Oxcitas Covid Challenge

Jon Perez

18-Jan-2023

1 Introduction

The Oxcitas Covid Challenge has the goal to develop a model to predict whether a patient is at high risk in relation to Covid-19. I intend to present the decision-making and the reasoning behind my approach.

2 Data

The data is provided in CSV format accompanied by a section describing each column. The very first thing I did was take a look at the data. In this initial look, it became apparent that a large number of rows had undefined or missing data for the 'ICU' and the 'INTUBED' columns. On top of that and continuing with these two variables, I realised that according to the dataset, a person could be intubated but not in the ICU. This seemed quite strange to me so I consulted a Doctor and a nurse on how possible it is for a patient to be intubated and not be in an ICU unit. Both responses were the same, it doesn't happen. More problems can be found when looking at the 'AGE' column which looks very unlikely given the distribution and the range.
Looking at the data correlation, this also seems to be a weak point for this dataset, for both the 'ICU' and the 'DATE_DIED' columns.

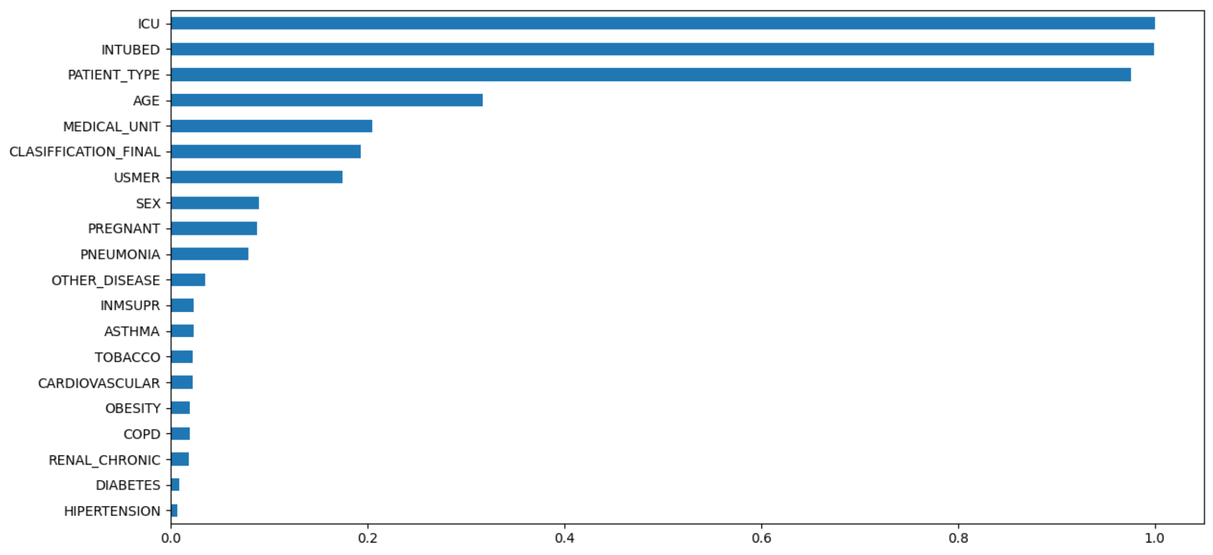


Figure 1: Correlation with ICU

As a final note on the data quality, I suspect the values 97, 98 and 99 are some other encoding of the data, like 97 for no and 99 for yes. However, I decided to approach the challenge by sticking strictly to the data values provided in the document.

2.1 Data pre-processing

Before moving ahead with any machine learning related pipeline I decided to reformat the dataset.

For the Boolean fields, I changed them to a more standard 0-1 format, where 0 is NO and 1 is YES. The 'DATE_DIED' column was changed to also become a 0-1 type column. the cases where the date was specified as '9999-99-99' were changed to 0, while the cases with an actual date were changed to 1. In any case, where the data was missing the value was set to -1. For the 'PREGNANT' column some extra work was done. For all the cases where the row was referring to a male and data was missing in the column the value was set to 0.

In this dataset, we have two different columns that represent the high-risk scenario,

’ICU’ and ’DATE_DIED’. To get a more straightforward target I decided to combine this two into a single column representing the target. The target was set to 1 for the cases where the person had either died or was admitted to the ICU. If the person has not died and was not admitted into the ICU the target was set to 0. For all other cases where the person has not died and the data is missing for ICU admission, the target was set to -1.

When it came to splitting the data for train and test sets, this was done in a stratified manner to ensure the class proportion was adequate for both training and validation. However, before this was done all rows with a missing target variable were discarded.

3 Machine Learning approach

Given that the data was in its greatest part categorical tree models became my classical machine learning model of choice. However, some more data processing was done before feeding the data to the models.

I created two pipelines, one for numerical data, ’AGE’, and one for categorical data.

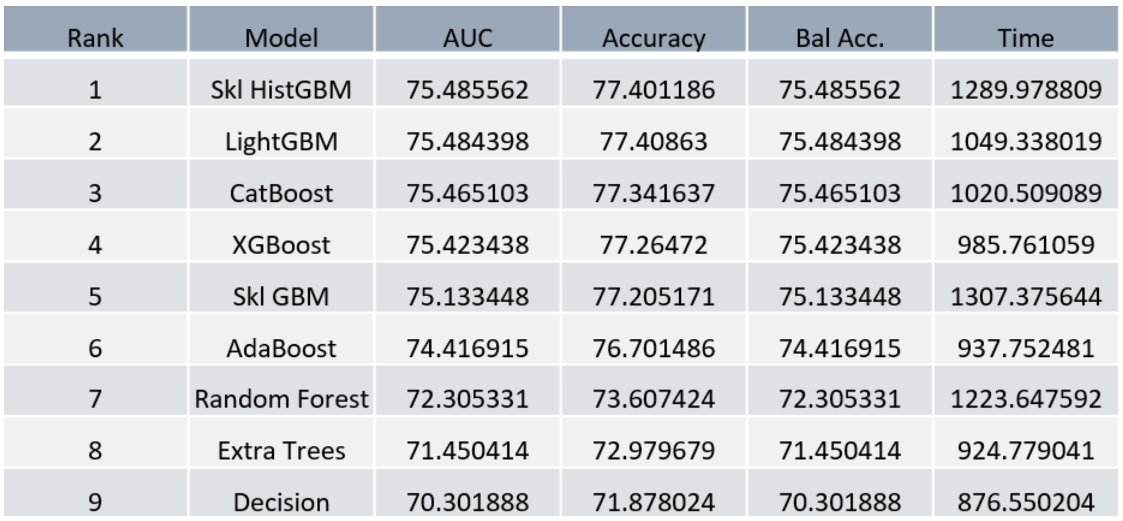
These are the steps for the numerical pipeline:

- Impute: KNN imputer
- Scale: Robust Scaler
- Normalize: Yeo-Johnson Power Transformer

However, for the categorical columns, there was only a single step, a KNN imputer.

These pipelines were put together into a column transformer with the details on which columns should be targeted by each. For each of the tree models selected a pipeline was created consisting of the previously mentioned column transformer and the model.

Each model was trained and evaluated by the following metrics: accuracy, balanced accuracy and AUC ROC.



Rank	Model	AUC	Accuracy	Bal Acc.	Time
1	Skl HistGBM	75.485562	77.401186	75.485562	1289.978809
2	LightGBM	75.484398	77.40863	75.484398	1049.338019
3	CatBoost	75.465103	77.341637	75.465103	1020.509089
4	XGBoost	75.423438	77.26472	75.423438	985.761059
5	Skl GBM	75.133448	77.205171	75.133448	1307.375644
6	AdaBoost	74.416915	76.701486	74.416915	937.752481
7	Random Forest	72.305331	73.607424	72.305331	1223.647592
8	Extra Trees	71.450414	72.979679	71.450414	924.779041
9	Decision	70.301888	71.878024	70.301888	876.550204

Figure 2: Performance table

As seen in figure 2, all of the top models performed similarly, however, only the best performer in the AUC ROC metric is saved, including the columns transformer. The model is saved in a pickled format to be loaded for future uses.

4 Deep Learning Approach

On top of the classical machine learning approach a Multi-Layer Perceptron was also trained and evaluated. Different architectures were tried for the model with a small variation in performance. Here are some relevant details.

- Activation function: ReLu
- Output Layer: Sigmoid
- Optimizer: Adam with a stepLR scheduler
- Criterion: BCE
- Dropout: 0.4
- Epochs: 100

This model was evaluated using the same data and similar metrics. However, in this case, three models are saved.

- The model with the lowest loss validation value

- The model with the highest validation accuracy
- The model with the highest validation AUC ROC score

The results coming from the experiments done with the MLP were very similar to the ones I got with the tree models. I will elaborate more on this in the conclusion sections.

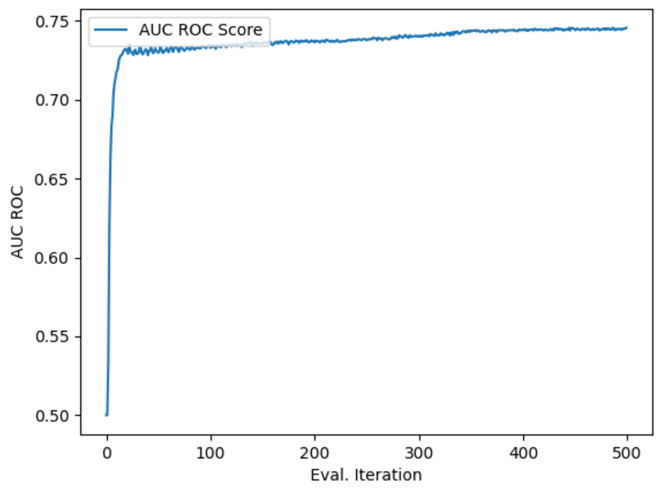


Figure 3: AUC ROC during validation

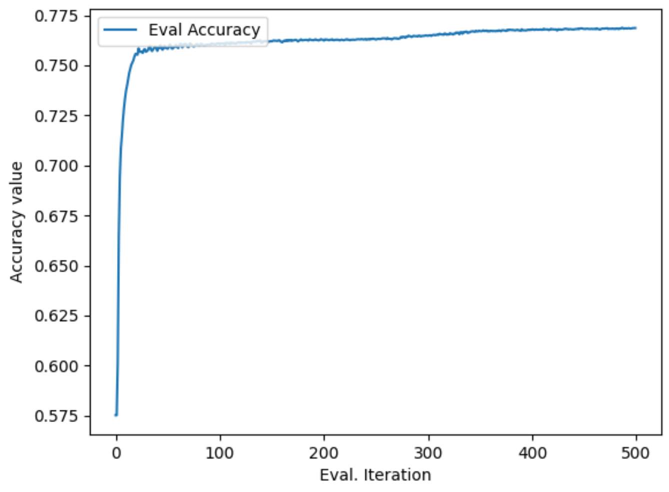


Figure 4: Accuracy during validation

5 Conclusions

I will keep the conclusions short. I think, given the results and the things mentioned in the Data section, the data quality is the biggest limitation to the performance of these models. The main reasons are the poor correlation of the variables, the issues mentioned with the 'AGE' and 'INTUBED' and the potential missing data.