1 Introduction

Every year thousands of pregnancies end in death [1] and many more in health complications. Risk factors to health complications include, for example, age and existing health conditions like diabetes or high blood pressure. [2] Being able to predict who may need extra care or surveillance during their pregnancy could potentially save lives. Simple measures like blood pressure, heart rate, blood sugar and the patient's age can help with predicting whether the patient is at risk or not. This is the goal of this machine learning application.

In this project section 1 is the introduction, and the problem is formulated into a machine learning problem in section 2. Section 3 (Methods) introduces the two machine learning algorithms used in this project and the necessary data preprocessing. In section 4 (Results) the results of these two algorithms are shown. Section 5 (Conclusions) contains these results' conclusions, and references are listed in section 6 (References).

2 Problem Formulation

In this project, data points are different pregnant patients, and their health information includes age, blood pressure, blood sugar, heart rate and body temperature. What we are interested in is whether a patient is at risk of health complications during pregnancy. This means the labels are low risk of health complication during pregnancy and risk of health complications during pregnancy.

3 Methods

3.1 Data and Preprocessing

The data set used in this project is Maternal Health Risk Data Set from UCI Machine Learning Repository [5]. It contains 1014 data points and there are no missing values. The data set has the following columns: 1. Age (patient's age, years), 2. SystolicBP (systolic blood pressure, mmHg), 3. DiastolicBP (diastolic blood pressure, mmHg), 4. BS (blood glucose levels, mmol/l), 5. BodyTemp (body temperature, F), 6. HeartRate (resting heart rate, beats per minute), 7. RiskLevel (risk level intensity)

The columns from one to six contain numerical data. BS and BodyTemp are floats and Age, SystolicBP, DiastolicBP and HeartRate are integers. I used all these columns as features because from a medical perspective the information in all of them can be significant when determining the risk intensity of health complications for a pregnant patient. [2]

The data type of RiskLevel is string and this column contains the labels. It can take three categorical values which are "low risk", "mid risk" and "high risk". Machine learning algorithms don't work with categorical values in the form of strings, so I converted these values to integers.

I converted "high risk" and "mid risk" to 1 and "low risk" to 0. This means the labels are in binary categorical values: 0 and 1. 0 means low risk of health complications during pregnancy and 1 means risk of health complications during pregnancy. After combining the two categories together, category 0 has 406 entries and category 1 has 608 entries.

I decided to simply split the data randomly into training, validation, and test sets. The optimal split for a data set is usually found with trial and error. [6] I compared a couple of simple splits by comparing the training and validation errors calculated with logistic loss on different splits (see section 3.2). I chose the ratio of 60-20-20 for the training, validation, and test sets, respectively. This split is simple, and it seemed fine. I also used this ratio because I wanted to have an adequate amount of data for training but also a good amount for validation and testing. For splitting the data, I used the function train_test_split from sklearn.model_selection module [4].

The features were on different scales which is why I used standardization for the features. For standardization I used class StandardScaler from sklearn.preprocessing package [4].

3.2 Logistic regression

I chose logistic regression as the first machine learning method because it is suitable for binary classification [4]. It is also simple to set up and train, which makes it a desirable choice for a beginner project. Logistic regression learns a linear hypothesis that works as a hyperplane. Data points that fall on either side of the hyperplane or the decision boundary are assigned to different classes.

As a loss function, I used logistic loss:

$$L_{\log}(y, p) = -(y \log(p) + (1 - y) \log(1 - p))$$

, where y is the true label (0,1) and p is the probability that y=1

I used logistic loss because it is normally used for logistic regression [3] and because it is the default function in sklearn.linear_model module for class LogisticRegression [4]. This function was used for learning linear hypothesis and computing the training and validation errors.

For measuring how well the model performed I also used F1-score. For computing it I used f1_score from sklearn.metrics [4], This is firstly because it is easier to interpret than logistic loss. Second, the data is a bit imbalanced. Other measures like accuracy, recall and precision might give misleading results. The third reason is that when determining a patient's health risk, we want to minimize both false positives and negatives, which is why F1-score is suitable for determining model performance.

3.3 K-nearest neighbors

In k-nearest neighbors (KNN) a label is assigned to a data point based on the k number of data points closest to it. If we introduce a new data point, the k data points closest to the new data point are found. Then the label which has the most data points near the new data point is predicted as the label of the new data. Different metrics can be used for the distance between data points, and I chose Minkowski metric because it is the default for distance in sklearn.neighbors [4].

I chose KNN as my second machine learning method because it can be used for classification problems. It is also usable on my data because distances between data points can be calculated. [7] I also chose this method, like logistic regression because it is suitable for a beginner project for its simplicity. Further, KNN is not too hard to understand, and you only need to choose the optimal value for k and a metric for distances.

I used 0/1 loss:

$$L(\hat{y}, y) = I(\hat{y} \neq y)$$

, where the value of function I is 1 if the predicted labels is the same as true label and 0 if it isn't

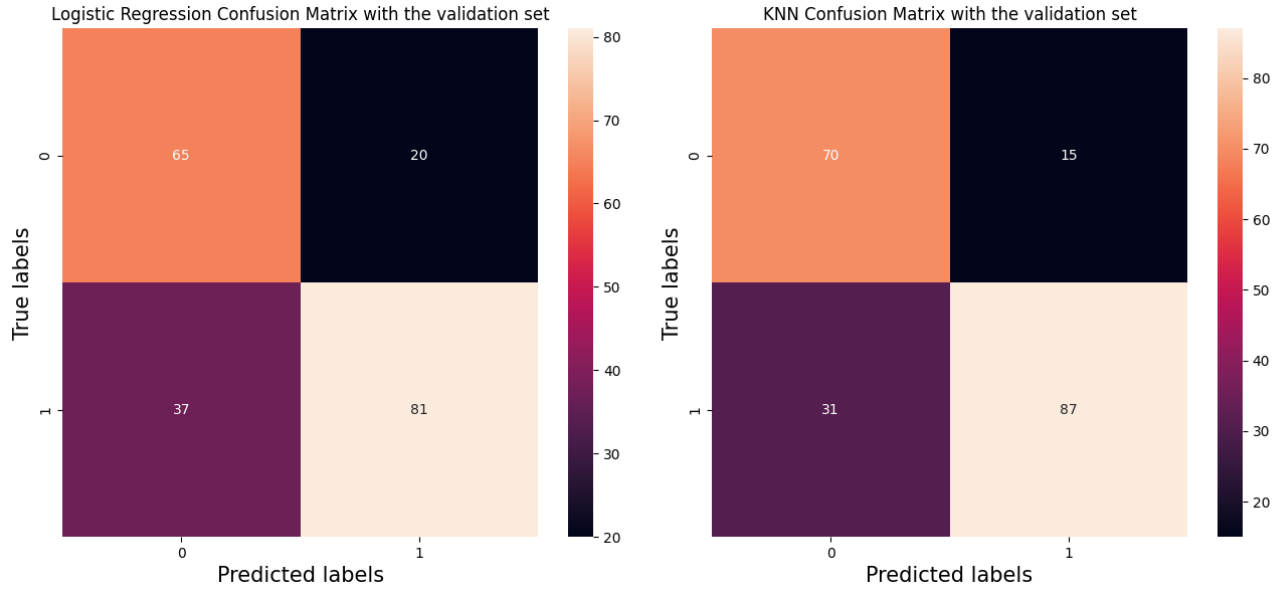
I used this loss because it is commonly used with k-nearest neighbors. I used function zero_one_loss from module sklearn.metrics module [4] to compute the loss.

For measuring the model performance, I used F1-score for the same reasons as with logistic regression. Also, 0/1 loss isn't easily compared with logistic loss or F1-score.

The optimal value for k is found with trial and error. I tried different values from three to 15 and with 6 the validation error was the lowest, so I chose k = 6.

4 Results

Confusion matrices:



Both models tended to give more false negatives than false positives.

Results with logistic regression:

```
Training error: 9.714157113294451
Validation Error: 9.698225810910474
F1-score: 0.7397260273972603
```

Surprisingly, the training error is higher than the validation error and this was the case with all the data splits I tried. The F1-score shows that the model did mostly make correct predictions but also plenty of wrong ones.

Results with KNN:

```
Training error: 0.16447368421052633
Validation Error: 0.2216748768472906
F1-score: 0.7999999999999999
```

The validation error is higher than the training error typical for a machine learning algorithm. This model does also make wrong predictions as we can tell from the F1-score and confusion matrix, but it performed better than logistic regression. Therefore, KNN is the final chosen method.

Results for the test with KNN:

```
F1-score: 0.8
Test error: 0.24630541871921185
```

Test error is a bit higher than validation error which is expected. The F1-score is close to the previous score. Confusion matrices for the test set for both machine learning methods are provided with the code.

5 Conclusions

I tried two machine learning methods for predicting the risk intensity of health complications for pregnant patients. The methods were k-nearest neighbors and logistic regression. I made the necessary modifications for the data and computed training errors, validation errors and F1-scores with both methods. By comparing the training errors, validation errors and F1-scores I was able to determine that KNN gives better results. Both models still had a lot of room for improvement as they both made wrong predictions.

The most obvious setback was that the dataset was not large, so I was not expecting significant or usable results as this was a beginner project. The results tell us that neither of the methods did an excellent job, but KNN was less off, and the results were more expected.

Second, the first method, logistic regression, wasn't the best for this data. Decision tree or some other non-linear classification method might have yielded better results. Logistic regression assumes that the data can be separated with a linear separator which isn't often the case. Logistic regression is also a method with high bias which means it tends to overfit. The results suggested that both problems were present in this project.

KNN yielded better results which makes sense as it doesn't have the same limits as logistic regression regarding the data. F1-scores with validation set and test set show that the model is quite consistent. I think using k-fold cross validation would have yielded better results with both models.

Finally, the data was a bit unbalanced which may have resulted in bias. Also keeping the original three labels might have been a better choice. Both models tended to give false negatives which suggest that many of the “mid risk” cases were likely assigned to wrong classes.

6 References

[1] https://data.worldbank.org/indicator/SH.STA.MMRT?year_high_desc=false

[2] https://www.nichd.nih.gov/health/topics/high-risk/conditioninfo/factors

[3] mlbook.cs.aalto.fi, chapter 3.6

[4] https://scikit-learn.org/stable/

[5] https://archive.ics.uci.edu/ml/datasets/Maternal+Health+Risk+Data+Set

[6] mlbook.cs.aalto.fi, chapter 6.2

[7] mlbook.cs.aalto.fi, chapter 3.13