Table of contents

Project Report - Comparing the performance of Decision Tree Classifier and Naive Bayes Classifier on Diabetes dataset

January 2022

Efim Novikov-Glushkov (s1034642)

Loek Geelen (s1056332)

Abstract

In this project, we have compared the performance of two models created by using two different classification algorithms, Naive Bayes and Decision Tree. Both models were compared on PIMA Indians diabetes dataset which can be found here:

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

Models are used to predict by a set of input data whether a person has diabetes or not.

We compared the performance of both models against each other and against a baseline by standard metrics like precision, recall and f-measure. We have also conducted a McNemar's test for model comparison.

At the end of our research we have concluded that both models can be used together to quite well predict diabetes; however, there are some metrics in which one model outruns another.

1. **Domain application and the research problem.**

   The project can be applied in the medical field to help medical experts and patients to help with diagnosing, predicting and possibly preventing Diabetes. In particular for women who are at least 21 years old.

   We compared two different methods, Decision Tree Classification and Naive Bayes Classification, and it would help if this project concludes that one method is more suitable than the other method.

   Both the Naive Bayes Classifier and the Decision Tree Classifier can be used to notice correlations between on one side whether a patient has or hasn’t diabetes and the other other side various other attributes.

Some of these attributes can have a strong correlation with whether a person has diabetes. For example, it could be the case that there is a high percentage of cases where people with a high BMI, high blood pressure and have and are about 65 years old also have diabetes and that there is a high percentage of cases, with a low BMI, low blood percentage and have an age which is everything besides an age around 65 who hasn't diabetes. In that case you may conclude that people with a high BMI, high blood pressure and are around 65 years old have a bigger chance of having/getting diabetes if there is a significant correlation.

Data mining projects like this can mainly be divided into two parts, namely:

- Performing the classification which determines correlations between outcome and the other attributes
- Testing whether the classification is reliable and accurate.

Performing the classification on its own is searching for correlations. Which……

This project however is more focused on the performance of the classifications. Does the classification make sense. Is it reliable, accurate, usable, etcetera. To determine whether such model makes sense we use various metrics to determine whether the classification makes sense and are helpful. The metrics that we use are:

All these metrics are compared for both the classifiers. We can take into account that some metrics are more suitable than others. For example "accuracy" is a metric that is relatively easy to implement but it's less reliable than, for example, a confusion matrix. Furthermore, different metrics can have different data they measure which can mean different things. This makes it difficult to determine which classification is more suitable for predicting diabetes.

2. Related previous work

We have found some useful related work here:
https://www.kaggle.com/uciml/pima-indians-diabetes-database/code

- Some insights of EDA before applying Naive Bayes:
  https://www.kaggle.com/saurav9786/naive-bayes
- An example of the Diabetes Prediction using a Decision Tree
  https://medium.com/edviconedu/pima-indians-diabetes-prediction-using-decision-tree-in-google-colab-419b443a4525
- A case study of the Pima Indian Dataset dataset with observations
  https://machinelearningmastery.com/case-study-predicting-the-onset-of-diabetes-within-five-years-part-1-of-3/

3. Dataset, and preprocessing methods

We used this dataset for our problem: https://www.kaggle.com/uciml/pima-indians-diabetes-database

3.1 EDA

Before data-preprocessing with have conducted some EDA. Here is what we have found during that process:

- Dataset consists of 8 input features and 2 target variables. Input features are measures of some human organism parameters and there are two continues features, namely BMI and Diabetes Pedigree function, all other features are discrete.
- There are a total of 768 records in the dataset and there are no missing values at a first glance and there are no duplicate values.
- Features like Glucose, BloodPressure, SkinThickness, Insulin, BMI contained zero values which is impossible to be found in a living person, so we treated those as missing values. We have also found that data contains several outliers.
- By a correlation heatmap (fig. 1) it can be seen that features like Glucose,
- The dataset almost as half more 0 outcomes than 1 outcomes making it imbalanced.
- Insulin and BMI correlated stronger with outcome than other features.

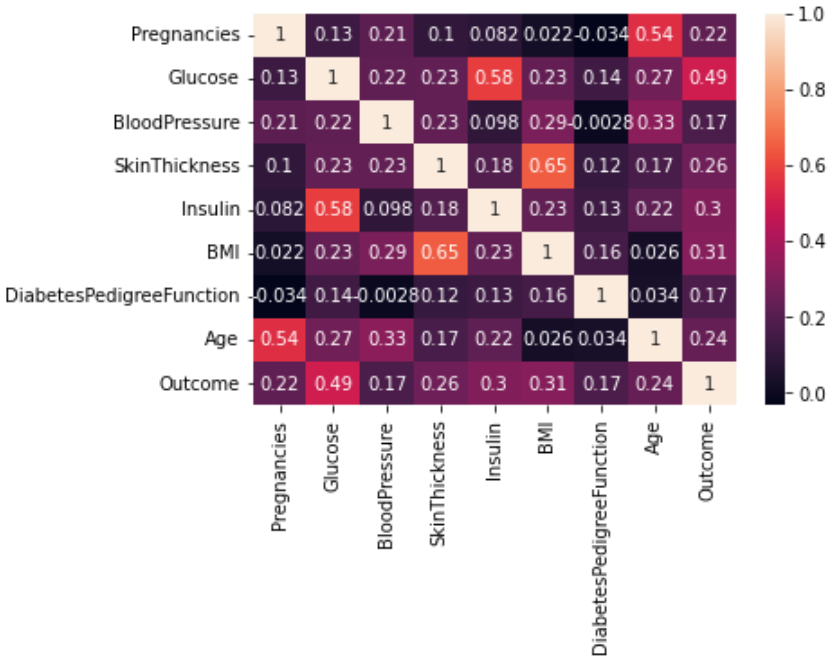

Fig. 1

3.2 Pre-processing

3.2.1 Missing values imputation

First, we had to decide what to do with missing values. There is maximum 374 missing values for insulin and simply deleting all of them is losing a lot of other

significant data. So, we decided to impute missing values by using mean or median techniques. The choice of a technic is mainly dependent on data skewness. If the data is normally distributed, then mean will work best and when the data is skewed the median is preferred. That is because the large or small values have a big impact on the mean. So, with a skewed distribution the median is better option. However, if the data is normally distributed then this hasn’t an impact on the mean since if there is a large value then there is also a small value, and these values are crossed out against each other. The mean is then a better option for a normal distribution since it is easier to calculate. The skewness

of data can be determined by a distance plot provided by seaborn package.

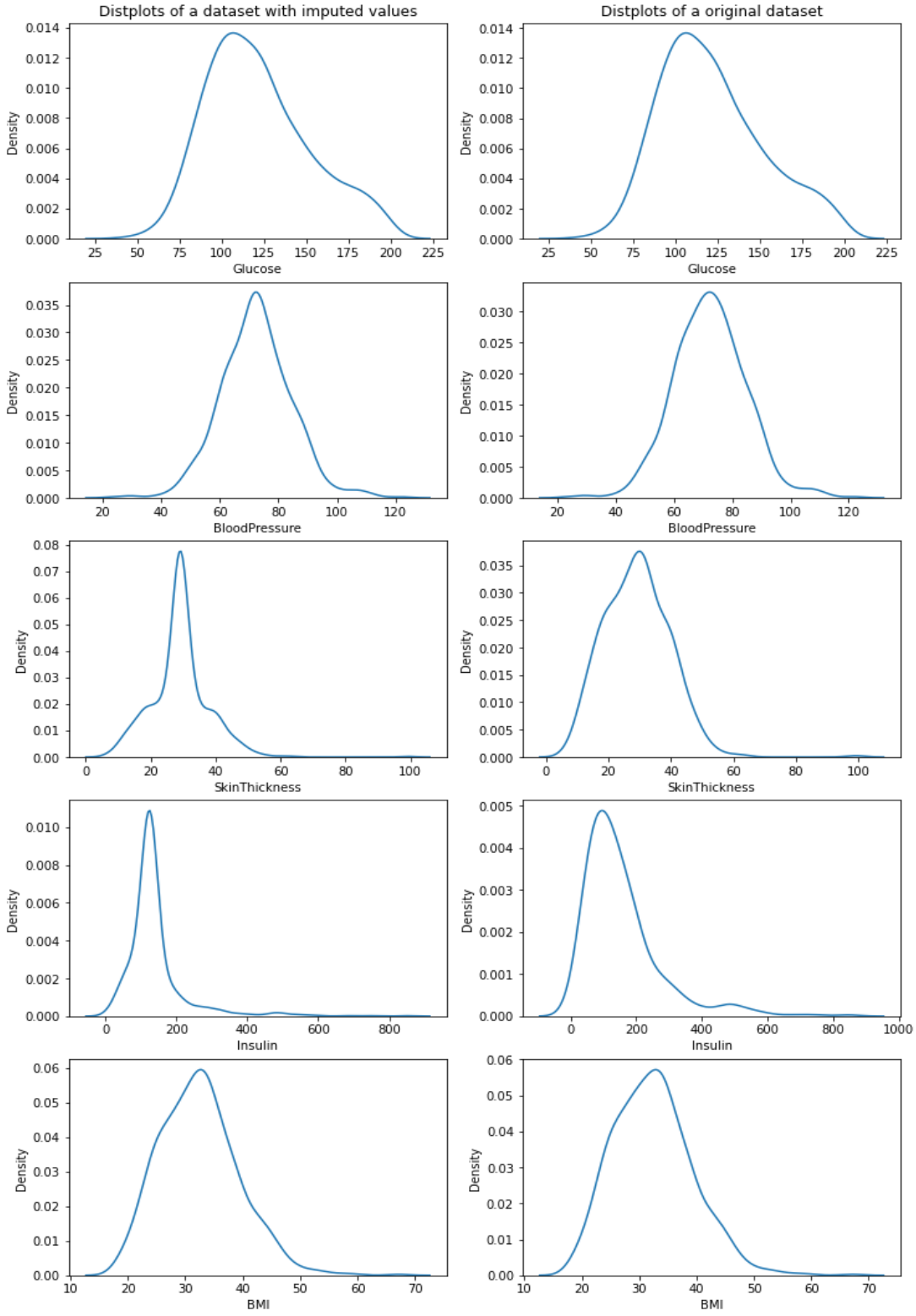

Fig 2

Fig 3

As we can see from the Fig. 2 the Glucose and BloodPressure distribution is close to normal, and all the rest of data has some degree of left skewness. Based on that we applied imputation by mean on first two features and by median for the rest. Imputation is a good technique to treat missing values, however it can add some bias to the dataset and thus making classification less effective. To check that correlation between features and outcome did not change dramatically, we compared the distance plots and correlation matrices of original dataset with the imputed one. If we compare Fig 2 and Fig 3 we can see that they are very similar which means that imputation did not change data at least in terms of it distribution. The only thing that changed more than others is a SkinThicnkess. Let's also check the correlation matrices.

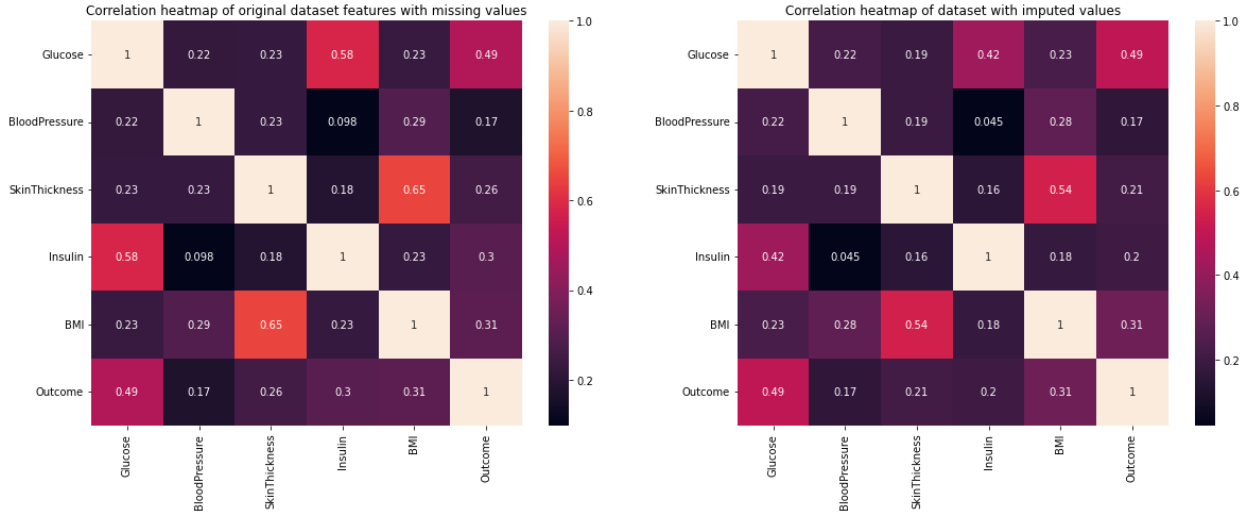

Fig. 4

Correlation matrices (fig 4) are almost the same. SkinThickness correlation with outcome and other features changed very slightly, so even if distance plots are more different for SkinThickness we can see that its correlation did not change much. This also applies to other features. We can also see that Insulin correlation is lowered than in the original dataset, but this is due there are 374 meaningful values more for that feature.

To conclude, by comparing the correlation and distance plots we can report that the relation between features and outcome did not change much after the imputation, and we did not introduce any bias in the dataset.

3.2.2 Dataset balancing

To display the imbalance, we have introduced a count plot of the target variable.

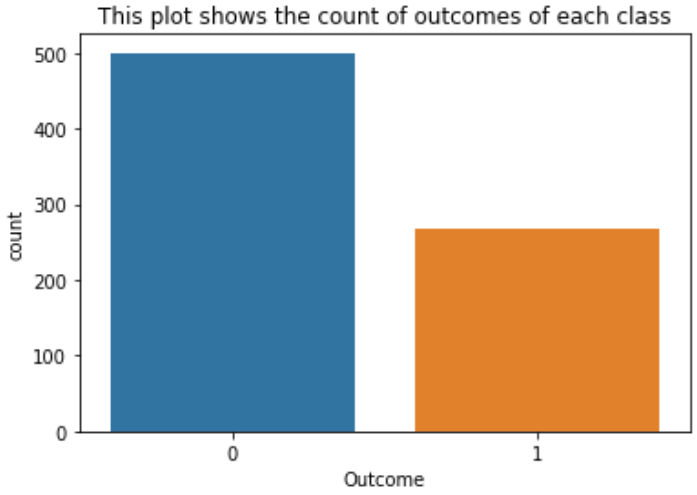

Fig. 5

From the fig 5 we see that the 1 outcome occurs almost half as less as outcome 0. This is not that good, assuming that our dataset is relatively small. We would expect some bias for 0 outcome in classification and, thus, a low recall rate, so it is better to balance the dataset before modelling. There are two main approaches to address dataset imbalance: undersampling and oversampling. As pima dataset is not very large, undersampling is not a desired approach as it will generate a huge loss of data and, as a result, a worse classification performance. We tried to do oversampling using SMOTE algorithm. In order to make sure, that oversampling did not generate some strong correlation between features, we will compare correlation matrices before and after oversampling. If the matrices are almost identical, then oversampling did not change much in the relation between features.

Fig 6

According to the correlation matrices before and after oversampling (fig 6) the relation between features and target variable was not much affected.

As a result of oversampling and imputing we now have a perfectly balanced dataset without any missing values. This new dataset preserved the original correlation between variables, so we can be sure that the classification model will be accurate.

4. Constructing classification models

All models require test and train data to work with. Model learns on a train data and then predicts the test data outcomes. For this dataset we decided to use a common 30/70 split where we randomly split our data and 30% of it becomes a test set and 70% as a train set. When it comes to measuring the model performance the main measures, we focused on was a recall and precision. The model we construct must predict the illness and it is extremely important to not predict a diabetic person as not diabetic. Therefore, recall is used as it shows the rate of correct diabetes predictions.

4.1 Gaussian Naïve Bayes

For this dataset we chose Gaussian Naïve Bayes classifier since it is suitable for continues data. Implementing this classifier is relatively straightforward as this does not require any hyperparameters tuning. After fitting train data to the model and predicting the test data outcome we scored the model using ROC and default sklearn scorings. Here are the results:

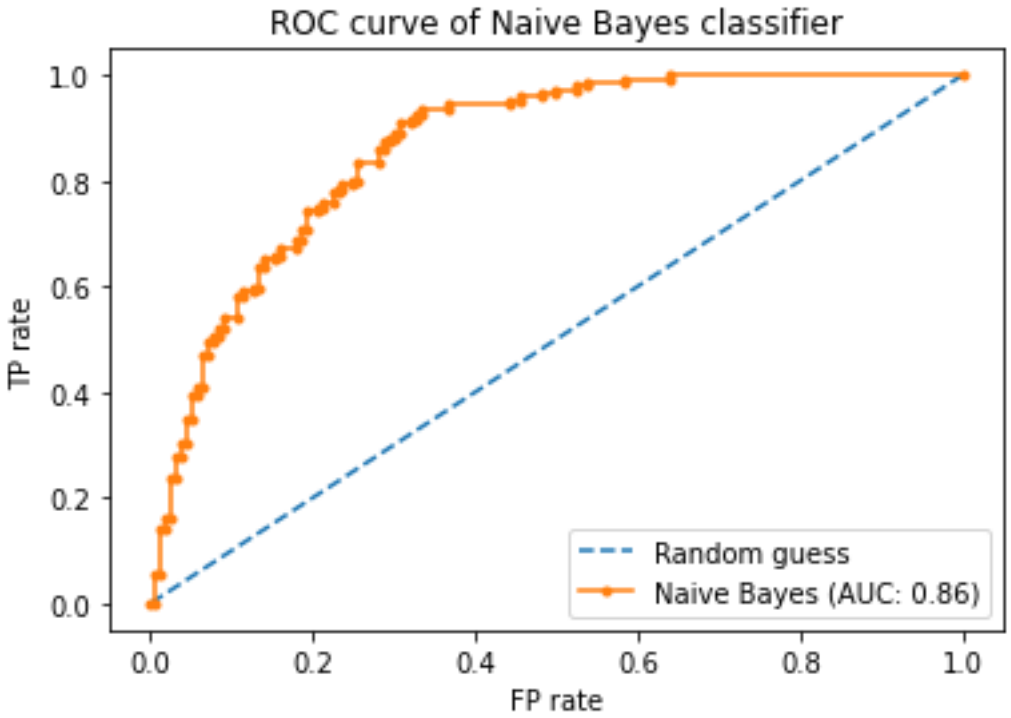

Fig. 7

According to the ROC curve (fig 7) our model is quite good at distinguishing diabetic and non-diabetic people. The recall of this model is 0.73 which means that in 73% of

cases the model will correctly show that diabetic person is indeed diabetic. This is quite good.

4.2 Decision Tree Classifier

4.2.1 Tuning hyperparameters

This classifier is a bit trickier as its performance depend on the values of hyperparameters. We decided not to optimize all of them but focus on the most important one: the depth of a tree. This parameter allows us to find a threshold where model starts overfitting. The overfitting is the main problem in decision trees as if it derives more and more relations on a train data it becomes more useless on a test data since the derived relations are specific for the used training data but not for data other than the specific training data.. We trained the Decision Tree model on using different max_depth parameters and derived a graph showing the absolute mean classification error for train and test data. Let's look at it (fig 8):

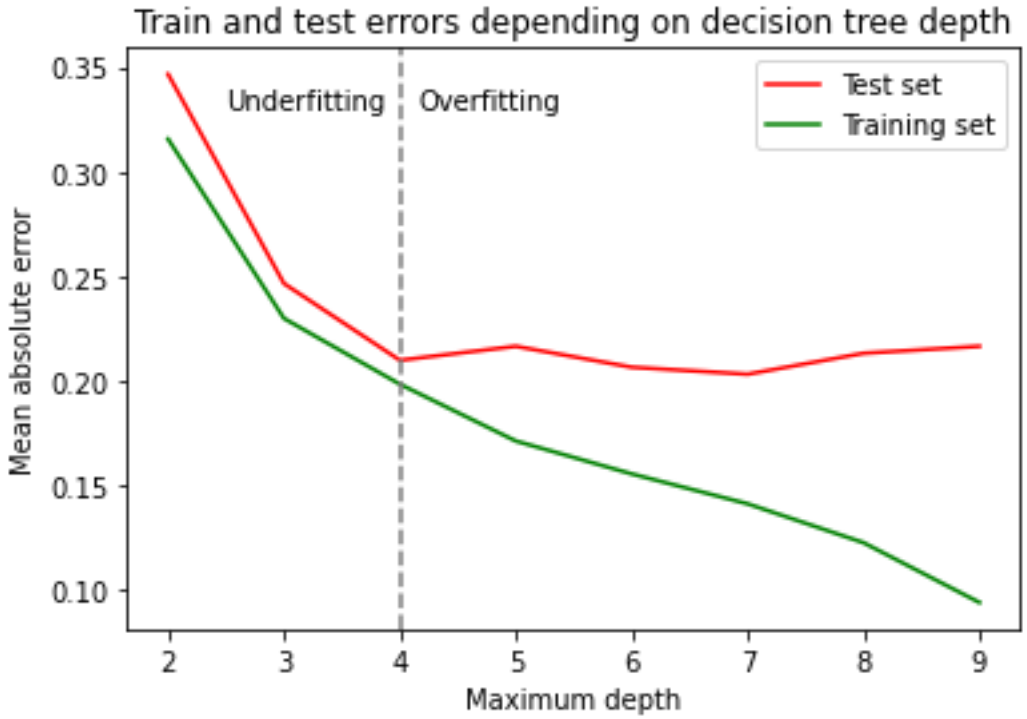

Fig. 8

At a max_depth = 4 the classification error of test set starts to grow again when it continues to drop for the training set. This is evidence of overfitting. Before the 4 max_depth, at the same time, the classification error both for training and test sets is high making it obvious that the tree is underfit at values less that 4 for max_depth. So, we decided to use max_depth = 4 as a hyperparameter for the Decision Tree.

4.2.2 Fitting the model and predicting outcomes

Let's train the Decision Tree classifier on the same sets that we used for Naïve Bayes. After the training and predictions, we derived the ROC curve and same metrics as for the previous classifier. Let's also look at them.

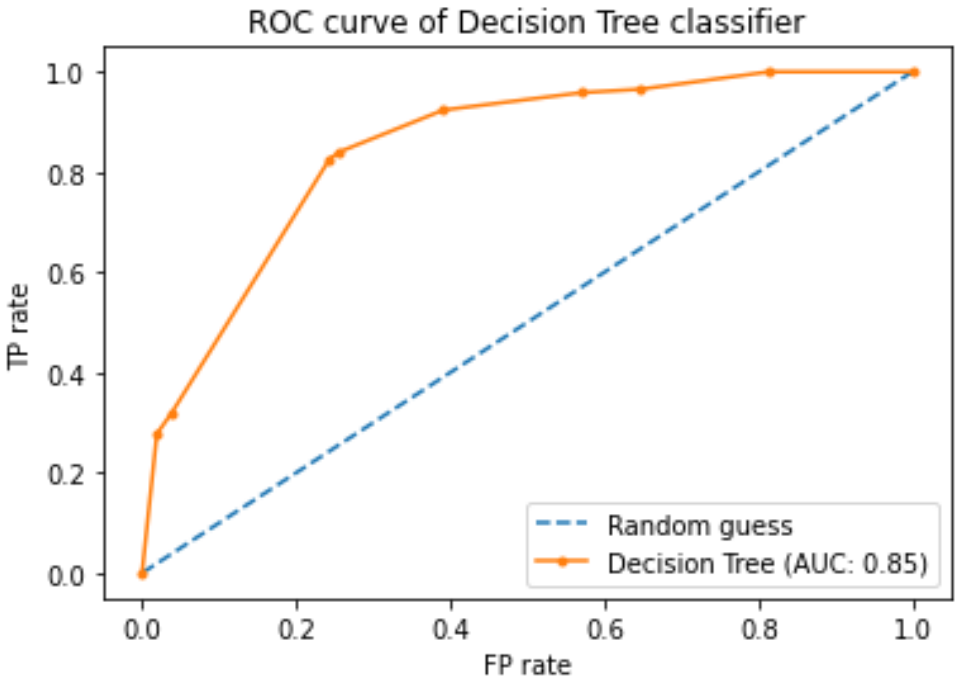

Fig. 8

The ROC curve again shows that the model is very good at distinguishing diabetic and non-diabetic people. The recall for Decision Tree is 0.84, which is by 10% higher than for Naïve Bayes.

To conclude this part, we have successfully trained and tested two different classifiers on our pre-processed data. We have derived some simple scores for both classifiers and in the next section they will be compared with each other.

5. Comparing the classifiers performance

5.1 Basic metrics evaluation

When it comes to comparison of performance for classifiers it is always good to compare them with basic sklrean scoring. First, let's compare their ROC curves (next page).

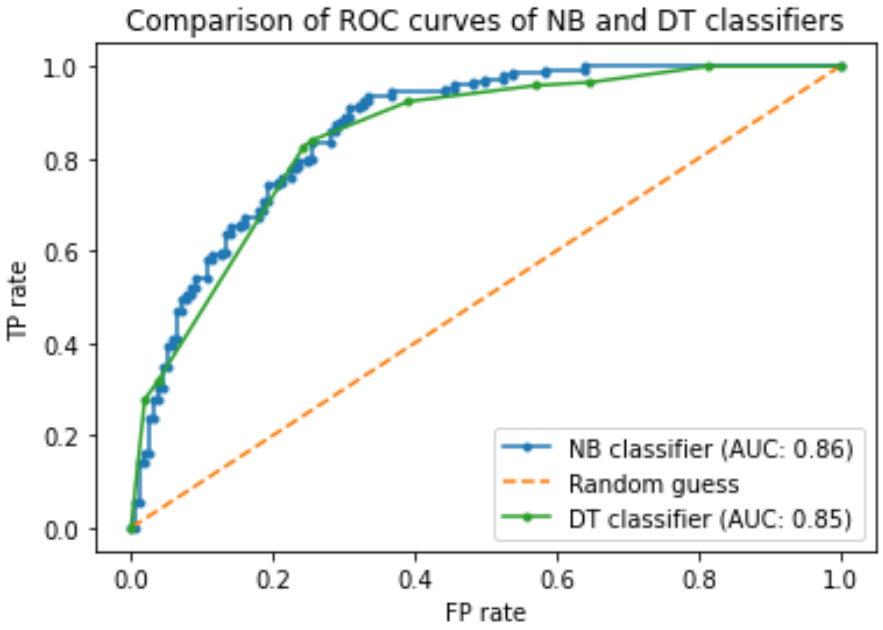

Fig 10.

From the comparison of ROC curves for Naïve Bayes and Decision Tree classifiers (fig 10) we see no significant difference. This means both classifiers are equally good at distinguishing class labels. Other standard measures are accuracy, recall, precision and f1. All of them are contained in confusion matrix, so let's plot confusion matrices for two classifiers. For better insight we also derived a confusion matrix for Dummy Classifier. This will be used as our baseline.

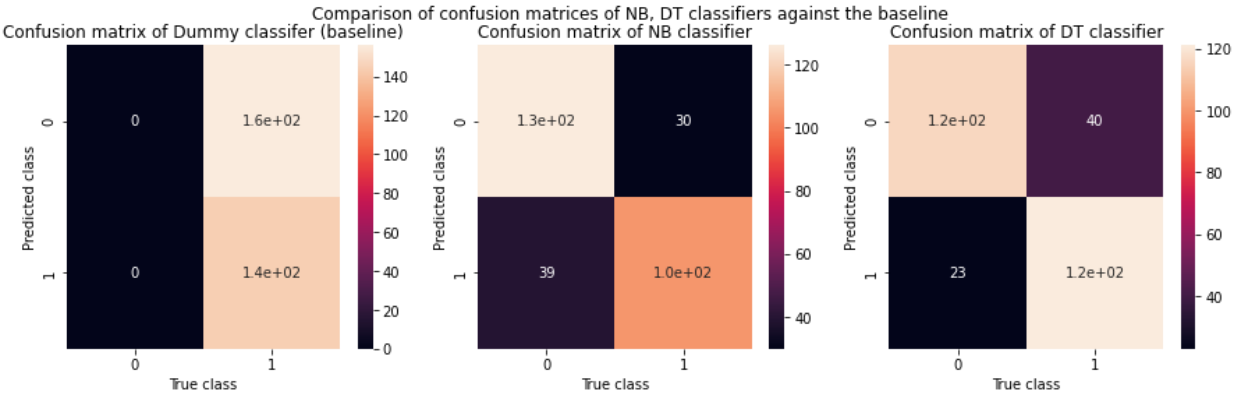

Fig. 11

Let's check the confusion matrices (fig. 11). Naïve Bayes classifier has a bit more false negatives – 39 than Decision Tree – 23. Dummy classifier at the same time has no false negatives at all since it predicted everyone having diabetes. False negatives are a most important parameter here since, as it was mentioned earlier, for predicting illness it is better to falsely predict a healthy person having diabetes rather than falsely predict that diabetic person does not have diabetes. However, it is also important for a model to predict healthy people as healthy, so

we don’t give treatment for people who do not need it. As we can see from the matrices, there are less false positives for Naïve Bayes compared to the Decision Tree. This means that this model is better at showing that healthy people are actually healthy. Again, it performs way better than the Dummy classifier which has a tremendous false positives rate.

5.2 McNemar test

We decided to use McNemar test to compare how statistically different the models are from each other as well as from the Dummy (baseline) classifier. If both models are significantly different from each other then we would expect the McNemar’s p-value to be less than 0.05. We used a 2x2 contingency matrix with the number of correct and incorrect predictions of two models. There is no need to plot this data, so here are the results.

McNemar compare DT and NB

pvalue 0.544289624429402

Here we tested the statistical difference of performance of Naïve Bayes and Decision Tree classifier. As we can see, the p-value is bigger than 0.05 significance level making both classifiers not statistically significantly different.

McNemar compare DT and baseline

pvalue 2.155375812484173e-11

McNemar compare DT and baseline

pvalue 6.02907129079485e-15

In the last two tests we compared Naïve Bayes and Decision Tree classifiers with the baseline. As we can see both p-values are way less than 0.05 significance level making these two classifiers significantly different from the baseline.

5.3 Results

McNemar test and comparison by default sklrean metrics did not show that both models are different from each other; however, there is a thing that must be noted. Decision Tree classifier showed greater results in a recall metric. The reason for that might be because of the outliers in dataset. Decision Trees are not affected by outliers, but Naïve Bayes is. Having this in mind, we can conduct that using Decision Tree classifier for this dataset is better than Naïve Bayes as the recall measure is most important for this problem. On the other side, Decision Tree precision is slightly less than the precision of Naïve Bayes, but the difference is not notable, only 3%.

To conclude this chapter, the research we made shows that both models are very good at predicting diabetes, however, Naïve Bayes might require some additional preprocessing like outliers removal which might increase its performance.

6. Possible future improvements

6.1 Cross-validation

Instead of just splitting the train and test sets by 0.3, cross validation may be applied.

This could improve the overall performance of models

6.2 Decision tree fine tuning

In this project we did not really focus on tuning the hyperparameters for the decision tree classifier. Tuning the hyperparameters could also increase the model performance.

Graphs and numbers were taken from our Project.ipynb Jupyter notebook. The notebook is self-explanatory, all the steps there were clearly explained.