Hospital Challenge

2021/06

Prepared By:
Guilherme Giuliano Nicolau

01 Introduction

One day at LinkdIn, a data scientist worker from a famous private Hospital in São Paulo sent me a message. He proposed me a challenge.

He gave a CSV file with 4 columns and asked me to provide and compare two predictive models using machine learning algorithms with my favorite programming language.

I ACCEPTED THE CHALLENGE, OF COURSE.

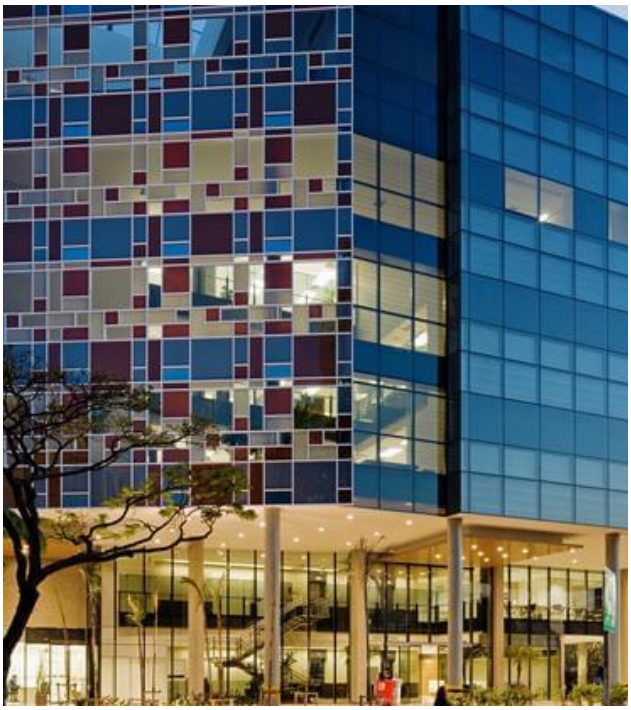
Technologies: Python version 3.9;
Tools: Atom, Jupyter IPython, Git;
Services: Github;
Python Libraries: pandas, warnings, numpy, seaborn, matplotlib, sklearn, pandas_profiling, IPython, pycaret, xgboost, GridSearchCV, shap, mlxtend, ipywidgets, mpl_toolkits

02 What about the Data?

As soon as I got the data in my hands, I've opened it on Atom, using Pandas library from Python. I had no info about it. As I opened, I started exploring it with some basic functions from Pandas.

WE HAVE:

- Three float columns (x1, x2, x3)
- One categorical Column ('target' - to predict)
- The target column has three hierarchical values (low, medium, high)
- 6000 rows (60%) are lows
- 3000 (30%) are mediums
- 1000 (10%) are highs
- 10000 rows on total
- All filled (no nulls)



CONCLUSION: MULTICLASS MODEL

02 What about the Data?

At this moment I've used my intuition and experience. Usually after this stage from checking the data I could already infer something, but this wasn't true here. There was no context. I couldn't tell the nature of the data at all. I couldn't tell what each column represented, I couldn't even figure out a metric for the values to check if they were in different scales.

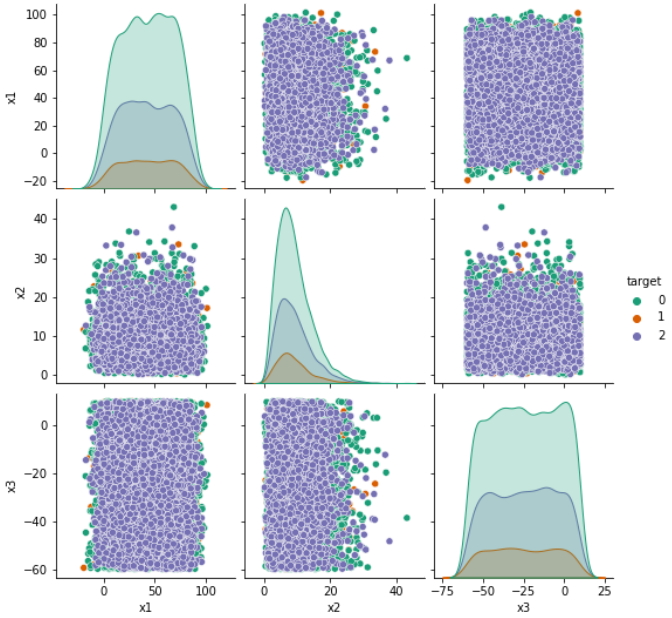
I decided to ask my colleague who challenged me about the nature of the data and the motive of the prediction of our target. He couldn't give me a clue and told me to treat the data as it is, pure and simple. I thought:

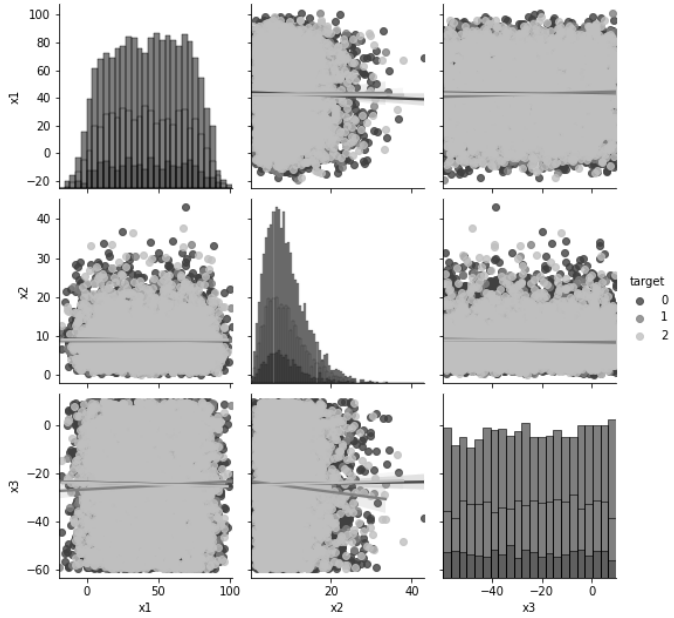
WELL, THIS CAN MAKE MY CHALLENGE HARDER AND MORE EXCITING. OR EVEN IT CAN MAKE IT EASIER! I CAN'T WAIT TO FIND OUT!

02 What about the Data?

A) PLOTING AND EXPLORING THE DATA

Our DataFrame isn't that big, so it was easy to generate a report with libraries such as Seaborn Pairplot and Pandas Profiling.





WHAT WE CAN SEE:

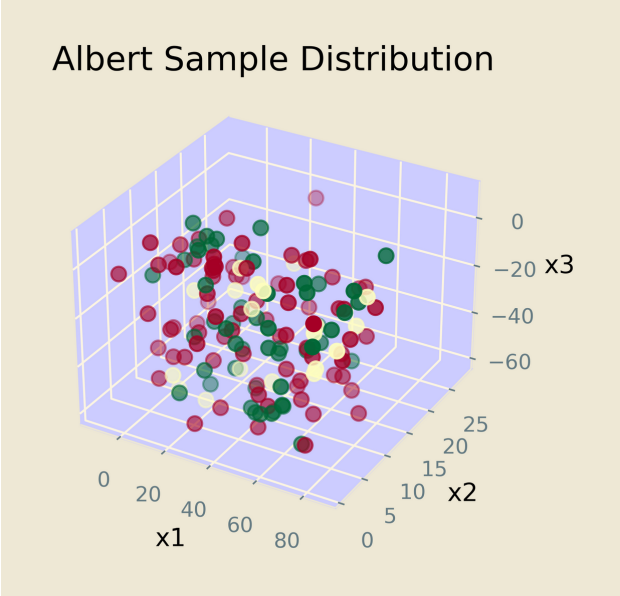
- Something close to a normal distribution on column x1;
- Left skewed distribution on column x2;
- Uniform distribution on column x3;
- Seems imbalanced for the values from our target variable in a geometric progression pattern;
- Data is small, no need for Lazy Execution with Dask, PySpark or Hadoop

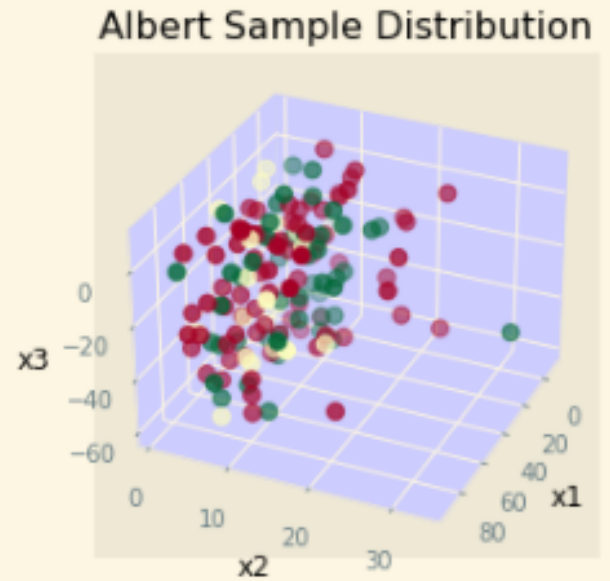
WE CAN INFER THAT THIS DATA WAS PROBABLY ARTIFICIALLY CREATED SO WE CAN HAVE SO CERTAIN SHAPES AND DISTRIBUTIONS THAT ARE WELL KNOWN FOR DATA ANALYSTS

02 What about the Data?

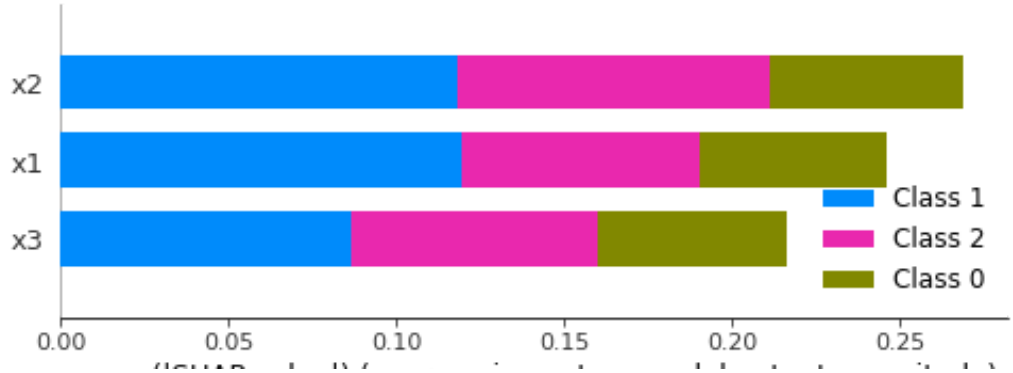
B) 3D PLOT AND FEATURE SELECTION

On our 3d plot from a sample of data, we can detect that the concentration of the distribution is on 40 for x1, between 0-10 for 'x2' and between -20-40 for 'x3'





FEATURES:

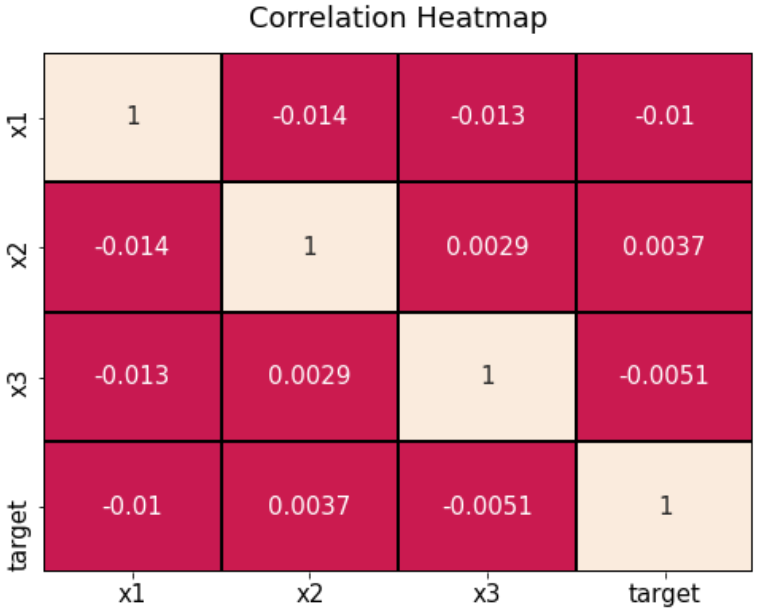


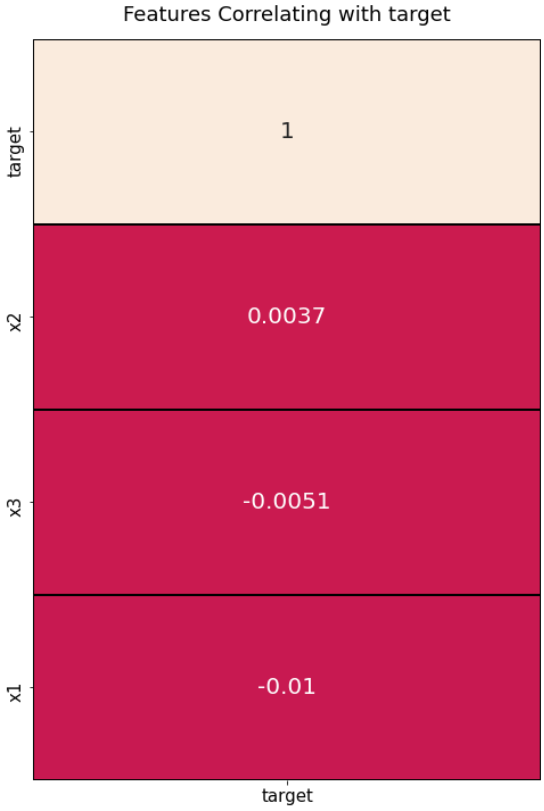
ALL THREE COLUMNS ARE SIMILARLY IMPORTANT

ilt was easy to notice; we didn't need to automate the search with RFE from sklearn library (RFECV)

02 What about the Data?

C) CORRELATION HEATMAP





WE CAN'T DETECT A CLEAR DIFFERENCE IN WEIGHT BETWEEN ONE ANOTHER

04 Preprocessing

	x1	x2	x3	target
0	2.71	10.39	-36.45	1
1	9.88	5.76	-54.63	1
2	82.87	1.73	0.83	1
3	12.99	10.40	-59.60	1
4	60.10	8.84	-45.87	1
...	...	...	...	...
9995	65.03	13.47	-28.30	2
9996	62.24	7.42	-50.47	2

First thing to do is to convert the Target column to ordinal labels. Low, medium and high turns 1, 2 and 3.

We could use ready-made functions from Sklearn, such as Ordinal Encoder, Label Encoder and One Hot Enconder.

SINCE IT WAS SIMPLE, WE DECIDED TO ENCODE OURSERLVES VECTORIZING THE CATEGORIES THROUGH PANDAS FACTORIZE COLUMN

```
# Order categories
categories = pd.Categorical(albert['target'],
                            categories=['low', 'med', 'high'], ordered=True)
categories
# Label your target with numerical values
labels, unique = pd.factorize(categories, sort=True)
albert['target'] = labels
albert['target'].value_counts()
```

03 Choosing the model

We tought first about using PyCaret. This is a framework (a set of Python libraries) that automatize all the modelling process, like choosing the model, tuning, ensemble and predict.

But that wouldn't be a challenge. I couldn't show my skills interpreting the models to choose.

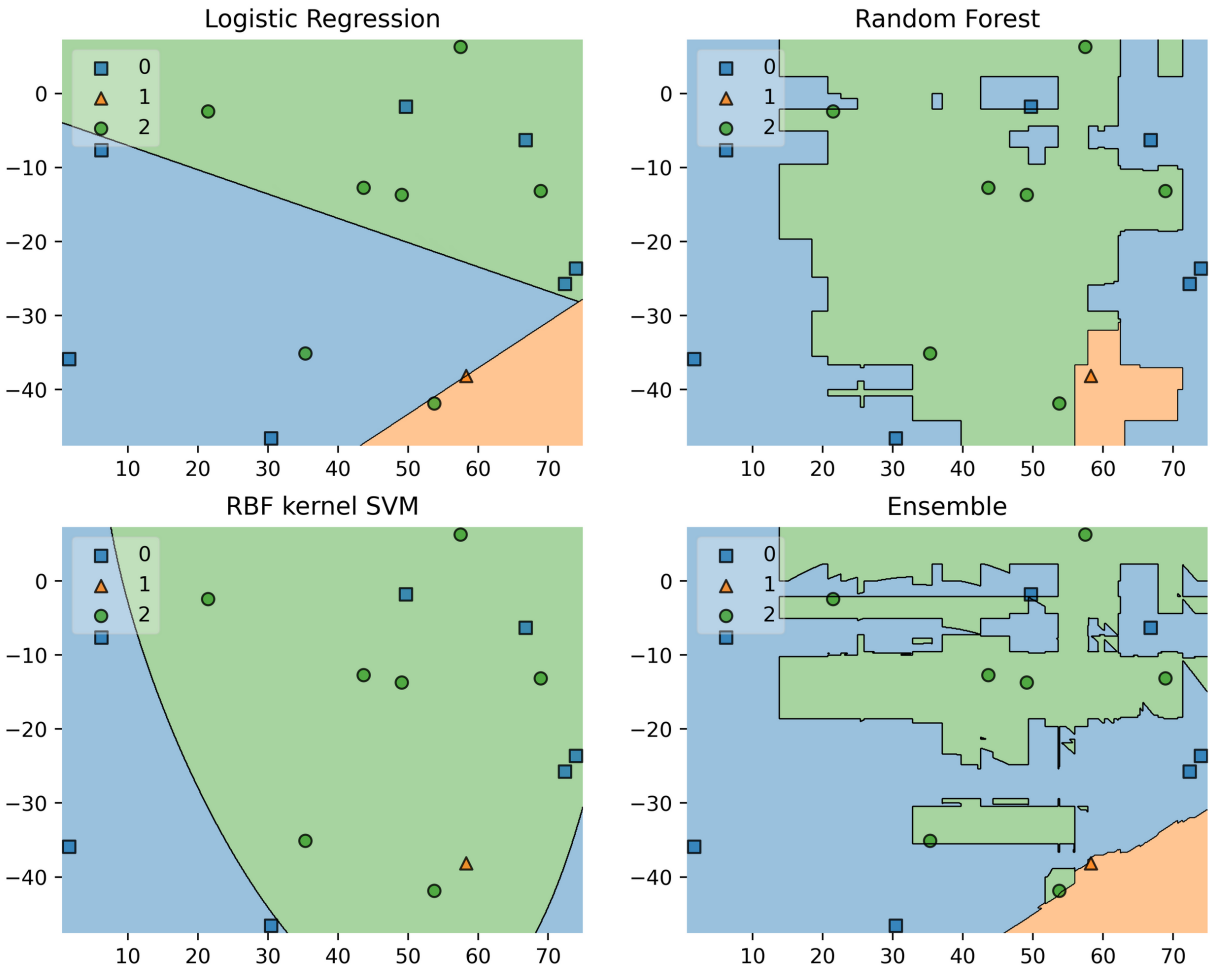
TWO MODELS TO COMPARE: SVM AND XGBOOST

We decided to choose two classification models to compare. Singular Value Machine (SVM) which calculates distance through regressions. And XGBoost which is an ensemble of tree type models (we will plot each one later so you can visualize)

03 Choosing the model

PLOTING THE MODELS

We used mxlextend library to plot 4 type of models using our data to show how each one behaves. We've 2 distance models on the left (logistic regression and SVM) and 2 tree models on the right.



The models we chose are SVM with Kernel RBF (for multiclass targets) and an Ensemble Tree model (XGboost)

SVM RBF KERNEL

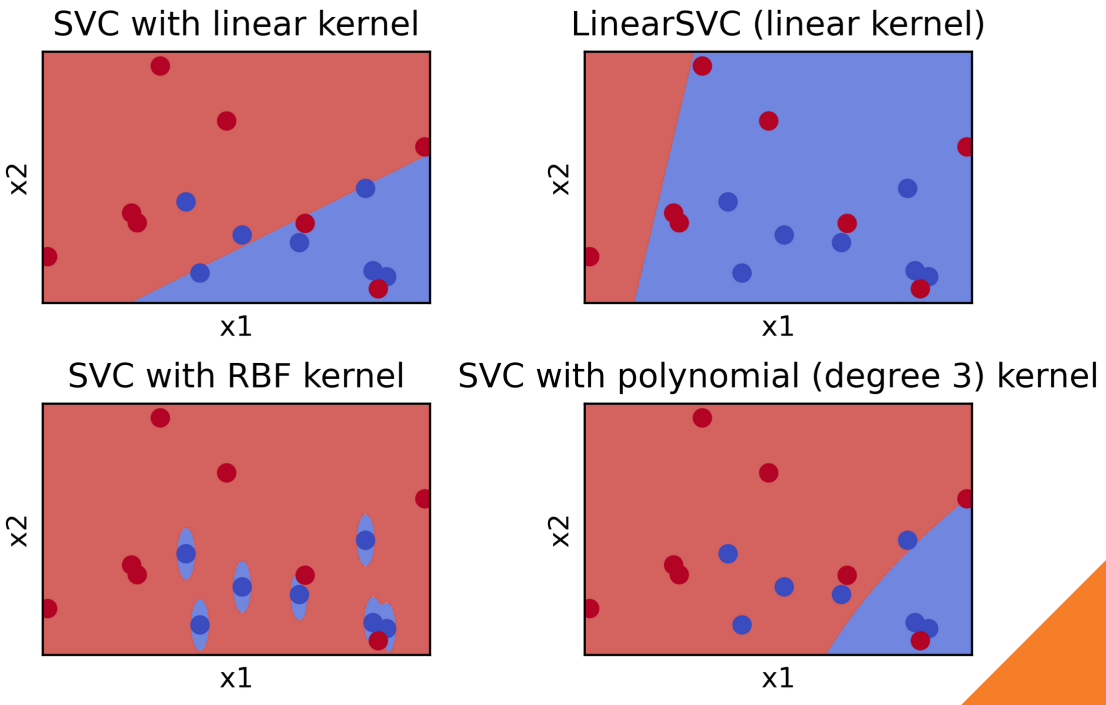
04

It's not easy to score a multiclass model since we have more than two boolean variables to distribute the probability. ROC AUC score, a popular method to score binary models, doesn't work quite right for multiclass.

There are two ways to evaluate a multiclass model: 1) through mean squared error, so that we can detect how many miss we have; 2) Through F-1 Score weightened by the variables, or F-1 Score, which is sometimes called 'Accuracy' for multiclass. But you shouldn't expect values as high as a binary accuracy.

The first thing we've made was to create and evaluate all types of kernels for SVM model and check their accuracy

```
Accuracy Linear Kernel: 0.605
Accuracy Polynomial Kernel: 0.605
Accuracy Radial Basis Kernel: 0.5935
Accuracy Sigmoid Kernel: 0.497
```



Ploting each model on our data so we can see how each one behaves;

SVM RBF KERNEL 04

We chose to compare RBF kernel with Linear Kernel. We used GridSearchCV from sklearn to find the best parameters and tune our models. We tried to use other Grids that are faster and better, but we had compability issues installing it on our virtual machine.

```
# Tuning hyper-parameters for precision



   GridSearchCV(estimator=SVC(),
                param_grid=[{'C': [1, 10, 100, 1000], 'gamma': [0.001, 0.0001],
                             'kernel': ['rbf']},
                            {'C': [1, 10, 100, 1000], 'kernel': ['linear']}],
                scoring='precision_macro')


Best parameters set found on development set:

{'C': 1, 'gamma': 0.001, 'kernel': 'rbf'}
```

```
Detailed classification report:

The model is trained on the full development set.
The scores are computed on the full evaluation set.

              precision    recall  f1-score   support

           0       0.60      1.00      0.75      1801
           1       0.00      0.00      0.00       318
           2       0.00      0.00      0.00       881

    accuracy                           0.60      3000
   macro avg       0.20      0.33      0.25      3000
weighted avg       0.36      0.60      0.45      3000
```

GridSearchCV chose RBF Kernel and got 0.60 F-1 Micro Score. That's not a big improvement after one hour of tuning. Further, it seems to not have taken in account mediums and highs.

SVM RBF KERNEL

04

Ok. Our Squared error are reasonable, but not quite enough. We want to improve our score.

But we want to avoid scaling since we don't know our metrics. What if we do a weightening trick on our ordinal hierarchy (low to high)?

```
SEE YOUR roc_auc SCORE: 0.480815


SEE YOUR RMSE SCORE: 1.131665
SEE YOUR RMSEL SCORE: 0.636686

SEE YOUR F1_weight SCORE: 0.450406
SEE YOUR F1_macro SCORE: 0.250087
SEE YOUR F1_micro SCORE: 0.600333

SVM's accuracy score: 0.6003333333333334
```

```
   SVC(C=1, class_weight='balanced'


SEE YOUR roc_auc SCORE: 0.478954




SEE YOUR RMSE SCORE: 1.282316
SEE YOUR RMSEL SCORE: 0.717899

SEE YOUR F1_weight SCORE: 0.373947
SEE YOUR F1_macro SCORE: 0.305431
SEE YOUR F1_micro SCORE: 0.344667

SVM's accuracy score: 0.344666666666
```

Terrible! It got even worst!!!

```
   StandardScaler()


   SVC(C=1, class_weight='balanced'


SEE YOUR roc_auc SCORE: 0.478954




SEE YOUR RMSE SCORE: 1.206372
SEE YOUR RMSEL SCORE: 0.673846

SEE YOUR F1_weight SCORE: 0.469744
SEE YOUR F1_macro SCORE: 0.286214
SEE YOUR F1_micro SCORE: 0.556667

SVM's accuracy score: 0.55666666666
```

So we tried to scale it and, weel, it got better, but still wors than our first tuned model...

SVM Predict 04

	x1	x2	x3	predict
9394	32.60	5.90	-14.50	0
898	37.38	15.80	-39.15	0
2398	9.80	4.58	-50.60	0
5906	53.71	3.67	-29.92	0
2343	35.05	13.57	-14.26	0
...	...	...	...	...
1037	71.21	9.76	-3.96	0
2899	0.53	3.24	-10.25	0
9549	14.91	8.33	-29.43	0
2740	35.93	3.49	-10.70	0
6690	67.43	5.21	2.22	0

2000 rows x 4 columns

```
0    1806
2     154
1      40
Name: predict, dtype: int64
```

	x1	x2	x3	target	predict	result
0	34.86	7.86	-47.64	1	0	False
1	15.13	9.67	-37.22	1	0	False
2	18.28	3.85	-43.69	1	0	False
3	5.50	24.23	-45.81	1	0	False
4	76.97	8.30	-25.64	1	0	False
...	...	...	...	...	...	...
495	9.52	3.59	-19.32	2	0	False
496	24.00	5.25	-22.56	2	0	False
497	85.92	2.89	-4.30	2	2	True
498	22.87	4.84	-9.08	2	0	False
499	49.80	12.34	6.84	2	0	False

500 rows x 6 columns

```
True     269
False    231
Name: result, dtype: int64
```

We couldn't achieve a good model using SVM. It's almost random. We tried to predict on our test sample (when we separated train and split for our model) and we tried to predict on an unseen data.

The distribution between low, medium and high is tottally different from the original DataFrame. It predicted right only 269 rows against 231 wrong.

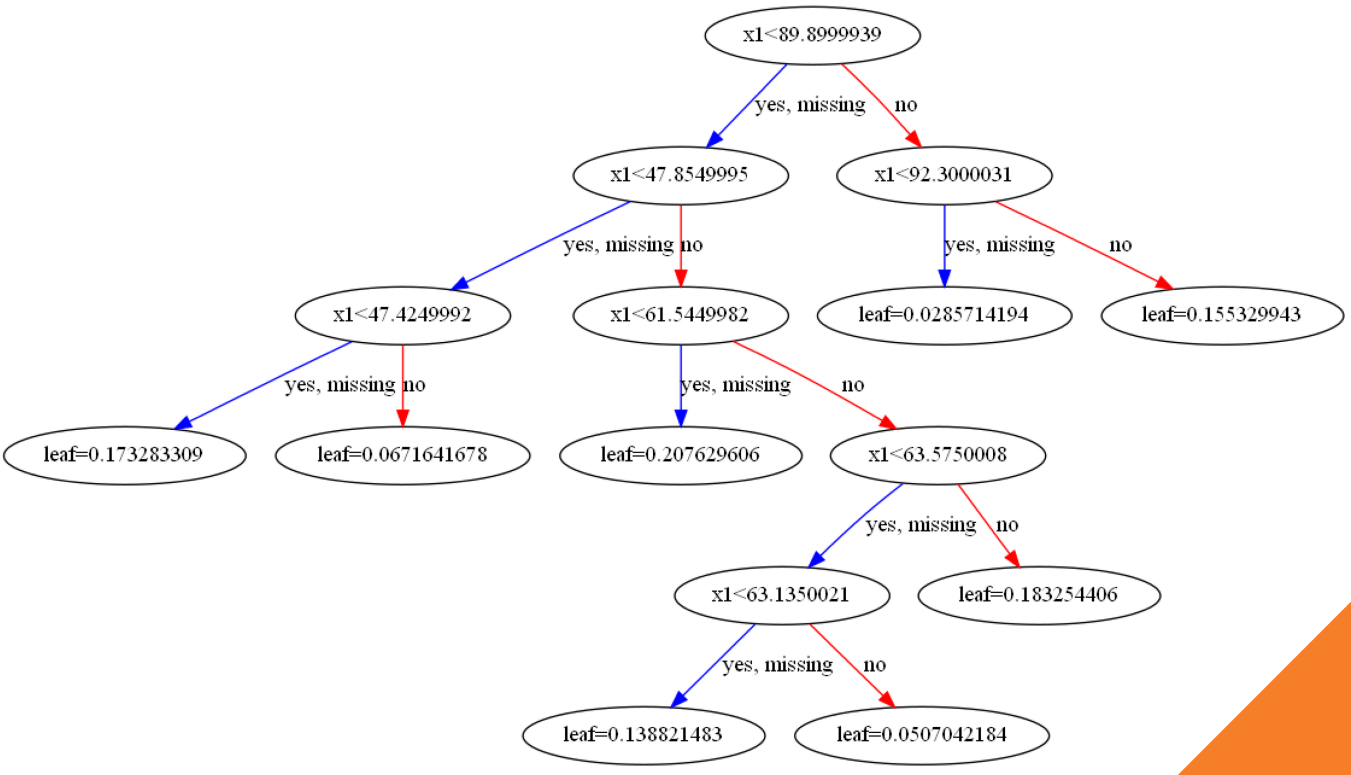
We should note that it's good to compare True and False but with parcimony: since it's a hierarchy from low/medium/high, True or False doesn't always has the same weight.

XGBoost Tree Model

04

We did the same as before. We split our data on train and split and also got an unseen data to predict (another kind of split). We instantiated our model, fit it, run and got the scores. After that we tuned our model and compared with the first score. After that we tried to predict on our unseen data and on our test data.

> This time we kept our attention to mean squared error since it seems to be a better metric for performance of the model. Now we didn't need to scale it because it's an ensemble of tree models which its decision it's not affected considerably by scale. This is how it looks on our data:



XGBoost Score

04

	train-logloss-mean	train-logloss-std	test-logloss-mean	test-logloss-std
0	0.677632	0.000523	0.678158	0.002154
1	0.664979	0.000882	0.666014	0.004174
2	0.654679	0.001225	0.656016	0.006038
3	0.646232	0.001572	0.647921	0.007796
4	0.639128	0.001778	0.641391	0.009381

```
18    0.619537
Name: test-logloss-mean, dtype: float64
```

	train-rmse-mean	train-rmse-std	test-rmse-mean	test-rmse-std
0	0.917757	0.001677	0.917739	0.015519
1	0.914344	0.001604	0.914427	0.015106
2	0.911540	0.001497	0.911715	0.014694
3	0.909270	0.001440	0.909523	0.014323
4	0.907377	0.001422	0.907768	0.013995

```
16    0.901816
Name: test-rmse-mean, dtype: float64
```

	train-rmsle-mean	train-rmsle-std	test-rmsle-mean	test-rmsle-st
0	0.501335	0.000551	0.501379	0.004671

	train-mae-mean	train-mae-std	test-mae-mean	test-mae-std
0	0.803878	0.001678	0.80396	0.014154

XGBoost Predict 04

	x1	x2	x3	target	predict	result
0	34.86	7.86	-47.64	1	1	True
1	15.13	9.67	-37.22	1	1	True
2	18.28	3.85	-43.69	1	0	False
3	5.50	24.23	-45.81	1	2	False
4	76.97	8.30	-25.64	1	1	True
...	...	...	...	...	...	...
495	9.52	3.59	-19.32	2	2	True
496	24.00	5.25	-22.56	2	2	True
497	85.92	2.89	-4.30	2	2	True
498	22.87	4.84	-9.08	2	2	True
499	49.80	12.34	6.84	2	2	True

```
True     424
False     76
Name: result, dtype: int64
```

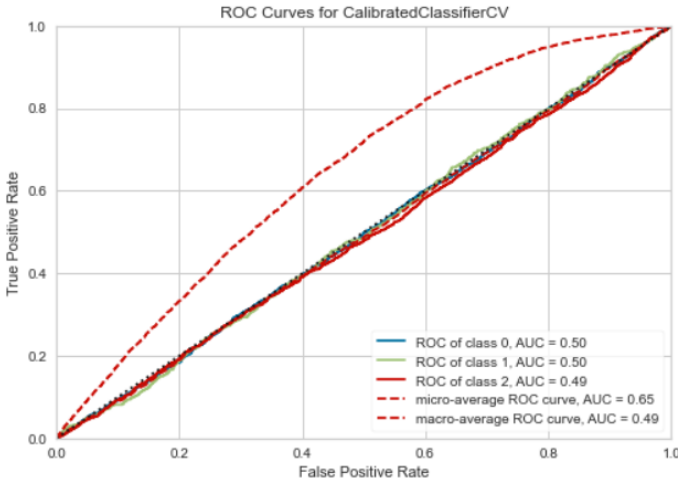
Well. It predicted really well! Our model got it right on 424 of 500! And we should consider that False doesn't always have the same weight, since if it predicted 'high' it isn't the same mistake if the correct answer is 'medium' or 'low'.

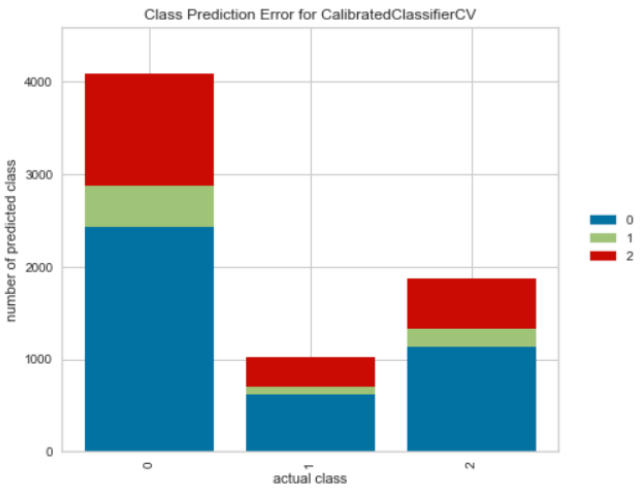
WE HAVE OUR MODEL! XGBOOST WITH THE FOLLOWING PARAMETERS:

"objective":"reg:squarederror",'colsample_bytree': 0.3,'learning_rate': 0.1, 'max_depth': 25, 'alpha': 10, 'gamma':0.5, 'min_child_weight':10, 'reg_lambda': 5, 'scale_pos_weight': 1

BONUS: PyCaret 05

I already have a template I've made for use on any classification model with PyCaret. It automates all the modelling process much further from what we did for this challenge. It automatess preprocess, tune, ensemble, calibration, threshold, prediction, SHAP interpratation, plots, save the model and finalize the model, also you can inspect your model with Mlflow. I've applied this template for our Hospital Challenge. The best model PyCaret chose was a Tree Ensemble.





BONUS: PyCaret 05

It was pretty fast to set and run. I didn't have many errors to debug. The results were consistent and it was much faster then setting our previous models. Although our XGBoost model performed better than our PyCaret Template, It could get better if more work would be done on the template our the application on this dataset.

	x1	x2	x3	target	Label	Score	result
0	34.86	7.86	-47.64	1	1	0.7990	True
1	15.13	9.67	-37.22	1	0	0.4600	False
2	18.28	3.85	-43.69	1	2	0.5647	False
3	5.50	24.23	-45.81	1	0	0.7732	False
4	76.97	8.30	-25.64	1	1	0.7929	True
...	...	...	...	...	...	...	...
495	9.52	3.59	-19.32	2	0	0.6942	False
496	24.00	5.25	-22.56	2	2	0.8180	True
497	85.92	2.89	-4.30	2	0	0.4576	False
498	22.87	4.84	-9.08	2	2	0.6863	True
499	49.80	12.34	6.84	2	2	0.4321	True

500 rows x 7 columns

```
True     308
False    192
Name: result, dtype: int64
```

PyCaret Prediction

	x1	x2	x3	target	predict	result
0	34.86	7.86	-47.64	1	1	True
1	15.13	9.67	-37.22	1	1	True
2	18.28	3.85	-43.69	1	0	False
3	5.50	24.23	-45.81	1	2	False
4	76.97	8.30	-25.64	1	1	True
...	...	...	...	...	...	...
495	9.52	3.59	-19.32	2	2	True
496	24.00	5.25	-22.56	2	2	True
497	85.92	2.89	-4.30	2	2	True
498	22.87	4.84	-9.08	2	2	True
499	49.80	12.34	6.84	2	2	True

500 rows x 6 columns

```
True     424
False     76
Name: result, dtype: int64
```

XGBooost Prediction

BONUS: SHAP 06

We could use a beautiful library called SHAP to interpret our data with our model. But in this case we should know the nature and meaning of our data as we should know our objective. Actually, this is essential for any data analysis and to statistic modelling. If we are modelling to predict cancer, for example, we should know which weight we need to give for false negatives inside our confusion matrix because the cost of a false negative is really high, much higher than a false positive.

HOW SHAP LOOKS:

CONCLUSION 07

There are many ways to select a model for specific datas. There are also many ways to improve our model. Also, from time to time, our model needs to be retrained as we get more data or the behaviour of our data changes on consequence of a structural change on ints nature. We explored only some of those methods. We compared SVM model with XGBoost model and we've got a better prediction with XGBoost. Actually, almost always for a multiclass problem it's better to use a Tree Model or an ensemble from Tree Models or Neural Networks (that are more resource consuming from our computational equipment and can take more time to process it).

LINKS 08

- **Scripts**
- **Repository**: https://github.com/ggnicolau/Albert_Einstein
- **Python Files**: https://github.com/ggnicolau/Albert_Einstein/tree/main/py_files
- **Profiling Report**: https://github.com/ggnicolau/Albert_Einstein/tree/main/Profiling_report
- **Pictures**: https://github.com/ggnicolau/Albert_Einstein/tree/main/Pictures
- **Pickle Files**: https://github.com/ggnicolau/Albert_Einstein/tree/main/Models_backup

Technical articles

- https://towardsdatascience.com/machine-learning-multiclass-classification-with-imbalanced-data-set-29f6a177c1a
- https://medium.com/analytics-vidhya/multi-class-ml-models-evaluation-103c9fdadb41
- https://towardsdatascience.com/xgboost-for-multi-class-classification-799d96bcd368
- https://medium.com/swlh/support-vector-regression-explained-for-beginners-2a8d14ba6e5d
- https://towardsdatascience.com/machine-learning-multiclass-classification-with-imbalanced-data-set-29f6a177c1a
- https://machinelearningmastery.com/xgboost-for-imbalanced-classification/
- https://towardsdatascience.com/multiclass-classification-with-support-vector-machines-svm-kernel-trick-kernel-functions-f9d5377d6f02
- https://medium.com/pursuitnotes/support-vector-regression-in-6-steps-with-python-c4569acd062d
- https://www.baeldung.com/cs/svm-multiclass-classification
- https://medium.com/@b.terryjack/tips-and-tricks-for-multi-class-classification-c184ae1c8ffc
- https://towardsdatascience.com/machine-learning-multiclass-classification-with-imbalanced-data-set-29f6a177c1a
- https://towardsdatascience.com/multi-class-metrics-made-simple-part-ii-the-f1-score-ebe8b2c2ca1
- https://medium.com/swlh/visualizing-svm-with-python-4b4b238a7a92
- https://stackoverflow.com/questions/41592661/determining-the-most-contributing-features-for-svm-classifier-in-sklearn
- https://antonhaugen.medium.com/intro-to-support-vector-machines-2bfbfec6517a
- https://towardsdatascience.com/svm-kernels-what-do-they-actually-do-56ce36f4f7b8
- https://stackoverflow.com/questions/43284811/plot-svm-with-matplotlib
- https://edisciplinas.usp.br/pluginfile.php/4144383/mod_resource/content/1/svm.pdf
- https://jakevdp.github.io/PythonDataScienceHandbook/05.07-support-vector-machines.html
- https://stackoverflow.com/questions/31265110/does-gridsearchcv-not-support-multi-class
- https://towardsdatascience.com/20x-times-faster-grid-search-cross-validation-19ef01409b7c
- https://stackoverflow.com/questions/59666138/sklearn-roc-auc-score-with-multi-class-ovr-should-have-none-average-available
- https://medium.com/analytics-vidhya/root-mean-square-log-error-rmse-vs-rmlse-935c6cc1802a
- https://stackoverflow.com/questions/31421413/how-to-compute-precision-recall-accuracy-and-f1-score-for-the-multiclass-case
- https://towardsdatascience.com/multi-class-metrics-made-simple-part-ii-the-f1-score-ebe8b2c2ca1
- https://medium.com/mlearning-ai/shap-force-plots-for-classification-d30be430e195
- https://medium.com/fiddlerlabs/case-study-explaining-credit-modeling-predictions-with-shap-2a7b3f86ec12