Over vs Under-sampling a imbalanced dataset for ML models

Gil Teixeira, 88194, Student
https://bearkillerpt.xyz
Ana Tomé, Professor
https://www.ua.pt/pt/p/10307429

Departamento de Eletrónica, Telecomunicações e Informática,
Universidade de Aveiro

Abstract

This project was developed for the Data Mining course of my Informatics Engineering Masters. First a paper ,*et al*. [1], was analyzed and a different approach, over-sampling, than what is suggested, under-sampling, and are compared!

1 Introduction

The paper links the dataset used by them and it's the same used here. To start there is a need to do some pre-processing to the data. Dropping the id column because it doesn't add relevant information for the models, encoding categorical variables and finally dealing with the high imbalance of the dataset. The amount of entries with risk of stroke is very small compared to the ones with no risk, the variable we want to predict.

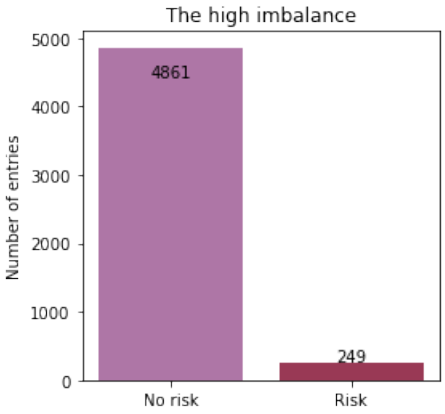

Figure 1: The data imbalance

The paper proposed under-sampling:

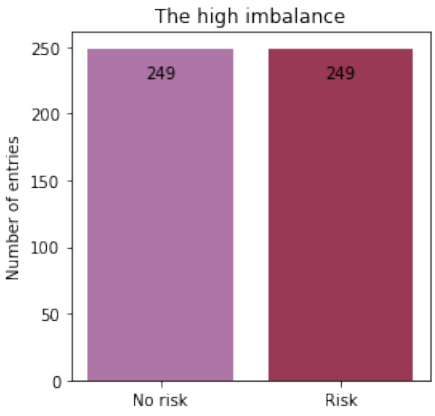

Figure 2: Under-Sampling the majority class

We propose over-sampling the minority class!

1.1 Machine Learning Algorithm Models

We will discuss the results of 6 ML algorithms:

- Logistic Regression
- Decision Tree

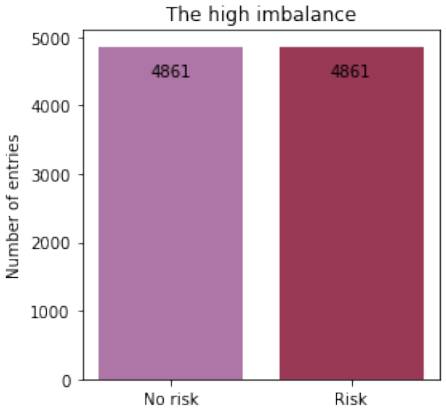

Figure 3: Over-Sampling the minority class

- Random Forest
- K Nearest Neighbors
- Support Vector Machine (RBF Kernel)
- Gaussian Naive Bayes

All tested on both the over and under-sampled datasets and are then compared! The Professor *(view top of the document)* questioned the choice of the Gaussian Naive Bayes as most variables follow either a Bernoulli distribution as their value is binary! The original paper [1] didn't specify the choice made by them. Neither did they specify the **Kernel** for the SVM algorithm or the **K** for the K Nearest Neighbors. As such the models were trained with the paramethers specified here and any others were implicitly calculated by the *sklearn* library!

2 ML Algorithm Models Metrics

2.1 Logistic Regression

Metrics and difence in the balancing method

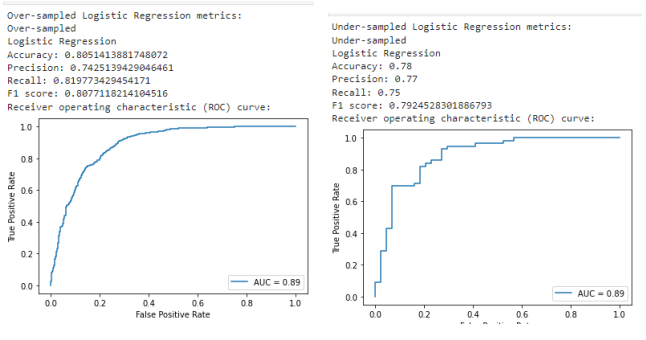

Figure 4: Over vs Under-Sampling Logistic Regression

2.2 Decision Tree

Metrics and difence in the balancing method

2.3 Random Forest

Metrics and difence in the balancing method

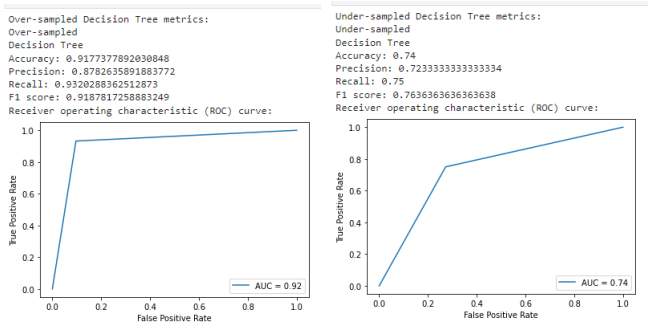

Figure 5: Over vs Under-Sampling Decision Tree

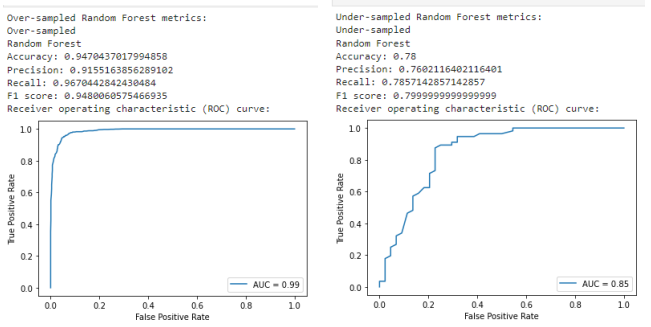

Figure 6: Over vs Under-Sampling Random Forest

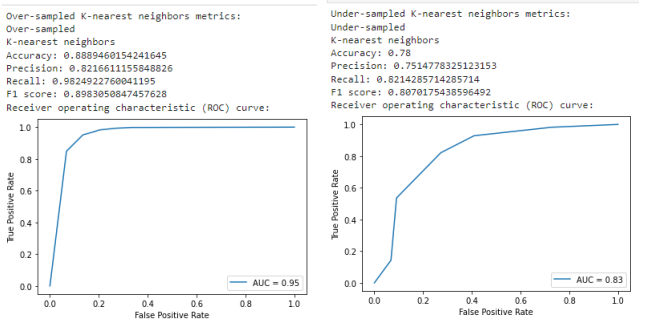

Figure 7: Over vs Under-Sampling K Nearest Neighbors

2.4 K Nearest Neighbors

Metrics and difence in the balancing method

2.5 Support Vector Machine (RBF Kernel)

Metrics and difence in the balancing method

2.6 Gaussian Naive Bayes

Metrics and difference in the balancing method

3 Results analysis

In this section multiple metrics are compared for all 6 ML Algorithms and two balancing sample techniques with the following bar plots!

You might already be on my website but if you're not feel free to come by and test the outputs of this models:
Test your inputs and use the models to predict the risk of stroke in this link!

The paper built a website just with flask. I'm personally very fond of js and particularly ReactJs and so I used Flask to build an api that I can query to get the results that are then displayed on the frontend! The notebook is also embedded as an iframe on the website [3] If you wish to use it an example of a request, as available on the dev console of your browser:
Public api get example!

For more information on how the parameters were encoded check the web-api.py script on my github page:
API Source Code!

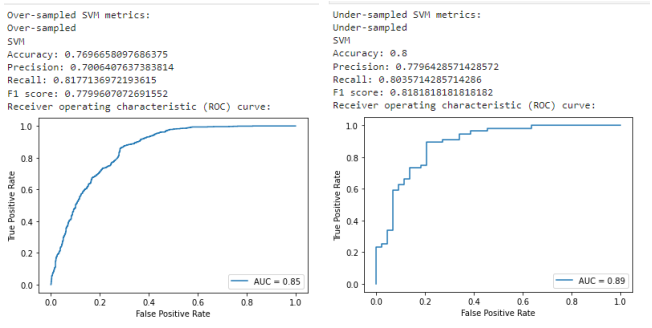

Figure 8: Over vs Under-Sampling SVM

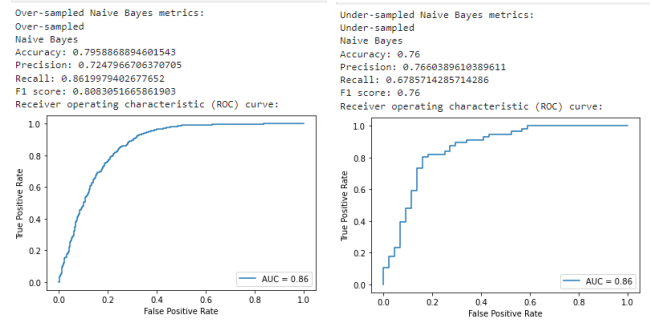

Figure 9: Over vs Under-Sampling Naive Bayes

For more information on how the parameters were encoded check the web-api.py script on my github page:
API Source Code!

4 Conclusion

The proposed balancing technique produces way better models in every ML model created except SVM! It is important to note that over-sampling might be result in a very heavy computationally heavy task and worsen the complexety of model fitting! In this specific dataset the majority class had 4861 entries and so over-sampling the 249 entries of the minority class is not very time consuming. As an example all the models were built on my Raspberry Pi 2B and took around 1,5 minutes for the over-sampled models and around 10 seconds for the under-sampled! The Api [2] is running on said hardware!

References

[1] Gangavarapu Sailasy and Gorli L Aruna Kumari. Analyzing the performance of stroke prediction using ml algorithms. URL https://thesai.org/Downloads/Volume12No6/Paper_62-Analyzing_the_Performance_of_Stroke_Prediction.pdf.

[2] Gil Teixeira. Data mining api usage example, . URL https://edapi.bearkillerpt.xyz/?age=20&gender=0&hypertension=0&heart_disease=0&ever_married=0&work_type=0&Residence_type=20&avg_glucose_level=20&bmi=20&smoking_status=0.

[3] Gil Teixeira. Data mining website to predict inputs, . URL https://ed.bearkillerpt.xyz/.

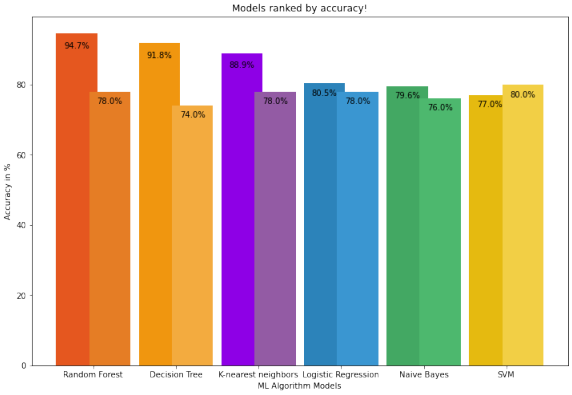

Figure 10: Over vs Under-Sampling Accuracy

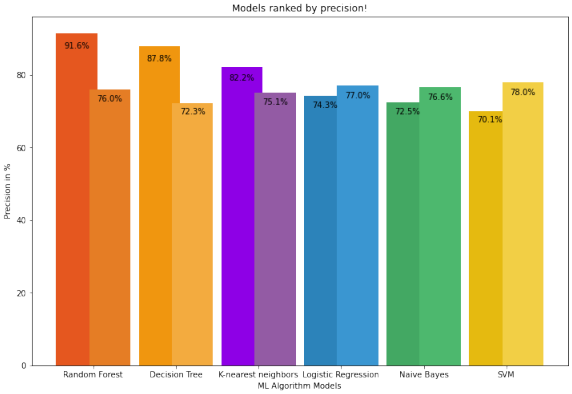

Figure 11: Over vs Under-Sampling Precision

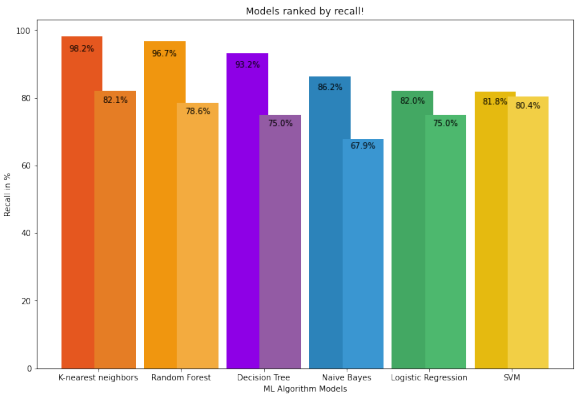

Figure 12: Over vs Under-Sampling Recall

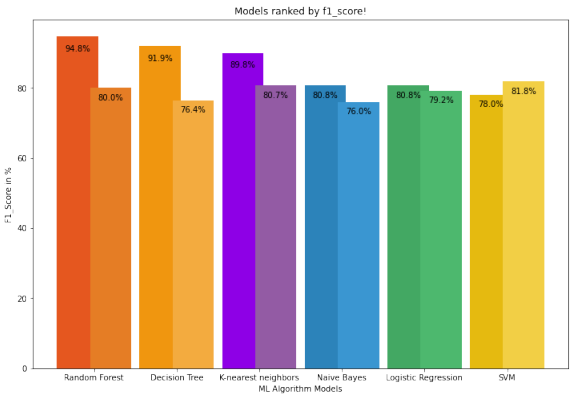

Figure 13: Over vs Under-Sampling F1 Score