CS 483 - Big Data Mining
Final Project Report - Fall 2024

Instructor: Prof. Lu Cheng

1 Group Information

Group Name: Spaghetti Coders

Group Members: Eleonora Cabai (665321925), Filippo Corna (658181569), Davide Ettori (655693998), Patrik Poggi (651290827)

Group Leader: Filippo Corna

2 Introduction

The intersection of health and Artificial Intelligence has unveiled significant opportunities, in particular in the domain of predictive modeling and decision-making. With this purpose, traditional machine learning algorithms, such as logistic regression and decision trees, have been widely used for medical datasets to predict health outcomes.

In this project, we aim to investigate the predictive power of Machine Learning algorithms on the CDC Diabetes Health Indicators dataset, exploring whether these models can help identify key risk factors and support healthcare decisions.

The aim of our project is to train a neural network to predict the probability of diabetes presence. Then, we analyze the neural network with the SHAP method and extract the interpretability coefficients for each feature. At this point, we pass the patient's data, the neural network's prediction, and the interpretability information to an LLM model. Using this, we generate a medical report for the patient, highlighting the positive and negative aspects of their health and concluding with some recommendations to mitigate risks.

3 Problem and Goal

Diabetes is an increasing public health issue, and early identification of individuals at risk is essential for effective prevention and management. Our primary goal is to classify individuals as healthy, pre-diabetic, or diabetic based on a variety of health, lifestyle, and demographic factors. We also aim to:

- **Identify significant risk factors** contributing to the onset of diabetes, such as physical activity levels and other health indicators.
- **Ensure fairness and unbiased predictions**, particularly across different demographic groups, by assessing the model's performance on sensitive variables like income and education.
- **Conduct cluster analysis** to discover subgroups with distinct health profiles or behaviors, which could reveal underlying patterns in diabetes susceptibility.
- **Develop a predictive health score** that ranks individuals based on their risk of developing diabetes, supporting healthcare professionals in early diagnosis and intervention.

By pursuing these goals, we seek to enhance the understanding of diabetes risk and improve strategies for its prevention and treatment.

4 Dataset Information

In this project, we aim to investigate the predictive power of Machine Learning algorithms on the CDC's Diabetes Health Indicators Dataset, exploring whether these models can help identify key risk factors and support healthcare decisions. Table 1 presents the dataset's features along with their corresponding values or ranges.

Feature	**Values**	**Range**
Diabetes	0 / 1 / 2	-
High Blood Pressure	0 / 1	-
High Cholesterol	0 / 1	-
Cholesterol Check (last 5 years)	0 / 1	-
BMI	-	12 - 98
Smoker	0 / 1	-
Stroke	0 / 1	-
Heart Disease or Attack	0 / 1	-
Physical Activity	0 / 1	-
Fruits	0 / 1	-
Veggies	0 / 1	-
Heavy Alcohol Consumption	0 / 1	-
Healthcare Coverage	0 / 1	-
No Doctor because of Cost	0 / 1	-
General Health	-	1 - 5
Days of not good Mental Health	-	0 - 30
Days of not good Physical Health	-	0 - 30
Difficulties in Walking	0 / 1	-
Sex (female / male)	0 / 1	-
Age category (from 18-24 to 80+)	-	1 - 13
Education level	-	1 - 6
Income level	-	1 - 8

Table 1: Feature Values

5 Dimensionality Reduction

Dimensionality reduction is the process of reducing the number of features in a dataset while preserving important information. This is often achieved through techniques like Principal Component Analysis (PCA) and t-Distributed Stochastic Neighbor Embedding (t-SNE), which transform the original high dimensional data into a lower dimensional space. More recently Neural Networks, specifically Autoencoders have also proven to be very a effective method for non-linear Dimensionality Reduction.

In diabetes data analysis, dimensionality reduction can improve computational efficiency, reduce noise, and enhance the interpretability of models by focusing on the most relevant features. This is particularly useful in data mining, where managing large, complex datasets is crucial for effective predictive modeling and pattern discovery.

Each analysis aims at going from the 21 initial features to 3 reduced features, which is the the highest number of dimensions that can be visualized.

5.1 Principal Component Analysis

Principal Component Analysis (PCA) is a linear dimensionality reduction technique that finds directions (principal components) capturing the greatest variance in a dataset. Given a centered data matrix $X \in \mathbb{R}^{n \times d}$ (where each column has zero mean), PCA can be derived using Singular Value Decomposition (SVD). Performing SVD on X gives $X = U\Sigma V^T$, where U and V are matrices of left and right singular vectors, and Σ is a diagonal matrix of singular values. The principal components correspond to the columns of V, ordered by the magnitude of singular values in Σ. To reduce dimensionality, we project X onto the first k components using XV_k, where V_k includes the first k columns of V. This projection captures the maximum variance, minimizing reconstruction error, but it has only access to linear transformations of the features. An important advantage of this method is that we do not have to choose any hyperparameters, beside the number of components. Finally, as we will see, it's naturally interpretable, since when can clearly see the contribution of each feature to each Principal Component.

The PCA analysis makes the dataset easily visualizable in three dimensions. We can observe that subjects without diabetes are clustered to the left (purple), while those with diabetes are more towards the right (yellow), as shown in Figure 1. The three most influential Principal Components capture, respectively, 16.3%, 8.5%, and 6.3% of the total data variance, revealing over 31% of the variance with only three features (instead of 21). This reduction can be utilized later to train more precise models that are less affected by noise present in the data.

Finally, we note that the reconstruction error, measured by MAE (Mean Absolute Error), is 0.59. This represents the average absolute error per feature when attempting to return to the 21-feature representation from the three selected components.

Regarding interpretability, we can analyze the contributions of each feature to each component, as shown in the plots below. For instance, we observe that the first component represents medical features: General Health, Physical Health, and Difficulty in Walking. The second component reflects the patient's social situation: Age, ability to see a doctor, and healthcare coverage. The third component focuses on lifestyle habits, such as frequency of eating fruits, vegetables, and smoking. These insights align with our expectations and already offer an interesting view of the situation, as well as providing a method for dimensionality reduction for future analyses.

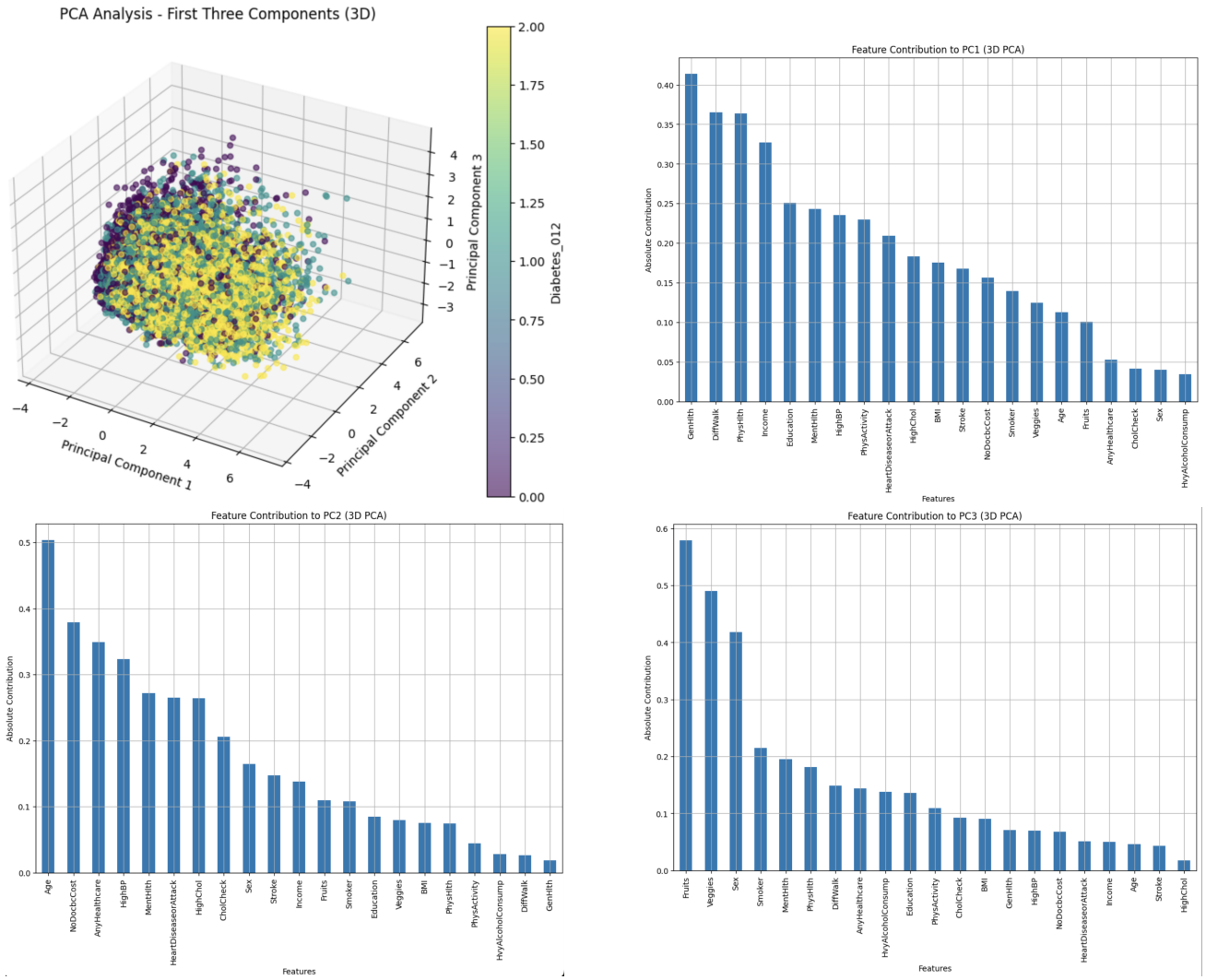


Figure 1: PCA Data Visualization and individual Feature Contributions

5.2 t-Distributed Stochastic Neighbor Embedding (t-SNE)

t-SNE is a non-linear dimensionality reduction technique designed to visualize high-dimensional data by preserving local similarities. The main idea is to model the probability of similarity between points in high-dimensional space and map it to a lower-dimensional space while maintaining these relationships. For points x_i and x_j in the original space, t-SNE calculates the probability p_{ij} of x_j being a neighbor of x_i using a Gaussian distribution centered on x_i. In the low-dimensional space, the probability q_{ij} is calculated similarly, but with a Student's t-distribution to handle crowding. The objective is to minimize the Kullback-Leibler (KL) divergence between p_{ij} and q_{ij}, defined as:

$$\mathrm{KL}(P||Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

This ensures that nearby points in the high-dimensional space remain close in the lower-dimensional embedding, creating a visual representation that captures local structure. The **perplexity** is the most important hyperparameter and it balances the focus between local and global aspects of the data. Lower perplexity values emphasize local relationships, while higher values capture broader, more global structures.

The t-SNE analysis shows the data clustered in a less regular manner, which results from the non-linearity of this probability-based method. One of the main challenges lies in finding the right value for the perplexity

parameter, which depends on the specific context and is not easy to determine a priori. In this case, we conducted an exhaustive search across all values from 1 to 100 in intervals of 10. The best value turned out to be 70, excluding the value 1, which seems to represent an overfitting case, overly focused on local characteristics of the data. Unfortunately this method does not permit us to analyze the captured variance or the reconstruction error, since once compressed the data cannot be brought back to the original dimensions.

Finally, we observe that no particular pattern seems to emerge in the data, suggesting that t-SNE might not be well-suited for this specific problem or data distribution, even if it's in theory more general than PCA. This is also evidenced by the final Kullback-Leibler Divergence value of 1.28, whereas a well-executed t-SNE analysis usually achieves a value below 1.

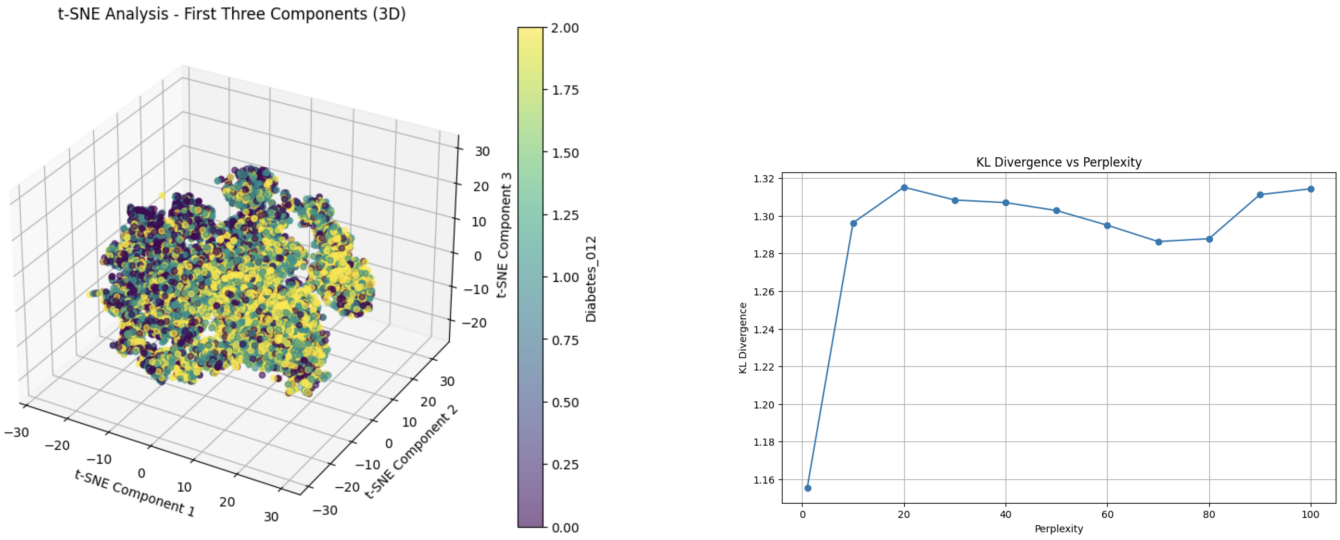


Figure 2: t-SNE Visualization (Left) and KL Divergence Variation with Perplexity (Right)

5.3 Autoencoders

Autoencoders are neural networks used for dimensionality reduction by learning efficient representations of data. They consist of an encoder, which maps input x to a lower-dimensional latent space z, and a decoder, which reconstructs x from z, aiming to minimize reconstruction error. Given an input x, the encoder function $f(x) = z$ and decoder function $g(z) = \hat{x}$ are optimized to minimize $L(x, \hat{x})$, typically using mean squared error. By constraining the latent dimension z to be smaller than the input, autoencoders capture essential data features. This enables denoising, compression, and effective dimensionality reduction, especially in cases where linear methods like PCA are insufficient. It's a very powerful method and being a Neural Network it inherits all their typical strength (very general) and weaknesses (high variance).

The autoencoder consists of a neural network with the following layers: 21, 16, 8, 3, 8, 16, 21. The activation functions are Leaky ReLU, the loss function is mean squared error, and the optimizer is Adam. We observe that the training converges after approximately 30 epochs, and the process is then stopped at the 40th epoch.

We note that the data distribution is entirely different from the previous methods, with multiple clusters scattered in the 3D space. This results from the neural network's ability to model highly complex shapes, thanks to its structure combining linear layers and non-linear functions. Here, we observe a much clearer separation compared to previous methods and a significantly lower reconstruction error, with a Mean Absolute Error (MAE) of 0.47. This indicates that the reduction has been quite successful and that this representation effectively captures the true characteristics of the data. The downside is that using a neural network has resulted in a loss of interpretability, as these architectures are nearly black-box models.

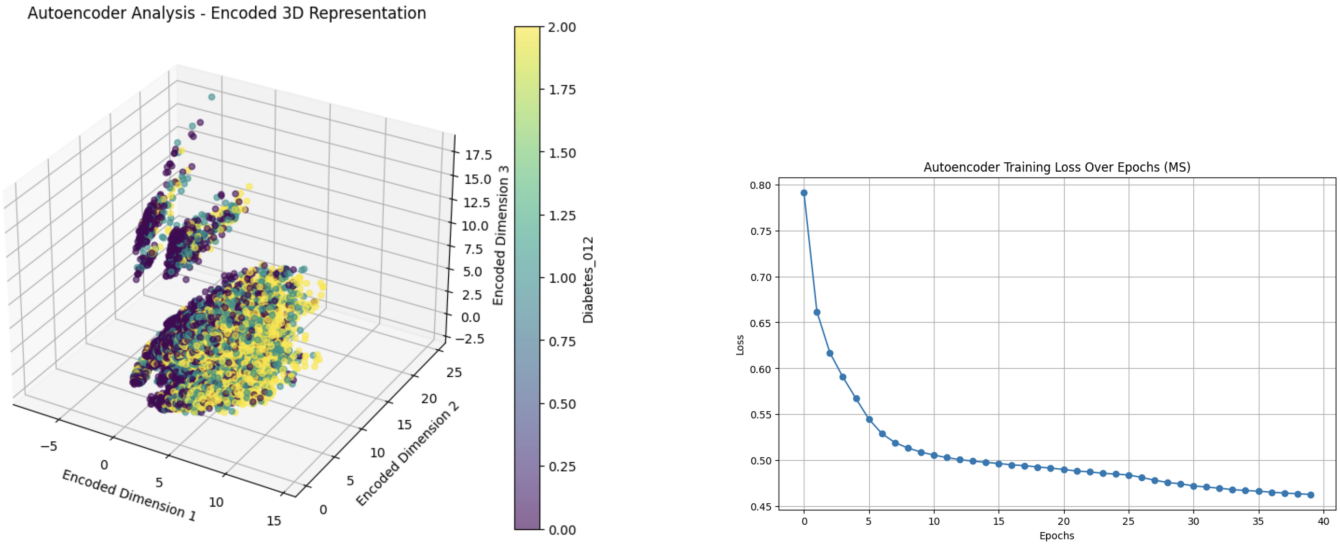


Figure 3: Left: Compressed Data with Autoencoder; Right: Loss per Epoch

6 K-means Clustering Analysis

Clustering is an unsupervised learning technique used to group data points into clusters that share similar characteristics. K-means clustering partitions data by minimizing the within-cluster variance, a process governed by the number of clusters k. Each data point is assigned to the nearest cluster center, iteratively refining these centers until convergence. The goal is to assign similar health profiles to the same cluster.

6.1 K-Means Clustering for Health Profile Identification

In this study, we utilized K-means clustering to identify distinct health profiles within the dataset, focusing on uncovering underlying patterns that may contribute to diabetes susceptibility. Specifically, we applied two versions of K-means clustering: traditional K-means and Stratified K-means.

6.1.1 Traditional K-Means Clustering

Traditional K-means clustering was employed to divide the dataset into three distinct groups based solely on feature similarity, without any prior knowledge of diabetes status. This clustering approach aimed to reveal natural groupings within the dataset, potentially illuminating health profiles that might be more susceptible to, or protected from, diabetes.

The resulting clusters show clear differences in average feature values across several health indicators, as outlined in Table 2. For instance, individuals in Cluster 1 exhibited higher BMI and a greater likelihood of having high blood pressure, high cholesterol, and a history of stroke, which are well-established diabetes risk factors. Meanwhile, Clusters 0 and 2 displayed lower average BMI, fewer incidences of high blood pressure, and generally more active lifestyles. These groupings provide insight into distinct health profiles within the population and underscore certain features associated with potential risk factors for diabetes.

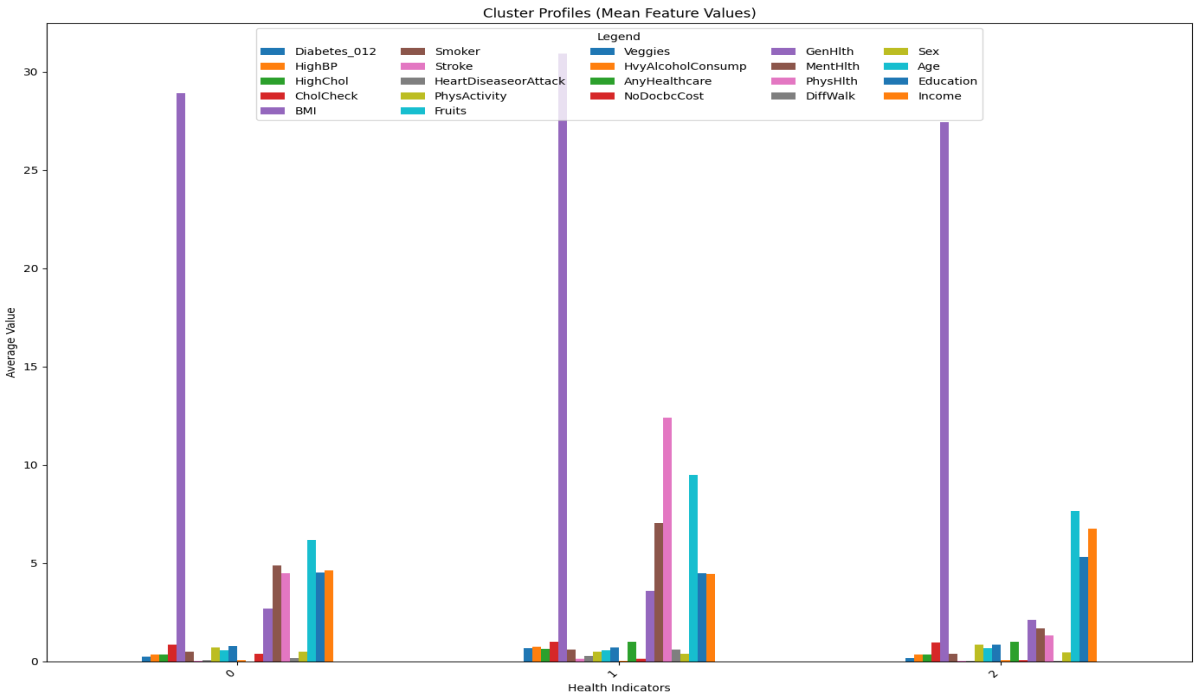


Figure 4: Cluster Profiles (Mean Feature Values)

Feature	Cluster 0	Cluster 1	Cluster 2
Diabetes	0.2456	0.6531	0.1727
High BP	0.3428	0.7332	0.3259
High Chol	0.3298	0.6387	0.3537
Chol Check	0.8639	0.9847	0.9617
BMI	28.8975	30.9095	27.4400
Smoker	0.4938	0.6070	0.3809
Stroke	0.0302	0.1290	0.0096
Heart Disease	0.0673	0.2730	0.0319
Phys Activity	0.6912	0.4950	0.8549
Fruits	0.5677	0.5444	0.6711
Veggies	0.7612	0.6980	0.8556
Hvy Alcohol	0.0670	0.0354	0.0629
Healthcare	0.0000	0.9992	1.0000
No Doc bc Cost	0.3698	0.1441	0.0428
Gen Hlth	2.6959	3.5942	2.1102
Ment Hlth	4.8715	7.0258	1.6896
Phys Hlth	4.4705	12.3878	1.3044
Diff Walk	0.1533	0.5796	0.0217
Sex	0.4835	0.3930	0.4543
Age	6.1581	9.4977	7.6370
Education	4.5204	4.4747	5.2938
Income	4.6165	4.4295	6.7366

Table 2: Cluster Profiles (Mean Feature Values)

6.1.2 Stratified K-Means Clustering

The second approach, Stratified K-means, involved clustering based explicitly on diabetes status: Cluster 0 for no diabetes, Cluster 1 for pre-diabetes, and Cluster 2 for diabetes. This stratified approach offers a focused comparison of feature averages across diabetes categories, allowing for an in-depth analysis of how each health indicator varies between non-diabetic, pre-diabetic, and diabetic populations.

Table 3 provides a summary of the average feature values across the diabetes status clusters. Results indicate that diabetes status correlates strongly with several key features: BMI, physical activity, general health , and age. Cluster 2 (diabetes group) showed significantly higher average BMI, lower physical activity, and worse self-reported general health compared to the other clusters. Additionally, it had a higher proportion of individuals with high cholesterol, high blood pressure, and lower income and education levels, further reinforcing established risk factors in diabetes research.

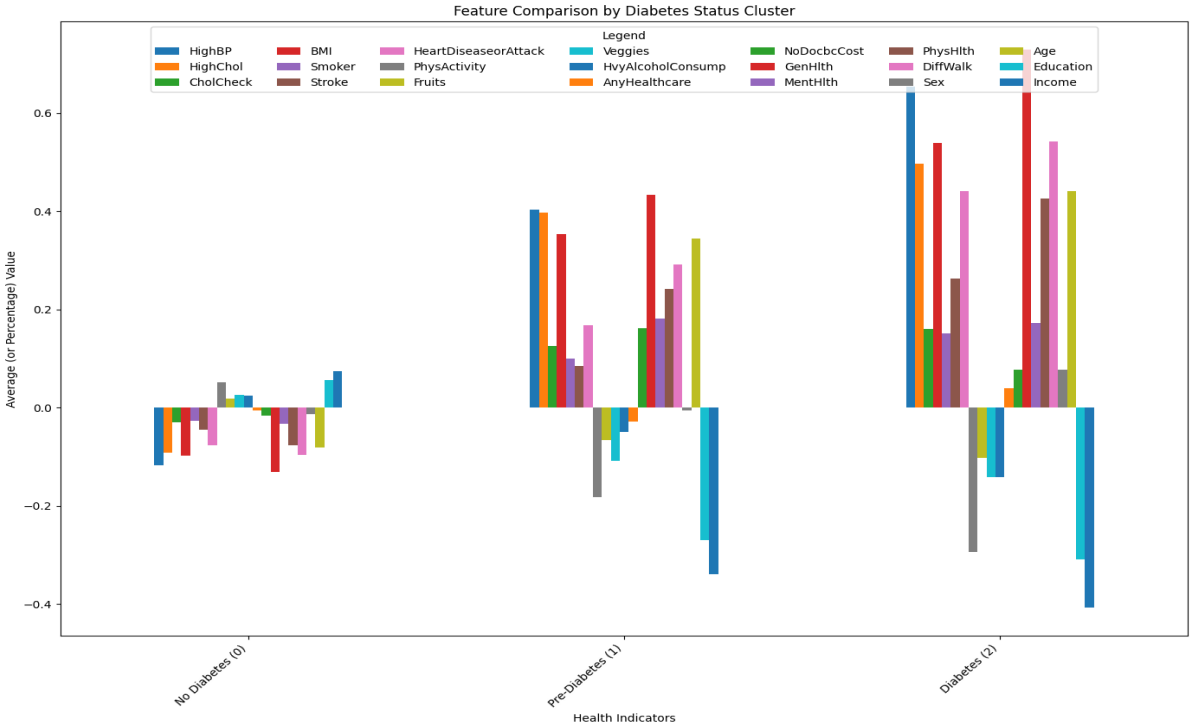


Figure 5: Features Comparison by Diabetes Status

Feature	No Diabetes	Pre-Diabetes	Diabetes
High BP	-0.1169	0.4041	0.6540
High Chol	-0.0910	0.3980	0.4978
Chol Check	-0.0294	0.1263	0.1610
BMI	-0.0968	0.3544	0.5389
Smoker	-0.0272	0.0998	0.1511
Stroke	-0.0453	0.0844	0.2630
Heart Disease	-0.0765	0.1684	0.4406
Phys Activity	0.0525	-0.1819	-0.2936
Fruits	0.0182	-0.0665	-0.1014
Veggies	0.0256	-0.1086	-0.1406
Hvy Alcohol	0.0245	-0.0490	-0.1418
Healthcare	-0.0061	-0.0273	0.0404
No Doc bc Cost	-0.0164	0.1627	0.0781
Gen Hlth	-0.1301	0.4345	0.7296
Ment Hlth	-0.0324	0.1815	0.1723
Phys Hlth	-0.0757	0.2416	0.4258
Diff Walk	-0.0961	0.2921	0.5427
Sex	-0.0128	-0.0053	0.0781
Age	-0.0804	0.3442	0.4410
Education	0.0570	-0.2698	-0.3093
Income	0.0747	-0.3393	-0.4074

Table 3: Feature Comparison by Diabetes Status

6.2 Cluster Profiles and Feature Analysis

Across both clustering methods, we observed several important patterns. In the traditional K-means clusters, diabetes status was not a distinguishing factor, allowing health characteristics alone to dictate group membership. In contrast, the stratified clusters clearly highlighted the differences between individuals at various stages of diabetes. These insights not only underscore the importance of health profiles in diabetes prevention but also support the development of targeted interventions based on health indicators that show strong associations with diabetes progression.

6.3 Feature Distributions by Cluster

To further interpret these clusters, we analyzed the distribution of each feature within them. The distributions, visualized as side-by-side histograms, provide insights into the unique characteristics of each group. For instance:

- Blood Pressure and Cholesterol: Clusters associated with pre-diabetes and diabetes show higher frequencies of elevated blood pressure and cholesterol levels.
- Lifestyle Factors: Features such as physical activity, fruit and vegetable consumption, and smoking habits differ noticeably between clusters, indicating a correlation between these lifestyle choices and diabetes risk.
- Demographics: Age and education levels also vary across clusters, with Cluster 2 (diabetes) showing a higher proportion of older individuals.

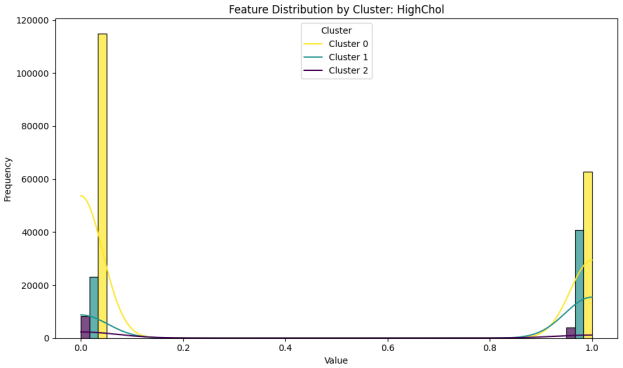


Figure 6: High Cholesterol Activity Histogram

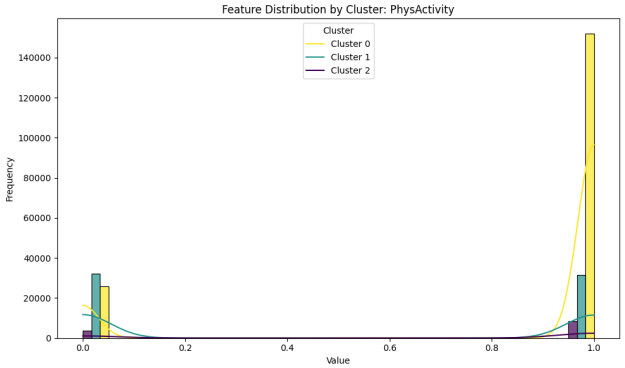


Figure 7: Physical Activity Histogram

6.4 Conclusion

In summary, the K-means clustering analysis provides valuable insights into the dataset, highlighting the diverse health profiles associated with different diabetes susceptibility levels. This information could be useful for identifying high-risk groups and informing targeted prevention efforts based on lifestyle and health indicators.

7 Tree Based Classification

7.1 Introduction

In machine learning, one of the most common approaches towards classification based on tabular data is what in this work will be referred to as 'Tree Based Classification'. this machine learning technique exploits branches in decision trees to classify inputs based on combination of features values. In general, it may be the case that not all features are needed to classify a data point and decision trees are built choosing the combination of splits that maximizes the information gain from the root to the leaves.

In the following subsections we will carry out a preliminary investigation of this class of algorithms when applied to our classification problem. In particular we will discuss different methods to apply this theoretical model and explore a few different parametrization for each of them.

7.2 Decision Trees

Decision Trees are the core of Tree Based Classification as they represent the main theoretical model behind this class of machine learning methods.

As anticipated in the introduction they are built choosing the optimal feature split to maximize information gain. Nonetheless, different parametrizations can yield different results as, for instance, they might imply a different exploration in the space of possible features splits combination. Furthermore, manipulating the way the trees are built, like constraining maximum depth, can significantly impact its performance.

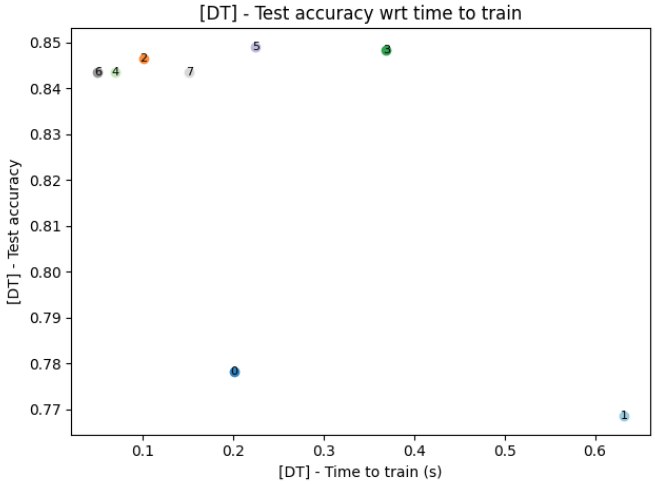


Figure 8: Test Accuracy vs Time to Train

In figure 8 we can observe the results obtained when changing parameters in the Decision Tree classifier under an **accuracy** point of view. This setting is meant to highlight a trade-off: that of accuracy (efficacy) versus time required to train the model (efficiency).

The best accuracy is achieved by setting number 5, with parameters shown in table 4

Setting Code	Max Depth	Max Features	Random State
5	5	21	42

Table 4: DT: setting achieving optimal accuracy

Where, in particular, the number of features is not constrained and we are allowing the model to use all

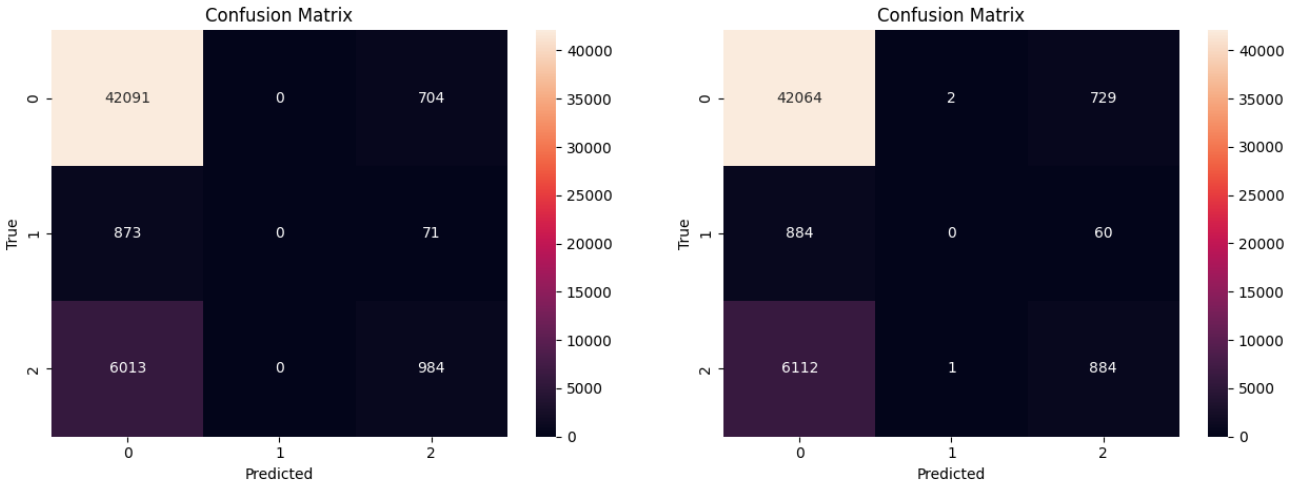
21 of them. the random state has been set to 42 as in the K-Means discussion for reporting consistency purposes.

Even though the model that takes the longest to be trained doesn't require much time, we could still be interested in the efficacy-efficiency trade-off, in which case model number two would be a nice alternative. It would require the parametrization shown in table 5. Mind that 'max features = sqrt' means that when considering the possible features split only a subset of features of dimension sqrt(21) is considered.

The most accurate setting, setting number 5, achieves 84.9% accuracy and presents the confusion matrix depicted in figure 9

Setting Code	Max Depth	Max Features	Random State
2	9	`sqrt`	42

Table 5: setting 2



(a) Setting 5 Confusion Matrix **(b)** Setting 2 Confusion Matrix

Figure 9: Comparison of Confusion Matrices for Different Settings

7.3 Bagging TBC: Random Forest

Bagging is an ensemble method which as such exploits the combined training of multiple, simple, models to achieve good performances without increasing model complexity. Its main merit lies in the fact that it typically lowers variance as it builds a set (hence the name 'bagging') of models, each trained on a subset of the instances or, in this case, on a subset of features (i.e., **feature bagging**. This will lead to less sensitivity to outliers and to a 'democratic' classification, since a quorum of results is considered.

In Tree based classification, the most famous approach is called Random Forest as it ensembles different Decision Trees. In this section we'll briefly discuss the results obtained with different parametrization of a Random Forest Classifier.

Table 6 presents the settings used for tuning the best performing Random Forest classifier, which is the one with code 23 as shown in figure 10a.

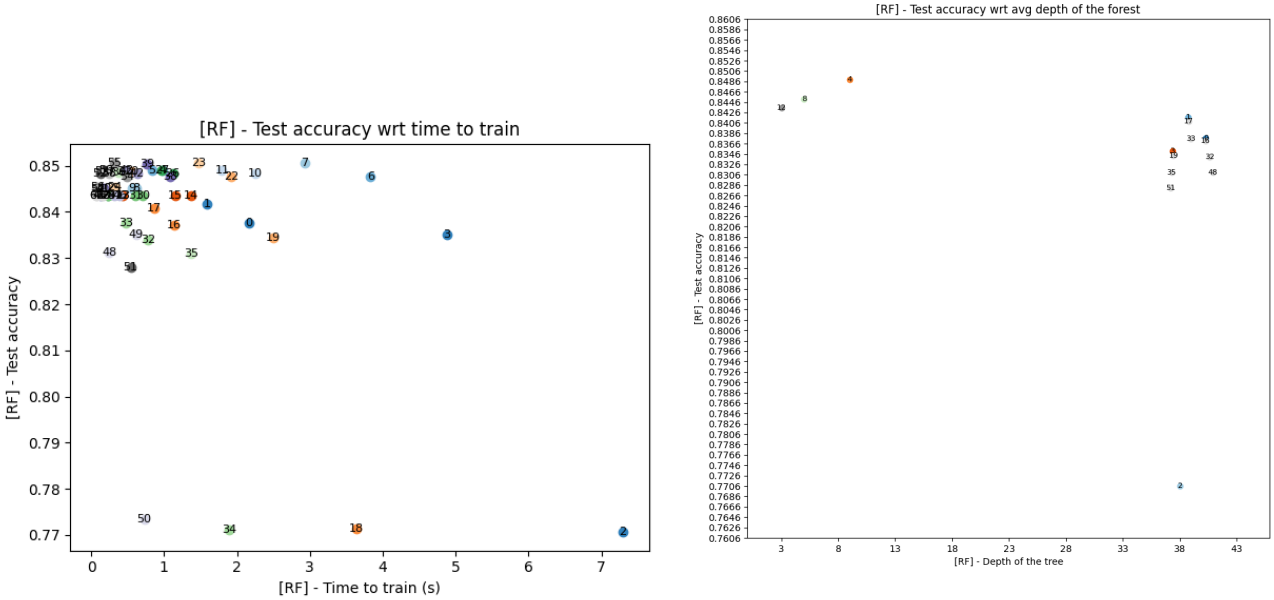
Setting Code	N Estimators	Max Depth	Max Features	Random State	Bootstrap
23	50	9	21	42	True

Table 6: Model Settings for Setting 23

As figure 10a shows there is not much difference in accuracy between setting number 23 and 55, while a model trained with setting 55 takes almost half the time.

Let's further investigate these two parametrizations based on another trade-off: accuracy versus depth of the tree. The depth of the tree can be a nice proxy to evaluate the delay in classifying a single data instance.

Since different parametrizations lead to models performing i a very similar way, in figure 10b we hve displayed only one representative for each number of layers (i.e., depth). However, since figure 10b only presents value 4 on the y coordinate corresponding to both setting 23's and 55's accuracies we conclude that they have the same depth (shared also with setting 4). Such a result means that in this particular case the two settings do not differ as far as the depth of the tree is considered and also their accuracy is practically the same, leading to the choice of setting 55 as it implies faster training.



(a) Random Forest - Test accuracy vs time to train **(b)** RF: accuracy vs depth

Figure 10: Random Forest Model Performance Analysis

7.4 Boosting TBC: AdaBoost

Boosting is another important ensemble methods. In this case we train different models by focusing, each time, more on the points is classified by the previous trained model. In the end each model will be taken into account as a vote in the classification, weighted by its accuracy.

The briefly explained algorithm is **AdaBoost**, which in this work has been used as an ensemble alternative to Random Forest to investigate. Obviously, the core model used was the decision tree, although AdaBoost is not strictly requiring that the classifier is a tree based one.

Figure 11 shows, as usual, the test accuracy performance of every model with respect to the time it required to be trained.

Each of the presented models is, naturally, the result of an AdaBoost ensemble of several models and they differ from each other based on different parametrization, concerning attributes such as the number of ensembled models and the learning rate.

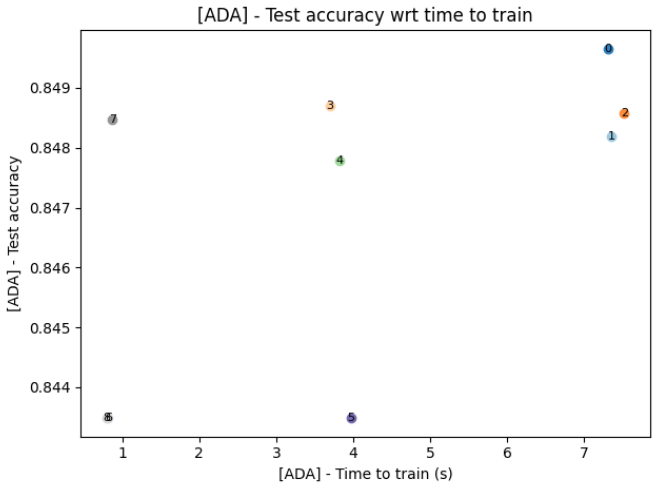


Figure 11: AdaBoost - Accuracy vs time to train

Here it is easy to see that the most accurate model is obtained with setting 0, reported in table 7

Setting Code	N Estimators	Learning Rate	Random State
0	100	1	42

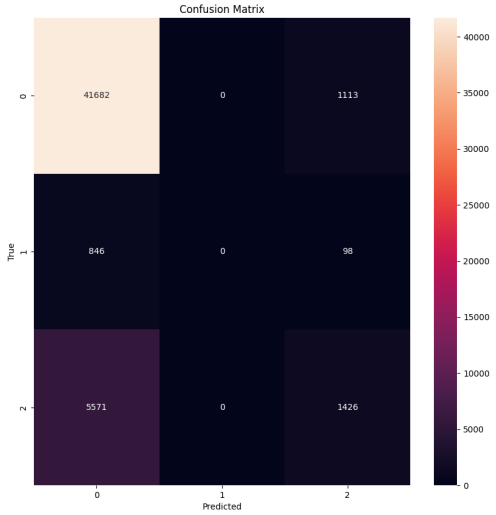
Table 7: Model Settings for Setting Code 0

However the accuracy of model setting 0 is not far apart from that of model setting 7, which takes roughly one seventh of the time to be trained. As a matter of fact, they yield the two confusion matrices reported in figures 12a and 12b. Hence, setting 7 might be preferable.

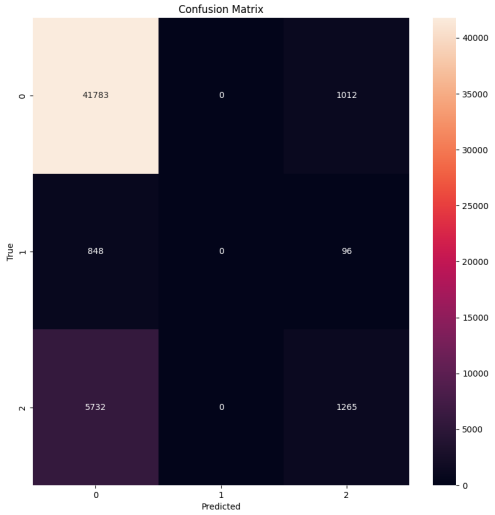
7.5 Conclusion

The final accuracies obtained with the three models are very similar and all of them round up to 85%. For this reason, no model is strictly better than the other ones. Decision Trees are generally lighter (especially in this case since setting 5's depth was 5 layers) but ensemble methods usually lead to better variance or bias.

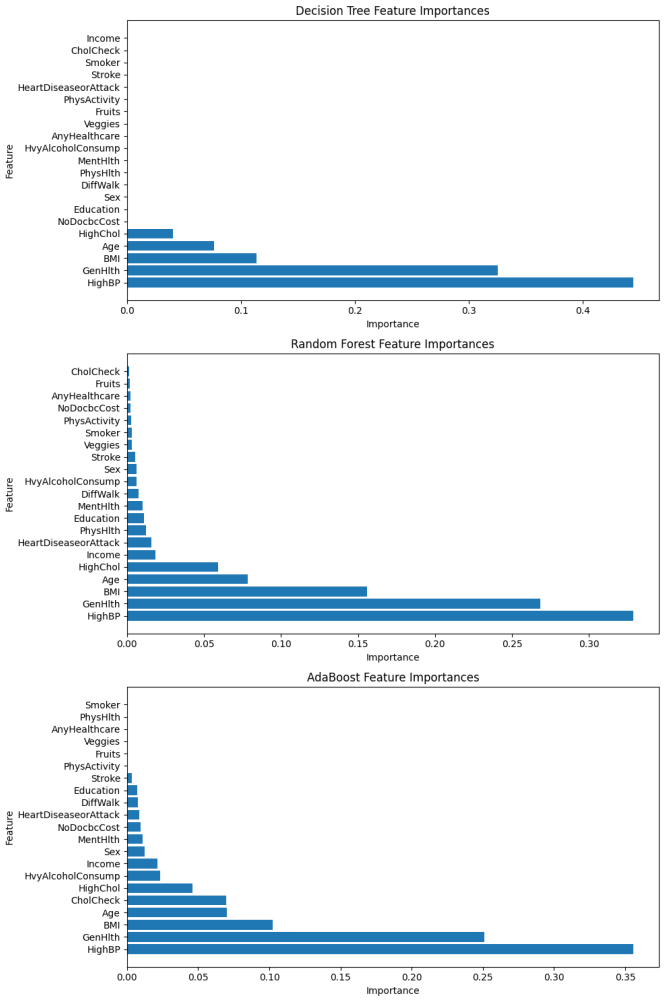
Further investigations of this class of machine learning methods on this dataset might take into account a subset of selected features. Therefore, we present the importance of each feature for each model class in figure 12c (in addition to the model-agnostic PCA analysis).



(a) Confusion Matrix for Setting 0



(b) Confusion Matrix for Setting 7



(c) Feature Importances in All Models

Figure 12: Comparison of Confusion Matrices for Settings 0 and 7, and Feature Importances

8 Neural Network

We designed a neural network to address a multiclass classification problem, aiming to predict three different levels of diabetes risk. The classes were defined as follows:

- Class 0: represents patients without diabetes;
- Class 1: represents patients at risk or in a pre-diabetic phase;
- Class 2: represents patients diagnosed with diabetes.

The objective of the model was to distinguish between these three states, providing classification support based on different health indicators present in the dataset.

8.1 Neural Network Configuration

8.1.1 Network Architecture

The architecture is composed of a series of densely connected, or fully connected, layers. In each layer, every neuron is connected to all neurons in the previous layer, allowing the network to learn complex patterns by combining information from all features. The network includes an initial dense layer with 128 neurons, followed by layers with 64, 32, and finally 16 neurons. Each hidden layer uses the ReLU (Rectified Linear Unit) activation function, promoting signal propagation that accelerates learning, while the output layer uses the softmax activation function. The softmax function converts the values into relative probabilities across different classes, assigning a probability to each class so that the sum of probabilities equals one. This makes it ideal for multi-class classification problems, such as this one, where classes represent different levels of diabetes risk (0, 1, or 2).

8.1.2 Model Compilation

The Adam (Adaptive Moment Estimation) optimizer was chosen for its ability to update model weights more efficiently than simple gradient descent. Adam combines the benefits of the momentum optimizer (which takes into account the rate of change of gradients) with those of the RMSprop optimizer (which adjusts the learning rate based on how often features are updated). This allows the learning rate to adapt for each network weight, improving stability and convergence speed. For multi-class classification, the sparse_categorical_crossentropy loss function was chosen. This function calculates the difference between the probabilities predicted by the network and the correct labels, aiming to minimize it during training. Compared to other loss functions, sparse_categorical_crossentropy is efficient for numerical classes, making the computation less demanding and enhancing performance.

8.2 Model Training

8.2.1 Hyperparameters

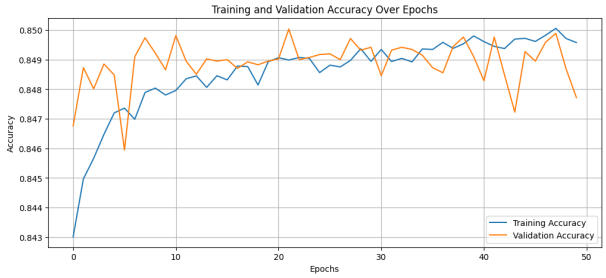
The following hyperparameters were selected for model training:

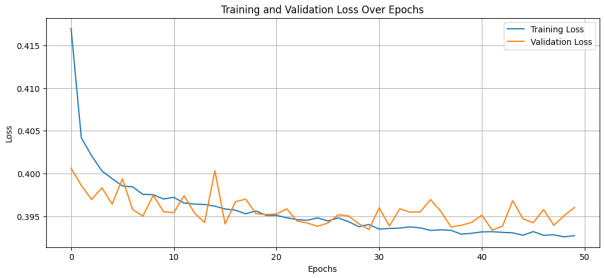
- **Epochs = 50:** The model was trained for 50 epochs, meaning 50 complete passes over the training data. This number is sufficient to observe a stabilization in performance without risking excessive training.
- **Batch Size = 32:** Model weights are updated after each batch of 32 examples. This value is a balance between stability and training speed, allowing the model to learn meaningful patterns without requiring high memory usage.
- **Validation Split = 0.2:** 20% of the training data was set aside for validation, enabling monitoring

of model performance on unseen data during training. This helps to identify potential overfitting issues.

8.2.2 Training Progress

In the graph, we can observe the accuracy trends for both the training and validation sets over the 50 epochs.





The curves show:

- **Initial Phase (0-10 epochs):** At the beginning, training set accuracy increases rapidly, while validation set accuracy fluctuates, indicating that the model is learning the general features of the dataset.
- **Stabilization (10-40 epochs):** After about 10 epochs, both curves start to stabilize with slight oscillations. This behavior suggests that the model is finding a balance between learning and generalization, with no evident signs of overfitting.
- **End of Training (40-50 epochs):** Towards the end, validation set accuracy shows some oscillations but remains close to the training set accuracy. This indicates that the model has generalized well on the validation data without suffering from overfitting or underfitting.

The final accuracy achieved is similar for both the training and validation sets, settling around 85%. This consistency between the two curves suggests that the model has learned useful patterns for classification without overly adapting to the training data.

8.3 Model Evaluation

8.3.1 Final Accuracy

The model's final accuracy on the test set is 84.78%. This value indicates that the model correctly classifies most samples overall. However, accuracy alone does not provide complete information on how the model handles each diabetes risk class, especially considering that the classes may be imbalanced.

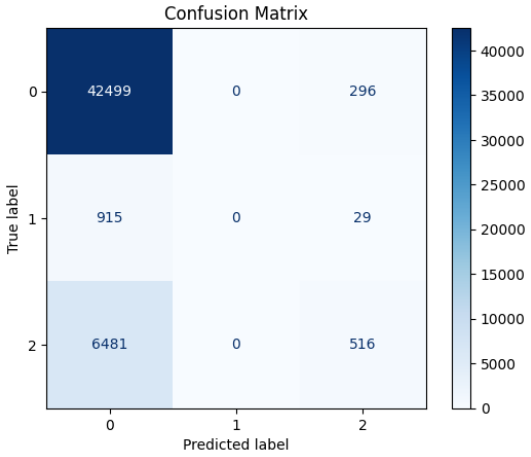
8.3.2 Confusion Matrix

The confusion matrix provides a detailed analysis of the model's performance for each class. In the attached matrix:

- **Class 0 (No Diabetes):** The model correctly identifies most Class 0 samples, with 42,499 samples correctly classified and only 296 misclassified into other classes. This reflects high accuracy for Class 0, but also suggests a possible bias toward the majority class.
- **Class 1 (Pre-Diabetes):** Class 1 is much less represented, with a total of 944 samples, of which 915 are mistakenly classified as Class 0. This result indicates that the model struggles to correctly

distinguish Class 1 samples, often misclassifying them as Class 0.

- **Class 2 (Diabetes):** Of the 6,997 Class 2 samples, only 516 are correctly classified, while 6,481 are mistakenly classified as Class 0. This behavior suggests that the model also has difficulty with Class 2, frequently confusing it with Class 0.



8.3.3 Conclusions

These results suggest that the model is highly effective at classifying Class 0, which is likely the majority in the dataset. However, performance is limited for Classes 1 and 2, indicating an issue with class imbalance or a lack of representation for higher-risk classes (1 and 2). This could potentially be improved with techniques such as undersampling Class 0 or oversampling Classes 1 and 2, or by using loss metrics that penalize errors more heavily on less represented classes.

8.4 Binary Classification with the Neural Network

8.4.1 Problem Reformulation

Since the original neural network achieved good overall accuracy but failed to correctly identify Class 1, we decided to reformulate the problem as a binary classification task. In this new configuration, we use a dataset divided into two classes:

- **Class 0:** Represents samples without a diabetes diagnosis.
- **Class 1:** Represents samples with a diabetes diagnosis.

8.4.2 Model Adjustments

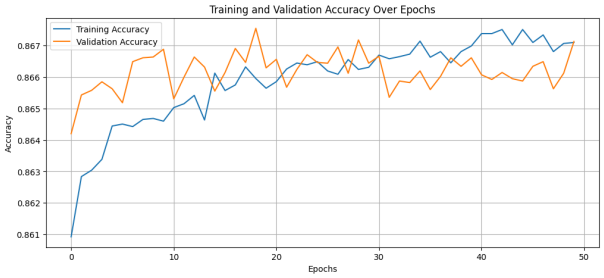
To adapt the model for binary classification, a few minor changes were made:

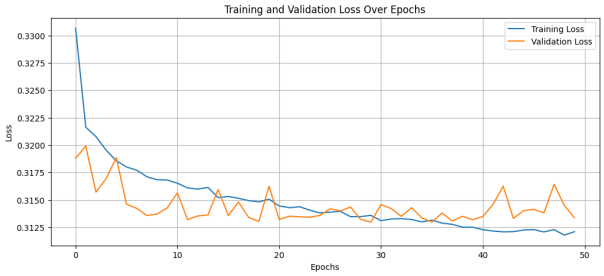
- **Model Architecture:** The output layer was modified to have a single unit with a sigmoid activation, appropriate for a binary problem.
- **Loss Function:** The loss function was changed to binary_crossentropy, ideal for binary classification.

8.4.3 Results and Analysis

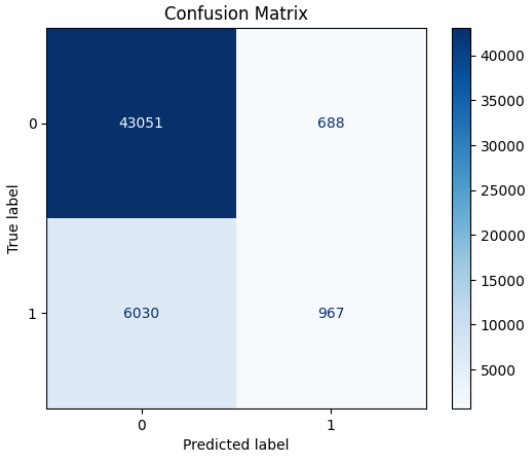
Observing the Training and Validation Loss and Accuracy graphs, we note the following:

- The loss for both the training and validation sets progressively decreases, stabilizing toward the end. This indicates that the model is converging, with no obvious signs of overfitting.
- The accuracy on the validation set stabilizes around 87%, an acceptable value but still reflecting difficulties in correctly classifying Class 1.





The confusion matrix shows a strong predominance of correct predictions for Class 0, but many instances of Class 1 are still classified as Class 0.

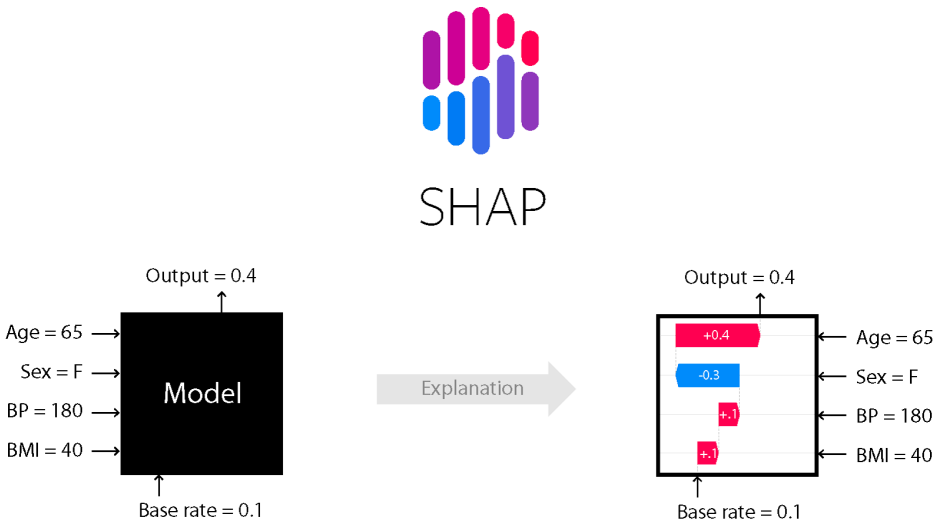


8.4.4 Summary

In summary, we have a better accuracy than the one with three classes, but while the model performs well for Class 0 samples, it continues to show some limitations for Class 1. This may suggest the need for further data rebalancing.

9 Interpretability (SHAP)

Neural networks (NNs) are powerful tools for modeling complex relationships in data. However, their black-box nature makes it challenging to understand the rationale behind their predictions, especially in high-stakes fields such as healthcare. SHAP (SHapley Additive exPlanations) is a method rooted in game theory that provides a unified framework for interpreting machine learning models, including neural networks.



SHAP is based on the concept of Shapley values, which originate from cooperative game theory. The Shapley value is a way of fairly distributing the total gain (or cost) of a game among its participants (or players). In the context of machine learning, the "players" are the features of the input data, and the "gain" corresponds to the model's prediction. The Shapley value of a feature represents its contribution to the prediction, calculated by considering all possible subsets of features and marginalizing over their contributions. The sign of a Shapley value indicates whether the feature increases or decreases the predicted outcome. A positive value means the feature contributes positively, pushing the prediction higher, while a negative value indicates a negative contribution. The magnitude of the Shapley value quantifies the strength of this contribution, with larger absolute values signifying greater influence. This property makes SHAP a powerful tool for explaining individual predictions by attributing importance to each feature in a manner consistent with the underlying model.

Applying SHAP to neural networks enables the extraction of feature importance rankings, which can help refine models and improve their interpretability. For example, in healthcare, identifying the most influential features in predicting a condition such as diabetes can guide medical practitioners to understand the model's rationale and validate its decisions against clinical knowledge. Moreover, SHAP can highlight potential biases or errors in the model, fostering trust and facilitating its integration into decision-making processes. However, while SHAP is highly insightful, it has some limitations. Calculating exact Shapley values is computationally expensive, particularly for complex models like deep neural networks, as it requires evaluating all possible feature combinations. Approximations, such as those provided by many SHAP implementations, help mitigate this issue but may introduce inaccuracies. Additionally, interpreting Shapley values still requires domain knowledge to ensure that the explanations align with real expectations.

In conclusion, SHAP offers a robust method for interpreting neural networks by providing insights into feature contributions and importance. These insights not only improve the transparency and validity of neural network predictions but also bridge the gap between human understanding and model outputs. This is particularly critical in fields like healthcare, where understanding and trust in predictive models are paramount.

10 LLM Integration

Large Language Models (LLMs) represent a significant advancement in natural language processing. Their ability to generate coherent and contextually relevant text makes them invaluable for tasks requiring complex communication and explanation. In this project, an LLM was integrated to improve diabetes predictions by generating professional and patient-specific reports. The generated report serves multiple purposes:

- **Patient Communication:** the LLM translates the technical results into easily understandable language, enabling patients to understand their health status;
- **Medical Insights:** the report supports healthcare providers in validating predictions and tailoring interventions;
- **Actionable Recommendations:** the reports highlight risk factors and suggest lifestyle changes or medical follow-ups.

To integrate the LLM, a structured process was implemented:

1. **Prompt Engineering:** a detailed template was designed to summarize the SHAP analysis and prediction results, ensuring that all critical information was included and formatted for clarity;
2. **API Integration:** the LLM was accessed through the Google Generative AI API, configured with the appropriate security and performance settings;
3. **Error Handling:** mechanisms were incorporated to manage failures in text generation, including fallback strategies to notify users or request manual interpretation.

While the integration of an LLM adds significant value, it is not without challenges: LLM-generated outputs depend heavily on the quality and clarity of the input prompt. Additionally, while the LLM can provide contextually rich explanations, its outputs must be reviewed by domain experts to ensure medical validity and avoid potential inaccuracies. Ethical considerations around the confidentiality and sensitivity of health data should also be addressed, ensuring compliance with data protection standards.

In conclusion, the LLM integration bridges the gap between machine learning predictions and human decision-making by providing clear, actionable, and professional reports. This approach underscores the potential of combining predictive analytics with generative AI to improve healthcare communication and outcomes.