Factors influencing stroke: a simple classifier

Raffaele Nolli

In this project I have performed data cleaning and preprocessing on the proposed dataset, with the goal of building a simple classifier.

Features preprocessing

The dataset contains both numerical and categorical variables. Two columns, "*bmi*" and "*smoking_status*", contain null values. As regards "*bmi*", roughly 3% of the values are missing, and I have replaced them with the average value for the sample. An alternative approach could be to calculate average values for each gender, and check a correlation with other variables, such as age, but that could introduce bias. Regarding "*smoking_status*", more than 30% of the values are missing, and I decided to fill the missing values with the *"never smoked"* value, and to create a variable _missing_smoking_status_, with values (0,1), equal to 1 if the _smoking_status_ information was missing. This is because the information could be relevant for the outcome, and I do not think it should be disregarded, or that the cases with "*smoking_status*" information missing, being a considerable fraction, should be dropped.
For the other non-numerical categorical values, I used the LabelEncoder() function to convert them into numerical values.
The data is very unbalanced when it comes to the "*stroke"* target variable, as only 1.8% of the values are equal to 1. As a result, a predictor would most certainly predict only 0 values. I opted to rebalance it through over-sampling by using the Synthetic Minority Oversampling Technique (SMOTE), which generates synthetic data. In the resampled dataset, the "*stroke"* target variable is equally distributed with 29470 samples per value. Oversampling can lead to over-fitting, as the synthetic samples match the distribution already present in data, and that has to be taken into consideration.

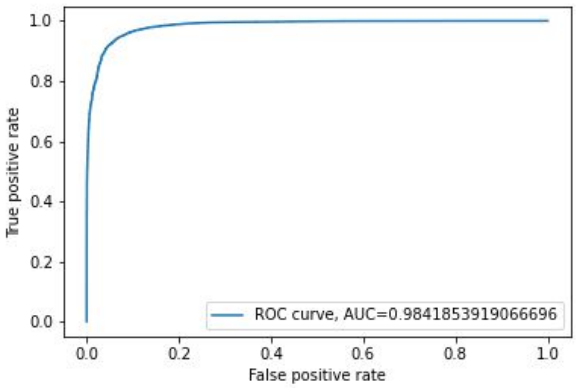
Choice of predictive model

As a predictive model, I chose a random forest classifier, as it handles categorical variables well and it is more resistant to overfitting than single decision trees, since it takes a random subset of features and samples during each iteration. I optimised the model parameter through the GridSearchCV, and the outcome is reported in the attached notebook. One drawback of random forest classifiers is the training time, especially when using GridSearchCV, and in my case the whole process took roughly 20 minutes, and would take much longer for a more complete search through parameter space. Training however happens once, and the benefit of increased precision is maintained. Another advantage of random forest is that it requires little preprocessing, whereas other methods, e.g. SVM, are based on geometric definitions of a "distance" between data points, and require scaling or one-hot encoding.

Model performance

The performance of the classifier is summarised in the table produced by the classification_report function, hereby reported (please refer to the function guide for the definition of the metrics mentioned in the table). With respect to an un-optimised model, i.e. without using GridSearchCV, all values improve, and in particular F1 increases from 0.85 to 0.93.

	precision	recall	f1-score	support
0	0.97	0.87	0.92	7323
1	0.88	0.97	0.93	7412
accuracy			0.92	14735
macro avg	0.93	0.92	0.92	14735
weighted avg	0.93	0.92	0.92	14735



The ROC curve is reported above on the right, and the value of the area under the curve is 0.98, which can be interpreted as the probability that the model separates output classes correctly.

	Feature Importance
age	0.386340
avg_glucose_level	0.193029
bmi	0.133780
ever_married	0.088285
work_type	0.055685
missing_smoking_status	0.040249
smoking_status	0.033349
Residence_type	0.022581
gender	0.022081
hypertension	0.015527
heart_disease	0.009094

Finally, we can use the feature_importances_ attribute of the model to identify the dataset features that have a higher influence on the classifier, and are thus more useful to the task.

The table reporting the dataset features and their relative importance (i.e. the sum of the importance values is 1) is hereby reported. We can see how "*Age", "avg_glucose_level", "bmi", "ever_married", "work_type"* make up roughly 85% of the relative importance. It should be noted that "*smoking_status"* is not amongst the most important features, and actually the fact that the information is missing is more important than the value itself.

It is somewhat interesting that *"ever_married"* has more importance than other features which are directly linked to someone's health, however it might be that the *"ever_married"* is correlated to other features such as "*Age"*.

Conclusions

I have built a simple random forest classifier for the provided dataset which reliably classifies patients that suffered a stroke. I preprocessed the data and optimised the model parameters in order to maximise performance, and the final model has a probability of 98% of distinguishing between the classes. The analysis allowed me to identify a subset of features that impact the

most on the model, which means that an accurate model may be built with a lower number of features, 5 instead of 10 in this case.
Such a model could be used to identify patients with potential risks of suffering a stroke, or to study potential correlations between occurrence of stroke and other health or lifestyle factors.

References:

During the development of the assignment code and the writing of the report I have consulted my previous project work, some of which available on my Github profile (https://github.com/RaffaToSpace), and the following sources:

- the discussion page and the notebooks submitted on the Kaggle database page (https://www.kaggle.com/asaumya/healthcare-dataset-stroke-data)
- The documentation on Random Forest Classifiers (https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html)
- The documentation on the imblearn package and the SMOTENC method (https://imbalanced-learn.readthedocs.io/en/stable/generated/imblearn.over_sampling.SMOTENC.html)
- The documentation on the sklearn.metrics module (https://scikit-learn.org/stable/modules/classes.html#module-sklearn.metrics)
- An article on Medium, *"An Implementation and Explanation of the Random Forest in Python"* (https://towardsdatascience.com/an-implementation-and-explanation-of-the-random-forest-in-python-77bf308a9b76)