We seek to determine whether a person is or isn’t afflicted by diabetes (irrespective of the type), using various physiological measurements.

In order to form the dataset, we have combined three different datasets, from the following links. Each of them presents the same predictors.

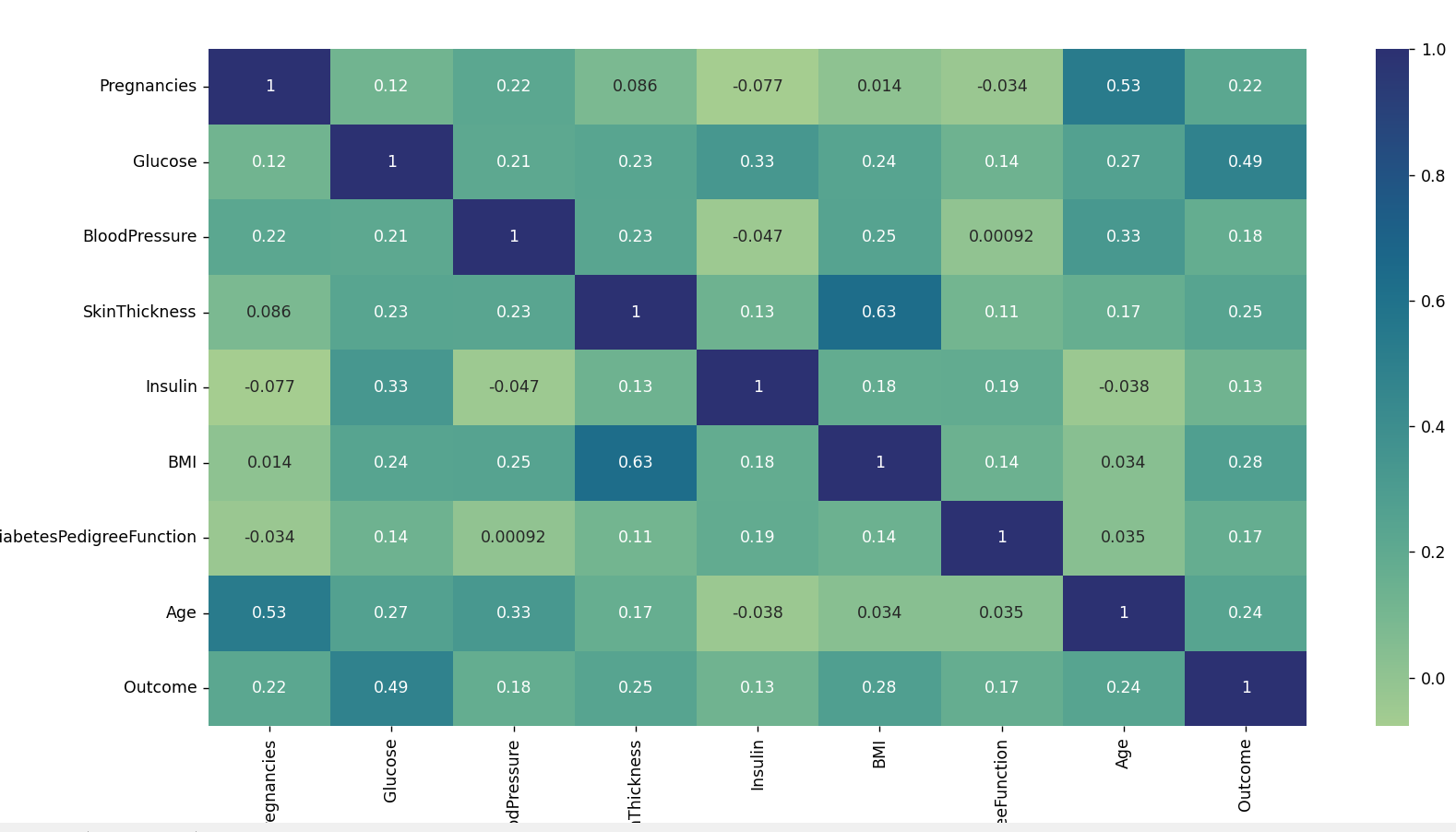
<https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database>

<https://www.kaggle.com/datasets/johndasilva/diabetes>

<https://www.kaggle.com/datasets/nanditapore/healthcare-diabetes/discussion>

However, after combining them into one dataset, after dropping the duplicates, the number of samples is reduced from 5539 to 778. It is likely that the datasets had samples deliberately inserted multiple times. From thereon, the obtained results use the dataset with the duplicate samples removed

Given that some of the predictors, more precisely 'Glucose', 'BloodPressure', 'SkinThickness', 'BMI' had values that were equal to 0, which is illogical, we have analyzed three different scenarios when testing the performance for the model. In the first case, we left the data as it were, and in the other cases, we calculated the mean\median values for each predictor, after splitting the dataset in patients that have diabetes and patients that do not, then recombined the dataframes into one. This was done in order to, hopefully, obtain more accurate results.



We provided the correlation matrix. We can see that the amount of multicollinearity present is fairly low, with the exception of SkinThickness and BMI, and the age and the number of pregnancies of the patient.

In order to measure the performance of the model, we used the accuracy, precision, recall, f1 and the area under the roc curve. Scores are provided for both the best score obtained, across the k-folds, along with the average scores.

In the case of both models, stratified kfolds has been used as part of the performance evaluation. This is a variation of KFold that returns stratified folds. The folds are made by preserving the percentage of samples for each class. This is important, because in the dataset the numbers of patients that suffer from diabetes is equal to roughly half of those that do not.

For the linear model, these are the results for the case where no data replacement has taken place, for any of the predictors. A seed of value 42 has been used, in order to guarantee the reproducibility of the results. The “metric index” refers to the k-fold in which the best value has been observed. For cross validation, 10-fold cross-validation was used.

We assume that the classes can be separated using a line, defined by the following function:

Where w is the weight vector, x is the input vector, and b is the bias term.

These are the test results.

accuracy: 0.8333333333333334

accuracy index: 7

precision: 0.85

precision index: 7

recall: 0.6296296296296297

recall index: 7

f1: 0.723404255319149

f1 index: 7

roc\_auc: 0.9353667392883078

roc\_auc index: 7

Mean accuracy: 0.7634865134865134

Mean precision: 0.7185322280059123

Mean recall: 0.5476190476190476

Mean f1: 0.6178884978765136

Mean roc\_auc: 0.8225593941280215

We can see that there is a considerable difference between the average scores and the best scores. This can happen as a result of the small number of samples (778).

We can see that the recall score of the model is very low, which means that the number of patients that are successfully “diagnosed” with diabetes is unsatisfactory. However, the model has a slightly better precision score, which tells us that the number of false negatives is lower than that of false positives.

The F1 score is really low, which translates into overall poor performance for the model.

For the case when the values are replaced with the mean, these are the results. 289 instances had at least one of the predictor variables mentioned, with a value of 0.

accuracy: 0.7948717948717948

accuracy index: 5

precision: 0.8

precision index: 7

recall: 0.6296296296296297

recall index: 9

f1: 0.6666666666666666

f1 index: 5

roc\_auc: 0.915

roc\_auc index: 7

Mean accuracy: 0.7596070596070595

Mean precision: 0.7086046176046177

Mean recall: 0.544047619047619

Mean f1: 0.6113621264774056

Mean roc\_auc: 0.8397539682539682

We can see that the results are lower, across both the best scores in the k-folds, and the average scores obtained. This could be the result of the fact that, especially in the case of the people with diabetes, the values of some physiological characteristics are outliers with a high amount of variation, which cannot offer a reliable analysis using a simple machine learning model.

For the case when the values are replaced with the median, these are the results.

accuracy: 0.7948717948717948

accuracy index: 5

precision: 0.8

precision index: 7

recall: 0.6296296296296297

f1: 0.6666666666666667

f1 index: 9

roc\_auc: 0.9142857142857144

roc\_auc index: 7

Mean accuracy: 0.7621878121878122

Mean precision: 0.7131046176046175

Mean recall: 0.5477513227513227

Mean f1: 0.6156583621566201

Mean roc\_auc: 0.8391056644880175

The scores are marginally higher than in the previous case, but they are extremely similar, and still lower than in the first case. We can infer from this that the median and mean values of the metrics have very similar values:

|  | **Glucose** | **Blood Pressure** | **Skin Thickness** | **BMI** |
| --- | --- | --- | --- | --- |
| **No diabetes (Mean)** | 111.10 | 70.72 | 27.49 | 31.15 |
| **Diabetes (Mean)** | 142.27 | 75.37 | 33.26 | 35.41 |
| **No diabetes (Median)** | 108.0 | 70.0 | 27.0 | 30.4 |
| **Diabetes (Median)** | 140.0 | 74.5 | 32.5 | 34.3 |

We can see that is the case. The values are rounded to two decimal places for the mean values.

In the case of the neural network, we have theorized that the overall performance of the model will be lower than the one for the linear regression, due to the reduced number of samples. Due to this, we have used 5-fold cross validation instead of 10-fold CV, in order to partially mitigate this issue.

The neural network used has the following architecture

* An input layer with 8 neurons
* Three hidden layers, the first two with 12 neurons and the last with 8 neurons
* An output layer with one neuron

The Leaky Rectified Linear Unit activation function allows a small gradient when the unit is not active, i.e., it allows for a small negative slope (the default hyperparameter value of 0.01 was used) when the input is less than zero. It was used instead of normal ReLU in order to avoid the “dead ReLU” problem (the situation in training a neural network where some of the neurons in the network become inactive and only output 0 for any input.)

This architecture is a feed-forward neural network with 3 hidden layers. The LeakyReLU activation function is used to add non-linearity after each linear transformation except for the last one, where a Sigmoid activation function is used for binary classification.

The learning hypothesis for the neural network can be written as such:

 (for some reason, after uploading the document, the equation wouldn’t render, thus I have also provided a screenshot of it)

Where:

W1, W2, W3, and W4 are the weight matrices for each layer.

b1, b2, b3, and b4 are the bias vectors for each layer.

σ is the activation functions, Leaky ReLU and the Sigmoid.

x is the input vector.

The loss function used is Binary Cross Entropy, the optimizer is the AdamW variant of Adam, with a learning rate of 0.01.

The number of epochs used is 200, with a batch size of 16

Due to the way computations are handled on a GPU, reproducibility cannot be achieved by setting a seed value.

Below we can see the average results over 5 runs, of each used metric, for every way of handling the missing values (rounded to the 4th decimal):

| **Metric** | **Missing values Unchanged** | **Missing Values replaced with the mean** | **Missing values replaced with the median** |
| --- | --- | --- | --- |
| Best accuracy | 0.7645 | 0.8045 | 0.8295 |
| Best precision | 0.7315 | 0.8129 | 0.7934 |
| Best recall | 0.6148 | 0.7364 | 0.7887 |
| Best f1 | 0.6262 | 0.7264 | 0.7526 |
| Best roc\_auc | 0.7159 | 0.7890 | 0.8087 |
| Mean accuracy | 0.7391 | 0.7751 | 0.7622 |
| Mean precision | 0.6571 | 0.7192 | 0.6873 |
| Mean recall | 0.5459 | 0.6269 | 0.6069 |
| Mean f1 | 0.5921 | 0.6476 | 0.6401 |
| Mean roc\_auc | 0.6944 | 0.7363 | 0.7312 |

We can see that replacing the missing values with their mean gives us the best results, especially when compared with the case where the missing values are left unchanged. When comparing the case when the values are replaced with mean and the one in which they are changed with the median, the difference in performance are only marginally in favor of the first case.

Below we’ve provided the average values over 5 runs, for each metric, in order to compare the performance when using the AdamW/Adam optimizers

| **Metric** | **AdamW (Median)** | **AdamW (Mean)** | **AdamW (Unchanged)** | **Adam (Median)** | **Adam (Mean)** | **Adam (Unchanged)** |
| --- | --- | --- | --- | --- | --- | --- |
| Accuracy | 0.7648 | 0.7956 | 0.7584 | 0.7649 | 0.7519 | 0.7545 |
| Precision | 0.6732 | 0.7153 | 0.7038 | 0.6624 | 0.7338 | 0.7003 |
| Recall | 0.6508 | 0.6950 | 0.5439 | 0.6993 | 0.4637 | 0.5298 |
| F1 Score | 0.6585 | 0.7046 | 0.6113 | 0.6717 | 0.5612 | 0.6014 |
| ROC AUC | 0.7386 | 0.7724 | 0.7087 | 0.7499 | 0.6854 | 0.7087 |

The AdamW model with missing values replaced with the mean has the overall best performance. It has the highest scores in accuracy (0.7956), recall (0.6950), F1 score (0.7046), and ROC AUC (0.7724). However, when it comes to precision then the Adam model with missing values replaced with the mean would be the best as it has the highest precision (0.7338).

In summary, AdamW performs better in most categories when missing values are replaced with the mean, while Adam performs better when missing values are replaced with the median.

However, it is likely that, due to the small number of samples, the values provided by the model are too volatile to be able to draw a conclusion with a high degree of certainty.

When comparing the neural network and the logistic regression models, we compared the best, overall performing scenario for both models, meaning the scenario when the missing values are unchanged, for the logistic regression, and the scenario when the missing values are replaced with the mean, for the neural network:

Mean Accuracy: The neural network has a higher mean accuracy (0.7751) compared to Logistic regression (0.7635).

Mean Precision: The mean precision is almost the same for both models, with the second (0.7192) being slightly higher than the first (0.7185). This indicates that both models are almost equally precise in predicting people without diabetes.

Mean Recall: The NN has a significantly higher mean recall (0.6269) than the Logistic regression (0.5476). This means that the NN is better at identifying all positive instances (people with diabetes).

Mean F1 Score: The mean F1 score is higher for the neural network (0.6476) compared to logistic regression (0.6179). The higher F1 score for the NN suggests that it has a better overall performance.

Mean ROC\_AUC: The LR has a higher mean ROC\_AUC (0.8226) than the NN (0.7363). A higher ROC\_AUC score for the regression suggests that it is better at distinguishing between positive and negative instances.