Machine Learning in Computational Biology

Georgios Leventis

7115172100024

Assignment #2

https://github.com/geoleven/Assignment-2

The exercise was done according to its description as much as possible.
At the relevant github you can find the final model, as well as the two notebooks and the python code file.

Exploratory Data Analysis:

For the data exploration task I did all that is considered normal, from showing info and description of the datasets. The data had no duplicates but quite some null values which would reduce the sample to 384 if we remove all the relevant lines.

```
Duplicate Values:

0
Missing Values:

id                         0
diagnosis                  0
radius_mean                7
texture_mean               5
perimeter_mean             5
area_mean                  4
smoothness_mean            6
compactness_mean           1
concavity_mean            10
concave points_mean        8
symmetry_mean              5
fractal_dimension_mean     3
radius_se                  2
texture_se                 7
perimeter_se               1
area_se                    3
smoothness_se              6
compactness_se             4
concavity_se               2
concave points_se          4
symmetry_se                5
fractal_dimension_se       5
radius_worst               5
texture_worst              2
perimeter_worst            7
area_worst                 7
smoothness_worst           5
compactness_worst          9
concavity_worst            4
concave points_worst      11
symmetry_worst             3
fractal_dimension_worst    7
dtype: int64
```

As such for the next tasks, instead of choosing to remove them we fill them with median.

The diagnosis is converted with one-hot encoding for better compatibility.

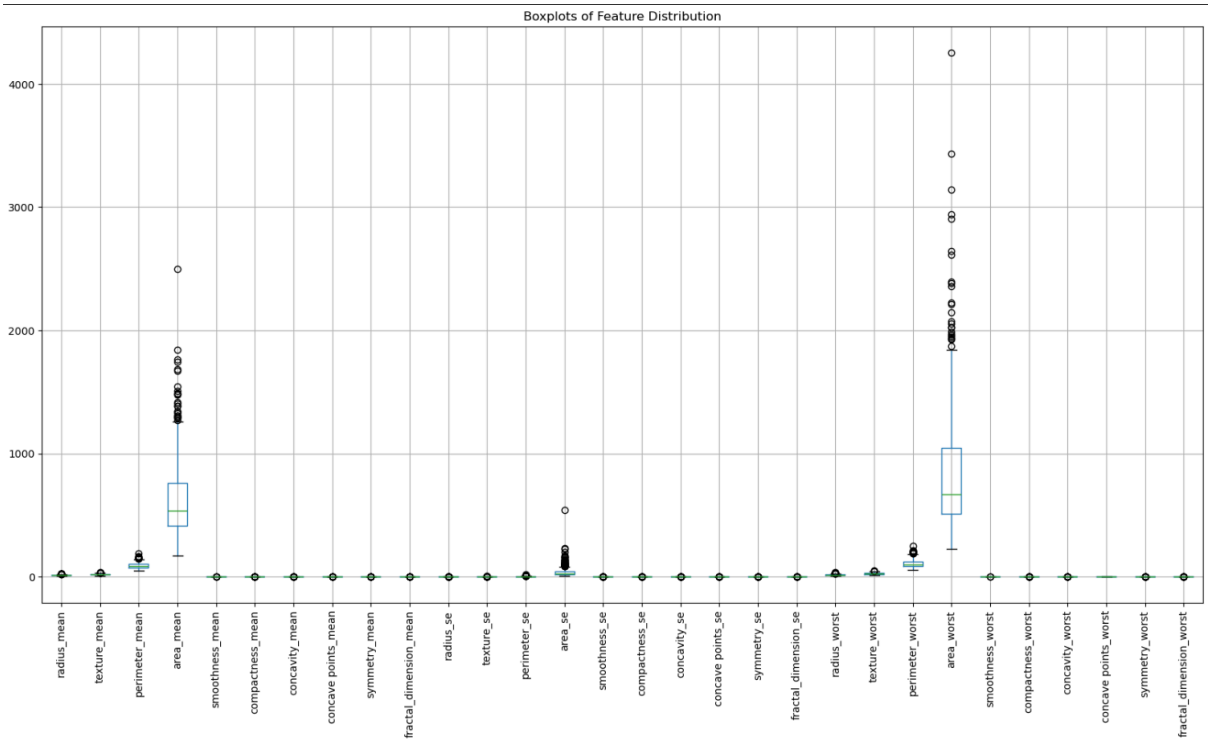
Then I calculated the class balances (which are not exactly balanced, something that plays role to the decisions made later with the models).

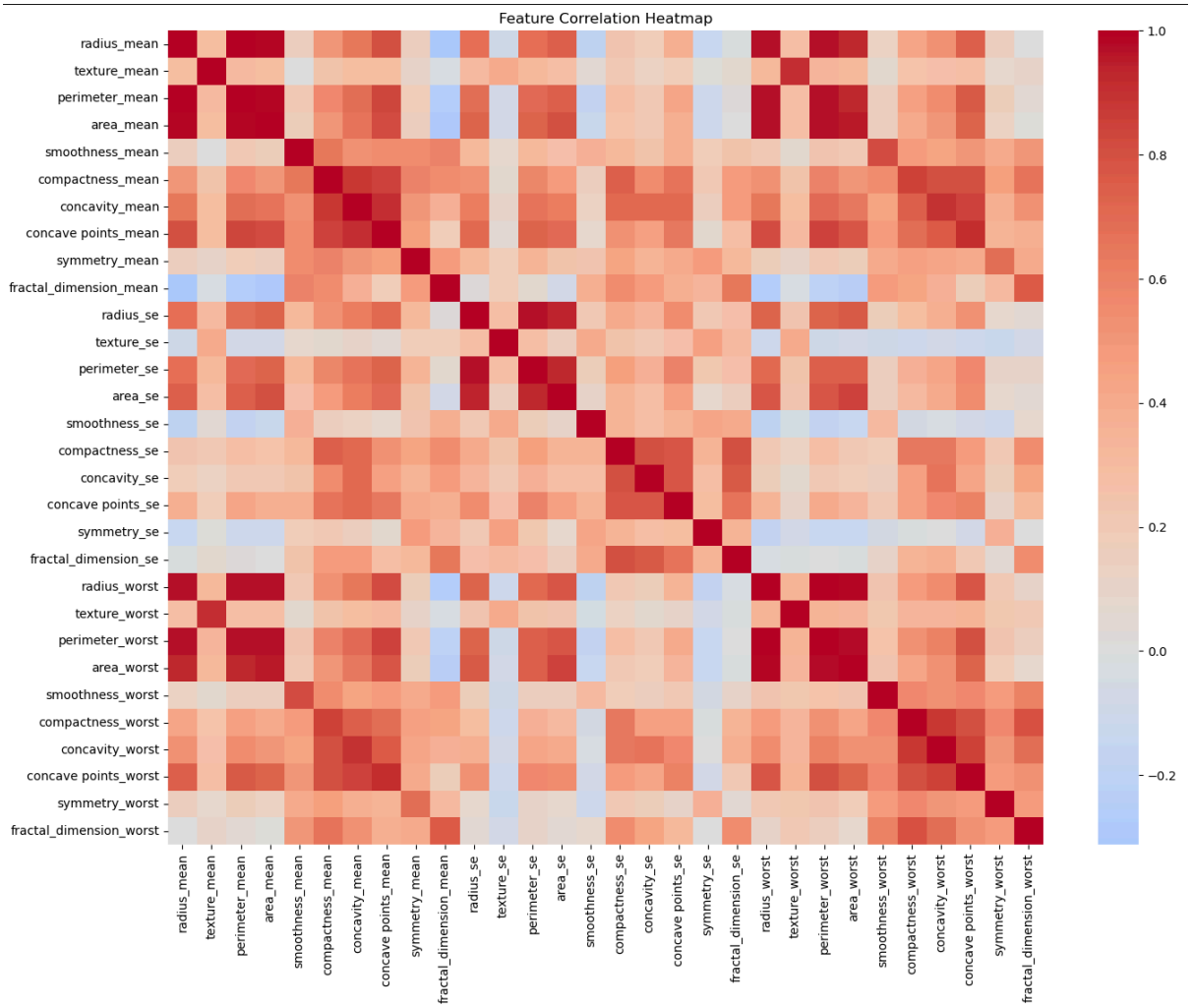
```
Class imbalance:

diagnosis
0    0.640625
1    0.359375
Name: proportion, dtype: float64
```

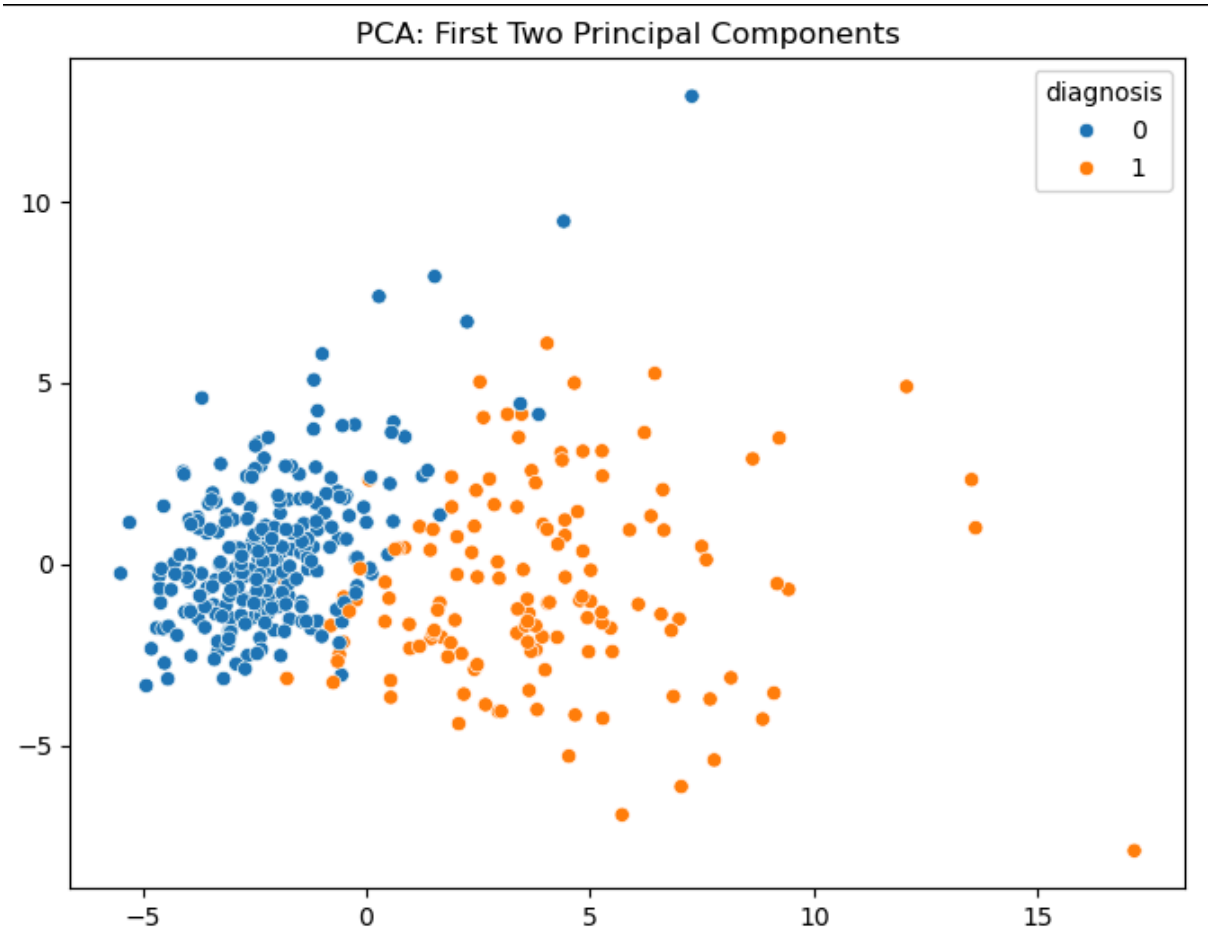
Then I calculated the IQR to see the outliers which were few:

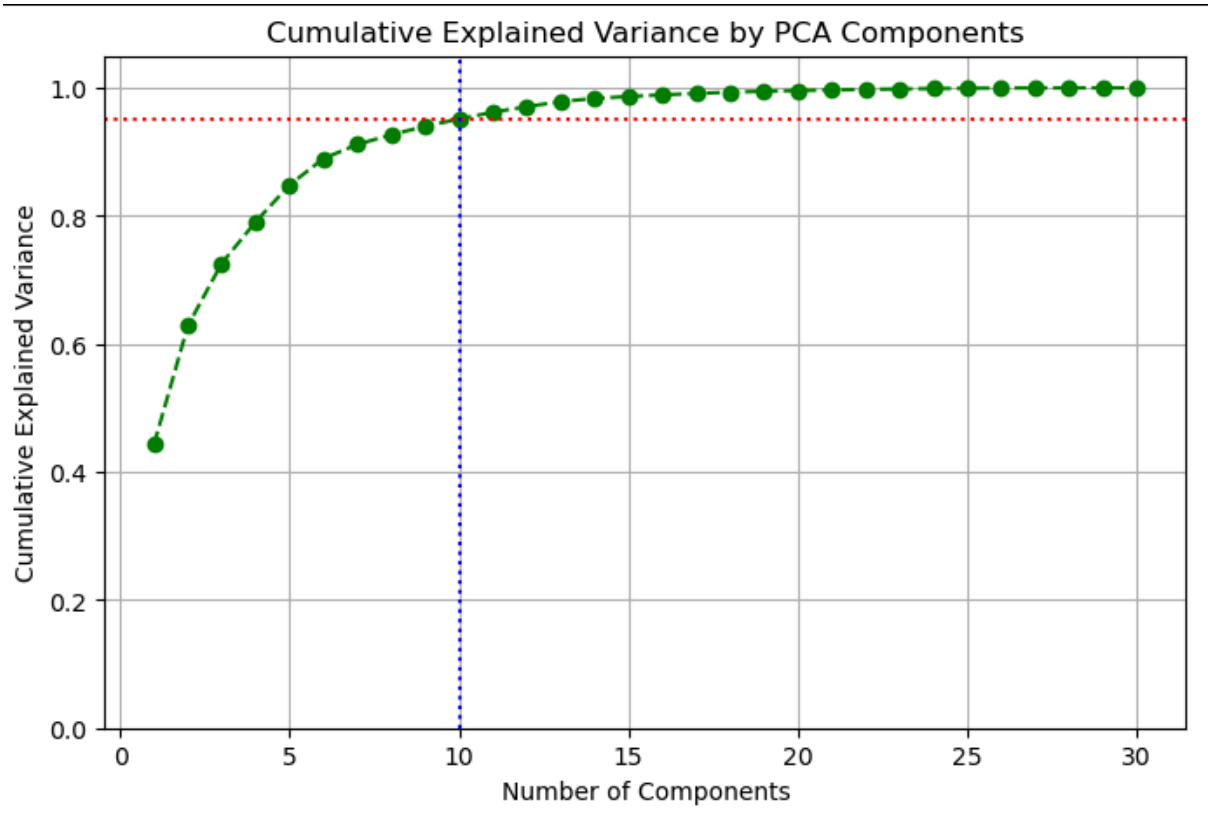
```
{'radius_mean': 4.017500000000002,
 'texture_mean': 5.495000000000001,
 'perimeter_mean': 28.427499999999995,
 'area_mean': 342.425,
 'area_se': 26.095,
 'radius_worst': 5.540000000000003,
 'texture_worst': 8.504999999999999,
 'perimeter_worst': 40.3275,
 'area_worst': 533.5}
```

And finally made the relevant plots to show the correlation heatmap but also the PCA and its relevant values.



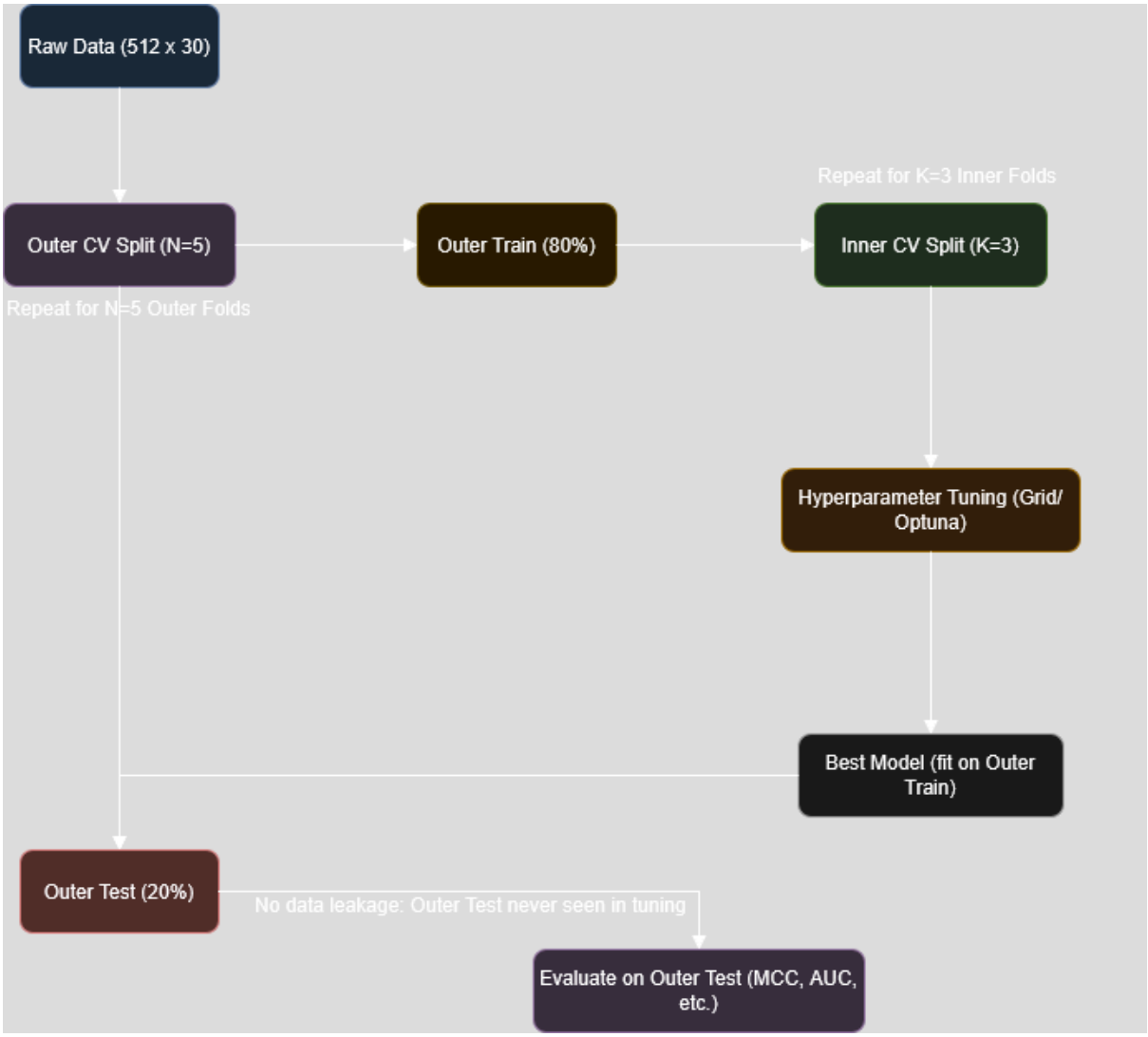

Feature Correlation Heatmap
radius_mean
texture_mean
perimeter_mean
area_mean
smoothness_mean
compactness_mean
concavity_mean
concave points_mean
symmetry_mean
fractal_dimension_mean
radius_se
texture_se
perimeter_se
area_se
smoothness_se
compactness_se
concavity_se
concave points_se
symmetry_se
fractal_dimension_se
radius_worst
texture_worst
perimeter_worst
area_worst
smoothness_worst
compactness_worst
concavity_worst
concave points_worst
symmetry_worst
fractal_dimension_worst
1.0
0.8
0.6
0.4
0.2
0.0
−0.2


PCA: First Two Principal Components
diagnosis
0
1
10
5
0
−5
−5
0
5
10
15


Cumulative Explained Variance by PCA Components
Cumulative Explained Variance
1.0
0.8
0.6
0.4
0.2
0.0
0
5
10
15
20
25
30
Number of Components

We can clearly see that 10 of the feats are enough. However, as per the assignment we keep everything till the next few steps.

Intuition & Preparation for the nCV implementation:



Nested Cross Validation implementation:

For step 1 I wrote all the relevant code (given at my repo) that does all what is asked for with a wrapper class to adhere to the OOP that is being asked. I initially tried to use ATOM-ml but it ended up being more of a hassle (major incompatibilities with my system's libstdcxx.so.6 and no proper conda make) and after I decided that it had already wasted more time than it would ever "win" I just used optuna and scikit learn for everything.
For the LightGBM I used the standalone version which meant I had to still install and tinker with some parts of my system.

For step 2 I get the results for all the metrics for all the estimators and print the median as well as the range of the 95 percentile. I made any further decisions according to these.
I am attaching here one of the multiple different runs that I had to get these results in multiple ways:

	LogisticRegression	GaussianNB	LDA	SVM	RandomForest	LightGBM
MCC	(0.8185610964404444, [0.7081759933729 16, 0.876...	(0.8545865420294878, [0.7888634213269922, 0.93...	(0.8986132476675199, [0.8179669523347212, 0.95...	(0.8955476521397039, [0.7931990809775066, 0.93...	(0.8959004429201081, [0.8126711590701251, 0.97...	(0.9169613440442888, [0.8526078664776954, 0.97...
AUC	(0.9614488444669366, [0.9281052315 2834, 0.9884...	(0.9878700657894737, [0.9688939144736842, 0.99...	(0.9932154605263158, [0.9754214638157894, 0.99...	(0.9901579369095816, [0.9687088815789474, 0.99...	(0.9884868421052 63, [0.971225645242915, 0.9994...	(0.9923930921052632, [0.9747841282894736, 1.0])
BalancedAccuracy	(0.8999209261133604, [0.8370147499156546, 0.94...	(0.9258392375168691, [0.8845383666497976, 0.96...	(0.9395559210526316, [0.881578947368421, 0.979...	(0.9449013157894737, [0.8797800164473684, 0.97...	(0.9488075657894737, [0.9045358362854251, 0.98...	(0.9568256578947368, [0.9197882401315 79, 0.990...
F1	(0.8828571428571428, [0.8024324324324325, 0.92...	(0.9078787878787878, [0.8580357142857142, 0.95...	(0.9315068493150684, [0.8656716417910447, 0.97...	(0.9333333333333333, [0.8571832579185 52, 0.961...	(0.9350649350649 35, [0.8807012987012987, 0.983...	(0.9473684210526315, [0.9046849315068493, 0.98...
F2	(0.8597950268817205, [0.7489639037433155, 0.92...	(0.8994708994708994, [0.8227459016393442, 0.95...	(0.9090909090909091, [0.8011049723756906, 0.97...	(0.9259259259259259, [0.8120396495396495, 0.96...	(0.9317467754382096, [0.8666988416988417, 0.97...	(0.9358288770053476, [0.8834203036053131, 0.99...
Recall	(0.8421052631578947, [0.7121963562753036, 0.94...	(0.8947368421052632, [0.7953947368421053, 0.94...	(0.8947368421052632, [0.7631578947368421, 0.97...	(0.9210526315789473, [0.7809210526315 79, 0.974...	(0.9230769230769231, [0.8480263157894736, 0.97...	(0.9230769230769231, [0.8684210526315 79, 1.0])
Specificity	(0.961298076923077, [0.9108834134615384, 0.996...	(0.96875, [0.90625, 0.9965384615384615])	(1.0, [0.984375, 1.0])	(0.96875, [0.921875, 1.0])	(0.9765625, [0.922145432692307 8, 1.0])	(0.984375, [0.953125, 1.0])
Precision	(0.9260752688172043, [0.8435406698564593, 0.99...	(0.9444444444444444, [0.8486263736263736, 0.99...	(1.0, [0.96796875, 1.0])	(0.9473684210526315, [0.8738553113553114, 1.0])	(0.9575551782682512, [0.8725160256410257, 1.0])	(0.9722222222222222, [0.9189189189189189, 1.0])
PRAUC	(0.8395793354859089, [0.7486422910005747, 0.89...	(0.8754104512618445, [0.8001999363776586, 0.93...	(0.9188896107949882, [0.8507292825152545, 0.96...	(0.9022744666436993, [0.8041296538858459, 0.94...	(0.9006777675508325, [0.8201063861464685, 0.97...	(0.9260673563421618, [0.8614062979388057, 0.97...
NPV	(0.9111062335381914, [0.8481512762762763, 0.96...	(0.9384615384615385, [0.8895142916321459, 0.96...	(0.9402985074626866, [0.8767123287671232, 0.98...	(0.9538461538461539, [0.8835597826086956, 0.98...	(0.9545454545454546, [0.9155632411067193, 0.98...	(0.9555531167690957, [0.9256200614574188, 1.0])

I choose to go with LightGBM as I think the range of its 95 percentile is better than other good options such as the LDA.

I finally found the best hyperparameters for the LightGBM:

```
Best hyperparameters: {'learning_rate': 0.5, 'n_estimators': 100, 'num_leaves': 31}
```

And trained my model which can be found in my repo.

AI disclosure:

I use perplexity.ai (with a mixture of models -> Chatgpt, Claude, etc).
The main usage (as you ask about it) is helper functions (especially data edits for plots) and cutting on the writing in general (AI is the best anti-tenontitis drug to this day 😀). Also, I sometimes use it to quickly learn about some bits and bytes that I may be missing to understand something, have it formulate some hands on examples I may need to understand a data structure, etc.