Stroke prediction: a Fundamental of Data Science Final Project

Andrea Potì, Amedeo Rinaldi, Giulio D'Erasmo e Onur Ergun

Fall Semester 2021

1 Abstract

According to the World Health Organization (WHO) stroke is the 2nd leading cause of death globally, responsible for approximately 11% of total deaths. The aim of this project is to investigate and tune the parameter of some machine learning models to predict better whether a patient is likely to get stroke based on input parameters like age, smoking status and possible heart diseases. This kind of researches can be useful to have as knowledge for change the lifestyle of a person that the models could predict to them a stroke. We focus on a dataset of about 5000 row gathered from Kaggle [2] using as a programming language Python and mostly the Sklearn library. We first investigate the dataset and act a phase of preprocessing the data like standardization and label encoding and then we tuned parameters of the Logistic Regression and KNeighbors model based on the data by performing a cross-validation and compare them finding the best model for our dataset. We discovered that also after tuning the parameters on the smaller and unbalanced dataset it can't give us a better accuracy and we needed to get at least a decent balanced data sample to perform a right decision.

2 Introduction

A stroke is a serious life-threatening medical condition that happens when the blood supply to part of the brain is cut off. The complexity of this disease involve a large variety of variables building a well framework for work with machine learning models. The purpose of this project is that we want to predict if a patient could have a stroke using different models and try to tune the parameters to better fit and predict the data of our dataset. Image you are a doctor who knows nothing about machine learning. If a patient visit your hospital telling you the symptoms they have, you could do compare similar medical records and came to a medical decision. This is essentially the **KNeighbors** model, a naive algorithm in the class of the classification models that we had seen during the class together with the **Logistic Regression**.

3 Exploratory Data Analysis and Preprocessing

3.1 The dataset

For our work we used a dataset taken from Kaggle, called **Stroke Prediction Dataset** [2]. This dataset it counts 5110 observations with 12 attributes containing medical, professional and biographical information about the patients.

We begin inspecting the dataset, checking the ratio between the actual strokes and not stokes in our dataset. We realize that the data about the strokes were a few percentage of the total. Then we realized also that the chosen dataset was quite small. This two aspects were quite critical for the analysis, so we decided to check if at least the data about the strokes were correct. We referred to the generically cause of strokes [1] and than we compared it with our data.

We saw that the stroke is frequent after the age of 55(75% of strokes occur in people over the age of 65). An high body-mass index and average glucose level are also important factor. We read about Hemorrhagic stroke that accounts for approximately 3% -5% of all strokes). This rare type of stroke is normally caused by hypertension.

We saw that there was effective consistency of age, bmi, and average glucose in the stroke cases in our dataset and only 26.5% of patients who had stroke had hypertension.

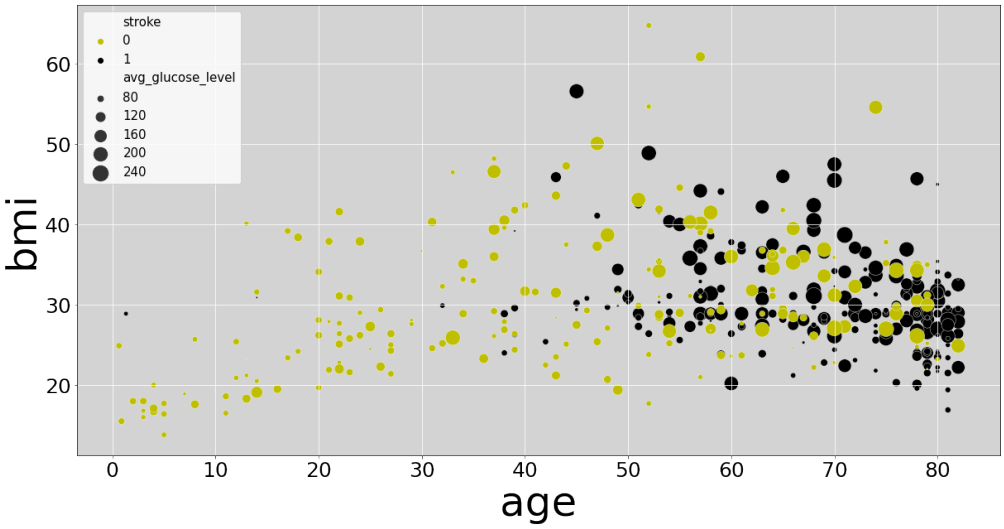

Figure 1: Scatterplot visualizing age and bmi in the axis, the size of the dot are the average glucose in the blood. In black we represent a patient who have a stroke, yellow otherwise.

In conclusion we saw that the data were correct and we decided to continue the analysis.

3.2 Preprocessing of our dataset

Our dataset is composed by numerical and categorical data that needs to be treated separately for the preprocessing of the data. For the numerical variables:

- we found some nan-values on the 'bmi' column and we decided to fill them with their mean;
- the values on 'age', 'glucose level' and 'bmi' also belong to different ranges, we decide to standardize the numeric values using the *sklearn.preprocessing.StandardScaler* sklearn class.

For the categorical variables we didn't want to assign a number for each label because that would have implied weight different the categories inside a column. We decide to go for:

- One-Hot-Encoding for the features 'ever_married' , 'gender', 'Residence_type' who deal with two categories only;
- for the multiple categorical features we used the class *sklearn.feature_extraction.DictVectorizer* to one-hot-encode 'work_type' and 'smoking_status' with cons adding more columns to the dataset.

After the preprocessing we ended up with 17 features against the 11 (if we don't consider the ID-index column) starting features.

The last bit of our preprocessing was deal with the imbalanced structure of the dataset, i.e the gap between the number of strokes and no strokes we have to train and test. Because only the 5% of our patients had a stroke we decide to compare an **oversample** dataset built using the *imblearn.over_sampling.SMOTE* library and the original unbalanced dataset.

4 The method: cross validation for tuning the parameters

The next step is tune the parameter to better predict the target on our dataset. To tune the parameter we perform a cross-validation using as cross-validator the Stratified K-fold algorithm (taken from sklearn). This method allow us to split in a target-uniform way the folders, avoiding the problem of choosing only "zeros" in the training set. Then we used the validation curve to get the best parameters, visualizing the bias-variance trade off.

4.1 KNeighbors

The first model we choose to make predictions on our dataset is the k-neighbors (KN) model. There are several reasons why we decided to implement it: first of all, it is a simple algorithm, very easy-to-implement and it can be used to solve both classification and regression problems. Our idea behind is to embody

a doctor watching similar medical records in the lab. We implemented the KNeighbors with the class from the sklearn package *sklearn.neighbors.KNeighborsClassifier* and we tried to set the best value for k in order to achieve the best accuracy for our model. We divide our work focusing on two datasets: the original one and the oversampling one.

As shown in the Figure 4 For the original dataset we choose as a best k a value of 15. In fact the gap between the training curve and the validation curve converge to a particular score, meaning that the model doesn't need more data to predict the target.

On the other hand for the oversampled dataset, after observing the validation curve (on the right of the Figure 4) we decided that a big compromise between bias-variance was choosing k = 10. With k = 1 we risk to have overfitting because it means each sample is using itself as reference and on the other side if we pick a value greater than 10, we loose too much accuracy.

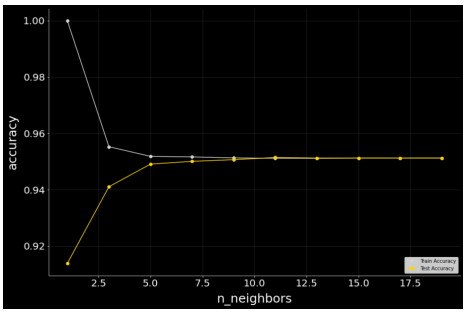

Figure 2: Validation curve for the original dataset

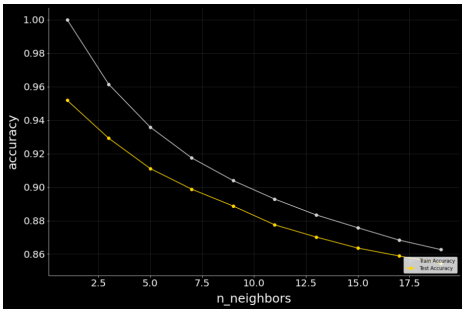

Figure 3: Validation curve for the oversample dataset

Figure 4: Validation curve for the parameter k in the KNeighbors.

4.2 Logistic Regression

The K-Neighbors works very well with a small amount of data, as in our case. But if one day the dataset will increase its examples, we will have to implement a stronger method. Our alternative is the Logistic Regression that is used to model the probabilities of our classes "stroke" and "not stroke" by estimating it using the logistic function. Using the sklearn library *sklearn.linear_model.LogisticRegression* to implement the Logistic Regression we try to tune the following parameters: degree of the polynomial and the selection of the features.

- For the **degree** of the polynomial, as we can see in Figure 7 in the case of the original dataset (on the left) the best choice is to pick degree = 2 because with degree equals to 1, the performance will result high biased. On the contrary, choose degree = 3 will produce high variance because

the performance of the model on the validation set is far worse than the performance on the training set.

For the oversample dataset (on the right) the degree 2 is too much accurate, a safer decision is to choose degree 3.

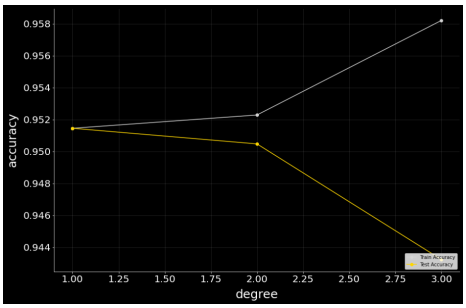

Figure 5: Validation curve for the original dataset

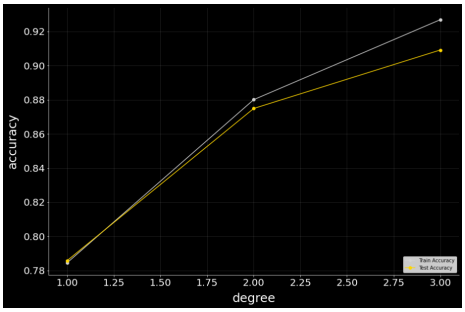

Figure 6: Validation curve for the oversample dataset

Figure 7: Validation curve for the parameter degree of the Logistic Regression.

- About the **feature selection** we use a technique called Sequential Forward Selection
  (*mlxtend.feature_selection.SequentialFeatureSelector*) to reduce an initial d-dimensional feature space to a k-dimensional feature subspace where $k < d$. Starting with an empty set of features, the algorithm will add sequentially only features that improve the performance in the validation set. We perform that on the Logistic regression of degree one.

  Taking a look to the Figure 10, for the original dataset (on the left) due to the unbalance nature of the dataset that provide a not so visible plot, we chose a total of nine features. Using the oversample dataset, we notice an improvement in the plot and we saw also in the number of features needed for the logistic regression of degree 1 is six.

5 Benchmark and conclusion

The final step of our analysis is to decide which of the models we tune is the most suitable for our purpose. As we can see in the plots on Figure 11, we compared the models looking at the AUC (Area Under The Curve) and we get that the best model is the Logistic Regression with the oversampled dataset. It highlights another time the importance of the data quantity, in fact the more data we get, the more precise and accurate our model become. Because of that, we can notice that the K-Neighbor model with oversampled dataset is even better then the simple Logistic Regression, unless the simplicity of this last model. A final consideration is that the resampling, given the initial conditions

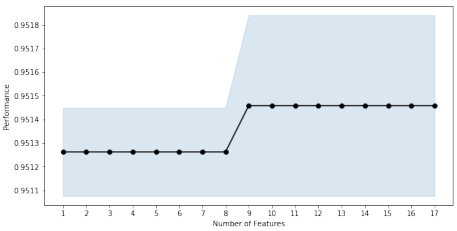

Figure 8: Validation curve for the original dataset

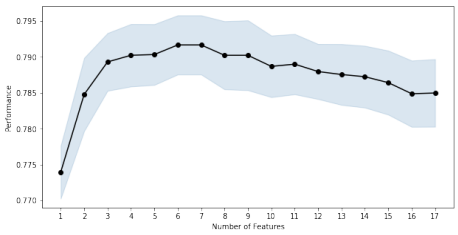

Figure 9: Validation curve for the oversample dataset

Figure 10: Validation curve for the number of feature on the Logistic regression of degree one.

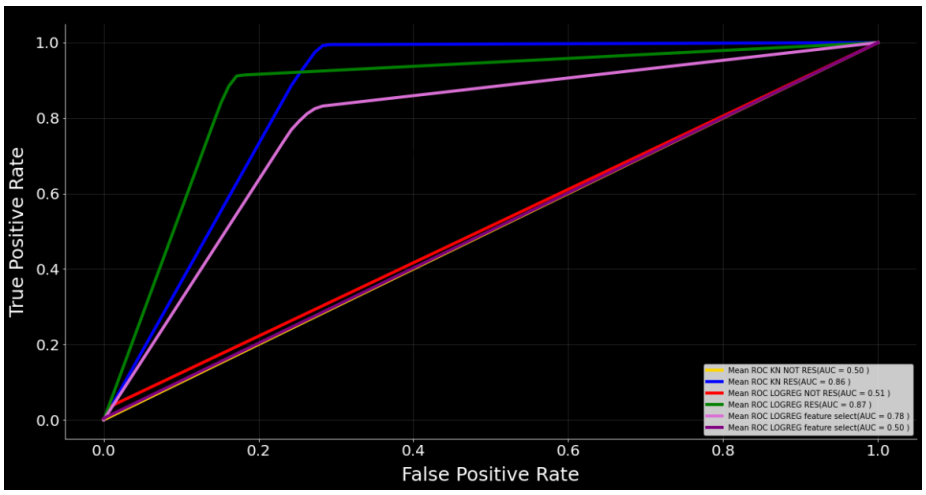

Figure 11: Comparison of the Roc Curve of the multiple models we analize

of the dataset, helps to obtain a better AUC value for all the models and as we expected the worst model is the K-Neighbor without resampling.

6 Future work

In this project we tuned separated parameter but we are sure that if we had used a grid of parameters tuning simultaneously, for example the degree of the polynomial, the regularization term and the number of feature we could have get a better accuracy. For example we try to a sketch of the algorithm in the notebook provided.

References

[1] *American Stroke Association*. https://www.stroke.org/en/about-stroke/stroke-risk-factors.

[2] *Stroke Prediction Dataset*. https://www.kaggle.com/fedesoriano/stroke-prediction-dataset.