further split into training and testing sets (a train size of 0.8). For this analysis, we used models such as Linear Discriminant Analysis (LDA), Quadratic Discriminant Analysis (QDA), Logistic Regression, and Naive Bayes. Each model was trained, utilized for predictions, and its error rate was calculated. We implemented a function to smooth the visual appeal of the results, as the graph before appeared rough (smooth function). The results obtained were presented below:

Error Rate:

- Error for **LDA**: 0.25
- Error for **QDA**: 0.27
- Error for **Logistic Regression**: 0.25
- Error for **Naive Bayes**: 0.28

LDA and Logistic Regression models show lower error rates, indicating better classification performance. QDA and Naive Bayes have slightly higher error rates.

Area Under the Curve (AUC):

- AUC for **Logistic Regression**: 0.820
- AUC for **LDA**: 0.819
- AUC for **QDA**: 0.778
- AUC for **Naive Bayes**: 0.783

The AUC values show how well the methods can distinguish positive and negative cases. So we want to maximize AUC. By analyzing the AUC values, we note that Logistic Regression and LDA display similar AUCs, suggesting both models perform well. However, Logistic Regression might be preferred due to its slightly lower error rate. On the other hand, QDA and Naive Bayes have slightly lower AUCs and exhibit higher error rates.

In this project, prioritizing the AUC makes the most sense, and it appears that QDA is likely the optimal method among the basic approaches.

The ROC curve obtained is presented below:

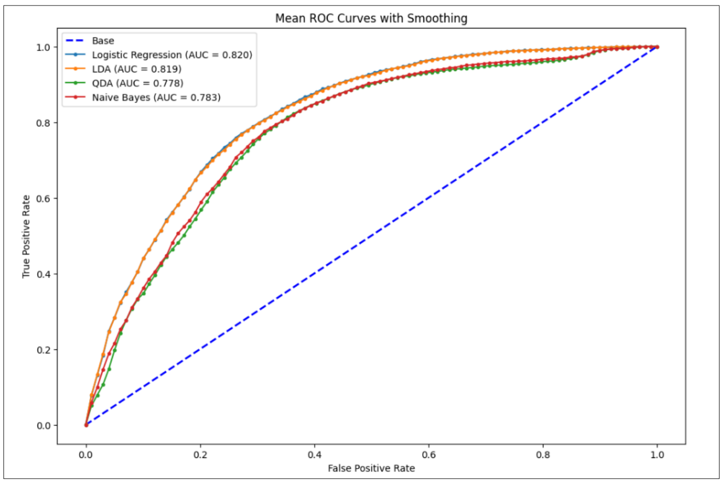


Figure 1: Roc Curve for Basic Methods

3.2 Support Vector Machine

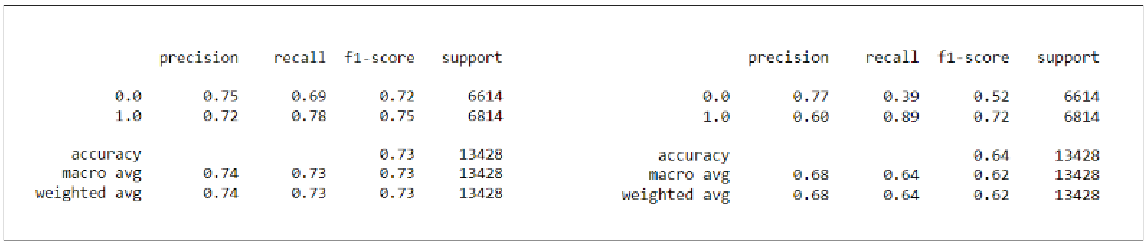
Support Vector Machines (SVM) constitute an extremely useful supervised learning algorithm for dealing with either classification or regression problems. Their functioning relies on the algebraic concepts of hyperplanes, which they use to separate the n-th dimensional data points depending on the target variable value.

The fundamental function of SVM is to find the optimal hyperplane in a high-dimensional space that best separates data points belonging to different classes. This hyperplane is positioned to maximize the margin, which is the distance between the hyperplane and the nearest data points from each class, known as support vectors.

In our project, this method was applied to the diabetes dataset to develop models that can predict this disease in a patient, given some relevant parameters.

Firstly, in terms of preprocessing, the data set was divided into training/testing data, preserving 20% for the testing of the model. There was no need for any further treatment of the information since the data was already balanced with 50% of positive cases and no missing arguments (NaN).

The application of the method was performed using different conditions and parameters. Two types of model were trained, using different types of kernel: linear (‘linear’) and non-linear (‘rbf’). Initially, a model of each kind was created using default values for the hyperparameters C (=1), the Cost, and gamma (=1), with the latter applying to the non-linear case. Afterward, besides testing the models, we also used GridSearchCV to evaluate which combination of parameters (from a designated range) originated the best results, whose quality was determined based on the F1 Score. The choice of this scoring metric relies on the fact that, from the medical perspective, the existence of false negatives should be avoided as it dangerously delays the treatment of the patient, with the F1 Score taking into consideration this phenomenon.



```
              precision    recall  f1-score   support

         0.0       0.75      0.69      0.72      6614
         1.0       0.72      0.78      0.75      6814

    accuracy                           0.73     13428
   macro avg       0.74      0.73      0.73     13428
weighted avg       0.74      0.73      0.73     13428
```

```
              precision    recall  f1-score   support

         0.0       0.77      0.39      0.52      6614
         1.0       0.60      0.89      0.72      6814

    accuracy                           0.64     13428
   macro avg       0.68      0.64      0.62     13428
weighted avg       0.68      0.64      0.62     13428
```

Figure 2: Classification Report for the linear (left) and non-linear (right) kernels

Regarding the initial tests, we obtained the results shown above using `classification_report` from `sklearn`. We can infer through observing the results that, overall, the linear kernel presented better results, however consisting of a simpler model. This is further proven by the ROC Curve in Figure 3 (linear and rbf AUC’s are 0.81 and 0.72, respectively). The linear kernel having a higher performance than the non-linear one is a considerably unexpected result that is probably explained by the values chosen for the hyperparameters. In regards to the F1 Score, the linear model also outperformed the non-linear one (0.75 vs 0.72), meaning that it has a better predictive capability by balancing well the precision and recall. Therefore, it is probable that a positive prediction made by the linear model will be correct.

Nevertheless, in terms of recall, it's essential to note that the non-linear model exhibited a notably high recall value (0.89 compared to 0.78), signifying its success in identifying diabetes cases. This aligns with the medical perspective of prioritizing sensitivity to avoid missing cases, as explained previously. Wrongly overestimating is preferable instead of missing a sick patient (as commonly said, "better safe than sorry").

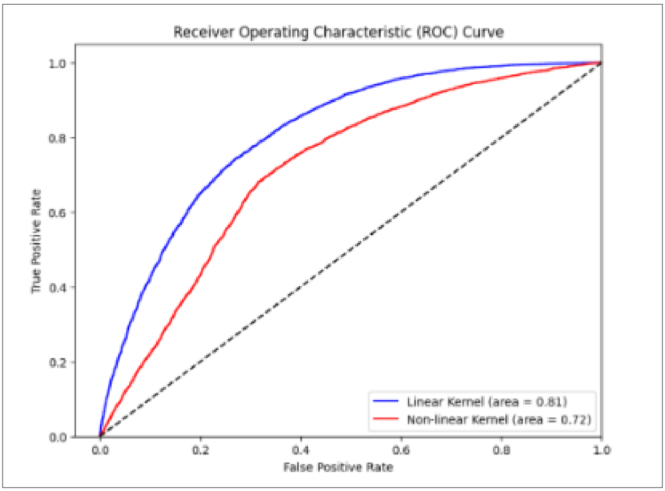
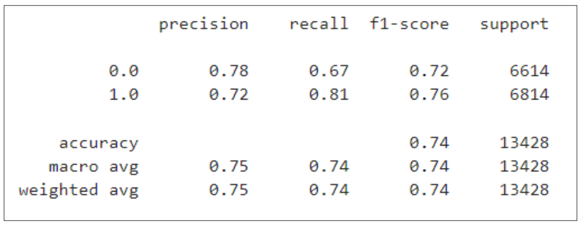


Figure 3: ROC Curve for SVM

Knowing that the non-linear solution has greater flexibility and thus more predicting power than the linear one further motivates the use of GridSearchCV to optimize the models and obtain their best performances in terms of F1 Score. Regarding the inputs of the grid search, we opted to use 2-Fold Cross Validation[2], since this operation is very time-consuming for SVM, even while using the maximum number of cores (`n_jobs = -1`).

The aforementioned optimization was performed similarly for both kernels. Lists containing different values of each hyperparameter were considered: {`'C': [0.01, 0.1, 1, 10, 100]`} for the linear kernel and {`'C': [0.01, 0.1, 1, 10, 100], 'gamma': [0.001, 0.01, 0.1, 1, 10]`} for the rbf kernel. The GridSearchCV executed the Cross Validation, compared the F1 Score of each model created, and returned the values that produced the best results. The best parameters were `C = 0.01` for the linear kernel and `C = 100, gamma = 0.001` for the non-linear one.

Using the optimized hyperparameters, the program was executed anew, obtaining the same score for the linear model, which may be explained by the SVM finding a robust decision boundary less sensitive to the Cost, and the score below (Figure 4) for the rbf kernel.



```
              precision    recall  f1-score   support

         0.0       0.78      0.67      0.72      6614
         1.0       0.72      0.81      0.76      6814

    accuracy                           0.74     13428
   macro avg       0.75      0.74      0.74     13428
weighted avg       0.75      0.74      0.74     13428
```

Figure 4: New Classification Report for the non-linear kernel

By observing the report and comparing the values to Figure 2, we can conclude that the new model surpassed the previous ones. Every scoring metric is equal (precision) or higher than the previous models', demonstrating a very good overall performance. The optimized parameter, **F1 Score**, has a

[2]The generally better options for Cross Validation are 5- and 10-folds, which would have been chosen if it was not for the "never-ending" waiting time.

greater value than before (**0.76** $\leftarrow$ 0.72), as intended, but the Recall parameter was penalized (0.81 $\leftarrow$ 0.89). To have a better general result, the model's ability to cover all positive cases decreased.

On the other hand, resuming the overall product of the optimization, the ROC Curve below (Figure 5) is a further indicator of the mentioned success. However close the AUC values are, the non-linear model is slightly higher.

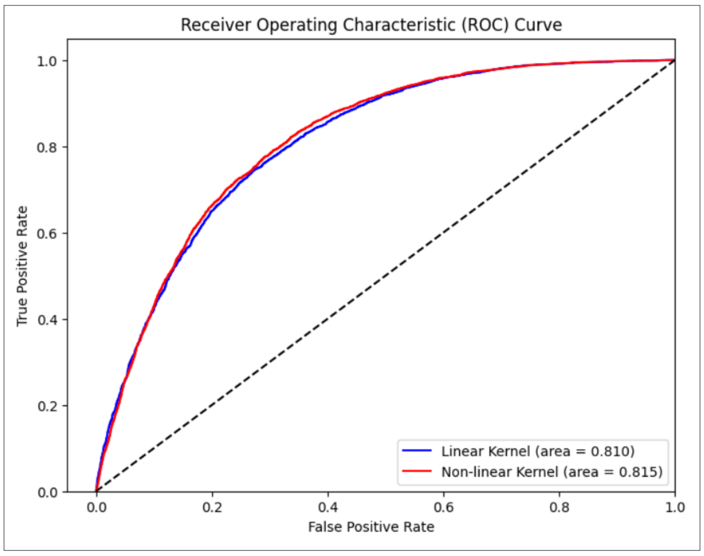


Figure 5: New ROC Curve for SVM

3.3 Neural Network

Neural networks are a type of machine learning algorithm inspired by the structure and function of the human brain. They are composed of interconnected nodes called neurons, which are organized into layers. Neural networks can learn from data by adjusting the weights of the connections between neurons during backpropagation. This allows them to perform a wide variety of tasks, such as image recognition, natural language processing, and speech recognition. A basic neural network is showcased in Figure 6.

Activation functions are used to introduce non-linearity into neural networks. This is important because it allows the networks to learn more complex patterns than they could otherwise. Some common activation functions include the sigmoid function, the tanh function, and the ReLU function.

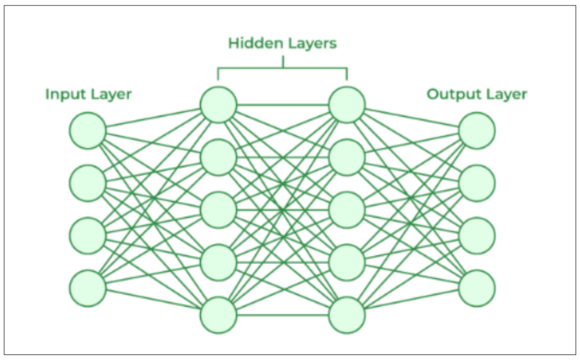


Figure 6: An image of a fully connected neural network

The Keras library is utilized to construct the neural network. Given the need to process the data sequentially, the network is designed as a sequential multilayer perceptron network. It consists of an input layer with a size of 21 and an output layer with one neuron and sigmoid activation. This architecture is chosen for binary categorization. As it aims for binary classification, the loss is set to binary cross-entropy, commonly employed in such applications. Adam is chosen as the optimizer due to its faster and more efficient convergence compared to stochastic gradient descent (SGD).

Since neural networks exhibit significant unpredictability, performing feature engineering for determining hyperparameters is challenging. Therefore, grid search is employed. Grid search is a brute-force optimization method that exhaustively evaluates all possible combinations of hyperparameter values to find the best-performing model. The grid search is conducted with the following values:

- Neurons in the input layer: 16, 32, 64
- Neurons per layer: 16, 32, 64, 128, 256
- Number of hidden layers: 0, 1, 2
- Learning rate: 0.0001, 0.001, 0.01
- Batch size: 128
- Epochs: 10

Following this, the optimal configuration achieved in terms of validation accuracy was with 64 neurons in the input, 0 hidden layers, 32 neurons per layer, and a learning rate of 0.001. These values were then saved, and the neural network was retrained, this time with 100 epochs and a batch size of 32. As the code lacks any form of L2 regularization or dropout, checkpoints were employed to save the best model in terms of validation accuracy. The model with the best weights was then loaded back, and three different metrics were obtained for the test set: Accuracy, Loss, and F1 score. The F1 score is a measure of predictive performance that balances both precision (correctly predicted positive cases) and recall (correctly identified positive cases), making it a useful metric for evaluating the performance of binary classification tasks. The final test metrics are showcased in Table 1. Additionally, with the assistance of the `Matplotlib Pyplot` library, the metrics were plotted for each epoch, and the plots can be seen in Figure 7.

Final test accuracy	Final test loss	Final test F1-score
0.740	0.520	0.757

Table 1: Final test metrics for the best model

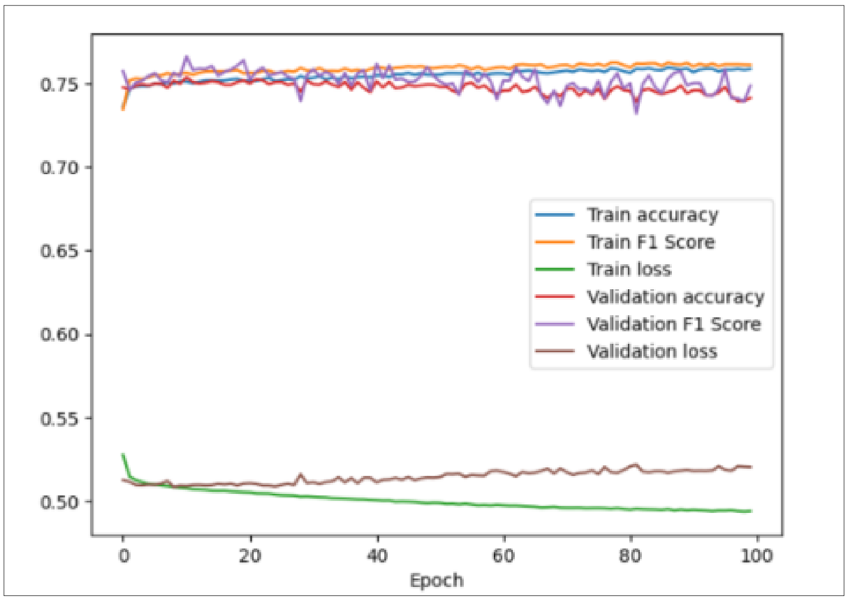


Figure 7: Training and validation: Accuracy, Loss and F1 score plot

Based on the values in Table 1 the model performed relatively well. The 0.740 accuracy means that the model classified 74% of the test right. The loss of 0.52 is mediocre, and it indicates the closeness to true labels. Lastly, a decent F1 score was achieved. It shows that the model is performing quite well in terms of both precision and recall. When observing Figure x3 we can see that the values aren't changing a lot through the 100 epochs. The biggest difference which is expected is the training loss. After around 10 epochs the model starts to minorly overfit and the results start slowly worsening.

3.4 Decision Tree Methods

3.4.1 Decision Tree

Decision tree methods, such as Decision Trees, Random Forests, and Boosting, represent a powerful machine learning approach for both classification and regression tasks. Decision Trees create transparent decision structures, while Random Forests enhance robustness through an ensemble of trees, and Boosting combines weak learners sequentially to boost accuracy. These methods are especially good at working with different types of data and figuring out complicated patterns.

Given this information, we opted to develop a program, focusing on each of the methods, allowing us to draw conclusions based on its performance and outcomes. The program is designed for binary classification in predicting diabetes.

Initially, the dataset was divided into input features ('X') and the target variable ('y'), followed by a further split into training and testing sets (train size of 0.8).

Subsequently, a Decision Tree Classifier was implemented with a specific maximum depth (`max_depth = 6`), and training was conducted on the designated training data. Predictions were then made on the test set, and the F1 score was calculated to assess the model's performance (F1 = 0.7423). We also chose to visualize the generated decision tree, a segment of which can be observed in Figure 8, and to construct the confusion matrix for a comprehensive evaluation.

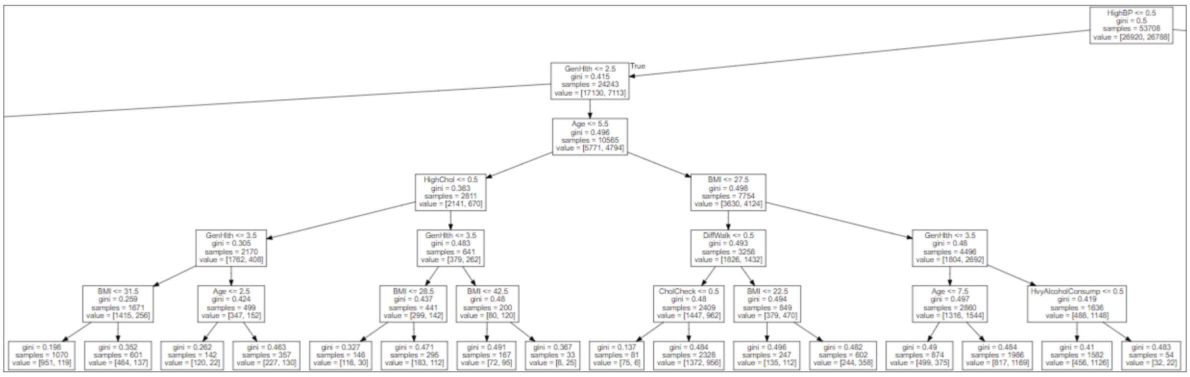
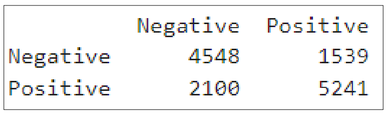

HighBP <= 0.5 gini = 0.5 samples = 53708 value = [26920, 26788]
True
GenHlth <= 2.5 gini = 0.415 samples = 24243 value = [17130, 7113]
Age <= 5.5 gini = 0.496 samples = 10565 value = [5771, 4794]
HighChol <= 0.5 gini = 0.363 samples = 2811 value = [2141, 670]
BMI <= 27.5 gini = 0.498 samples = 7754 value = [3630, 4124]
GenHlth <= 3.5 gini = 0.305 samples = 2170 value = [1762, 408]
GenHlth <= 3.5 gini = 0.483 samples = 641 value = [379, 262]
DiffWalk <= 0.5 gini = 0.493 samples = 3258 value = [1826, 1432]
GenHlth <= 3.5 gini = 0.48 samples = 4496 value = [1804, 2692]
BMI <= 31.5 gini = 0.259 samples = 1671 value = [1415, 256]
Age <= 2.5 gini = 0.424 samples = 499 value = [347, 152]
BMI <= 28.5 gini = 0.437 samples = 441 value = [299, 142]
BMI <= 42.5 gini = 0.48 samples = 200 value = [80, 120]
CholCheck <= 0.5 gini = 0.48 samples = 2409 value = [1447, 962]
BMI <= 22.5 gini = 0.494 samples = 849 value = [379, 470]
Age <= 7.5 gini = 0.497 samples = 2860 value = [1316, 1544]
HvyAlcoholConsump <= 0.5 gini = 0.419 samples = 1636 value = [488, 1148]
gini = 0.198 samples = 1070 value = [951, 119]
gini = 0.352 samples = 601 value = [464, 137]
gini = 0.262 samples = 142 value = [120, 22]
gini = 0.463 samples = 357 value = [227, 130]
gini = 0.327 samples = 146 value = [116, 30]
gini = 0.471 samples = 295 value = [183, 112]
gini = 0.491 samples = 167 value = [72, 95]
gini = 0.367 samples = 33 value = [8, 25]
gini = 0.137 samples = 81 value = [75, 6]
gini = 0.484 samples = 2328 value = [1372, 956]
gini = 0.496 samples = 247 value = [135, 112]
gini = 0.482 samples = 602 value = [244, 358]
gini = 0.49 samples = 874 value = [499, 375]
gini = 0.484 samples = 1986 value = [817, 1169]
gini = 0.41 samples = 1582 value = [456, 1126]
gini = 0.483 samples = 54 value = [32, 22]

Figure 8: Decision Tree

The confusion matrix is presented below:



	Negative	Positive
Negative	4548	1539
Positive	2100	5241

Figure 9: Confusion Matrix

The confusion matrix reveals valuable insights into the model's performance. In this matrix, the diagonal elements represent the correct predictions, while off-diagonal elements indicate misclassifications. The upper-left quadrant corresponds to true negatives, where non-diabetic instances are correctly identified, totaling 4548. on the other hand, the lower-right quadrant signifies true positives, capturing the correct identification of diabetic cases, totaling 5241. The upper-right quadrant denotes false positives, indicating instances where non-diabetic individuals were incorrectly classified as diabetic (1539 cases). Meanwhile, the lower-left quadrant represents false negatives, illustrating instances where diabetic individuals were incorrectly classified as non-diabetic (2100 cases). While classifications are evident, this analysis underscores the model's ability to correctly identify a substantial number of non-diabetic and diabetic cases.

3.4.2 Random Forest

Moving forward, a Random Forest Classifier was implemented with defined parameters, including the number of estimators (estimators=100) and maximum depth (`max_depth = 6`). The model underwent training, predictions were generated on the test set, and the resulting F1 score was computed (F1

=0.7534). Additionally, we produced a horizontal bar chart, to show the feature importance of the random forest model. This visualization helps in identifying the most influential features contributing to the model's predictive performance. The key features were: HighBP, GenHlth, Age, BMI and HighChol.

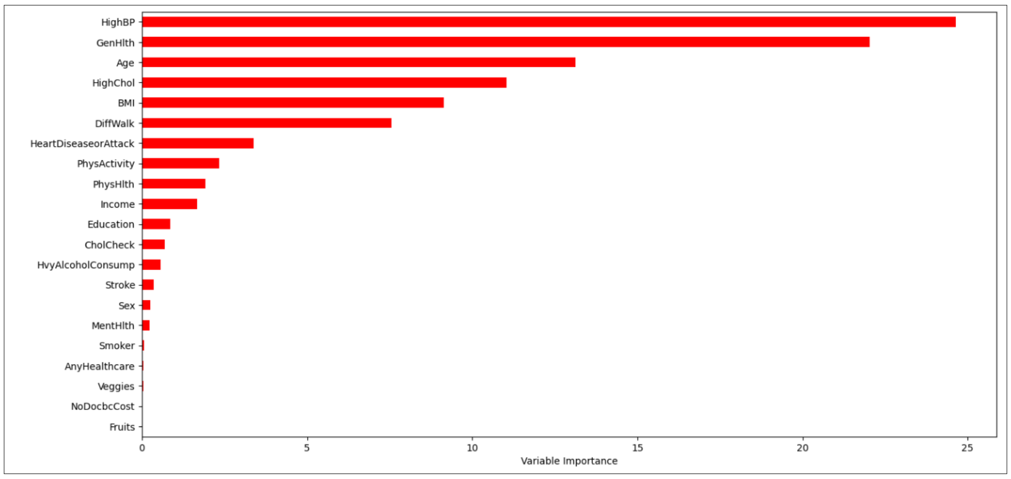


Figure 10: Feature importance for Random Forest

3.4.3 Gradient Boosting

At last, we decided to implement a Gradient Boosting Classifier, incorporating specified parameters such as the number of estimators(estimators=100), learning rate (learning_rate = 0.01), and maximum depth (`max_depth = 6`). The model underwent training, predictions were made, and the corresponding F1 score was determined (F1 = 0.7561). Likewise, as we did for random forests, we created a horizontal bar chart to illustrate the importance of features. This allowed us, once again, to identify the key features influencing the model's predictive performance. The key features were: HighBP, GenHlth, Age, BMI and HighChol, as in random forests.

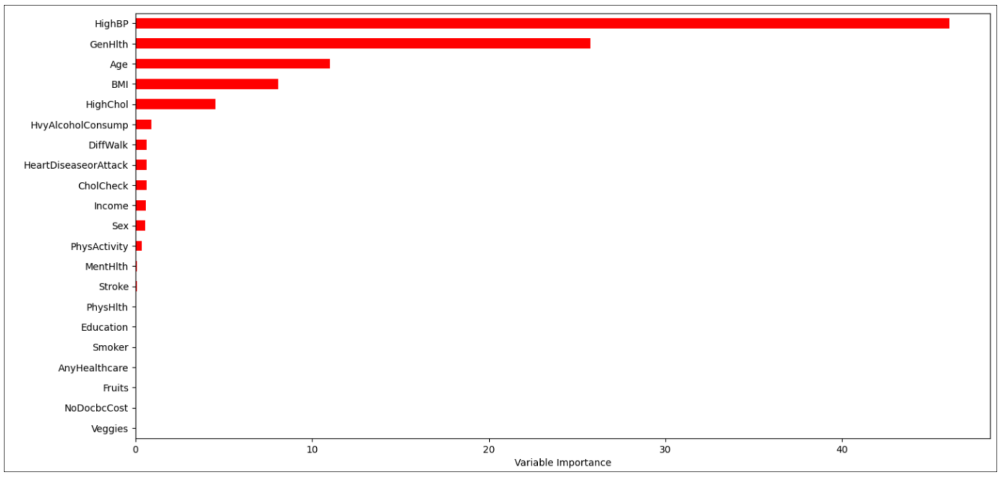


Figure 11: Feature importance for Gradient Boosting

3.4.4 Analysis

We choose F1 score to compare the 3 methods because it offers a more balanced evaluation, particularly in the context of predicting diabetes outcomes. A larger F1 score is desirable as it indicates a better balance between a model's precision and recall.

In this diabetes prediction program, the Decision Tree Classifier yielded an F1 score of **0.7423**, indicating moderate accuracy. The Random Forest Classifier achieved a slightly higher F1 score of

0.7534, suggesting an improvement in predictive performance with its approach. Similarly, the Gradient Boosting Classifier demonstrated a comparable F1 score of **0.7561**, affirming the effectiveness of the method. Succinctly, the F1 scores imply that all three methods perform reasonably well in predicting diabetes outcomes, with Gradient Boosting Classifier having a slightly better F1 score. However, it's essential to consider other factors such as interpretability and computational cost. Below there is a table with the F1 score values for each method.

Decision tree	Random forest	Boosting
0.7423	0.7534	0.7561

Table 2: F1 Score for Different Methods

Following this, we aimed to compare the three classifiers – Decision Tree, Random Forest, and Gradient Boosting – by generating ROC curves. We used KFold cross-validation (number of splits = 5) to make sure our evaluation was more reliable. Initially, we set specific hyperparameters and trained our models on various folds of the dataset. Then, ROC curves were generated for each fold, and their average was calculated. Ultimately, the resulting mean ROC curves, along with their respective AUC values, were plotted on a single graph for visual comparison, making it easy to directly compare their performance. The AUC values were also determined, and the results were **0.8** for the Decision Tree, **0.81** for the Random Forest, and **0.82** for Gradient Boosting. The AUC values show how well the methods can differentiate between positive and negative cases. To summarize, when we look at the AUC values, higher values indicate better performance. Based on this, we can conclude that Gradient Boosting is the most effective method, proving to be the best at distinguishing between different outcomes compared to the other two methods.

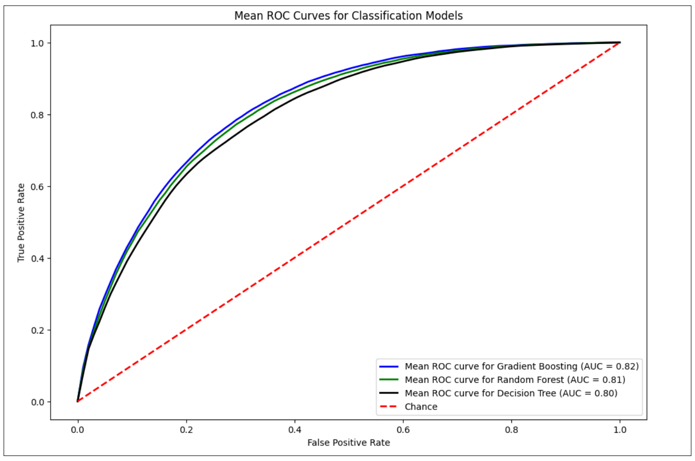


Figure 12: ROC Curve for Decision Tree Methods

In the final section, an independent grid search for hyperparameter optimization was carried out for the Decision Tree (best max depth), Random Forest (best number of estimators and max depth), and Gradient Boosting (best number of estimators, max depth, and learning rate). We compiled lists for individual parameters, assigning various values to each, in order to identify the optimal choice. The validation was done using cross-validation (number of folds $= 5$). The metric employed to examine the best parameters for each of the methods was the F_1 score. It was chosen in this case because it balances both the accuracy of the positive predictions made by the model. The results, including F_1 values for various hyperparameter configurations, were visualized to offer a comprehensive analysis of model performance under different settings. The list of values used in the GridSearch are presented below:

- **Decision Tree:**
  - Max Depth - $\{2, 6, 10\}$
- **Random Forest:**
  - Number of Estimators - $\{50, 100, 150\}$
  - Max Depth - $\{2, 6, 10\}$
- **Gradient Boosting:**
  - Number of Estimators - $\{50, 100, 150\}$
  - Learning Rate - $\{0.01, 0.1, 0.5\}$
  - Max Depth - $\{2, 6, 10\}$

The corresponding optimal parameter values were as follows:

- **Decision Tree:** Best Max Depth - 6
- **Random Forest:** Best Number of Estimators - 150 and Best Max Depth - 10
- **Gradient Boosting:** Best Number of Estimators - 150, Best Learning Rate - 0.5, and Best Max Depth - 2

Below, we present the graphics created from GridSearch:

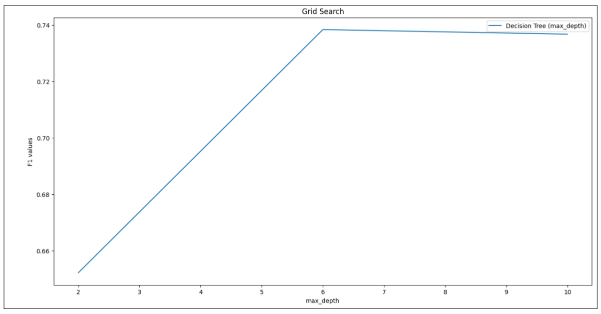


Figure 13: GridSearch for Decision Tree

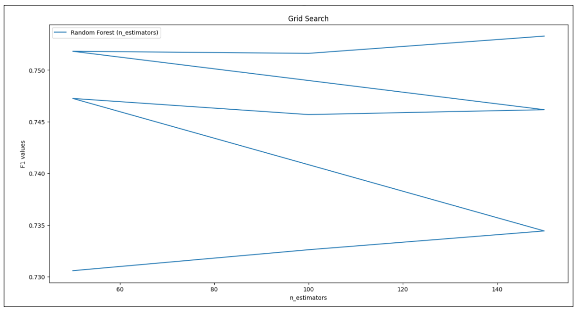


Figure 14: GridSearch for Random Forest

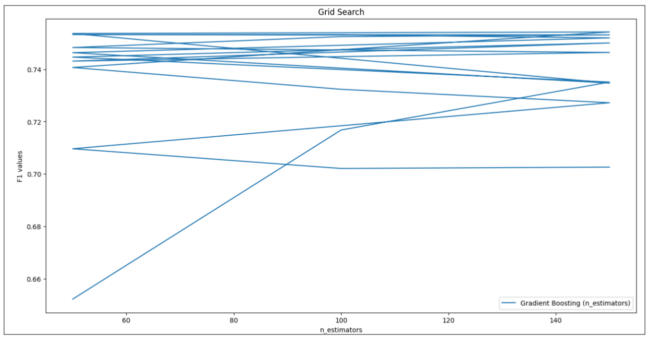


Figure 15: GridSearch Graphic for Boosting

For the Decision Tree, the F1 scores exhibit a discernible peak at a `max_depth` of 6, suggesting an optimal balance between model complexity and performance.

In the case of the Random Forest, the plot showcases the ensemble's sensitivity to the number of estimators, with the highest F1 scores achieved when `n_estimators` reach 150 and `max_depth` is set at 10.

The Gradient Boosting graphic clarifies the influence of parameters, indicating that the best performance is achieved with `n_estimators=150`, `learning_rate=0.5`, and `max_depth=2`.

We decided to rerun the programs of the methods once again to enhance the predictive performance of our models. The decision tree remained unchanged throughout this re-evaluation process. The F_1 score for the Random Forest increased from **0.7534** to **0.7614** after the grid search, and the Boosting method also showed improvement, with the F_1 score rising from **0.7561** to **0.7611**. This indicates an improvement in predictive accuracy which is the main reason why we do grid search in the first place.

We also graphed the mean curve using the same approach as previously. The AUC for Decision Tree and Boosting remained constant, whereas, in the Random Forest, there was an increase. We presume that in Boosting, there was also an increment, albeit in a decimal place that wasn't displayed. In summary, our analysis suggests that the models, particularly Random Forest and potentially Boosting, demonstrate improved predictive performance

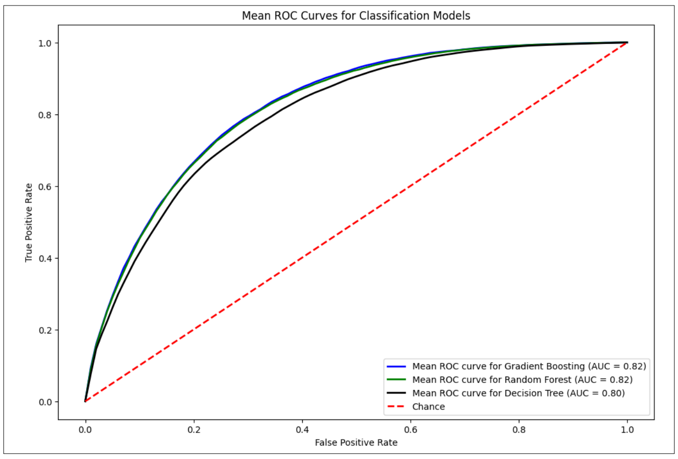


Figure 16: ROC Curve for Decision Tree methods after GridSearch

4 Conclusions

In this project, we explored several machine learning methods, including Neural Networks (NN), Support Vector Machines (SVM), and Decision Tree-based approaches, to predict the occurrence of diabetes based on relevant parameters.

The myriad of methods used allowed us to compare not only the results of each but also their functioning. Each model's approach to handling and generalizing the data gave us an insight into some of the advantages and disadvantages of these machine learning techniques which became apparent through our evaluation.

Firstly, the Basic Methods used offered a simple, yet rudimentary approach which is not as complex as the more advanced methods. Despite that, these models surprisingly displayed satisfactory results, with some methods almost achieving similar scores to the latter.

On the other hand, SVM proved to be a powerful algorithm for classification tasks with both linear and non-linear kernels. The linear SVM, despite its simplicity, demonstrated robust performance, emphasizing a good separation between classes. The non-linear kernel, while more complex, also exhibited its capacity to establish decision boundaries, despite requiring an optimization to compete with the linear model. However, since this method is so dependent on the algebraic relations between the points, the resolution of the problem is quite time-consuming, even with simpler Cross Validation (2-fold).

Regarding Neural Networks, they are known for their ability to model complex relationships within data and provided a high level of flexibility and adaptability to the intricate patterns present in the diabetes dataset. The multi-layered architecture allowed the network to learn hierarchical features and interactions among variables, although the processing is considerably lengthy, similar to SVM.

Finally, the Decision Tree-based methods, including Random Forest and Gradient Boosting, provided a different perspective. These models leveraged an ensemble of trees, offering insights into feature importance and interactions. Their ability to handle non-linear relationships and divide the data into branches facilitates the analysis and allows for faster computation of the models while maintaining competence and pertinence.

In conclusion, this project facilitated the application of methods elucidated in both theoretical and practical coursework to the specific context of diabetes. This hands-on project not only translated theoretical concepts into practical insights but also enhanced our proficiency in applying machine learning methodologies to real-life situations.