CSE454

Data Mining

Final project

171044095 – Djuro Radusinovic

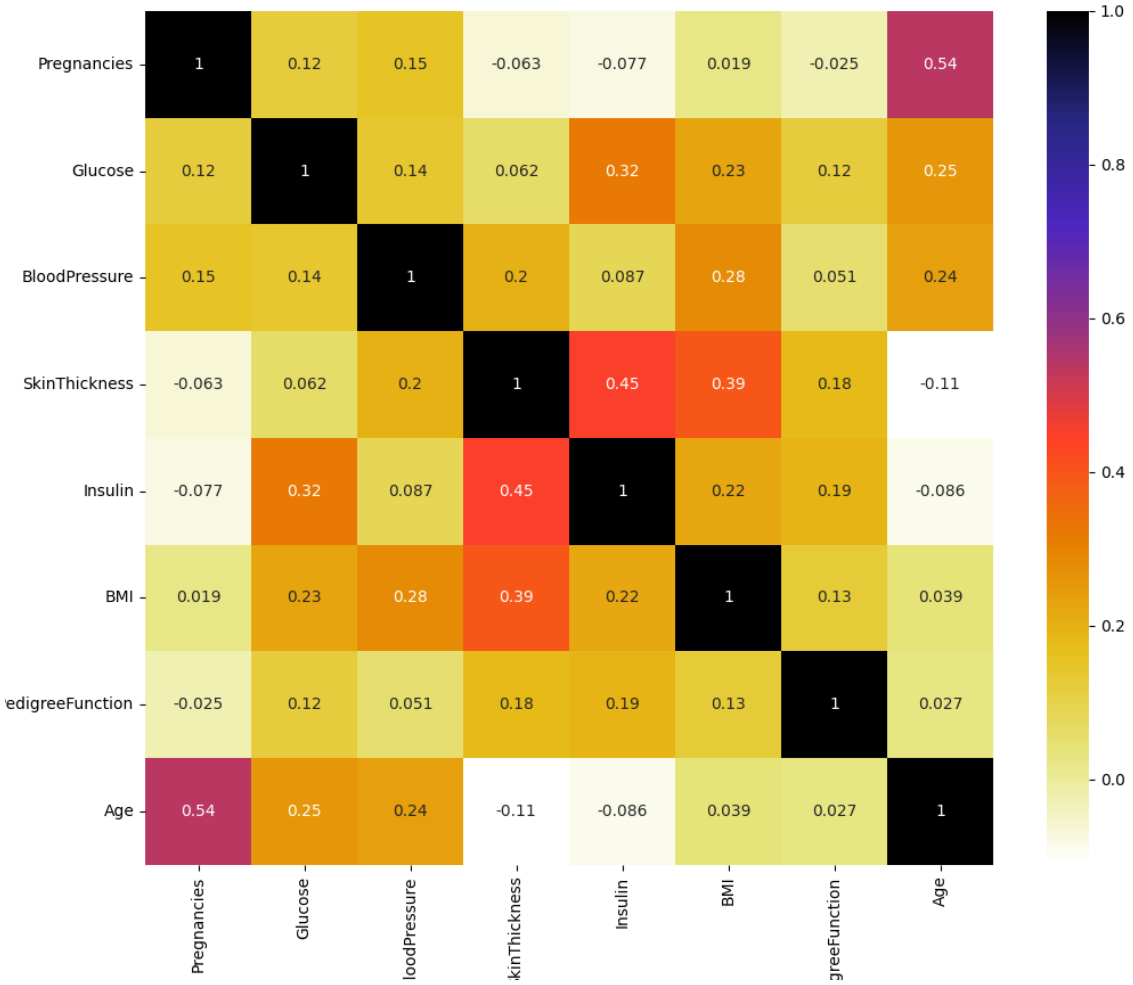
In this project I have chosen perform various classification algorithms and compare them. The main topic of this project is the analysis of the diabetics according to the data from the given dataset I have found on Kaggle.

Dataset

I have found and used a dataset I have downloaded from Kaggle. This dataset has 2000 columns for each feature it gives. In order to perform necessary operations I firstly performed many Preprocessing tasks .

One of the first Things I had to do was to solve the 'Curse of dimensionality'. Too many unnecessary features were given. I order to do that I used Feature Selection with respect to correlation factor each pair of features would give. I removed the ones that correlate to avoid duplicate usage and used a Threshold factor that would give the best results.

This results in reducing the dimensionality to only 2 factors which were in the end Insulin and age as the ones most important features we had to take into account.



Feature selection after finishing its job concluded that the 3 most important features are Age, BMI and Insulin and I have been working with them after that.

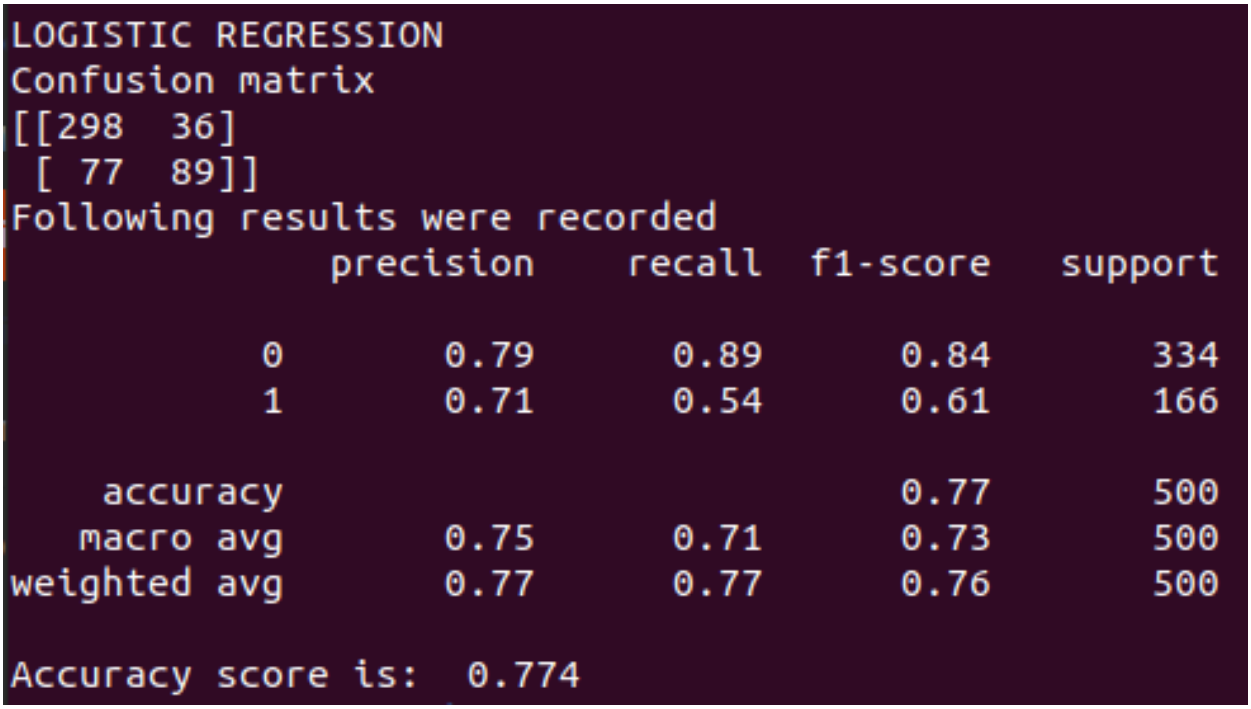
Classification algorithms used and tested:

- Linear regression
- K-NN
- SVM
- Kernel SVM
- Naïve Bayes

- Decision Tree
- Random Forest
- PERCEPTRON (self-implemented)

For each of these algorithms I tested them and got the results for their accuracy, precision, recall, F-score and support by using values from the confusion matrix in order to get these values.

Linear Regression:



Why I decided to use Linear Regiression?

Linear Regression is an ML algorithm used for supervised learning. Linear regression performs the task to predict a dependent variable(target) based on the given independent variable(s). So, this regression technique finds out a linear relationship between a dependent variable and the other given independent variables. Hence, the name of this algorithm is Linear Regression. Another thing

prefeormed in data preprocessing is Feature Scaling in order to make data compatible with each other.

Pros of Linear Regression:

- Linear Regression is simple to implement.
- Less complexity compared to other algorithms.
- Linear Regression may lead to over-fitting but it can be avoided using some dimensionality reduction techniques, regularization techniques, and cross-validation.

Cons:

- Outliers affect this algorithm badly.
- It over-simplifies real-world problems by assuming a linear relationship among the variables, hence not recommended for practical use-cases.

KNEARESTNEIGBOUR:

```
KNEARESTNEIGHBOUR
Confusion matrix
[[297  37]
 [ 56 110]]
Following results were recorded
              precision    recall  f1-score   support

           0       0.84      0.89      0.86       334
           1       0.75      0.66      0.70       166

    accuracy                           0.81       500
   macro avg       0.79      0.78      0.78       500
weighted avg       0.81      0.81      0.81       500

Accuracy score is:  0.814
```

The k-nearest neighbors (KNN) algorithm is a simple, easy-to-implement supervised machine learning algorithm that can be used to solve both classification and regression problems.

PROS

- It is extremely easy to implement
- As said earlier, it is lazy learning algorithm and therefore requires no training prior to making real time predictions. This makes the KNN algorithm much faster than other algorithms that require training e.g SVM, linear regression, etc.
- Since the algorithm requires no training before making predictions, new data can be added seamlessly.
- There are only two parameters required to implement KNN i.e. the value of K and the distance function (e.g. Euclidean or Manhattan etc.)

CONS

- The KNN algorithm doesn't work well with high dimensional data because with large number of dimensions, it becomes difficult for the algorithm to calculate distance in each dimension.
- The KNN algorithm has a high prediction cost for large datasets. This is because in large datasets the cost of calculating distance between new point and each existing point becomes higher.
- Finally, the KNN algorithm doesn't work well with categorical features since it is difficult to find the distance between dimensions with categorical features.

SVM

```
SVM
Confusion matrix
[[294  40]
 [ 74  92]]
Following results were recorded
              precision    recall  f1-score   support

           0       0.80      0.88      0.84       334
           1       0.70      0.55      0.62       166

    accuracy                           0.77       500
   macro avg       0.75      0.72      0.73       500
weighted avg       0.77      0.77      0.76       500

Accuracy score is:  0.772
```

SVM or Support Vector Machine is a linear model for classification and regression problems. It can solve linear and non-linear problems and work well for many practical problems. The idea of SVM is simple: The algorithm creates a line or a hyperplane which separates the data into classes.

PROS

- It’s more efficient in spaces which are high dimensional. Makes adequate use of memory.
- Suited for classes that have clear separation margins. Also works well when total dimensions are more than samples.

CONS

- Doesn’t work well when every data point's total features exceed total training data samples. Also for larger data sets, it’s algorithm isn’t suited.
- When target classes overlap or there’s more noise in a data set, it starts to lag. Plus there’s a lack of probabilistic explanation for support vector classifier's classification.

KERNEL-MODE SVM

Linear SVM is a parametric model, but an RBF kernel SVM isn’t, so the complexity of the latter grows with the size of the training set. Not only is more expensive to train an RBF kernel SVM, but you also have to keep the kernel matrix around, and the projection into this “infinite” higher dimensional space where the data becomes linearly separable is more expensive as well during prediction. Furthermore, you have more hyperparameters to tune, so model selection is more expensive as well! And finally, it’s much easier to overfit a complex model.

```
SVM KERNEL
Confusion matrix
[[312  22]
 [ 67  99]]
Following results were recorded
              precision    recall  f1-score   support

           0       0.82      0.93      0.88       334
           1       0.82      0.60      0.69       166

    accuracy                           0.82       500
   macro avg       0.82      0.77      0.78       500
weighted avg       0.82      0.82      0.81       500

Accuracy score is:  0.822
```

We can observe that SVM-Kernel is 5% higher in accuracy with defaulted paramters C, gamma, etc.

NAIVE-BAYES

```
NAIVE BAYES
Confusion matrix
[[285  49]
 [ 72  94]]
Following results were recorded
              precision    recall  f1-score   support

           0       0.80      0.85      0.82       334
           1       0.66      0.57      0.61       166

    accuracy                           0.76       500
   macro avg       0.73      0.71      0.72       500
weighted avg       0.75      0.76      0.75       500

Accuracy score is:  0.758
```

Naive Bayes is a probabilistic algorithm that's typically used for classification problems. Naive Bayes is simple, intuitive, and yet performs surprisingly well in many cases. For example, spam filters Email app uses are built on Naive Bayes

PROS

- Even though the naive assumption is rarely true, the algorithm performs surprisingly good in many cases
- Handles high dimensional data well. Easy to parallelize and handles big data well
- Performs better than more complicated models when the data set is small

CONS

- The estimated probability is often inaccurate because of the naive assumption. Not ideal for regression use or probability estimation
- When data is abundant, other more complicated models tend to outperform Naive Bayes

DECISION TREE

```
DECISIONTREE
Confusion matrix
[[325   9]
 [  4 162]]
Following results were recorded
              precision    recall  f1-score   support

           0       0.99      0.97      0.98       334
           1       0.95      0.98      0.96       166

    accuracy                           0.97       500
   macro avg       0.97      0.97      0.97       500
weighted avg       0.97      0.97      0.97       500

Accuracy score is:  0.974
```

Decision tree is a type of algorithm that includes conditional ‘control’ statements to classify data. A decision tree starts at a single point (or ‘node’) which then branches (or ‘splits’) in two or more directions. Each branch offers different possible outcomes, incorporating a variety of decisions and chance events until a final outcome is achieved.

PROS

- Good for interpreting data in a highly visual way.
- Good for handling a combination of numerical and non-numerical data.
- Easy to define rules, yes, no, if, then, else
- Requires minimal preparation or data cleaning before use.
- Great way to choose between best, worst, and likely case scenarios.
- Can be easily combined with other decision-making techniques.

CONS

- Overfitting (where a model interprets meaning from irrelevant data) can become a problem if a decision tree’s design is too complex.

- They are not well-suited to continuous variables (i.e. variables which can have more than one value, or a spectrum of values).
- In predictive analysis, calculations can quickly grow cumbersome, especially when a decision path includes many chance variables.
- When using an imbalanced dataset (i.e. where one class of data dominates over another) it is easy for outcomes to be biased in favor of the dominant class.

RANDOM FOREST

```
Random Forest
Confusion matrix
[[332   2]
 [ 22 144]]
Following results were recorded
              precision    recall  f1-score   support

           0       0.94      0.99      0.97       334
           1       0.99      0.87      0.92       166

    accuracy                           0.95       500
   macro avg       0.96      0.93      0.94       500
weighted avg       0.95      0.95      0.95       500

Accuracy score is:  0.952
```

Random Forest is a Supervised Machine Learning classification algorithm. In supervised learning, the algorithm is trained with labeled data that guides you through the training process. The main advantage of using a Random Forest algorithm is its ability to support both classification and regression

PROS

- Random Forest algorithm is less prone to overfitting than Decision Tree and other algorithms

- Random Forest algorithm outputs the importance of features which is a very useful

CONS

- Random Forest algorithm is less prone to overfitting than Decision Tree and other algorithms
- Random Forest algorithm outputs the importance of features which is a very useful

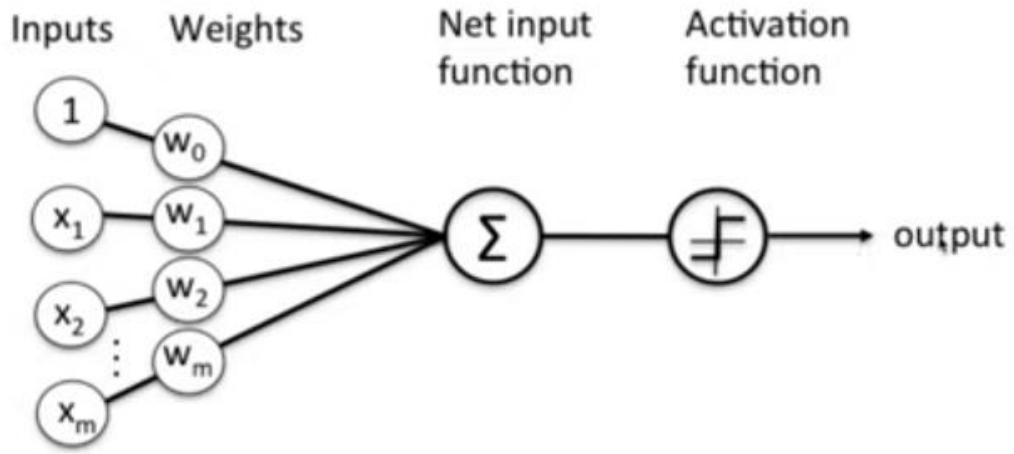
Implementation algorithm, preferably not from the class

PERCEPTRON

For this part I decided to implement Perceptron. The reason for this is that we didn't talk about it during class though we did talk about neural networks a bit . Next semester there is a Machine learning course I might take so I wanted to research more about it this way. I used Perceptron as a linear binary classifier.

In machine learning, the perceptron is an algorithm for supervised learning of binary classifiers. A binary classifier is a function which can decide whether or not an input, represented by a vector of numbers, belongs to some specific class. It is a type of linear classifier, a classification algorithm that makes its predictions based on a linear predictor function combining a set of weights with the feature vector.

Here is the model I implemented in here. Following excerpts are form



For the unit step function I used the following in order to represent my linear model:

$$g(\mathbf{z}) = \begin{cases} 1 & \text{if } \mathbf{z} \geq \theta \\ -1 & \text{otherwise.} \end{cases}$$

where

$$\mathbf{z} = w_1 x_1 + \cdots + w_m x_m = \sum_{j=1}^{m} x_j w_j$$
$$= \mathbf{w}^T \mathbf{x}$$

Also, for output approximation I followed this rule:

Approximation

$\hat{y} = g(f(w, b)) = g(w^T x + b)$

I used the following for weights update

$w := w + \Delta w$

$\Delta w := \alpha \cdot (y_i - \hat{y}_i) \cdot x_i$

The accuracy was behind other algorithms but changing the activation function and similar should be able to increase its accuracy.

```
PERCEPTRON
Confusion matrix
[[255  79]
 [ 66 100]]
Following results were recorded
              precision    recall  f1-score   support

           0       0.79      0.76      0.78       334
           1       0.56      0.60      0.58       166

    accuracy                           0.71       500
   macro avg       0.68      0.68      0.68       500
weighted avg       0.72      0.71      0.71       500

Accuracy score is:  0.71
```

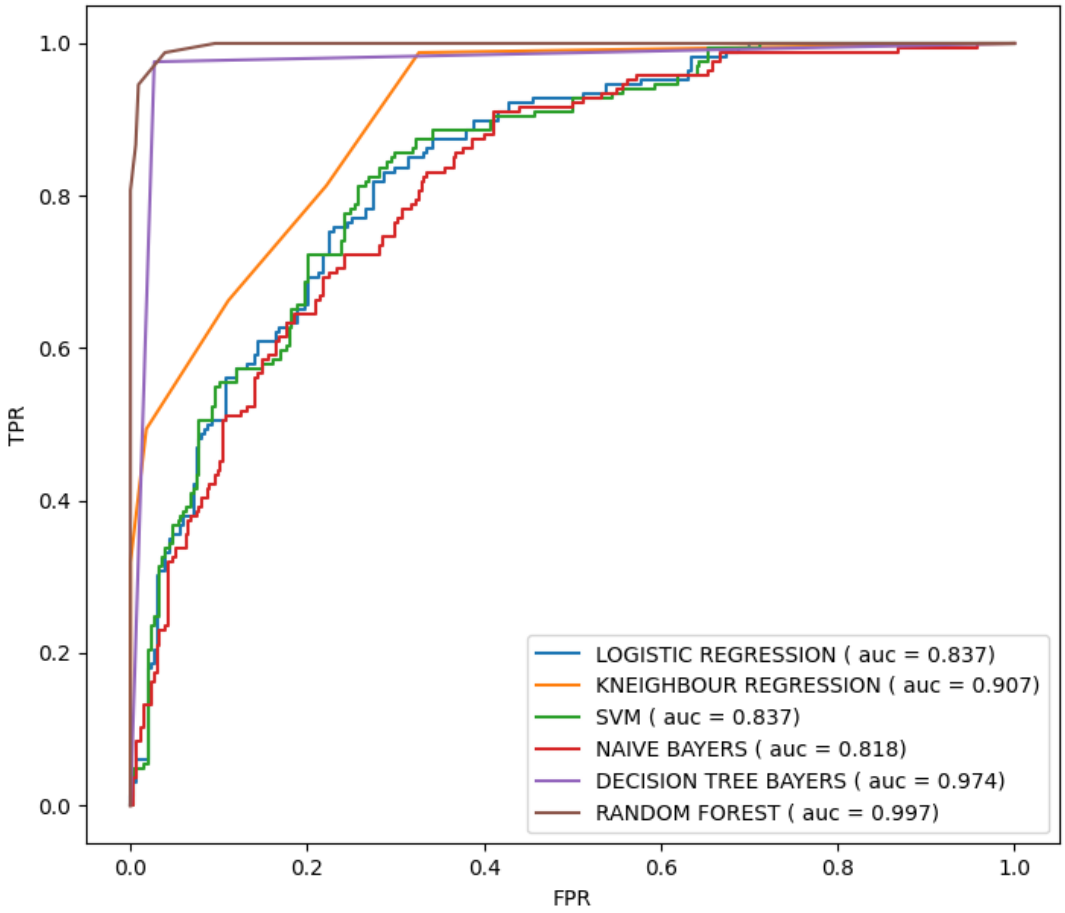
PROS

- Works well on linear data

CONS

- Works only well with linear data though it is possible to implement a multi-layered perceptron that will be able to learn to work well even with the non-linear data

CONCLUSIONS



ROC – CURVE FOR ALL THE ALGORITHMS MENTIONED ABOVE

We can see that best performer for this dataset were Random Forest followed by Decision tree. This was not expected as Decision tree usually falls behind other classification algorithms and is more of a tool for other algorithms. This is why it is important to notice and understand that ensemble techniques are very important in Data Mining since intuition isn't always the best tool for choosing the dataset. There are too many factors involved and that's why it is best to combine several classification algorithms before and perform at least Bagging, If possible even Boosting since that will provide the best results because there is no classification algorithm that works best on every dataset there is.