Computer Vision and Machine Learning project: AI prediction model

Márta Simon
au678406
202001346

***Abstract*—This is the project work of the course Computer Vision and Machine Learning. I solved the problem by building two different models and then I compared them.**

***Index Terms*—Computer Vision, Machine Learning, Logistic Regression, PCA, SVM**

I. Introduction

The aim of this project is to develop a clinical decision-making model for determining patient-specific risk factors for knee arthroplasty survival and complications based on patient demographics, lifestyle factors, radiographic evaluation, bone biomarkers, bone mineral density, and radiotelemetry measures. The problem is formulated as a binary classification problem aiming to predict if a future failure of the knee implant can be expected or not based on the potential risk factors of a patient.

II. Method description

The project was implemented in Python using the numpy [1], pandas [2], imblearn [4] and sklearn [3] libraries.

A. Database

The database is anonymized and includes data from 450 patients. It consists of potential risk factors such as the age, blood information, or exercise habits of the patients. There are 23 such variables in total containing both discrete and continuous values. In some cases, postoperative data was also provided, categorizing the migration of the knee implant using a threshold value to decide if the migration will lead to later implant failure or not. This is a single discrete variable.

Training, validation, and test sets made from the aforementioned database were provided in the form of Excel files. The training and validation sets consist of data from 300 and 50 patients respectively, describing their potential risk factors and also the postoperative data. The test set consists of 100 rows of potential risk factors for patients.

B. Data preprocessing

The provided datasets were imbalanced and contained some missing values. On Figure 1, we can see a big imbalance in the values of the ”MIG_group” variable from the training set. ”MIG_group” denotes the postoperative data and is used in the followings as the target variable for the classification models. Such an imbalance in the target variable can easily lead to a biased model, so it is important to do something with it. A popular way to go is to undersample the majority class or to oversample the minority class. Since there are only 300 training samples in total, I chose to oversample the minority class. This way, I worked with 427 training samples.

Some columns of the training set contained up to 9 missing values and there were 16 columns in total which had at least one missing value. In the validation and test sets, there were 1 and 2 missing values in the whole sets respectively. To approach this problem, I used the mean values of the training data columns to fill the NaN values in the different sets.

I also scaled the data before passing it to the models.

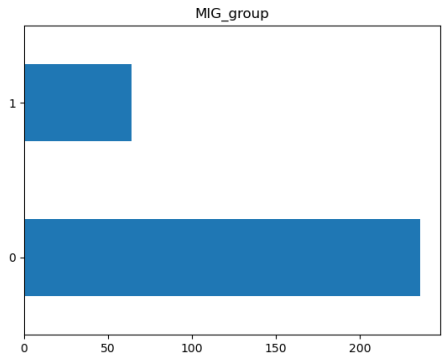

Fig. 1. Imbalanced training data

C. Models

The first model is a pipeline of the Principal Component Analysis (PCA) and the Logistic Regression algorithm. PCA is a popular unsupervised algorithm for dimensionality reduction, while Logistic Regression is a popular algorithm for binary classification problems. Since there are 23 predictor variables, it can be useful to reduce the dimensionality of the problem. The second model consists of a Support Vector Machine (SVM) with a kernel. SVM is a robust and effective algorithm even in high-dimensional spaces.

For tuning the hyper-parameters of these models, I used the function GridSearchCV from the scikitlearn library. GridSearchCV performs an exhaustive search using user-given hyper-parameters and chooses the best of them. I tried PCA with 1, 2, 5 and 15 components and SVM with different kernels such as ”linear”, ”poly” or ”rbf”. The algorithm chose 2 components for PCA and ”poly” kernel for SVM as the best parameters.

III. Results & Conclusion

The achieved accuracy of the first and the second model was 80% and 81.3% respectively. But since we worked with an imbalanced dataset, it is not exactly a good measure to use. Instead, it is more useful to look at the confusion matrices, ROC curves, and AUC values of the models.

The confusion matrix summarizes the performance of an algorithm based on its correct and incorrect predictions broken down by each class. The confusion matrices of the models made on the validation set are shown on Figure 2 and 3. We can see that any of the models were able to predict the True Positive values for the validation sets and the proportion of the False Negatives was quite high too.

The ROC curve shows the relationship between the True Positive Rate and the False Positive Rate of the predictions and the AUC is a single measure of this relationship. The ROC curves of the first model are shown on Figure 4 and 5, and of the second model on Figure 6 and 7. In Figure 6, we can see that the second model with the SVM was overfitting on the training data, and did poorly on the validation data in Figure 7. We can also see for both models, that the AUC values for the validation set are typically very low, indicating that the predictions are close to random guessing. These show that the built models became biased which resulted from working with an imbalanced dataset with a low number of samples. Although I tried to mitigate the imbalances with oversampling, it still wasn't effective enough.

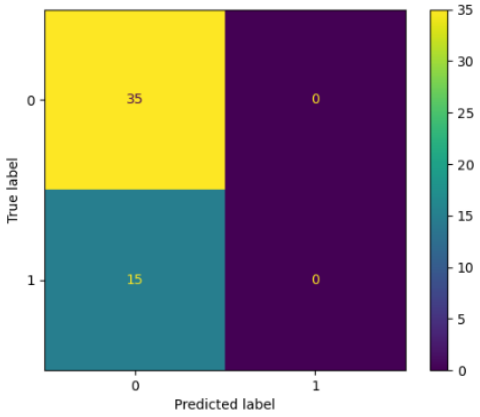

Fig. 2. Confusion matrix of the first model

References

[1] NumPy https://numpy.org/
[2] Pandas https://pandas.pydata.org/
[3] Scikit-Learn https://scikit-learn.org/stable/
[4] Imbalanced-Learn https://imbalanced-learn.org/stable/

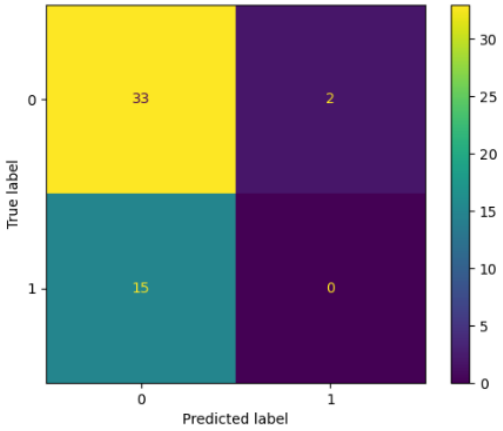

Fig. 3. Confusion matrix of the second model

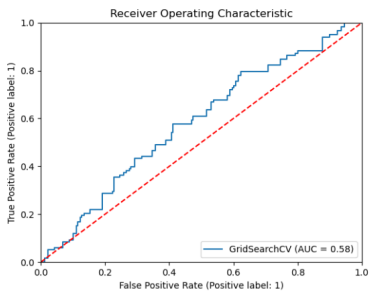

Fig. 4. ROC curve of the first model on the training data

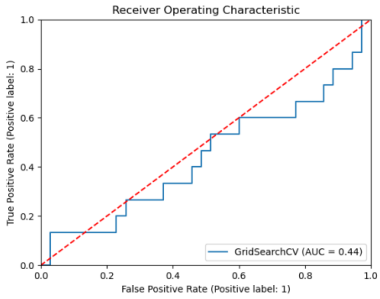

Fig. 5. ROC curve of the first model on the validation data

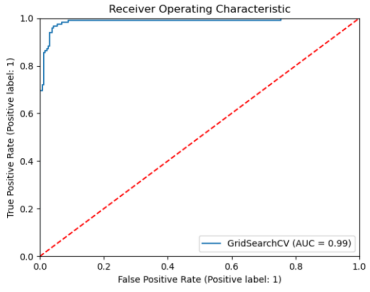

Fig. 6. ROC curve of the second model on the training data

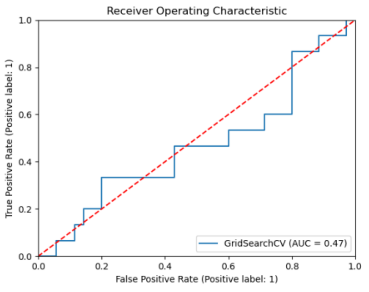

Fig. 7. ROC curve of the second model on the validation data