# Diabetes prediction using Multilayer Perceptron

by

Yussup Tumgoyev

Submitted to the School of Engineering and Digital Sciences
in partial fulfillment of the requirements for the degree of

Master of Science in Data Science

at the

NAZARBAYEV UNIVERSITY

May 2022



Author .....

Yussup Tumgoyev
School of Engineering and Digital Sciences
Apr 29, 2022

Certified by .....
Muhammed Fatih Demirci
Associate Professor of Computer Science
Thesis Supervisor

Accepted by .....
Vassilios D. Tourassis
Dean, School of Science and Technology

# Diabetes prediction using Multilayer Perceptron

by

Yussup Tumgoyev

Submitted to the School of Engineering and Digital Sciences
on Apr 29, 2022, in partial fulfillment of the
requirements for the degree of
Master of Science in Data Science

## Abstract

Diabetes mellitus is one of the most popular diseases that causes 1,5 million people to die each year. It is the major cause of blindness, kidney failure, heart attacks, stroke, and lower limb amputation. Being the world's top 9 severe diseases, diabetes puts a burden on the world's economy and the healthcare system. There are two types of diabetes. Type 1 is generic and in the vast majority of cases shows up early in life. This study focuses on type 2 diabetes which is around 90% of all diabetes cases, and that can be diagnosed. Early diagnosis of diabetes can prevent serious medical complications. In this literature, we are introducing a framework of diabetes prediction based on a Multilayer Perceptron of only 1 hidden layer and 8 neurons, which is lighter than the state-of-art framework which consists of 3 hidden layers and 144 neurons in total. The grid-search method was used for hyperparameter tuning to maximize Area Under the ROC Curve (AUC) that was chosen as a performance metric. Pima Indian Diabetes Dataset was used to conduct the experiments. The dataset was preprocessed with outlier rejection, missing values imputation, standardization, data scaling, and feature selection algorithms. K-fold cross-validation technique was used to train/test the classification model. The Multilayer Perceptron model was tuned with various hyperparameters as well as the dynamic learning rate. Finally, the best lightweight MLP model consisting of 1 hidden layer that reaches the AUC of 0.90 was obtained. The model performs as well as the state-of-art but is at least 5 times faster in training and more than 80 times more efficient in terms of memory usage.

Thesis Supervisor: Muhammed Fatih Demirci
Title: Associate Professor of Computer Science

# Acknowledgments

This work is dedicated to my mother - Ganzha Aisha, who insisted that I enter Nazarbayev Intellectual School and later Nazarbayev University. Thank you for all the sacrifices you have made for me and for doing everything possible to help me grow as a person. I would also like to express my gratitude to my father - Tumgoyev Lom-Ali, who instilled masculinity in me, provided me with everything I needed to study and continues to be a strong pillar of support.

Much appreciation to my mentor Seitzhan Sypabekov without whose foresight and the right words I would have dropped out of university.

The completion of this could not have been possible without the expertise of Professor M. Fatih Demirci. He was kindly sharing his expertise and support during all of this research work. I also want to thank Profesor Askar Boranbayev for his openness and faith in me. Professor Martin Lukac for his sense of humor during my study.

Thanks to Amanzhol, my roommate for the last 7 years for being next to me all the time.

A depth of gratitude is also owed to the Vice President for Student Affairs and International Cooperation Ms. Kadisha Dairova for her wisdom and support during the most difficult times of student life. Thanks to the old DSA team represented by Aseke Bekzhanov, Bakhtiyar Rahimov, and Kassymkhan Nurmukamedov. They were staying with us from the very beginning of the foundation year at Nazarbayev University.

# Contents

# Chapter 1

# Introduction

Diabetes is a severe disease that is considered to be on the top list of world healthcare major challenges with an ever-growing reach of the global population. World Health Organization (WHO) ranks diabetes to be the top nine causes of death among all the diseases in terms of the mortality rate worldwide [52]. At the same time diabetes puts an economic burden on the world's economy. American Diabetes Association's research shows that the estimated cost of diagnosed diabetes in 2017 is $327 billion [6]. The number of people suffering from this incurable disease has shown a huge increase in the past four decades. In its April report WHO mentioned the worldwide increase of diabetics from 108 million people in 1980 to 422 million people in 2019, with around 1,5 million death caused annually by diabetes [52]. Additionally, diabetes is claimed to be the major cause of blindness, kidney failure, heart attacks, stroke, and lower limb amputation [52].

Diabetes is a chronic disease that develops either when the person's body cannot produce enough insulin (type 1 diabetes) or the body's cells are resistant to perceiving the required amount of insulin (type 2 diabetes) for the proper body processes [42]. Type 1 diabetes is a genetic disorder that usually shows itself early in life, while type 2 diabetes is mostly associated with diet and develops in time [19]. Both type 1 and type 2 are incurable. However, type 2 is amenable to diagnosis and treatment. Therefore, early diagnosis of type 2 diabetes can prevent serious medical and financial complications.

For the past two decades, different methods and techniques were employed on Pima Indian Diabetes Dataset (PIDD) to make the best diabetes predicting model. Linear Discriminant Analysis (LDA) [21], Support Vector Machine (SVM) [60], Decision tree (DT) [63], Naive Based (NB) [20], Random Forest (RF) [11], AdaBoost (AB) [13], Logistic Regression (LR) [80] are a part of ML-based approach. General Regression Neural Networks (GRNN) [49], Higher Order Neural Networks (HONN) [4], Convolutional Neural Networks (CNN) [3] are examples of Neural Network-based architectures suggested for diabetes prediction. Some authors like [34, 81, 44, 1, 45] suggest ensemble models which are a combination of ML and NN approaches. Although there are numerous frameworks already been published with competitive preciseness, in recent years, still, the improvement requires in the performance and simplicity of the model for its implementation in the industry. The dire need for the tool that can help with early diabetes detection is strongly supported by [10, 42].

This ressearch is based on the Multilayer Perceptron (MLP) classification model that is a part of a NN approach toward diabetes prediction. An MLP model of 1 hidden layer and 8 neurons were proposed in this literature versus 3 hidden layers and a total of more than 140 neurons recently proposed as a state-of-art by [34]. The method described in this literature is at least 5 times faster in training and more than 80 times more efficient in memory usage. It consist of outlier rejection, missing values imputation and normalization for data preprocessing and seven steps of experiments that cover the grid-search [43] for optimal number of hidden layers and neurons, several feature selection techniques, hyperparameters tuning including number of hidden layers, number of neurons on each layer, batch size, learning rate, number of the epoch, dropout neurons, loss functions, and an MLP optimizer. The method is based on the publically available PIDD from the National Institute of Diabetes and Digestive and Kidney Diseases [51, 12]. The experiments were conducted to maximize the Area under the ROC curve (AUC) [16] which was chosen as a performance metric.

The rest of the paper is organized as follows: Chapter II is about the previous work done in diabetes prediction. Chapter III describes the dataset used, prepossessing techniques, and the proposed methodology. Chapter IV is about the experiments

done and the techniques used. In Chapter V the results are illustrated. The discussion can be found in Chapter VI. Limitations, further work and conclusion are described in the last Chapter VII.

# Chapter 2

# Related work

A major number of scientific work has been done in the area of diabetes prediction. Various ML and NN-based algorithms have been studied that boosted the development of this research area. This section will talk about some previous research on diabetes prediction based on machine learning techniques and/or NN-based algorithms to show the feasibility of using different data analysis techniques in the field of diabetes prediction.

Swapna G et al [75] conducted a study that showed that machine learning practices were effective for building diabetes prediction models using Heart Rate Variability (HRV) signals in the deep learning (DL) approach. The author was motivated by the fact of the growing mortality rate from diabetes in the world. The convolutional neural network(CNN) and a long-term-short-term memory (LSTM) were used in this research to build a new ensemble model for the identification of the complex chronological characteristics of HRV signals. The Support Vector Machine (SVM) algorithm was applied to the identified characteristics for data classification. The developed model can find its demand in the governmental and private healthcare sector to predict or identify diabetes using ECG [29]. The problem of the rapid development and the huge global impact of diabetes has also motivated Sajida P. et al. [55] to start working on a model to predict diabetes. Sajida P. compared three classification methods namely AdaBoost, bagging [17], and J48 [47]. The author claims that the AdaBoost method is better for diabetes prediction than bagging and J48 algorithms.

Sajida has introduced a model that showed improved results for diabetes prediction in the dataset related to the Canadian population. The author of [55] believes that AdaBoost may also show a good performance in the classification of heart diseases. The author of [24] had a broader scope of research compared to the previously mentioned work. He has evaluated six different machine learning techniques namely SVM, LR, Artificial Neural Network (ANN), Classification tree, and KNN. In [24] the author managed to achieve improvements in accuracy, specificity, and sensitivity. The highest accuracy achieved was 78% with a 0.22 misclassification rate with a 10-fold cross-validation data split technique. Many research works used accuracy as an evaluation metric. Among such studies are the following works that are based on the Pima Indian Diabetes dataset:

- Abu-Naser [25] has achieved an accuracy of 87% for diabetes prediction with an Artificial Neural Network [35].
- The authors of [61] have achieved an accuracy of 76% using the Naive Bayes algorithm which was the best compared to SVM and DT.
- Ram D. et al.[38] achieved an accuracy of 78% with the logistic regression algorithm.
- According to Sidong W. et al. [76] deep neural network model has performed better than SVM, DT, and NB achieving an accuracy of 78%.
- Aiswarya I. et al. [37] have achieved an accuracy of 80% with the Naive Bayes algorithm.
- Quan Z. et al. [82] used Decision Tree to achieve around 77% of accuracy.
- Roshan B. [14] created the models based on the Gradient Boosting, Logistic Regression, and Naive Bayes algorithms. The highest result of 86% of accuracy was shown by Gradient Boosting.

The list above could go for up to 30-50 articles that somehow used Pima Indian Diabetes dataset to predict diabetes. Much less research uses the area under

the ROC (Receiver Operating Characteristics) curve (AUC) as an evaluation metric. The author of [46], however, argues that AUC is statistically consistent and more discriminating than accurate in evaluating and comparing classification learning algorithms. Therefore, in this literature, the AUC was used as an evaluation metric for binary diabetes classifications.

Several articles about diabetes prediction were also written using AUC as a performance metric. In [62] authors employed three different classification algorithms namely DT, SVM, and NB to find the likelihood of diabetes, based on the PIDD. The author demonstrates that the NB algorithm achieves an AUC of 0.82. The state-of-art AUC for Multilayer Perceptron (MLP) algorithm was demonstrated in [34]. The authors were motivated to decrease the risk factor of diabetes by early detection of this severe disease. In [34] the author employed Machine Learning (ML) classifiers such as k-nearest Neighbour, Decision Trees, Random Forest, AdaBoost, Naive Bayes, XGBoost, and Multilayer Perceptron [28], as well as ensemble models (combinations) of the ML techniques mentioned. The author managed to achieve 0.90 of AUC for the MLP classifier with the best model of 3 hidden with N1 = 16, N2 = 64 and N3 = 64 neurons for each hidden layer respectively, and Independent Component Analysis (ICA) [36] feature selection technique, on the previously preprocessed dataset. The result of 0.89 AUC was achieved with Principal Component Analysis (PCA) [58] feature selection technique and 0.90 AUC with the ICA. The work done by [34] was taken as a baseline for this literature.

# Chapter 3

# Materials and Methods

## 3.1 Dataset Description

Pima Indians Diabetes Dataset is a publically available dataset from the National Institute of Diabetes [12]. The Pima Indian Population has been under scientific interest since 1965, because of its high incidence rate of diabetes [74].

It consists of the medical analysis of 768 women from the Pima Indian population near Phoenix, Arizona [74], and includes 9 features: pregnancy times, glucose, blood pressure, skin thickness, insulin, BMI, diabetic pedigree function, age, and a class label that identifies if a patient has diabetes (class label = 1) or not (class label = 0). Each feature is explained below with its type and description:

- Pregnancies (Numeric) — Number of times pregnant
- GlucosePlasma (Numeric) — 2 hours glucose concentration from glucose tolerance test
- Blood Pressure (Numeric) — Diastolic blood pressure (mm Hg)
- SkinThickness (Numeric) — Triceps skin-fold thickness (mm)
- Insulin (Numeric) — Two hours of serum insulin (mu U/ml)
- BMI (Numeric) — Body Mass Index (weight in kg/(height in m)$^2$)

- Diabetes Pedigree Function (Numeric) — Diabetes pedigree function
- Age (Numeric) — Age in years
- Outcome (Numeric) — Class variable (0 or 1)

| # | Atribute | Mean ± stdev | Range |
|---|---|---|---|
| 1 | Pregnancies | 3.85 ± 3.37 | 0 - 17 |
| 2 | Glucose | 120.90 ± 31.97 | 0 - 199 |
| 3 | BloodPressure | 69.11 ± 19.36 | 0 - 122 |
| 4 | SkinThickness | 20.54 ± 15.95 | 0 - 99 |
| 5 | Insulin | 79.81 ± 115.24 | 0 - 846 |
| 6 | BMI | 32.00 ± 7.88 | 0 - 67.1 |
| 7 | DiabetesPedigreeFunction | 0.47 ± 0.33 | 0.078 - 2.42 |
| 8 | Age | 33.24 ± 11.76 | 21 - 81 |

Table 3.1: The overview of Pima Indian Diabetes Dataset

The PIDD consist of the results of females from 21 to 81 years old. The dataset is imbalanced and has 500 negatively tested and 268 positively tested patients. The PIDD has no null values, all the values in the dataset are numeric. Previously several researchers considered PIDD to have no missing values, but later it was found out that the dataset has 51% of the missing values replaced with zeros. For example, Body Mass Index, which by definition is a number higher than zero. Figure 3-1 shows the population distribution of the Pima Indian Diabetes Dataset, where red and yellow colors mean non-diabetic and diabetic respectively.

## 3.2 Proposed framework

The Figure 3-2 illustrates the proposed framework. The PIDD were preprocessed with: outlier rejection (O), missing values imputation (M), and a Z-score standardization (Z) techniques with two different feature selection methods (PCA and ICA) of 4 and 6 principal components resulting in twelve differently preprocessed datasets.

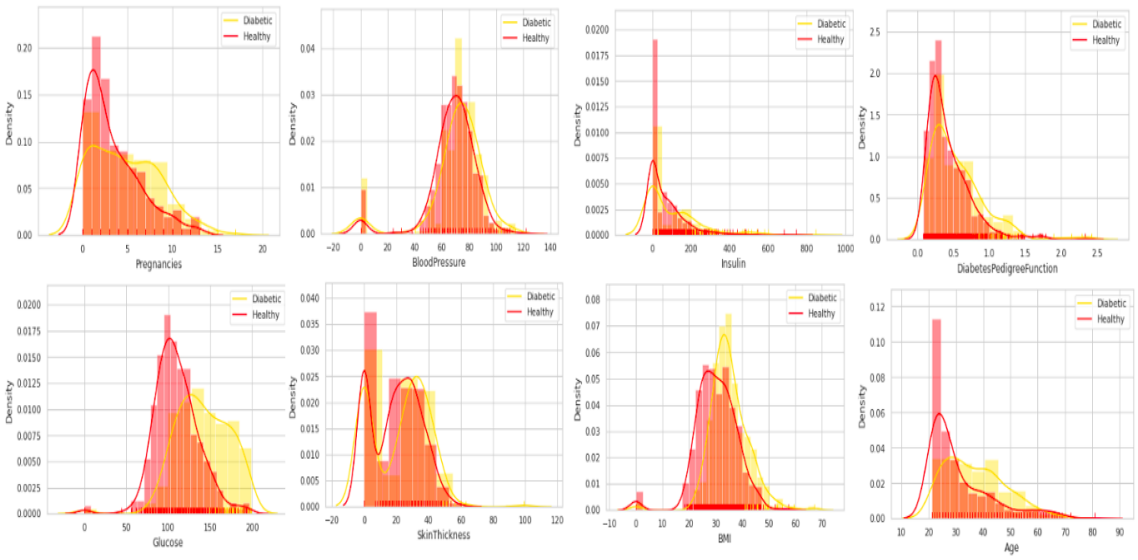


Figure 3-1: The population distribution of Pima Indian Diabetes Dataset, where red and yelow colors means non-diabetic and diabetic respectively

These datasets were tested on the MLP architecture with three hidden layers proposed by [34] and taken as a baseline. The MLP model from [34] was replicated and tested on the dataset preprocessed in 12 different ways to find the preprocessing technique for which an MLP model shows the best AUC results. Next, extensive experiments were done on the selected dataset to find out the best architecture and hyperparameters in terms of model complexity and performance.

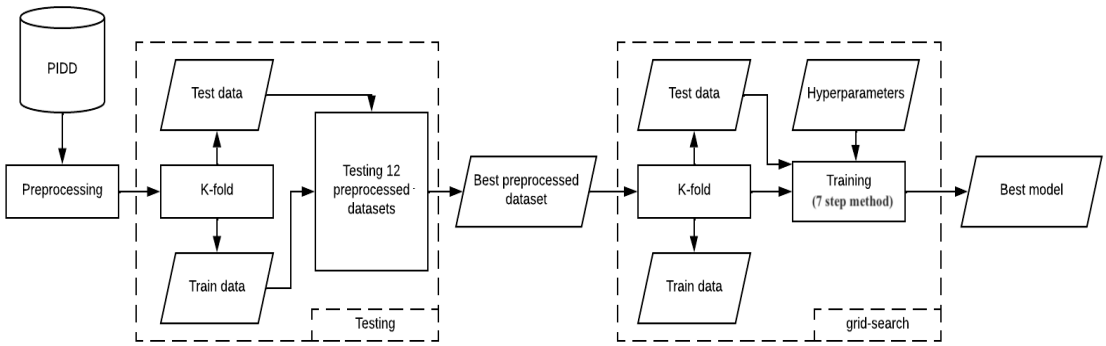


Figure 3-2: The proposed block diagram of a robust and automatic diabetes prediction.

### 3.2.1 Preprocessing

Before proceeding to classification it is crucial to preprocess the data, as MLP might be sensitive to missing values, outliers, and distribution of the data. Therefore the preprocessing consist of the following steps:

1. Outlier rejection (O)
2. Missing values imputation (M)
3. Data normalization (Z)

The following steps are briefly described below.

An outlier is a marked deviation of an observation from other observations [34]. In this literature, the rows with outliers are deleted from the dataset as classification models might be sensitive to it. The process of outlier rejection can be described by the formula (3.1) [9]

$$O(x) = \begin{cases} x, if Q_1 - 1.5 \times IQR \leq x \leq Q_3 + 1.5 \times IQR \\ reject, otherwise \end{cases}. \tag{3.1}$$

where $x$ represents the set of items of the feature vector that lies in n-dimensional space, $x \in R^{\mathrm{n}}$. $Q_1, Q_3$, and $IQR$ are the first quartile, third quartile, and interquartile range of the feature vector respectively. The $IQR$ is used to describe the spread of a distribution between third and a first quartile [77], where $Q_1$, $Q_3$, $IQR \in R^{\mathrm{n}}$. $O(x)$ is a function applied to each feature vector of the dataset that takes $x$ as the set of items of the feature vector and returns the set $x$ with rejected values based on the conditions of the system of equations in (3.1). In the first preprocessing stage the outlier rejected dataset (O) was obtained.

Next, the (O) dataset was processed to fill in missing values. The problem of missing values is common for most of the datasets and can have a significant effect on the final output of the classification model [31]. As it was mentioned before the dataset had no null values, but 51% of rows were containing missing values. In this literature, missing values for each item in the feature space were imputed by

the respective features mean, so no missing data were rejected. The advantage of the mean value imputation technique is that it does not introduce outliers to the imputed data. The way missing values were handled is described by the formula (3.2) [22]

$$M(x) = \begin{cases} mean(x), if x = missed \\ x, otherwise \end{cases} . \tag{3.2}$$

where $x$ represents the set of items of the feature vector that lies in n-dimensional space, $x \in R^{\mathrm{n}}$. $M(x)$ is a function applied to each feature vector of the dataset that takes $x$ as the set of items of the feature vector and returns the set $x$ with missing values imputed based on the conditions of the system of equations in (3.2).

After the outlier rejection and missing values imputation, the (M) dataset was normalized using Z-score normalization. The normalization technique changes the values of the attributes to use a common scale, without losing the difference in values range, making the dataset normally distributed with zero mean and unit variance [34]. The Z-Score normalization can be described by the formula (3.3) [54].

$$Z(x) = \frac{x - \overline{x}}{\sigma} \tag{3.3}$$

where x represents each item of the each feature vector in n-dimensional space, x $\in \mathrm{R}^{\mathrm{n}}$, $\sigma$, and $\overline{x} \in \mathrm{R}^{\mathrm{n}}$. $Z(x)$ is a function applied to each feature vector of the dataset that takes $x$ as the set of items of the feature vector and returns the standardized set $x$ based on the right hand side (3.2).

For the feature selection, the Principle Component Analysis (PCA) [58] and Independent Component Analysis (ICA) [36] methods were used to compare their performance on the PIDD.

### 3.2.2 Cross validation

The K-fold Cross-Validation (KCV) is used to evaluate the classification model's performance on the limited amount of data [5]. The k-fold method has a parameter $K$ that indicates the number of equal pieces the data is split to. KCV's advantage over

other methods like simple train/test split is that it is less biased as the whole dataset is used as training and testing data. In this literature, we use 5-fold cross-validation. The Figure 3-4. explains the techniques of the division of the data into train and test that are used in this work. The $K-1$ folds are used for training and the remaining fold is used for testing the MLP model performance. This procedure runs $K$ times with random data selected for training and testing, and the final performance metric is the average of all $K$ MLP performances. As PIDD is not balanced (see Figure 3-3) in terms of the number of positive and negative samples the stratified KCV [79] was employed to balance the percentage of each class in train and test data split for each fold. The final MLP performance metric for $K$ folds can be described by the formula:

$$S = \frac{1}{K} \times \sum_{n=1}^{K} P_{\mathrm{n}} \pm \sigma_{\mathrm{n}} \tag{3.4}$$

where $S$ is the final performance metric of MLP model for all of the $K$ combinations of train/test data split, $P$ is the performance metric and $\sigma$ is a standard deviation [18] of each of the $K$ experiments, where $P_{\mathrm{n}}$ , $\sigma_{\mathrm{n}} \in R$, for n = 1,2,...,$K$.

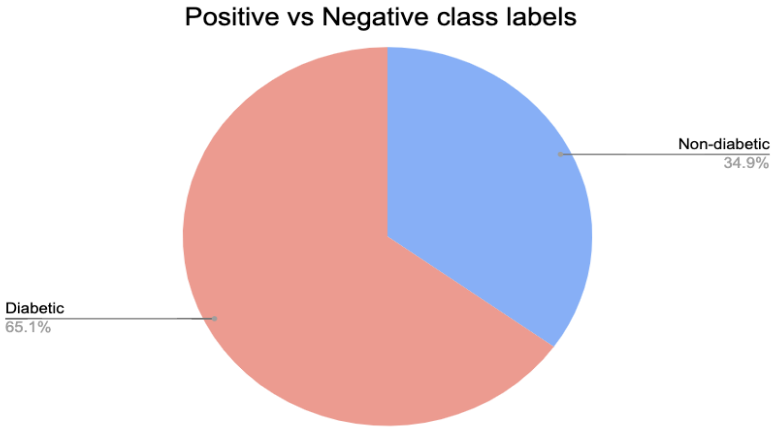


Figure 3-3: The imbalance in class labels

### 3.2.3 Multilayer Perceptron

Multilayer Perceptron is a part of an Artificial Neural Network that consists of three types of a layer such as input layer, hidden layer, and output layer. Figure 3-5 de-

scribes the architecture of the MLP as a system of interconnected nodes [28]. The nodes on each layer are neurons, except for the input layer where the nodes are the input vector from the dataset. The neurons uses a non-linear activation function to describe the input-output relation [2]. The neurons are trained with a backpropagation algorithm [59]. The prediction or classification is done on the output layer. The computation done on every neuron can be described by the formula (3-5)

$$h(x) = \Phi(\sum_{j} b + w_{\mathrm{j}} x_{\mathrm{j}}) \tag{3.5}$$

where the $x_{\mathrm{j}}$ is the inputs, $w_{\mathrm{j}}$ is weights, $b$ is a bias and $\Phi$ is a non-linear activation function. The weights of each neuron are updated using the back-propagation algorithm [48] during the training to decrease the error of the MLP model.

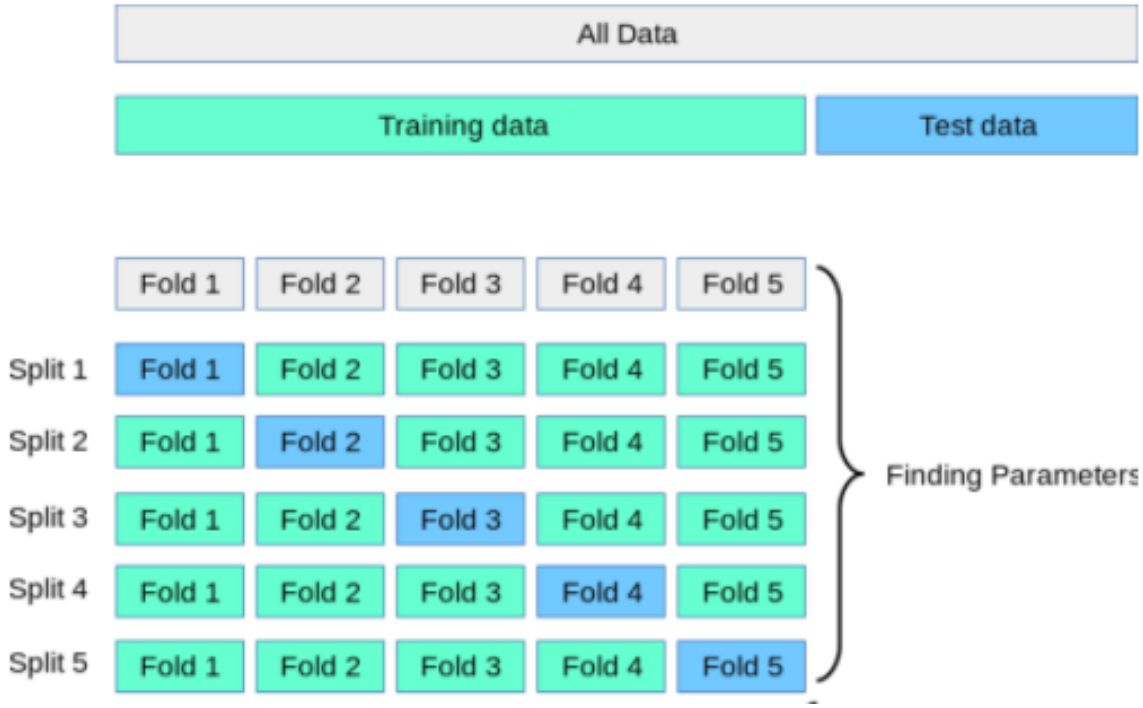


Figure 3-4: The partitioning of the PID dataset for KCV for both the hyperparameters tuning and evaluation [64].

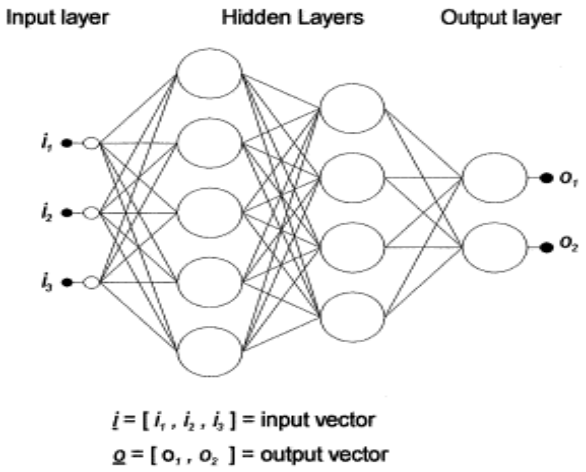


Figure 3-5: A multilayer perceptron with 2 hidden layers

### 3.2.4 Evaluation metric

The performance evaluation of the predicting algorithm was measured by Area Under the ROC Curve (AUC), where all the experiments were conducted to maximize this metric. At the same time, a careful analysis of the learning curves was performed to understand the classification model's behavior and avoid overfitting [23]. Finally, the model with the best performance and low complexity was selected as the best model. The confusion matrix with True Positives(Tp), True Negatives(Tn), False Positives(Fp), and False Negatives(Fn) was reported. The Sensitivity (SN) and specificity(SP) were calculated from Tp, Tn, Fp, and Fn. Sn and Sp are respectively used to quantify the type-II error (the patient has diabetes, but is classified as non-diabetic) and type-I error (the patient is classified as non-diabetic, but has diabetes).

### 3.2.5 Seven-step method for the best MLP architecture identification

In this literature, the goal is to find the best MLP architecture in terms of model complexity and the best AUC score. To achieve this more than 1000 different combinations of architectures and hyperparameters were tested to maximize the Area Under the ROC Curve (AUC) metric. The best hyperparameters such as batch size, learning rate, dropout layer, optimizer, loss function, number of epochs, and the number of

hidden layers and neurons on each layer were chosen with grid-search algorithm. No Python libraries, like sklearn.GridSearchCV [67], were used to implement the grid-search because they did not provide an opportunity to analyze the learning curves of each tested fold in the k-fold cross-validation procedure. Thus the grid-search method was hardcoded to find out the best combination of hidden layers and neurons that yield maximum AUC as well as the good fitted learning curve.

**Seven-step method:**

1. Replicate the model proposed in [34] and test it on the Pima Indian Diabetes Dataset previously preprocessed in twelve different ways described in Figure 4-1. Select the dataset on which MLP yelded the best performance for further steps.

2. Gridsearch the best combinations of the number of hidden layers as well as the number of neurons on each layer, with the constant hyperparameters proposed in [34]. Select the best combinations of hidden layers and neurons on each layer.

3. Test the obtained best combinations of hidden layers and neurons on the best performing preprocessed dataset, based on the results of Step 1, with different number of principal components for feature selection methods. Check if alternative feature selection techniques affect MLP in a positive or negative way. Choose architectures with respective feature selection technique for further steps, based on it's performance.

4. Gridsearch the best hyperparameters for the identified best-performing combinations of hidden layers and neurons. Obtain the best performing MLP architecture.

5. Use alternative data preprocessing techniques with early selected MLP architecture to find out if the performance changing positively or not.

6. Apply dynamic learning rate to test its effect on the MLP performance. Next, test the MLP model with different optimizers and loss functions. Finally, ana-

lyze the learning curves of obtained best architectures, find the optimal number of epochs based on the loss learning curve.

7. Select the best performing architecture with the obtained best hyperparameters and test its performance on the twelve datasets from Step 1. Record the results.

On every step of the seven-step method the results of the AUC, and the behavior of the loss learning curves [78] were analyzed to detect the overfitting, underfitting or unpredictable behavior of the model. As a result of Step 1, the best preprocessing technique for the baseline MLP model [34] were chosen in terms of the highest result for the AUC score as well as good fit of the loss learning curve. This dataset will be used in further steps of the experiments to find the best MLP architecture in terms of performance and model complexity. The result of the Step 2 are the combinations of hidden layers and neurons on each layer that show the best performance on the dataset selected in the Step 1. Step 3 introduce alternative feature selection technique to check whether more or less principal components of PCA/ICA will positevely or negatively affect MLP performance. In addition, this step is done to narrow down the list of the best architectures obtained in Step 2 for further experiments. The hyperparameters that show the best results in terms of AUC for each of the MLP architectures from Step 3 were identified as a result of the 4th Step of the Seven-step method. On this stage, when the best MLP architecture and hyperparameters are known the attempt to increase the performance of the previously selected model were done in Step 5 with alternative data preprocessing techniques. During the 6th Step the model optimization is performed by selecting the optimal number of epochs decreasing model's training time and train memory usage. The dynamic learning rate were applied to see if the performance of the MLP models on this stage would increase. Finally, the best MLP model was identified and tested with the 12 differently preprocessed datasets from Step 1, to evaluate it's overall performance [34].

# Chapter 4

# Experiments

The experiments were performed using Python [27] programming language with different libraries such as Keras [39], NumPy [50], Pandas [53], Skit-learn [65], Tensorflow [30], Math [26], etc. The classification models were evaluated using MacBook Air (13-inch, Early 2015) running on macOS Catalina v. 10.15 (19A603) with the CPU processor of 1.6 GHz Dual-Core Intel Core i5, memory (RAM) of 8 GB 1600 MHz DDR3. Moreover, for Stratified K-fold cross-validation the sklearn StratifiedKFold library was used which could be described as follow:

**StratifiedKFold(n_splits=5, *, shuffle=False, random_state=None)** (4.1)

**n_splits (int)** is the number of folds (must be at least 2), **shuffle (bool)** indicates whether to shuffle each class's samples before splitting into batches. **random_state (int)** affects the ordering of the indices, which controls the randomness of each fold for each class.

The number 162 was passed as an argument to the **random_state** for all of the experiments to make them run with the same ordering of indexes.

| Outlier rejection (O) | | | | (O) + Missing values imputation (M) | | | | (O)+(M)+Z standardization (Z) | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| PCA | | ICA | | PCA | | ICA | | PCA | | ICA | |
| 4 | 6 | 4 | 6 | 4 | 6 | 4 | 6 | 4 | 6 | 4 | 6 |
| dataset 1 | dataset 2 | dataset 3 | dataset 4 | dataset 5 | dataset 6 | dataset 7 | dataset 8 | dataset 9 | dataset 10 | dataset 11 | dataset 12 |

Figure 4-1: Twelve differently preprocessed datasets

## 4.1 Step 1: The best dataset

The author of [34] was testing eight different MLP models with 1-8 hidden layers where the number of neurons was a hyperparameter to select the optimum number. The MLP architecture of M = 3 hidden layers (H1, H2,and H3) with N1 = 16, N2 = 64, and N3 = 64 neurons was proposed as the best architecture proposed by [34]. This architecture was chosen as a baseline for this research paper. After outlier rejection (O), missing values imputation (M), and Z-score standardization (Z) the PCA and ICA feature selection techniques were implemented with 4 and 6 principal components each, resulting in 12 differently preprocessed datasets described in Fig. 4-1. The resulted datasets were tested with the baseline model [34] indicated above. The best-preprocessed dataset was chosen based on the resulted value of AUC and the best fit of the learning curve. All results for 12 datasets were recorded and shared in Chapter 5.

## 4.2 Step 2: The best architecture

After the best-preprocessed dataset was identified the experiments proceed to find the best architecture with a specific number of hidden layers and number of neurons on each layer. This was done using the hardcoded grid-search method described earlier. The grid-search was done with the constant hyperparameters proposed in the baseline model [34]. As the baseline model proposed by [34] had 3 hidden layers, the architectures of 1 to 3 hidden layers were tested in this experiment to keep the model as light as the state-of-art. Each layer could have 4, 8, 16, 32, 64, or 128 neutrons. Other numbers of neurons that are not in the power of two were also tested but

yielded unrepresentative results and unpredicted learning curves' behavior. In total, neglecting the unrepresentative experiments, 258 different combinations of layers and neurons were tested on the recently identified best-preprocessed dataset in Step 1. The best 15 architectures were highlighted at this Step based on the AUC score and learning curves' behavior. The difference in the performance of 0.006 AUC of the architecture with the best result for this Step was considered while selecting the best models. Table 5.2 in Chapter 5 illustrates the best architectures selected for this experimental step with their performance results.

## 4.3 Step 3: Alternative number of principal components

After the best combinations of hidden layers and neurons were selected on the previously identified best-preprocessed dataset with constant hyperparameters from [34], the three new datasets were generated under the same best preprocessing technique, but with different feature selection methods. The new datasets of 3,5 and 7 principal components were generated to test if they can perform better than the suggested 4 and 6 principal components from [34]. The best 15 architectures highlighted in the previous step were tested on the 3 new datasets. The performance was recorded and the best 8 architectures out of 15 were selected based on how they have performed on the datasets with 3, 5, 6, and 7 principal components. The priority was given to the models that performed better on the dataset indicated in Step 1 with its number of principal components. The best architectures were also selected based on the AUC score and learning curves' behavior. The difference in the performance of 0.006 AUC of the architecture with the best result was considered while selecting the best models. Table 5.3 in Chapter 5 illustrates the results for the newly selected best 8 architectures.

## 4.4 Step 4: The best hyperparameters

The goal of this experiment was to find out the hyperparameters for the previously selected architectures to increase their performance in terms of AUC score. The hyperparameters such as batch size, learning rate, and percentage of dropout neurons on each layer were grid-searched for the 8 architectures selected in Step 3. The best two architectures were selected and the activation function, optimizer, and the loss function were grid-searched for the two best architectures. 576 architectures with different hyperparameters were tested in this step.

## 4.5 Step 5: Alternative data preprocessing

In this experiment different feature selection techniques based on information gain[7], chi-squared value [56], fisher's score [32], and random forest importance [33]. All feature selection techniques were applied to the dataset without outliers and missing values (O+M). In total, 4 new datasets were obtained with each of the feature selection techniques described above. The best MLP model with one hidden layer and 8 neurons was tested with the 4 new datasets. The best feature selection technique was then used with different data scaling methods. For the data scaling the sklearn.preprocessing [68] module was used with a Normalizer [71], MinMaxScaler [70] , MaxAbsScaler [69], RobustScaler [72], and a StandardScaler [73] data scaling techniques. The results for the data scaling with the best feature selection technique from this experimental step and the architecture and hyperparameters from Step 4 are demonstrated in Table 5.5.

## 4.6 Step 6: More experiments

This Step was dedicated to improving the performance of the finally selected architecture. The dynamic learning rate technique was applied to find out if it positively affects MLP performance. The dynamic learning rate was applied such that at the beginning of the MLP model training, the higher learning rate was used and it was gradually decreased with increasing training epochs. The dynamic learning rate was

a hyperparameter that was tuned manually based on the learning curves' response. In total, 5 different dynamic learning rate algorithms were used to find optimal. Then different optimizers such as SGD [41], RMSprop [41], Adam [41], Adadelta [41], Adagrad [41], Adamax [41], Nadam [41], and Ftrl [41] were tested with the best MLP model from Step 4. Binary Cross-Entropy Loss [40], Hinge Loss [40], and Squared Hinge Loss [40] functions were applied to the MLP model and the results were recorded. Finally, the optimal number of epochs was found for the MLP model that shows the best performance for the number of epochs.

## 4.7 Step 7: Best MLP model

Finally, the best architecture was tested on the 12 differently preprocessed datasets described in Fig. 4-1. The results were recorded and shared in Table 5.7 in the Results section in Chapter 5.

# Chapter 5

# Results

This chapter presents the results of the experiments described in Chapter 4 as well as the results of the data preprocessing.

## 5.1 Results for the preprocessing

The class distribution illustrated in Figure 3-1 shows the difficulty in differentiating the diabetic and non-diabetic patients from PIDD. The majority of the attributes show positive and negative skewness. In addition, the distribution has a wide flatter shape with flatter tails that indicates a leptokurtic [8] behavior or outliers in the dataset. The outlier rejection technique proposed in [34] affect the kurtosis [8] of the data distribution curves.The effect of the outlier rejection technique could be noticed comparing Figure 5-1 and Figure 5-2, where it can be seen that the data with outliers has noticeably flatter tails and a bit flatter shape than the data without outliers.

Table 5.1 describes the number of outliers presented in each attribute and the number of rows that had outliers in the whole dataset. In total, 17% of the data (129/768 cases) with outliers were identified. Table 5.2 show the number of missing values for each attribute as well as the percentage of the data that were changed with the mean imputation technique.

Finally, the 17% of original data were rejected and 9.5% were changed with mean imputation technique after the data preprocessing step.

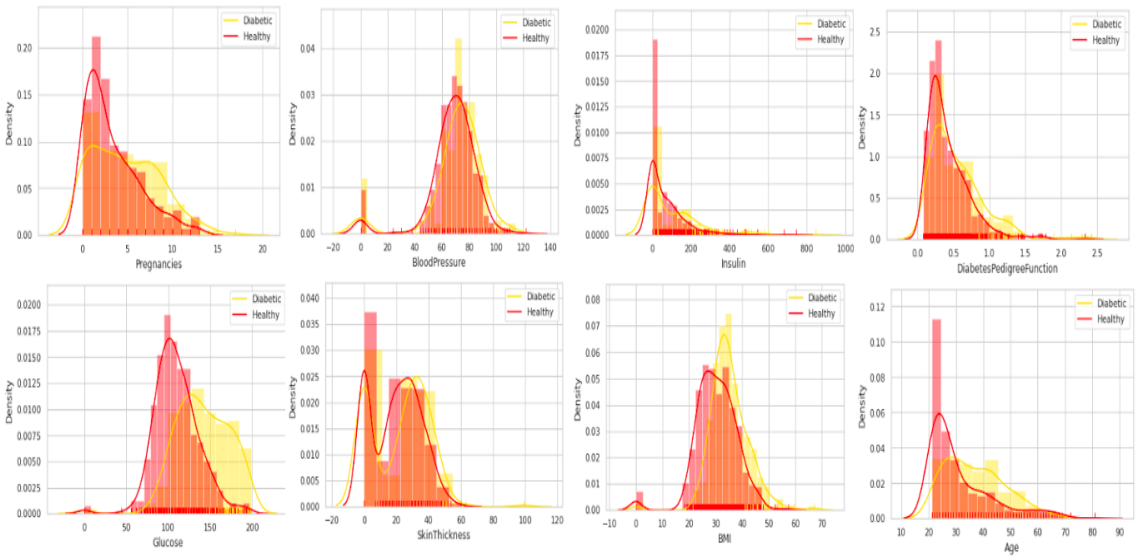


Figure 5-1: The original PIMA Indian Dataset

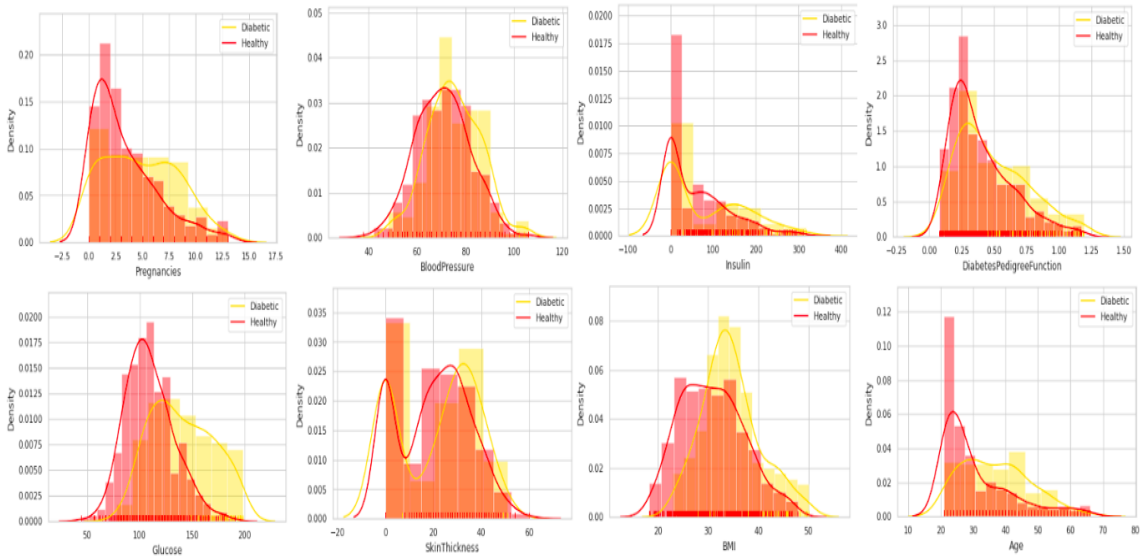


Figure 5-2: The PIMA Indian Dataset without outliers

## 5.2 Seven-step method for the best MLP architecture identification

The baseline MLP architecture of 3 hidden layers with 16, 64, and 64 neurons on each layer respectively was built based on Python Keras library as stated in [34]. The

| Feature name | # outliers |
|---|---|
| Pregnancies | 4 |
| Glucose | 5 |
| BloodPresure | 45 |
| SkinThickness | 1 |
| Insulin | 34 |
| BMI | 19 |
| PedigreeFunctio | 29 |
| Age | 9 |
| Total: 129 rows with outliers (17% of all data) | |

Table 5.1: The number of outliers in each row

| Feature name | # missing values |
|---|---|
| Pregnancies | 0 |
| Glucose | 0 |
| BloodPresure | 0 |
| SkinThickness | 179 |
| Insulin | 307 |
| BMI | 0 |
| PedigreeFunctio | 0 |
| Age | 0 |
| Total | 486 items (9.51% of all data) |

Table 5.2: Missing values after outlier rejection

experiments were conducted with epochs, learning rate, dropout layer, and batch size equal to 200, 0.001, 60%, and 8 respectively. The neurons on the hidden layers were initialized by a normal distribution with ReLU [57] activation function. The authors of [34] did not talk about the optimizer they used in their research, so Adam optimizer were chosen for this experiments.

### 5.2.1 Step1: The best dataset

Table 5.3 illustrates the performance results obtained by replicating the baseline model by [34], where the red and green highlight the worse or better performance of replicated MLP architecture over the results proposed by [34], respectively. From

Table 5-3 it can be seen that the replicated MLP has performed slightly better for the datasets 1, 2, and 3 that were obtained with only outlier rejection and feature selection techniques. However, the baseline model [34] shows a bit better results for the rest of the datasets preprocessed with missing values imputation technique only, missing values imputation combined with z-standardization preprocessing methods. On average the replicated MLP from this literature was worse by 0.04 AUC than the results proposed by [34]. However, Table 5.3 shows that the replicated MLP's performance is lower for the datasets where the ICA feature selection algorithm was used. For the datasets with the PCA feature selection technique, the replicated MLP performed 0.006 AUC worse and 0.065 AUC worse for the datasets preprocessed with the ICA feature selection method.

| Results | Outlier rejection (O) | | | | (O) + Missing values imputation (M) | | | | (O)+(M)+Z standardization (Z) | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PCA | | ICA | | PCA | | ICA | | PCA | | ICA | |
| | 4 | 6 | 4 | 6 | 4 | 6 | 4 | 6 | 4 | 6 | 4 | 6 |
| | dataset 1 | dataset 2 | dataset 3 | dataset 4 | dataset 5 | dataset 6 | dataset 7 | dataset 8 | dataset 9 | dataset 10 | dataset 11 | dataset 12 |
| This literature | 0.800 | 0.818 | 0.797 | 0.793 | 0.802 | 0.821 | 0.796 | 0.801 | 0.832 | 0.889 | 0.801 | 0.802 |
| Baseline model | 0.738 | 0.770 | 0.787 | 0.829 | 0.846 | 0.874 | 0.901 | 0.887 | 0.890 | 0.881 | 0.889 | 0.885 |
| Difference | 0.062 | 0.048 | 0.01 | -0.036 | -0.044 | -0.053 | -0.105 | -0.086 | -0.058 | 0.008 | -0.088 | -0.083 |

Table 5.3: Step 1: The results for the replicated baseline model [34]

In total, replicated MLP model show near the same results as the model proposed by [34] for PCA,and the best performance was recorded for the experiments with dataset 10 preprocessed with (O),(M),(Z), and PCA with 6 principal components. Thus 10th dataset was chosen to be used for the next 6 stages of the experiments.

### 5.2.2 Step2: The best architecture

Table 5.4 shows the performance of the top 15 architectures out of 258 obtained by grid search. It could be noticed that the difference between the performance of top 1 and top 15 architecture is 0.006 AUC. There are 2 MLP architectures with only one hidden layer, 7 architectures with 2 hidden layers, and 6 architectures with 3 hidden layers. The best performance of 0.889 ($\pm$ 0.02) was shown by the MLP with 2 hidden layers, which proves the ability of lighter models to classify diabetes with PIDD as well as deeper MLP models. All of the models from Table 5.4 show good-

| # of hidden layers | Neurons on each layer | Performance (AUC) | Loss curve behavior |
|---|---|---|---|
| 2 | 8-4 | 0.889 | good fit |
| 3 | 4-32-16 | 0.887 | good fit |
| 2 | 16-4 | 0.887 | good fit |
| 2 | 4-8 | 0.886 | good fit |
| 2 | 8-8 | 0.886 | good fit |
| 3 | 8-16-128 | 0.886 | good fit |
| 2 | 8-64-64 | 0.886 | good fit |
| 2 | 8-32 | 0.885 | good fit |
| 3 | 4-128 | 0.885 | good fit |
| 3 | 16-64-16 | 0.885 | good fit |
| 3 | 4-32-32 | 0.884 | good fit |
| 3 | 16-16-64 | 0.884 | good fit |
| 2 | 16-32 | 0.884 | good fit |
| 1 | 4 | 0.883 | good fit |
| 1 | 8 | 0.883 | good fit |

Table 5.4: Step 2: The performance of top 15 architectures with different combinations of hidden layers and neurons on each layer.

fitted learning curves with no overfit, underfit or unpredicted behavior. The example of the good-fitted learning curve is illustrated in Figure 5-3 where fluctuations in training loss are due to the dropout layer.

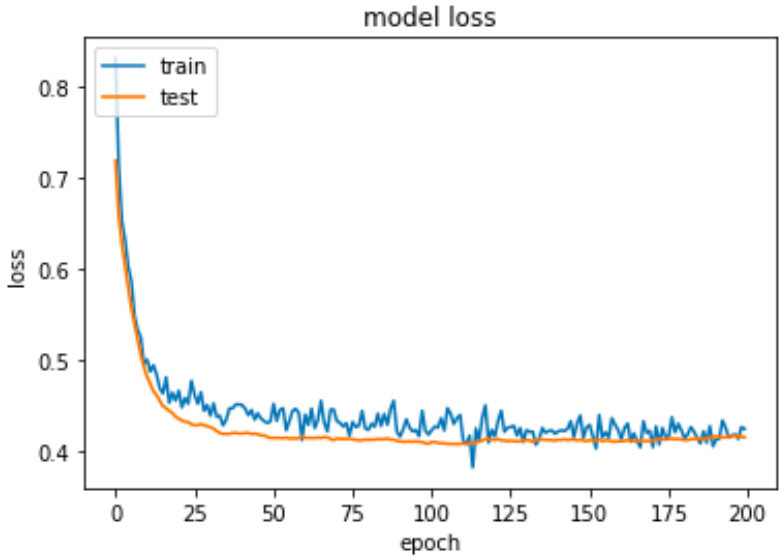


Figure 5-3: The good fitted loss learning curve

### 5.2.3 Step3: Alternative number of principal components

| # of layers | Neurons on each layer | Performance (AUC) PCA 3 | Performance (AUC) PCA 5 | Performance (AUC) PCA 6 | Performance (AUC) PCA 7 | Loss curve behavior |
|---|---|---|---|---|---|---|
| 2 | 8-4 | 0.828 | 0.848 | 0.889 | 0.883 | good fit |
| 3 | 4-32-16 | 0.828 | 0.829 | 0.887 | 0.873 | good fit |
| 2 | 16-4 | 0.832 | 0.841 | 0.887 | 0.888 | good fit |
| 2 | 4-8 | 0.831 | 0.817 | 0.886 | 0.875 | good fit |
| 2 | 8-8 | 0.837 | 0.839 | 0.886 | 0.882 | good fit |
| 3 | 8-16-128 | 0.831 | 0.838 | 0.886 | 0.879 | good fit |
| 3 | 8-64-64 | 0.821 | 0.828 | 0.886 | 0.877 | good fit |
| 2 | 8-32 | 0.834 | 0.839 | 0.885 | 0.886 | good fit |
| 2 | 4-128 | 0.834 | 0.842 | 0.885 | 0.887 | good fit |
| 3 | 16-64-16 | 0.838 | 0.836 | 0.885 | 0.876 | good fit |
| 3 | 4-32-32 | 0.827 | 0.820 | 0.884 | 0.874 | good fit |
| 3 | 16-16-64 | 0.837 | 0.831 | 0.884 | 0.887 | good fit |
| 2 | 16-32 | 0.841 | 0.843 | 0.884 | 0.891 | good fit |
| 1 | 4 | 0.836 | 0.828 | 0.883 | 0.884 | good fit |
| 1 | 8 | 0.840 | 0.840 | 0.891 | 0.888 | good fit |

Table 5.5: Step 3: The performance of top 15 architectures with different number of hidden layers and primcipal components of PCA.

The results of the performance of different MLP architectures on the datasets with alternative number of principal components are presented in Table 5.5. The red, yellow, and green color shadows in Table 5.5 highlight the performance of the same MLP architectures on the datasets with different PCA feature selection method, where red, yellow, and green means the worst, middle, and the best performance respectively. Table 5.5 illustrates that 6 out of 15 architectures show better performance on the datasets where the PCA with 7 principal components was applied, the rest 9 architectures show an increase in its performance with 6 principal components of PCA. All 15 models show worse performance results for PCA with 3 and 5 principal components compared to originally selected PCA with 6 attributes. All of the models show a good fit for the loss curve described in Figure 5-3. In this experiment, the small overfit for 1 out of 5 folds for k-fold cross-validation were accepted. At this stage 8 of 15 architectures with respective feature selection techniques that scored 0.886 AUC or above were selected for further experiments. Finally, 1, 5, and 2 architectures with

1, 2, and 3 hidden layers were selected for further experiments.

### 5.2.4 Step 4: The best hyperparameters

The best hyperparameters for the previously selected dataset from Step 1 and combinations of hidden layers and neurons for architectures from Step 3 are illustrated in Table 5.6. The results show that there are no unique hyperparameters for any model and they vary based on the MLP architecture selected. In general, the hyperparameter tuning did not show any sufficient improvement in the performance of previously selected MLP architectures. Due to the high learning rate and batch size, some experiments yielded AUC below 0.50 with unrepresentative learning curves. The average change in the top performance of 8 architectures was ± 0.005 AUC. All of the architectures score 0.88 AUC or above. However, the 1 hidden layer MLP model achieved 0.90 AUC in this experiment, which is equal to the state-of-art performance. Based on the results of Table 5.6 the model with one hidden layer of 8 neurons was chosen as the best architecture as it demonstrates the best result at this stage.

| # of layers | Neurons on each layer | Performance (AUC) PCA 3 | Batch Size | Learning Rate | Dropout | Loss curve behavior |
|---|---|---|---|---|---|---|
| 1 | 8 | 0.900 | 16 | 0.001 | 60% | good fit |
| 2 | 16-32 | 0.892 | 16 | 0.001 | 60% | good fit |
| 3 | 16-16-64 | 0.838 | 8 | 0.001 | 60% | good fit |
| 2 | 4-128 | 0.884 | 4 | 0.01 | 60% | good fit |
| 2 | 4-8 | 0.887 | 16 | 0.001 | 70% | good fit |
| 2 | 16-4 | 0.887 | 8 | 0.0001 | 50% | good fit |
| 3 | 4-32-16 | 0.888 | 16 | 0.01 | 70% | good fit |
| 2 | 8-4 | 0.886 | 32 | 0.001 | 50% | good fit |

Table 5.6: Best hyperparameters for the 8 architectures selected

### 5.2.5 Step 5: Alternative data preprocessing

Table 5.7 demonstrates the results for the performance of the best architecture from Step 4 on a dataset with alternative feature selection techniques. The results for the chi-squared value and fisher's score are really poor ranging between 0.64 to 0.71 AUC,

the 5 features selected based on information gain show the result of 0.82 AUC, which is still not competitive to the PCA with 6 principal components. So no alternative feature selection techniques to PCA were selected. The data scaling techniques from Table 5.8 show the range of performance from 0.54 to 1.00 AUC for StandardScaler and MaxAbsScaler respectively. However, the graphs for all of the data scaling techniques used in this Step show either huge overfitting which is the result of the perfect performance of MaxAbsScaler, or unrepresentative behavior of training data illustrated in Figure 5-4. Based on the results of Step 5 no changes were applied to the best architecture from Step 4.

| Architecture | Feature selection technique | # of features selected | AUC | Loss curve behavior |
|---|---|---|---|---|
| 1 hidden layer with 8 neurons | information gain | 4 | 0.789 | fits |
| | information gain | 5 | 0.824 | fits |
| | fisher's score | 4 | 0.640 | unrepresentative |
| | fisher's score | 5 | 0.712 | unrepresentative |
| | chi-squared | 4 | 0.643 | unrepresentative |
| | chi-squared | 5 | 0.545 | unrepresentative |

Table 5.7: The results for best architecture with alterantive feature selection techniques

| Architecture | Data Scaler | AUC | Loss curve behavior |
|---|---|---|---|
| 1 hidden layer with 8 neurons | StandardScaler | 0.545 | unrepresentative |
| | MaxAbsScaler | 1.000 | unrepresentative |
| | MinMaxScaler | 1.000 | unrepresentative |
| | RobustScaler | 0.606 | unrepresentative |
| | Normalizer | 1.000 | unrepresentative |

Table 5.8: Step 5: The results for (O + M) PIDD scaling with sklearn.preprocessing module.

### 5.2.6 Step 6: More experiments

In this experiment, the dynamic learning rate was a hyperparameter that was searched manually based on the learning curves' response. So almost 5 different combinations of the dynamic learning rate were used to find optimal. However, the dynamic learning rate did not sufficiently improve the performance of the MLP from previous steps, the average improve for the model from Step 4 was -0.001 AUC, so it was decided not to use in further steps. Table 5.9 shows the results for applying different optimizers to the selected MLP architecture. The RMSprop, Adamax, and Nadam optimizers show a competitive results of 0.889 AUC, 0.881 AUC, and 0.885 AUC respectively. The Hinge Loss and Squared Hinge Loss appeared not being applicable to the binary classification of the PIDD as they all show unrepresentative loss learning curves, so the Binary Cross-Entropy Loss was remaining the loss function for the MLP model. Analyzing the loss learning curves of the MLP with 1 hidden layer with 8 neurons the conclusion was made that the number of epochs between 50 and 75 is optimal for the classification model training which can be seen from Figure 5-3 also. Finally, the best architecture of 1 hidden layer and 8 neurons has achieved a performance of 0.90 AUC with 50 epochs.

| Architecture | Optimizer | AUC | Loss curve behavior |
|---|---|---|---|
| 1 hidden layer with 8 neurons | SGD | 0.875 | good fit |
| | RMSprop | 0.889 | good fit |
| | Adam | 0.897 | good fit |
| | Adadelta | 0.561 | unrepresentative |
| | Adagrad | 0.624 | unrepresentative |
| | Adamax | 0.881 | good fit |
| | Nadam | 0.887 | good fit |
| | Ftrl | 0.854 | unrepresentative |

Table 5.9: Step 6: Applying various optimizers with the best MLP model

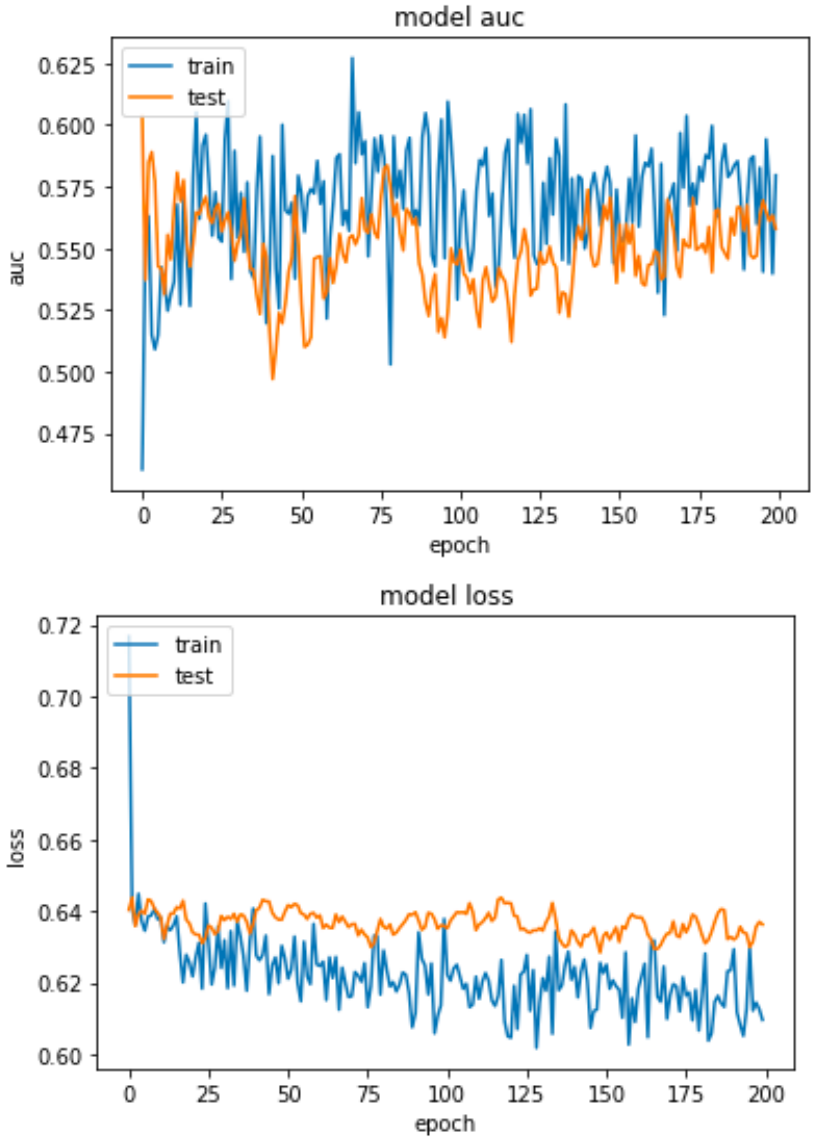


Figure 5-4: Unrepresentative behaviour of MLP training on the data scaled with sklearn.preprocessing.StandardScaler

### 5.2.7 Step 7: Best MLP model

Table 5.10 shows the performance results obtained by the selected best MLP model (BE) versus the Baseline model (BA) and Replicated baseline model (RE) from this literature. The baseline model consists of 3 hidden layers with 16, 64, and 64 neurons on each hidden layer respectively and trained with 200 epochs, while the experimentally identified best model consists of only one hidden layer with 8 neurons trained with 50 epochs only. In addition, Table 5.10 shows the difference in the performance

| Results | Outlier rejection (O) | | | | (O) + Missing values imputation (M) | | | | (O)+(M)+Z standardization (Z) | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PCA | | ICA | | PCA | | ICA | | PCA | | ICA | |
| | 4 | 6 | 4 | 6 | 4 | 6 | 4 | 6 | 4 | 6 | 4 | 6 |
| | dataset 1 | dataset 2 | dataset 3 | dataset 4 | dataset 5 | dataset 6 | dataset 7 | dataset 8 | dataset 9 | dataset 10 | dataset 11 | dataset 12 |
| Reolicated model (R) | 0.800 | 0.818 | 0.797 | 0.793 | 0.802 | 0.821 | 0.796 | 0.801 | 0.832 | 0.889 | 0.801 | 0.802 |
| Baseline model (Ba) | 0.738 | 0.770 | 0.787 | 0.829 | 0.846 | 0.874 | 0.901 | 0.887 | 0.890 | 0.881 | 0.889 | 0.885 |
| Best model (Be) | 0.805 | 0.829 | 0.805 | 0.833 | 0.807 | 0.832 | 0.804 | 0.834 | 0.841 | 0.901 | 0.807 | 0.841 |
| Difference (Be-R) | 0.005 | 0.011 | 0.008 | 0.04 | 0.005 | 0.011 | 0.008 | 0.033 | 0.009 | 0.012 | 0.006 | 0.039 |
| Difference (Be-Ba) | 0.067 | 0.059 | 0.018 | 0.004 | -0.039 | -0.042 | -0.097 | -0.053 | -0.049 | 0.020 | -0.082 | -0.044 |
| Difference (R-Ba) | 0.062 | 0.048 | 0.01 | -0.036 | -0.044 | -0.053 | -0.105 | -0.086 | -0.058 | 0.008 | -0.088 | -0.083 |

Table 5.10: Step 7: The results for the Best MLP model versus Baseline and Replicated baseline model

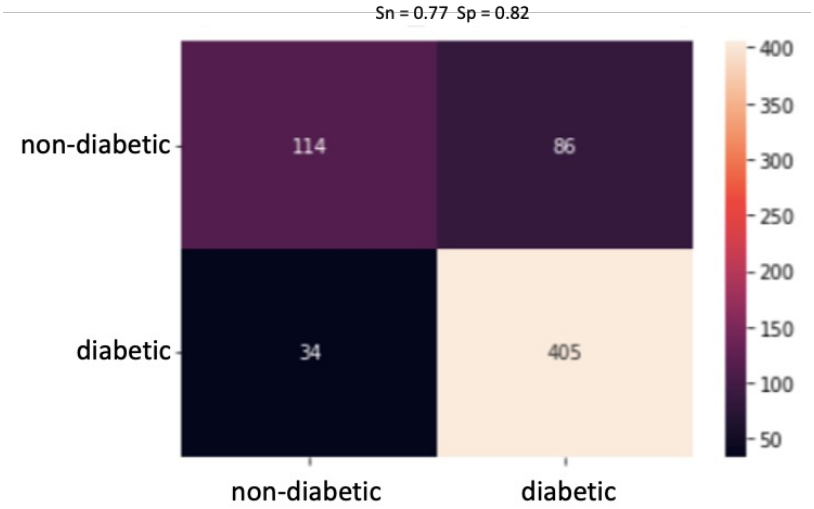


Figure 5-5: Step 7: The confusion matrix for the performance of the Best MLP with dataset 10

of BE, BA, and RE where the red and green highlight if the model performed worse or better than the other model. It can be seen that the newly identified BE with one hidden layer show better performance, on all 12 datasets, than the recently replicated MLP model from [34] with 3 hidden layers. Moreover, it shows better results than the baseline model for the datasets 1, 2, 3, and 4 preprocessed with only outlier rejection and ICA/PCA feature selection method and dataset 10. On average BE is better than RE by 0.02 AUC and worse than BA by 0.02 AUC which is better than the average difference in the performance of RE versus BA by 0.045 AUC. However, Table 5.10 still shows the lower performance of RE and BE versus BA for the ICA feature selection technique. For ICA the models perform 0.065 AUC and 0.042 AUC worse on average for RE and BE over BA, and for PCA the results were 0.006 worse and

| | Baseline model (BA) | Model proposed in this work (BE) |
|---|---|---|
| Number of hidden layers | 3 | 1 |
| Total number of neurons | 144 | 8 |
| Training time | 174 sec | 31 sec |
| Number of trainamble parameters | 5425 | 65 |
| Memory usage | 21.95 kb | 0.26 kb |
| Performance | 0.90 AUC | 0.90 AUC |

Table 5.11: Step 7: The comparison table of the BA versus BE

0.003 AUC better for RE and BE over BA. For the top performance BE scored 0.02 AUC better for PCA with 6 principal components than BA. The confusion matrix for the performance of BE with dataset 10 is reported in Figure 5-5. Table 5.11 shows the comparison of BA and BE. The training time was measured on the local machine consisting of the training time of the MLP model and the time to output learning curves. The memory usage was estimated from the trainable parameters of the MLP architecture [15]. From the Table 5.11, it can be seen that the BE is almost 84 times more efficient in terms of training memory consumption and more than 5 times faster in training than BA.

# Chapter 6

# Discussion

### Preprocessing

The results for the prepossessing part in this literature are the same as the results of [34]. This indicates that the outlier rejection, missing values imputation, and z-score standardization were perfectly employed.

### Step 1: The best dataset

The results for Step 1 were controversial. Despite the MLP model was assembled identically to the baseline literature [34], based on Keras, with 3 hidden layers and 16, 64, and 64 neurons activated with ReLU activation function, initialized with normal distribution and trained with the same hyperparameters described in [34] the results are almost similar for the PIDD preprocessed with PCA feature selection technique with a small difference between Ba and Be, but hugely different for the datasets preprocessed with ICA. Therefore, if the Be model is correct, the problem might be in ICA. Yet the ICA feature selection technique is done with sklearn.FastICA [66] which implementation is based on the same source [36] the [34] is referencing when implementing ICA in their work. Even the standard deviation for each of the 12 datasets proposed by [34] does not cover the range given by the difference between Be and Ba. This may question the validity of the results reported by [34], where the authors reported the best performance of the MLP model on ICA datasets.

**Step 2: The best architecture**

The best combination of 15 architectures was reported in Table 5.2. The results are such that there are only 6 out of 15 models of 3 hidden layers which are the minority. This may also question the results reported in [34] where the 3 hidden layer models were claimed to show the best performance for AUC among models from 1 to 8 hidden layers.

**Step 3: Alternative number of principal components**

In Step 3 the PCA with the different number of principal components was used to find out if the PCA with 4 and 6 principal components proposed by [34] gives the best results for MLP classification with the AUC metric. Table 5.3 shows that out of 15 architectures selected in Step 2, 8 architectures show better results for the PCA with 6 principal components (PCA 6) which is the majority. This indicates that PCA 6 is the best feature selection technique among other PCA alternatives.

**Step 4: The best hyperparameters**

Step 4 illustrates that each architecture of the different number of hidden layers and neurons on each layer has its values for the best hyperparameters. Yet the batch size of 128 and/or learning rate of 0.1 and above almost always gave insufficient results when applying with PIDD. In addition, the dropout above 60% has also resulted in unpredictable and unrepresentative learning curves of MLP models. The results of Step 4 demonstrate that the MLP models with only one hidden layer and proper hyperparameters tuning can show better performance than the models with more hidden layers for PIDD. At this stage, the final best-performing model was chosen and the rest of the experimental steps were dedicated to boosting its performance.

**Step 5: Alternative data preprocessing**

Unfortunately, Step 5 where alternative data preprocessing techniques were employed to boost the performance of MLP did not give any sufficient results. Surprisingly,

applying the PCA feature selection technique to the previously scaled dataset with sklearn.preprocessing algorithms give unrepresentative results. The experiments with the scaled data without feature selection techniques were done apart from this literature's experiments show low results with unrepresentative learning curves.

**Step 6: More experiments**

The dynamic learning rate applied in the 6th Step of a seven-step method for the best MLP architecture identification did not give any results. However, applying different optimizers to the MLP shows the potential for better performance in RMSprop, Nadam, and Adamax. Despite this optimizers were not properly tuned they show competitive results to our best MLP model. This step shows that the MLP model described in this literature may show a competitive to state-of-art performance with only one hidden layer and 50 epochs for model training.

# Chapter 7

# Limitations, Future work and Conclusion

## 7.1 Limitations and Future work

Despite the number of experiments done in this literature, there is a huge room for further work to improve the performance of MLP for diabetes prediction. One of the main limitations of this work is the way the dataset was preprocessed for MLP classification. The data normalization and PCA feature selection were applied to the entire dataset which leads to data leakage and more optimistic estimates of the final performance. Yet this research is aimed to prove that lighter MLP can perform better than deeper models, the future work should consider applying normalization and PCA feature selection to the training dataset only, to avoid optimistic estimates of the performance. Another limitation of this work is that it covers the experiments on 1 to 3 hidden layers with 4 to 128 neurons on each layer. The experiments from this literature prove that the architecture of 3 hidden layers with 16, 64, and 64 neurons proposed by [34] was not the best performing architecture. Therefore for future experiments, deeper MLP models with more hidden layers might be tested, because better AUC results for MLP may be worth increasing model complexity. Further work requires more powerful hardware than it was used in this literature, as running grid-search may take a lot of time. This experiment covers the usage of different opti-

mizers, yet only with the learning rate as hyperparameters with other parameters set to default. In addition, this literature covers only 5 different dynamic learning rate implementations, in the future more algorithms of the dynamic learning rate might be tried. This work is limited to the experiments with Multilayer Perceptron, so for future work, other Artificial Neural Networks like Residual Neural Network. The data provided by PIDD seems not enough to solve a real-world diabetes prediction problem, so other datasets with more data may be used to train the classification model. In this research, extensive experiments were performed on various hyperparameters and architectures. The data on every experiment was collected and put into a machine-friendly dataset. This dataset may be used to find the relationship between hyperparameters like batch size, dropout, and learning rate and to test it on other datasets, instead of once again performing the experiments described in this paper. This may optimize future research on diabetes prediction with Neural Networks. Finally, if this and further research would be considered for building real-life software for diabetes prediction the assistance from a qualified medical specialist is required, as in most of the literature, the PIDD is preprocessed from the machine learning point of view, rather than real life where it is not appropriate to perform a mean imputation, since equating the attributes of 51% of samples (patients) to common/mean value without considering their basic features (such as age, BMI, gender, etc.) introduces huge deviations and inaccuracies in the final result.

## 7.2 Conclusion

In this literature, extensive experiments were performed to find the best MLP model for diabetes prediction from Pima Indian Diabetes Dataset. The lightweight framework of Multilayer Perceptron with only one hidden layer and 8 neurons that show the best performance among 1 to 3 hidden layer models was proposed by this study. The outlier rejection, missing values imputation, z-score standardization, PCA, and ICA feature selection were implemented to preprocess the dataset. Extensive experiments were done to find out the best architecture of MLP based on the optimal number of

hidden layers and neurons. Hyperparameters tuning procedure has boosted the MLP model to achieve the performance of 0.90 AUC, which is the same performance as the state-of-art, but more than 5 times faster in training and almost 84 times more efficient in memory usage. The lightweight framework proposed in this study can be applied to the medical industry to make a real-life diabetes prediction.

# Bibliography


[1] Mani Abedini, Anita Bijari, and Touraj Banirostam. Classification of pima indian diabetes dataset using ensemble of decision tree, logistic regression and neural network. *Intern JAdvan Res CompCommunEngin*, 9:1–5, 2020.

[2] S Abirami and P Chitra. Energy-efficient edge based real-time healthcare support system. In *Advances in Computers*, volume 117, pages 339–368. Elsevier, 2020.

[3] Suja A Alex, J Nayahi, H Shine, and Vaisshalli Gopirekha. Deep convolutional neural network for diabetes mellitus prediction. *Neural Computing and Applications*, pages 1–9, 2021.

[4] Raj Anand, Vishnu Pratap Singh Kirar, and Kavita Burse. K-fold cross validation and classification accuracy of pima indian diabetes data set using higher order neural network and pca. *Int. J. Soft Comput. Eng*, 2(6):436–438, 2013.

[5] Davide Anguita, Luca Ghelardoni, Alessandro Ghio, Luca Oneto, and Sandro Ridella. The 'k'in k-fold cross validation. In *20th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)*, pages 441–446. i6doc. com publ, 2012.

[6] American Diabetes Association. Economic Costs of Diabetes in the U.S. in 2017. *Diabetes Care*, 41(5):917–928, 03 2018.

[7] B Azhagusundari, Antony Selvadoss Thanamani, et al. Feature selection based on information gain. *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*, 2(2):18–21, 2013.

[8] Kevin P Balanda and HL MacGillivray. Kurtosis: a critical review. *The American Statistician*, 42(2):111–119, 1988.

[9] Rashi Bansal, Nishant Gaur, and Shailendra Narayan Singh. Outlier detection: applications and techniques in data mining. In *2016 6th International conference-cloud system and big data engineering (Confluence)*, pages 373–377. IEEE, 2016.

[10] Nahla Barakat, Andrew P Bradley, and Mohamed Nabil H Barakat. Intelligible support vector machines for diagnosis of diabetes mellitus. *IEEE transactions on information technology in biomedicine*, 14(4):1114–1120, 2010.

[11] Sofia Benbelkacem and Baghdad Atmani. Random forests for diabetes diagnosis. In *2019 International Conference on Computer and Information Sciences (ICCIS)*, pages 1–4. IEEE, 2019.

[12] PeterH Bennett, ThomasA Burch, and Max Miller. Diabetes mellitus in american (pima) indians. *The Lancet*, 298(7716):125–128, 1971.

[13] Sourav Kumar Bhoi et al. Prediction of diabetes in females of pima indian heritage: a complete supervised learning approach. *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, 12(10):3074–3084, 2021.

[14] Roshan Birjais, Ashish Kumar Mourya, Ritu Chauhan, and Harleen Kaur. Prediction and diagnosis of future diabetes risk: a machine learning approach. *SN Applied Sciences*, 1(9):1–8, 2019.

[15] Nicolas Boullé, Yuji Nakatsukasa, and Alex Townsend. Rational neural networks. *Advances in Neural Information Processing Systems*, 33:14243–14253, 2020.

[16] Andrew P Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern recognition*, 30(7):1145–1159, 1997.

[17] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[18] Richard M Brugger. A note on unbiased estimation of the standard deviation. *The American Statistician*, 23(4):32–32, 1969.

[19] J.-L. Chiasson and Remi Rabasa-Lhoret. Prevention of Type 2 Diabetes: Insulin Resistance and Beta-Cell Function. *Diabetes*, $53(\text{suppl}_3) : S34 - -S38, 122004$.

[20] Dilip Kumar Choubey, Sanchita Paul, Santosh Kumar, and Shankar Kumar. Classification of pima indian diabetes dataset using naive bayes with genetic algorithm as an attribute selection. In *Communication and computing systems: proceedings of the international conference on communication and computing system (ICCCS 2016)*, pages 451–455, 2017.

[21] Manish; Shukla Vaibhav; Choubey, Dilip K.; Kumar. Comparative Analysis of Classification Methods with PCA and LDA for Diabetes. *Current Diabetes Reviews*, 16(8):833–850, 2020.

[22] Denis Cousineau and Sylvain Chartier. Outliers detection and treatment: a review. *International Journal of Psychological Research*, 3(1):58–67, 2010.

[23] Dictionary.com and Oxford University Press. Overfitting, 2022.

[24] Ashok Kumar Dwivedi. Analysis of computational intelligence techniques for diabetes mellitus prediction. *Neural Computing and Applications*, 30(12):3837–3845, 2018.

[25] Nesreen Samer El_Jerjawi and Samy S Abu-Naser. Diabetes prediction using artificial neural network. *International Journal of Advanced Science and Technology*, 121, 2018.

[26] Python Software Foundation. math — mathematical functions, 2022.

[27] Python Software Foundation. Python, 2022.

[28] Matt W Gardner and SR Dorling. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. *Atmospheric environment*, 32(14-15):2627–2636, 1998.

[29] David B Geselowitz. On the theory of the electrocardiogram. *Proceedings of the IEEE*, 77(6):857–876, 1989.

[30] Google. Tensorflow, 2022.

[31] John W Graham. Missing data analysis: Making it work in the real world. *Annual review of psychology*, 60:549–576, 2009.

[32] Quanquan Gu, Zhenhui Li, and Jiawei Han. Generalized fisher score for feature selection. *arXiv preprint arXiv:1202.3725*, 2012.

[33] Md Al Mehedi Hasan, Mohammed Nasser, Shamim Ahmad, and Khademul Islam Molla. Feature selection for intrusion detection using random forest. *Journal of information security*, 7(3):129–140, 2016.

[34] Md Kamrul Hasan, Md Ashraful Alam, Dola Das, Eklas Hossain, and Mahmudul Hasan. Diabetes prediction using ensembling of different machine learning classifiers. *IEEE Access*, 8:76516–76531, 2020.

[35] John J Hopfield. Artificial neural networks. *IEEE Circuits and Devices Magazine*, 4(5):3–10, 1988.

[36] Aapo Hyvärinen and Erkki Oja. Independent component analysis: algorithms and applications. *Neural networks*, 13(4-5):411–430, 2000.

[37] Aiswarya Iyer, S Jeyalatha, and Ronak Sumbaly. Diagnosis of diabetes using classification mining techniques. *arXiv preprint arXiv:1502.03774*, 2015.

[38] Ram D Joshi and Chandra K Dhakal. Predicting type 2 diabetes using logistic regression and machine learning approaches. *International Journal of Environmental Research and Public Health*, 18(14):7346, 2021.

[39] Keras.io. Keras, 2022.

[40] Keras.io. Losses, 2022.

[41] Keras.io. Optimizers, 2022.

[42] Akram T Kharroubi and Hisham M Darwish. Diabetes mellitus: The epidemic of the century. *World journal of diabetes*, 6(6):850, 2015.

[43] Damjan Krstajic, Ljubomir J Buturovic, David E Leahy, and Simon Thomas. Cross-validation pitfalls when selecting and assessing regression and classification models. *Journal of cheminformatics*, 6(1):1–15, 2014.

[44] Amit Kumar Dewangan and Pragati Agrawal. Classification of diabetes mellitus using machine learning techniques. *International Journal of Engineering and Applied Sciences*, 2(5):257905, 2015.

[45] Saloni Kumari, Deepika Kumar, and Mamta Mittal. An ensemble approach for classification and prediction of diabetes mellitus using soft voting classifier. *International Journal of Cognitive Computing in Engineering*, 2:40–46, 2021.

[46] Charles X Ling, Jin Huang, Harry Zhang, et al. Auc: a statistically consistent and more discriminating measure than accuracy. In *Ijcai*, volume 3, pages 519–524, 2003.

[47] Manish Mathuria. Decision tree analysis on j48 algorithm for data mining. *Intrenational Journal ofAdvanced Research in Computer Science and Soft-ware Engineering*, 3(6), 2013.

[48] AS Miller, BH Blott, et al. Review of neural network applications in medical imaging and signal processing. *Medical and Biological Engineering and Computing*, 30(5):449–464, 1992.

[49] Moeketsi Ndaba, Anban W Pillay, and Absalom E Ezugwu. An improved generalized regression neural network for type ii diabetes classification. In *International Conference on Computational Science and Its Applications*, pages 659–671. Springer, 2018.

[50] NumPy. Numpy, 2022.

[51] National Institute of Diabetes, Digestive, and Kidney Diseases. National institute of diabetes and digestive and kidney diseases, 20222.

[52] World Health Organization. Diabetes, 2021.

[53] The pandas development team. Pandas, 2022.

[54] S Patro and Kishore Kumar Sahu. Normalization: A preprocessing stage. *arXiv preprint arXiv:1503.06462*, 2015.

[55] Sajida Perveen, Muhammad Shahbaz, Karim Keshavjee, and Aziz Guergachi. Metabolic syndrome and development of diabetes mellitus: predictive modeling based on machine learning techniques. *IEEE Access*, 7:1365–1375, 2018.

[56] Nachirat Rachburee and Wattana Punlumjeak. A comparison of feature selection approach between greedy, ig-ratio, chi-square, and mrmr in educational mining. In *2015 7th international conference on information technology and electrical engineering (ICITEE)*, pages 420–424. IEEE, 2015.

[57] Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

[58] C Radhakrishna Rao. The use and interpretation of principal component analysis in applied research. *Sankhyā: The Indian Journal of Statistics, Series A*, pages 329–358, 1964.

[59] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[60] You S. and Minsoo K. CA Study on Methods to Prevent Pima Indians Diabetes Using SVM. *Korea Journal of Artificial Intelligence*, 8(2):7–10, 12 2020.

[61] Deepti Sisodia and Dilip Singh Sisodia. Prediction of diabetes using classification algorithms. *Procedia Computer Science*, 132:1578–1585, 2018. International Conference on Computational Intelligence and Data Science.

[62] Deepti Sisodia and Dilip Singh Sisodia. Prediction of diabetes using classification algorithms. *Procedia computer science*, 132:1578–1585, 2018.

[63] R Sivanesan and K Devika Rani Dhivya. A review on diabetes mellitus diagnoses using classification on pima indian diabetes data set. *International Journal of Advance Research in Computer Science and Management Studies*, 5(1), 2017.

[64] Skit-learn. 3.1. cross-validation: evaluating estimator performance, 2022.

[65] Skit-learn. Scikit-learn machine learning in python, 2022.

[66] Skit-learn. sklearn.decomposition.fastica, 2022.

[67] Skit-learn. sklearn.model$_s$*election.gridsearchcv*, 2022.

[68] Skit-learn. sklearn.preprocessing: Preprocessing and normalization, 2022.

[69] Skit-learn. sklearn.preprocessing.maxabsscaler, 2022.

[70] Skit-learn. sklearn.preprocessing.minmaxscaler, 2022.

[71] Skit-learn. sklearn.preprocessing.normalizer, 2022.

[72] Skit-learn. sklearn.preprocessing.robustscaler, 2022.

[73] Skit-learn. sklearn.preprocessing.standardscaler, 2022.

[74] Jack W Smith, James E Everhart, WC Dickson, William C Knowler, and Robert Scott Johannes. Using the adap learning algorithm to forecast the onset of diabetes mellitus. In *Proceedings of the annual symposium on computer application in medical care*, page 261. American Medical Informatics Association, 1988.

[75] G Swapna, R Vinayakumar, and KP Soman. Diabetes detection using deep learning algorithms. *ICT express*, 4(4):243–246, 2018.

[76] Sidong Wei, Xuejiao Zhao, and Chunyan Miao. A comprehensive exploration to the machine learning techniques for diabetes identification. In *2018 IEEE 4th World Forum on Internet of Things (WF-IoT)*, pages 291–295. IEEE, 2018.

[77] Dewey Lonzo Whaley III. *The interquartile range: Theory and estimation.* PhD thesis, East Tennessee State University, 2005.

[78] Louis E Yelle. The learning curve: Historical review and comprehensive survey. *Decision sciences*, 10(2):302–328, 1979.

[79] Xinchuan Zeng and Tony R Martinez. Distribution-balanced stratified cross-validation for accuracy estimation. *Journal of Experimental & Theoretical Artificial Intelligence*, 12(1):1–12, 2000.

[80] Changsheng Zhu, Christian Uwa Idemudia, and Wenfang Feng. Improved logistic regression model for diabetes prediction by integrating pca and k-means techniques. *Informatics in Medicine Unlocked*, 17:100179, 2019.

[81] Rahmat Zolfaghari. Diagnosis of diabetes in female population of pima indian heritage with ensemble of bp neural network and svm. *Int. J. Comput. Eng. Manag*, 15:2230–7893, 2012.

[82] Quan Zou, Kaiyang Qu, Yamei Luo, Dehui Yin, Ying Ju, and Hua Tang. Predicting diabetes mellitus with machine learning techniques. *Frontiers in genetics*, page 515, 2018.