Foundamentals of Data Science a.y. 22-23

Machine Learning Model for Stroke Prediction

Alessio Lucciola - lucciola.1823638@studenti.uniroma1.it
Domiziano Scarcelli - scarcelli.1872664@studenti.uniroma1.it
Danilo Corsi - corsi.1742375@studenti.uniroma1.it

December 23, 2022

Contents

1 Introduction

For this final project we decided to build a **stroke prediction model** that would allow us to predict if an individual is likely to get a stroke based on some parameters like gender, age, various diseases, smoking status and so on and so forth.

2 Motivations

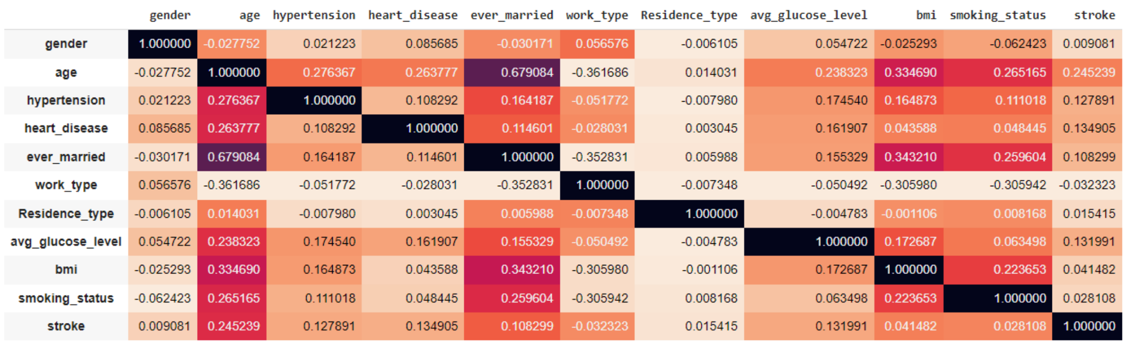
According to the *World Health Organization (WHO)* stroke is the 2nd leading cause of death globally, responsible for approximately 11% of total deaths. We found this task on a *Kaggle* competition and we eventually decided to choose it so that we could build a useful predictive model and hopefully help people to prevent this disease and improve their awareness. There are already several works that tried to give a solution to this problem and we hope to give a useful contribution as well.

3 Evaluation methods

To do this project we used the dataset already provided by *Kaggle* that contains about **5 thousand samples** and each sample has **12 features** one of which is called "stroke" that is the ground-truth so a binary value that represents whether the stroke occurred or not. Since it was a simple classification problem, we started by creating a **Logistic Regression** model and we evaluated it by testing the model on new samples. Also, we wanted to compare its performance with other models to see if they could give us better results, so we implemented other models: **K-Nearest-Neighbors**, **Gaussian Discriminant Analysis**, **Support Vector Classifier** and **Random Forests**. Some models were created from scratch in order to have more control on their executions and to avoid using them as black-box models while the remaining ones were implemented using *SKLearn*. Our aim was to **decrease the number of false negatives**, so the cases in which people that could have a stroke are predicted as safe. So we focused on increasing the **overall accuracy** and the **recall** computed making the comparison between our predictions on the test set and the true values provided by the dataset. Another metric we took into consideration was the way of splitting the dataset; since the dataset was quite small, we decided to use the 80% of the whole dataset for the training set hoping to add more generalization and get to a more robust model. The remaining data was used for the testing set.

4 Dataset, processing and feature extraction

We started working on the project by cleaning the data. First of all, the initial dataset had **5110 samples** of which 201 with absent *BMI* value; rather than imputing it naively with the mean or the median, we used a **simple decision tree** model which based on the age and gender of all other samples gave us a fair prediction for the missing values. We also deleted the *id* column since it had no useful information for the final prediction. Also, after having explored the data we noticed there were some features that carried less information with respect to others. So we measured the **correlation** between them and finally made a **feature selection** leaving only the 6 most discriminative ones and highly dependent on the response that were bmi, age, heart disease, hypertension, the average glucose level and the marital status.



	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
gender	1.000000	-0.027752	0.021223	0.085685	-0.030171	0.056576	-0.006105	0.054722	-0.025293	-0.062423	0.009081
age	-0.027752	1.000000	0.276367	0.263777	0.679084	-0.361686	0.014031	0.238323	0.334690	0.265165	0.245239
hypertension	0.021223	0.276367	1.000000	0.108292	0.164187	-0.051772	-0.007980	0.174540	0.164873	0.111018	0.127891
heart_disease	0.085685	0.263777	0.108292	1.000000	0.114601	-0.028031	0.003045	0.161907	0.043588	0.048445	0.134905
ever_married	-0.030171	0.679084	0.164187	0.114601	1.000000	-0.352831	0.005988	0.155329	0.343210	0.259604	0.108299
work_type	0.056576	-0.361686	-0.051772	-0.028031	-0.352831	1.000000	-0.007348	-0.050492	-0.305980	-0.305942	-0.032323
Residence_type	-0.006105	0.014031	-0.007980	0.003045	0.005988	-0.007348	1.000000	-0.004783	-0.001106	0.008168	0.015415
avg_glucose_level	0.054722	0.238323	0.174540	0.161907	0.155329	-0.050492	-0.004783	1.000000	0.172687	0.063498	0.131991
bmi	-0.025293	0.334690	0.164873	0.043588	0.343210	-0.305980	-0.001106	0.172687	1.000000	0.223653	0.041482
smoking_status	-0.062423	0.265165	0.111018	0.048445	0.259604	-0.305942	0.008168	0.063498	0.223653	1.000000	0.028108
stroke	0.009081	0.245239	0.127891	0.134905	0.108299	-0.032323	0.015415	0.131991	0.041482	0.028108	1.000000

Figure 1: Correlation Matrix (the darker, the more correlated)

Moreover, as the dataset was *highly imbalanced* concerning the occurrence of stroke (meaning that there were more negative examples then positive ones), we balanced it using an *over-sampling technique* on the training set, in particular we applied the **Synthetic Minority Oversampling Technique (SMOTE)**.

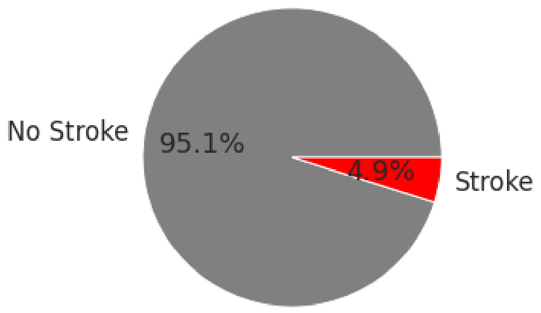


Figure 2: Stroke Distribution Before Smote

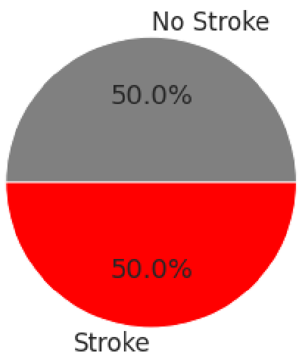


Figure 3: Stroke Distribution After Smote

We compared the results before and after the data cleaning operation and we surprisingly found out we had a much higher accuracy on all models we used.

5 Application of models

We started with a **Logistic Regression** model. We implemented it from scratch in order to have more control on its execution. We applied **gradient descent** in order to try to minimize the error the most, then we saw how the metrics changed varying the threshold. Since we applied *SMOTE* on the training set (we didn't do it on the testing set to avoid synthetic samples to possibly poison the testing operation), the metric values were different when compared to the one of the testing set (still the trend was the same). So, based on the different accuracy and the recall values computed with the testing set, we selected a threshold (t=0.55) that allowed us to get a low number of false negatives while keeping the accuracy high. Logistic Regression performance was quite good but we wanted to try different models and compare them all. We also implemented **Gaussian Discriminant Analysis**, from scratch. In particular, both Linear Discriminant Analysis and Quadratic Discriminant Analysis, in order to see if the latter performed better. We then implemented **K-Nearest Neighbors** (with an optimal selection of the K value using **Grid Search**), a **Support Vector Classifier** and **Random Forest** using the *SKLearn* library. The results were the following:

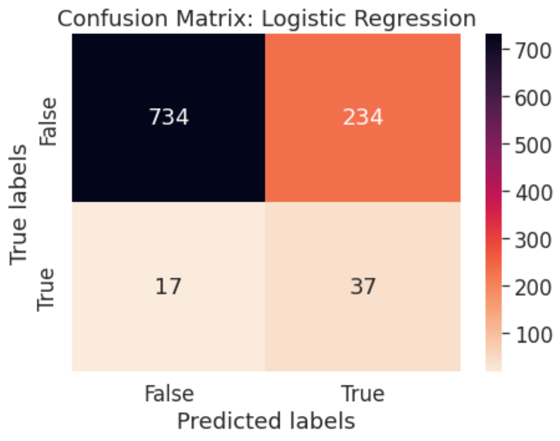


Figure 4: Confusion Matrix: LR (thres=0.55)

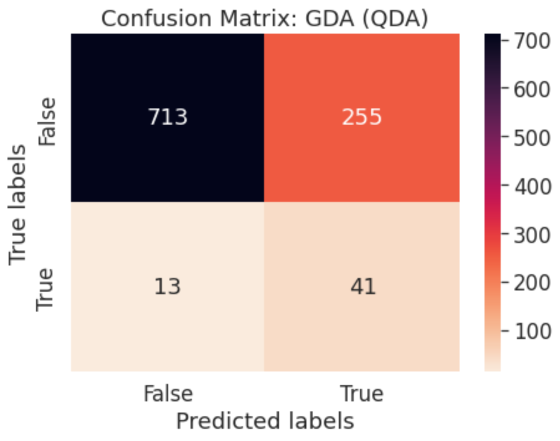


Figure 5: Confusion Matrix: GDA

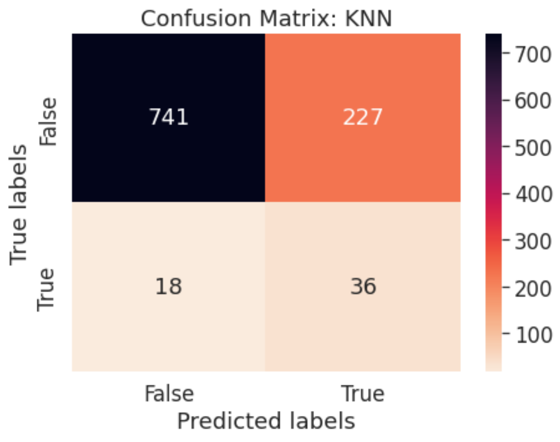


Figure 6: Confusion Matrix: KNN (k=18)

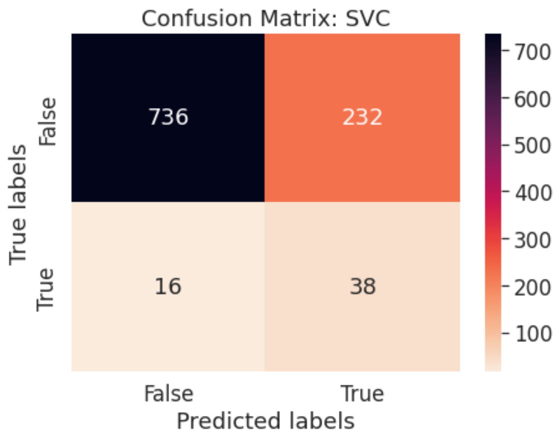


Figure 7: Confusion Matrix: SVC

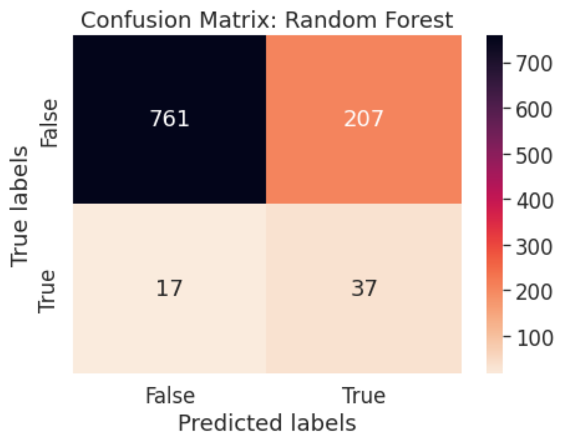
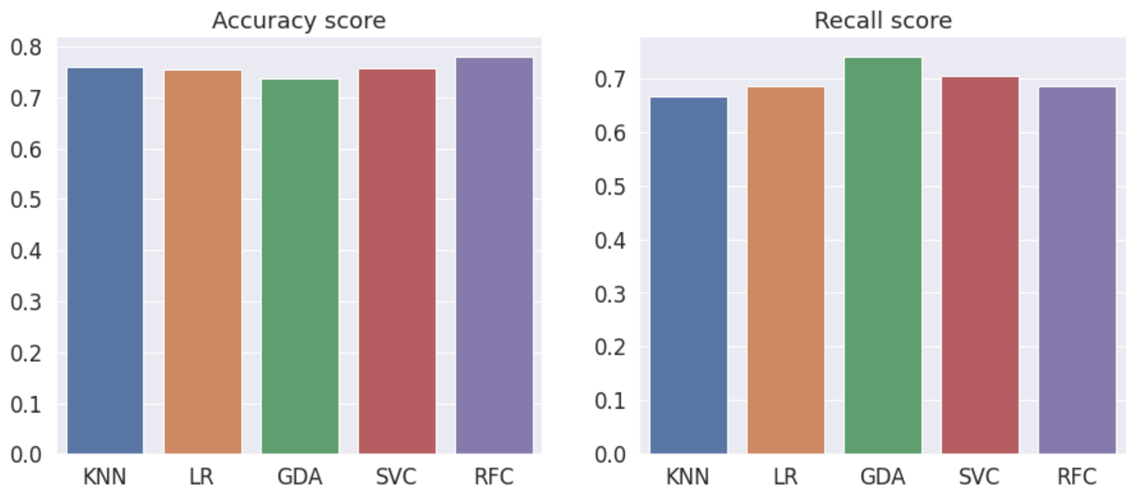


Figure 8: Confusion Matrix: RFC
(max-depth=5)

We analyzed the results for the different models, and came out with a conclusion. ***KNN*** with **k=18** and ***RFC*** have the best accuracy but a low recall. **Logistic Regression** with a threshold of **0.55** and **Support Vector Machine** has a higher recall with a little lower accuracy. **Random Forest** has the highest accuracy but gives a higher number of false negatives. Concerning our goal ***GDA***, in particular Quadratic Discriminant Analysis, is the best model we built since it has the highest recall while maintaining high accuracy.



6 Conclusion

This project gave us the opportunity to build a useful predictive model that helped us and, hopefully, other people to increase their awareness about this topic. The dataset we used had a lot of problems so we had to fix it before we could actually use it to make predictions. There were a bunch of features that carried little information, so we applied feature selection to leave only the most discriminative ones and also reduce the dimensionality of the database. Another huge problem was the fact the dataset was unbalanced. We tried several approaches to deal with the problem and finally decided to apply an oversampling technique on the training data to balance the number of negative and positive examples. *SMOTE* is generally used to gain an improvement in operational performance (in the training phase), while the testing set is used to provide an estimate of operational performance. Although it is common practice to apply *SMOTE* on the whole dataset, we decided to use it only on the training data to avoid synthetic samples to possibly poison the testing operation. Finally we applied several classification methods we thought they could give the best results and concluded that *GDA* gives the best performance because it has the lowest number of false negatives while keeping the accuracy high.