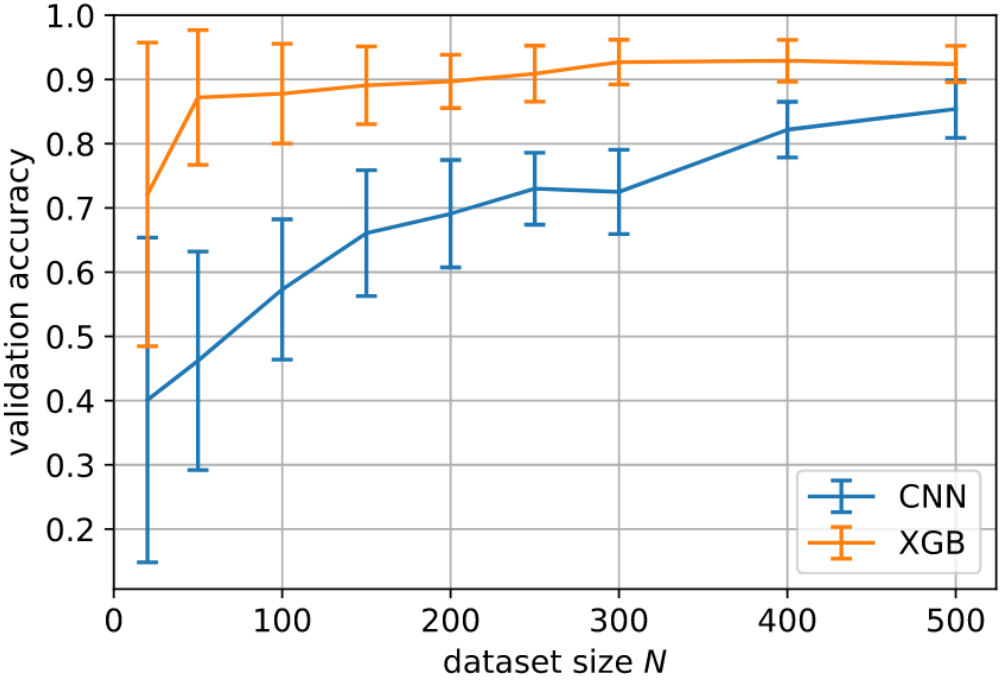
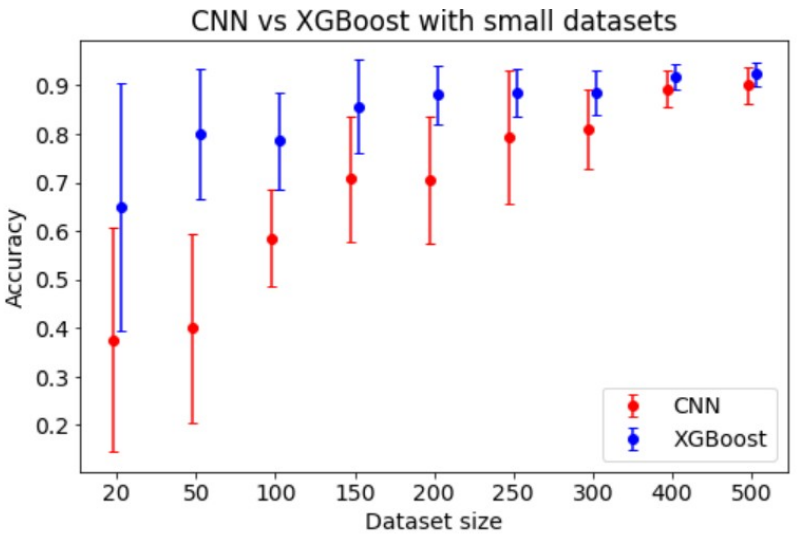
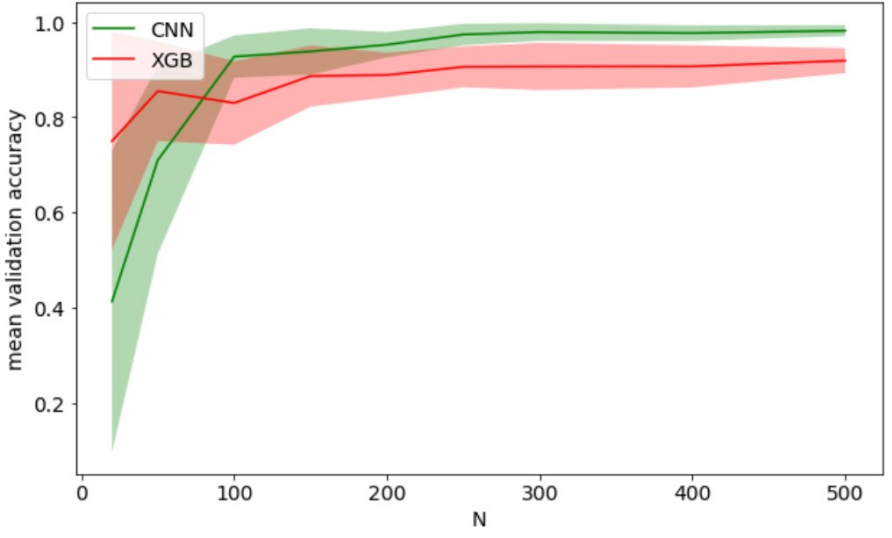
XGBoost

CNN vs XGB

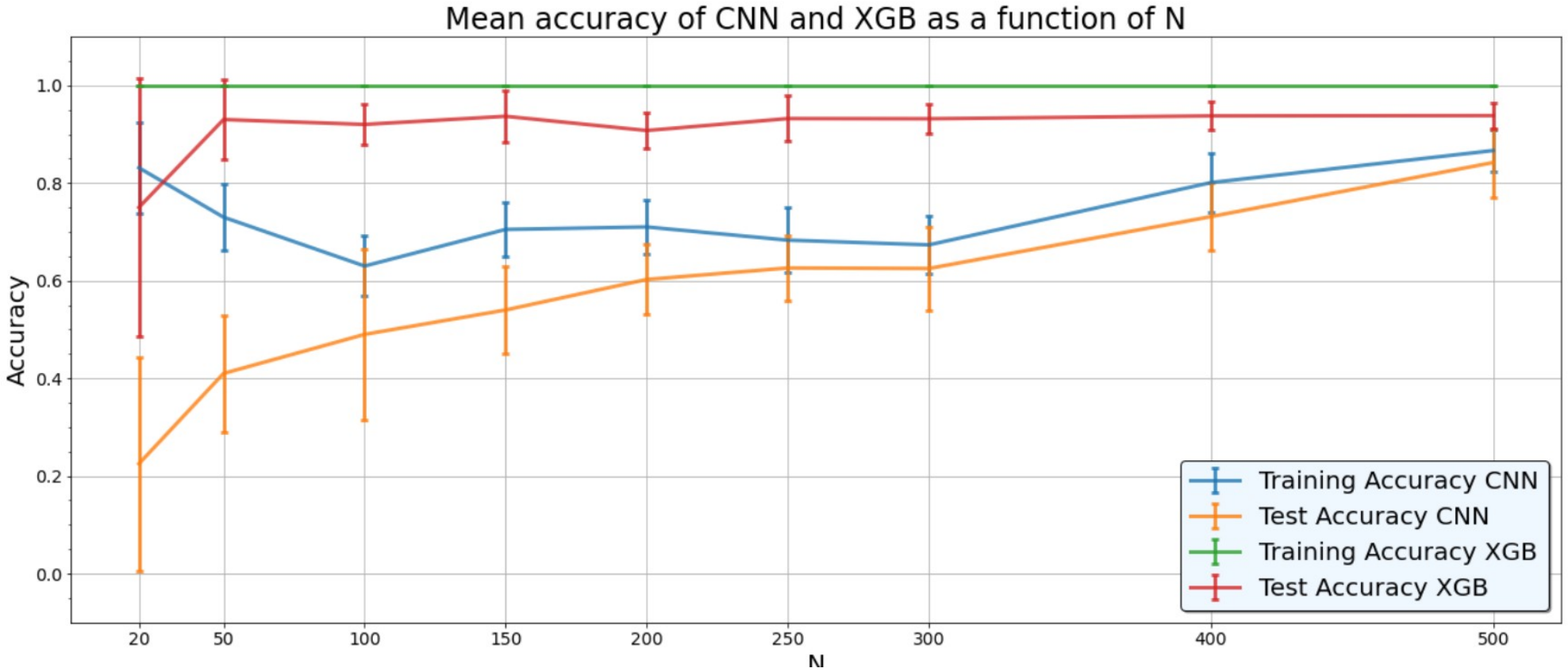

validation accuracy
dataset size N
CNN
XGB


CNN vs XGBoost with small datasets
Accuracy
Dataset size
CNN
XGBoost

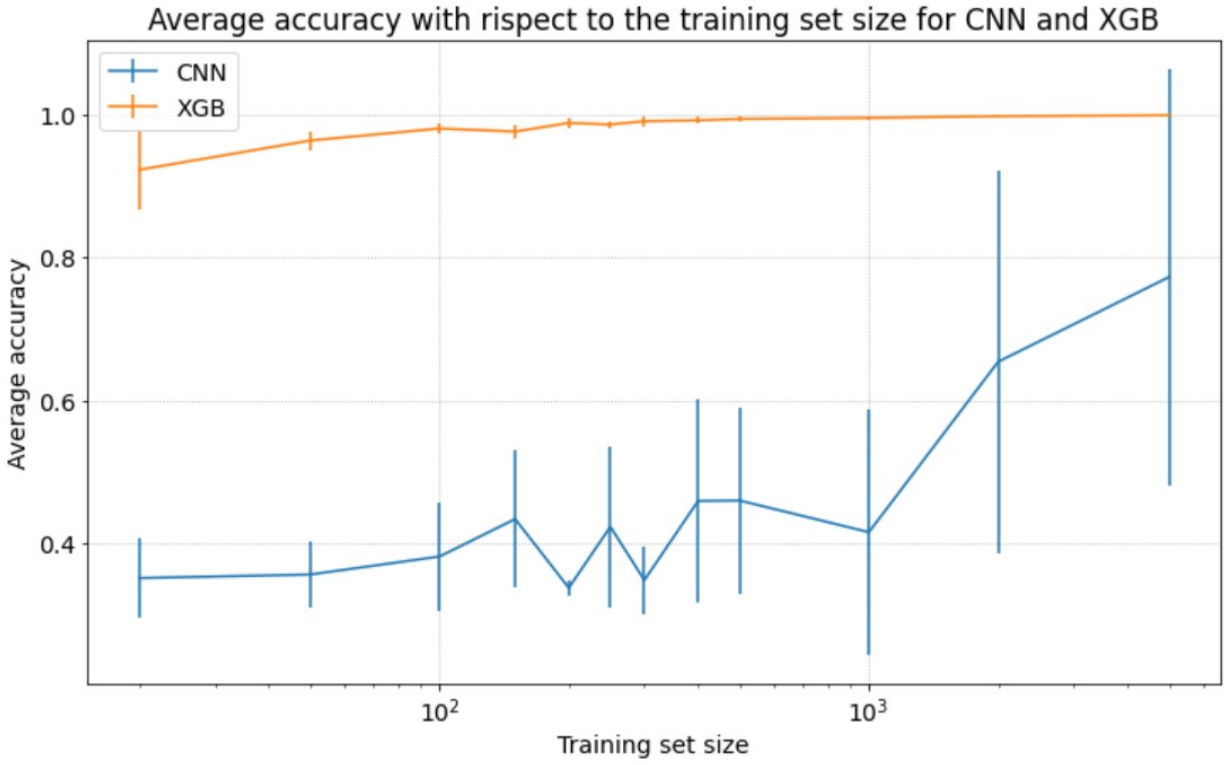

mean validation accuracy
N
CNN
XGB

CNN vs XGB: overfit

2213



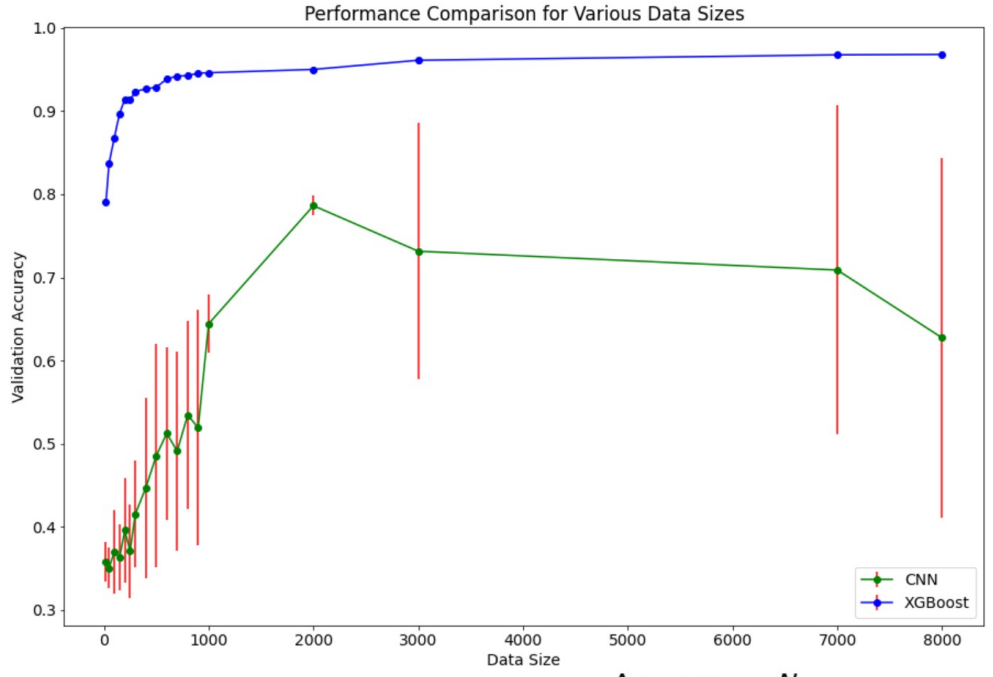
2216



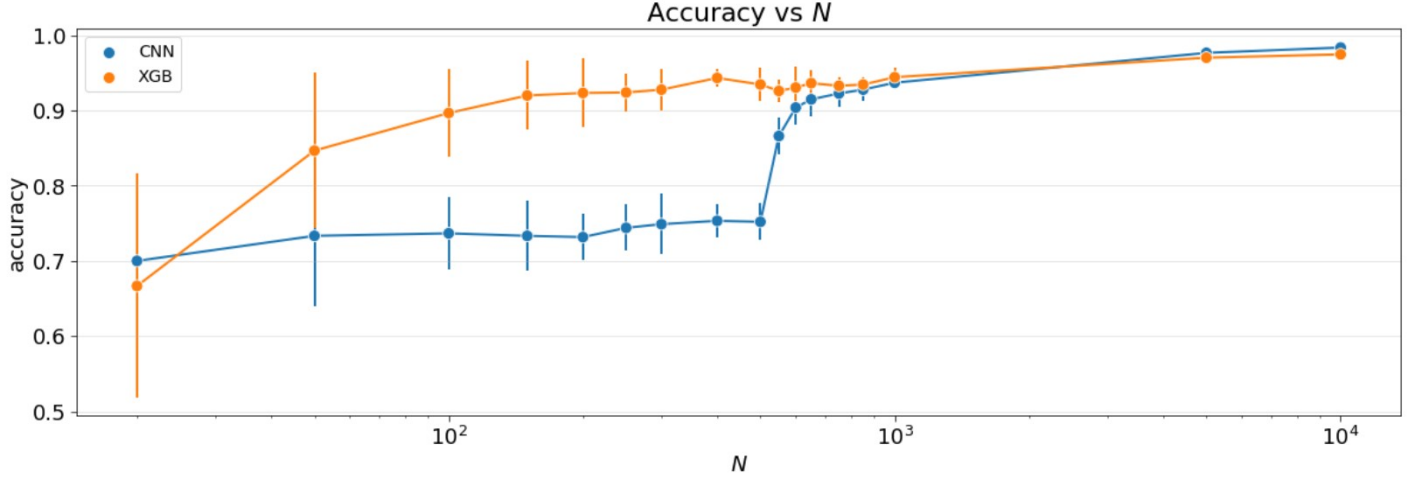
CNN vs XGB

CNN vs XGB

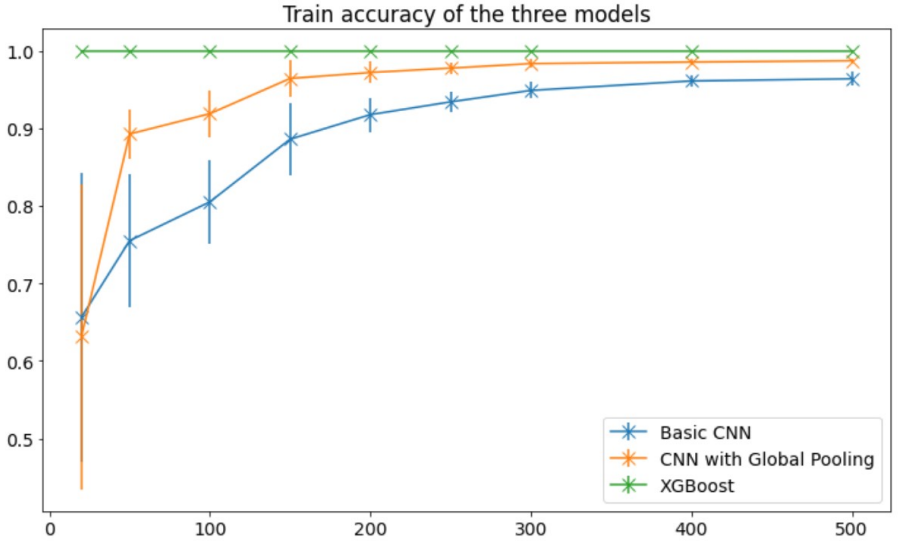
2220



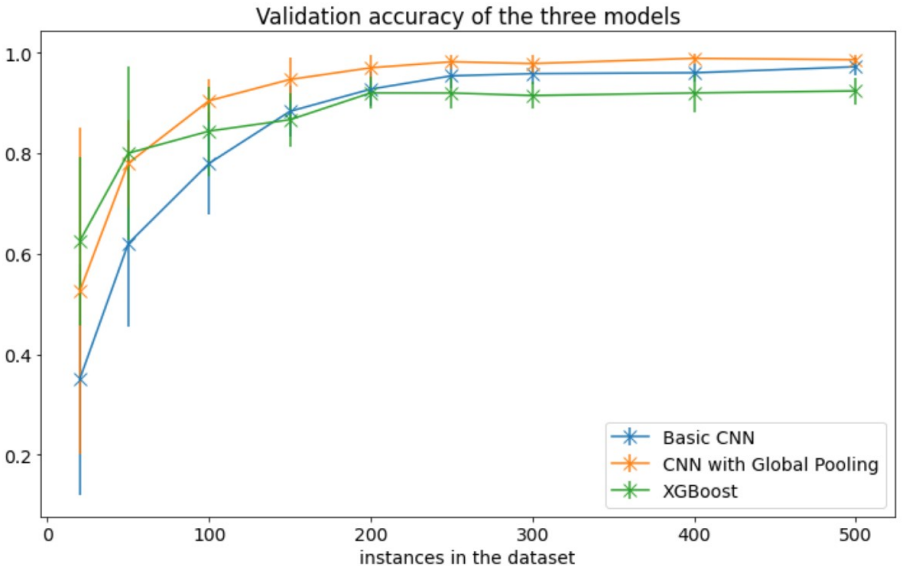
2207



train



test

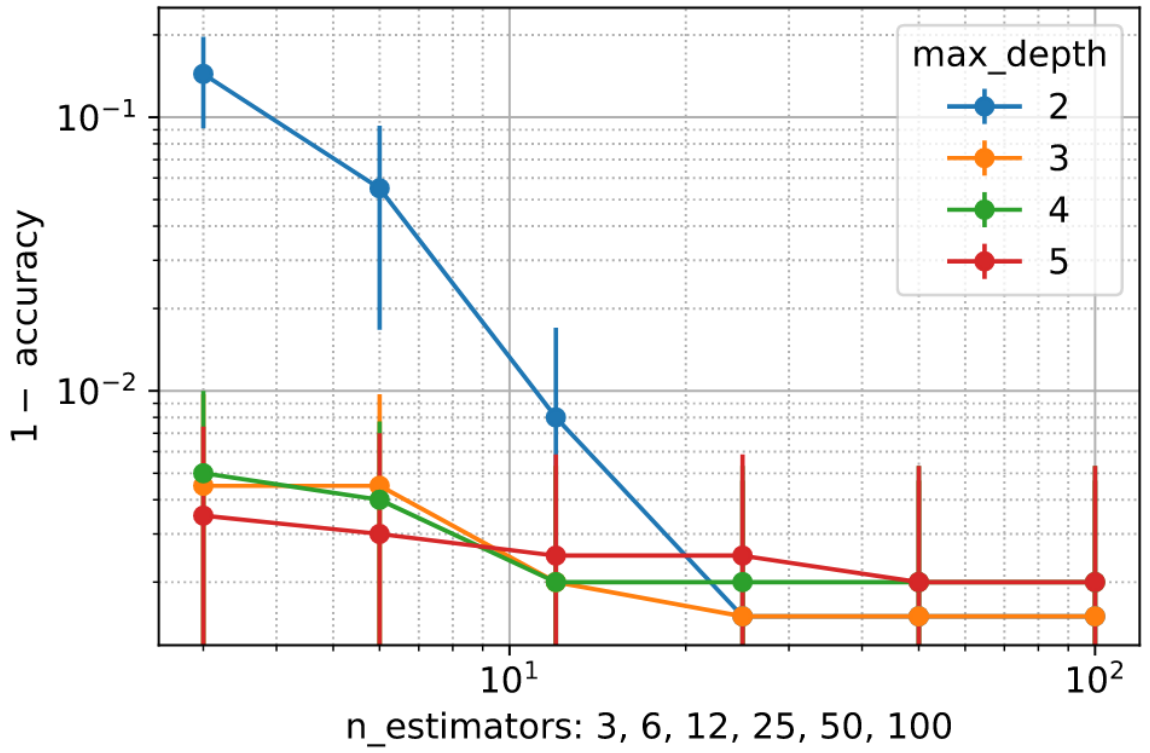


CNN vs XGB

2202

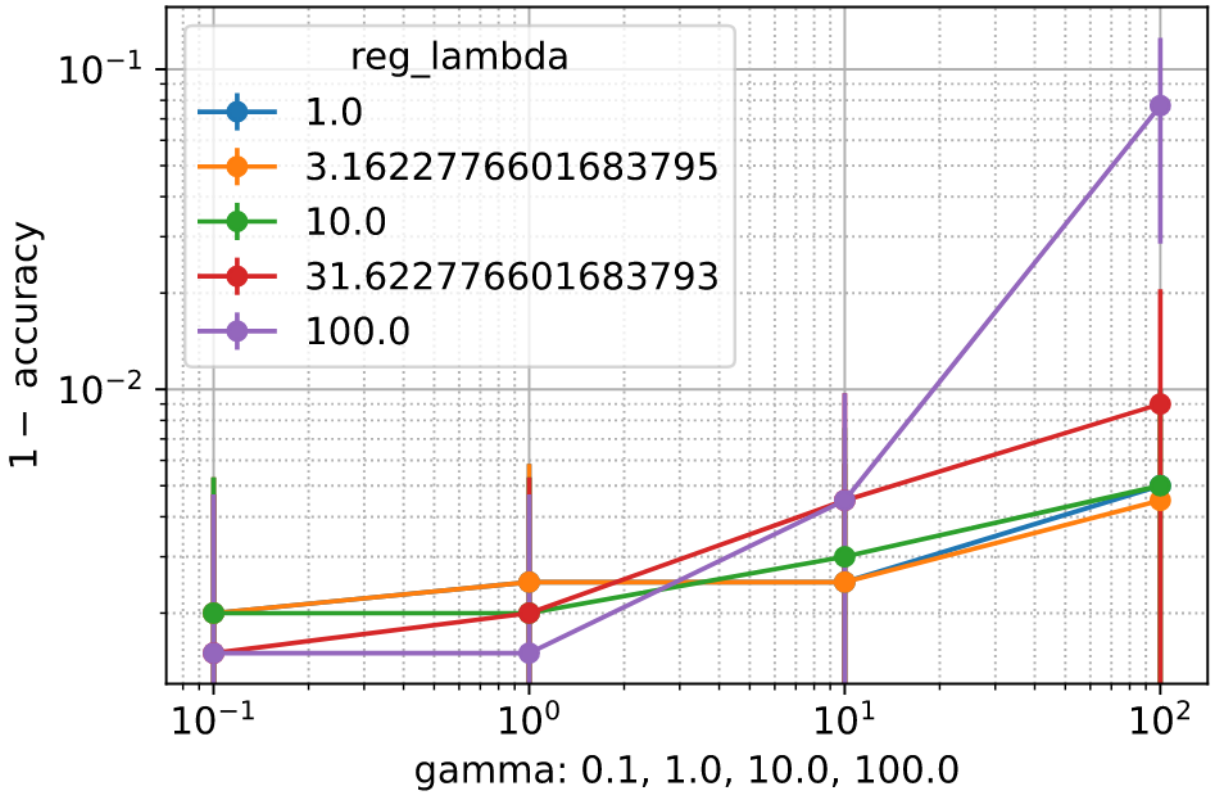
From the plots it is possible to see that the default XGBoost model outperforms the convolutional neural networks (even the best model with global pooling) on the small datasets. On bigger datasets instead the neural networks perform better, in particular the one with global pooling.

Depth of XGB



PhD

Regularization of XGB



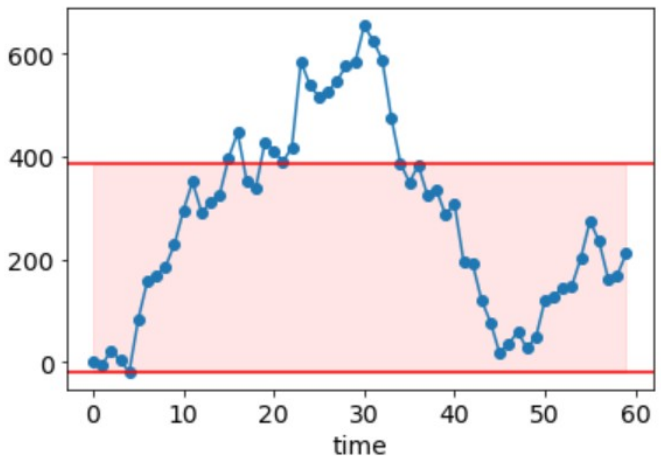
PhD

```
value__change_quantiles__f_agg_"var"__isabs_False__qh_1.0__ql_0.0
value__change_quantiles__f_agg_"var"__isabs_False__qh_0.6__ql_0.0
value__change_quantiles__f_agg_"var"__isabs_False__qh_0.4__ql_0.0
value__change_quantiles__f_agg_"var"__isabs_False__qh_1.0__ql_0.4
value__change_quantiles__f_agg_"var"__isabs_False__qh_1.0__ql_0.6
```

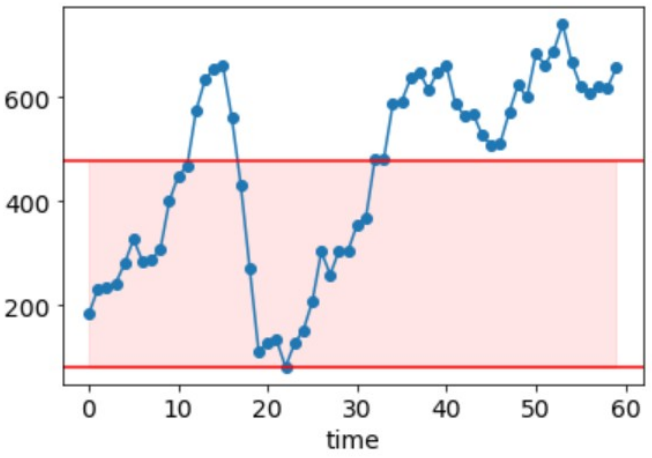
Feature relevance

2212

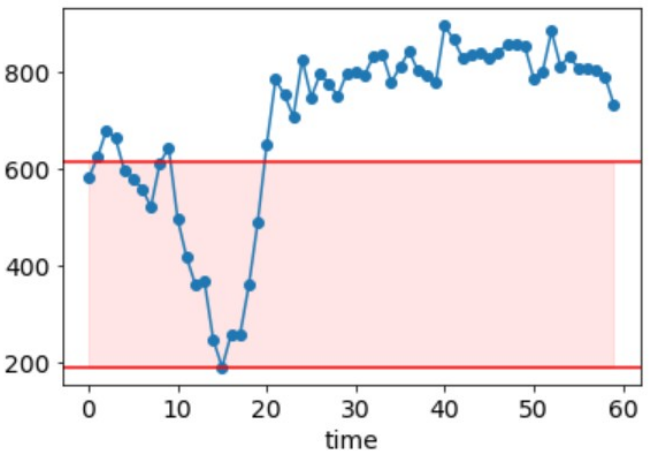
label: 0
feature value: 2228.55859375
feature value by hand: 2228.55859375



label: 1
feature value: 3931.572084481176
feature value by hand: 3931.572084481176



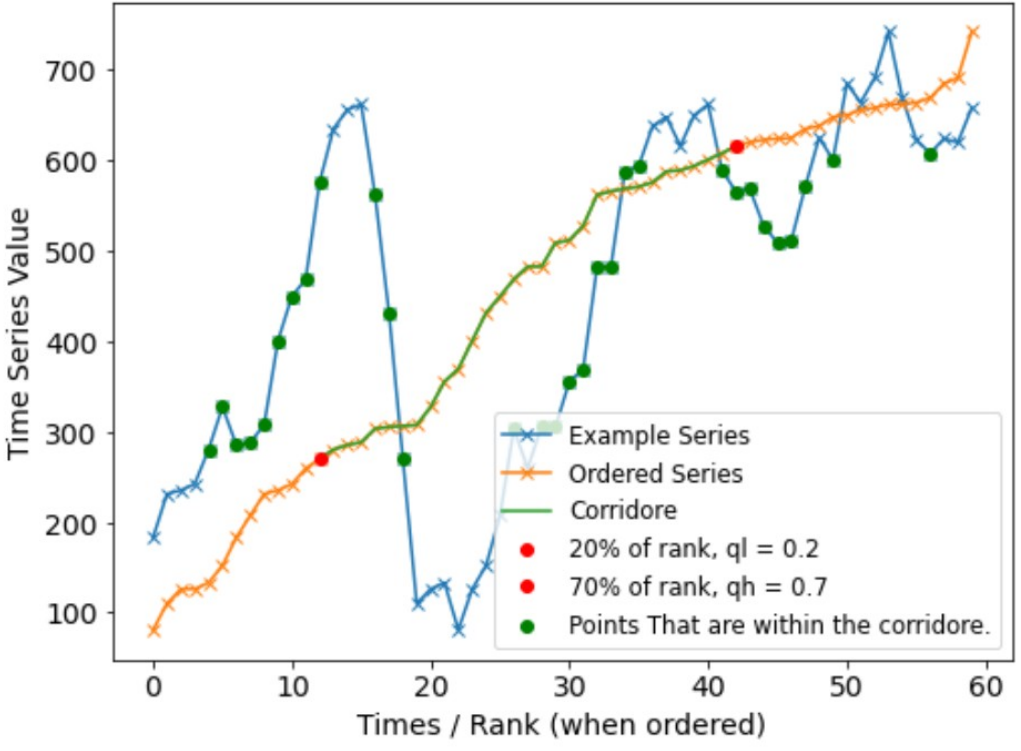
label: 2
feature value: 5419.743162901308
feature value by hand: 5419.743162901308

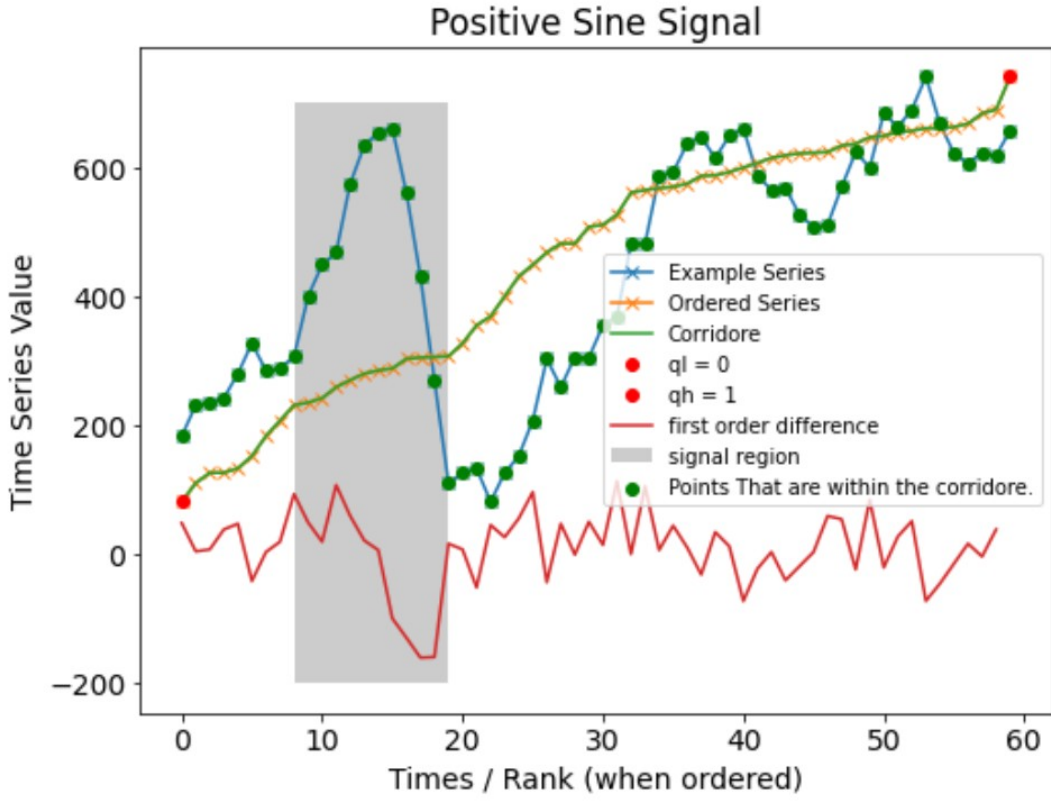


```
value__change_quantiles__f_agg_"var"__isabs_False__qh_1.0__ql_0.0
value__change_quantiles__f_agg_"var"__isabs_False__qh_0.6__ql_0.0
value__change_quantiles__f_agg_"var"__isabs_False__qh_0.4__ql_0.0
value__change_quantiles__f_agg_"var"__isabs_False__qh_1.0__ql_0.4
value__change_quantiles__f_agg_"var"__isabs_False__qh_1.0__ql_0.6
```

Feature relevance

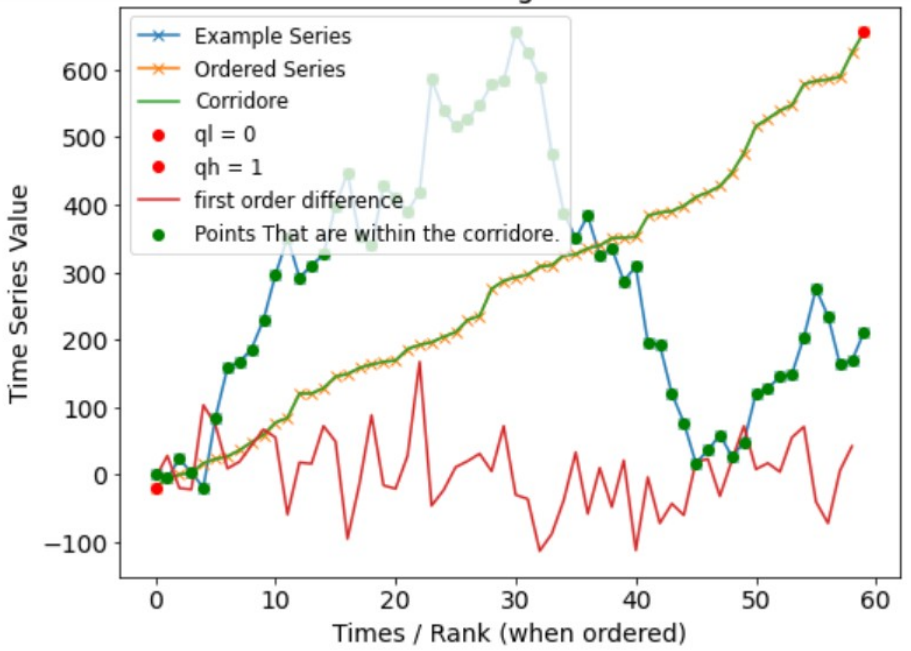
2220

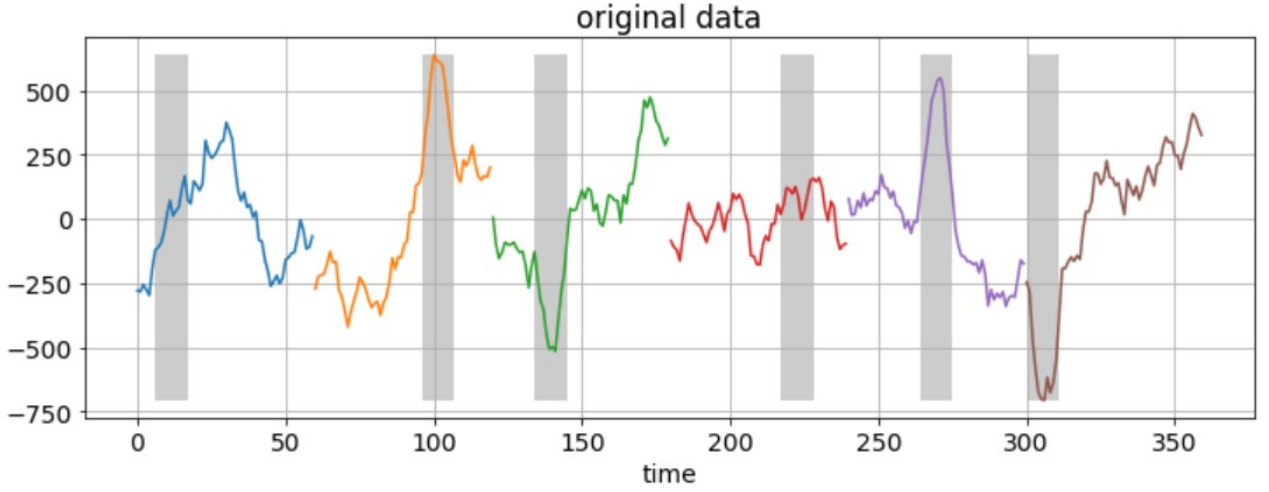




2220

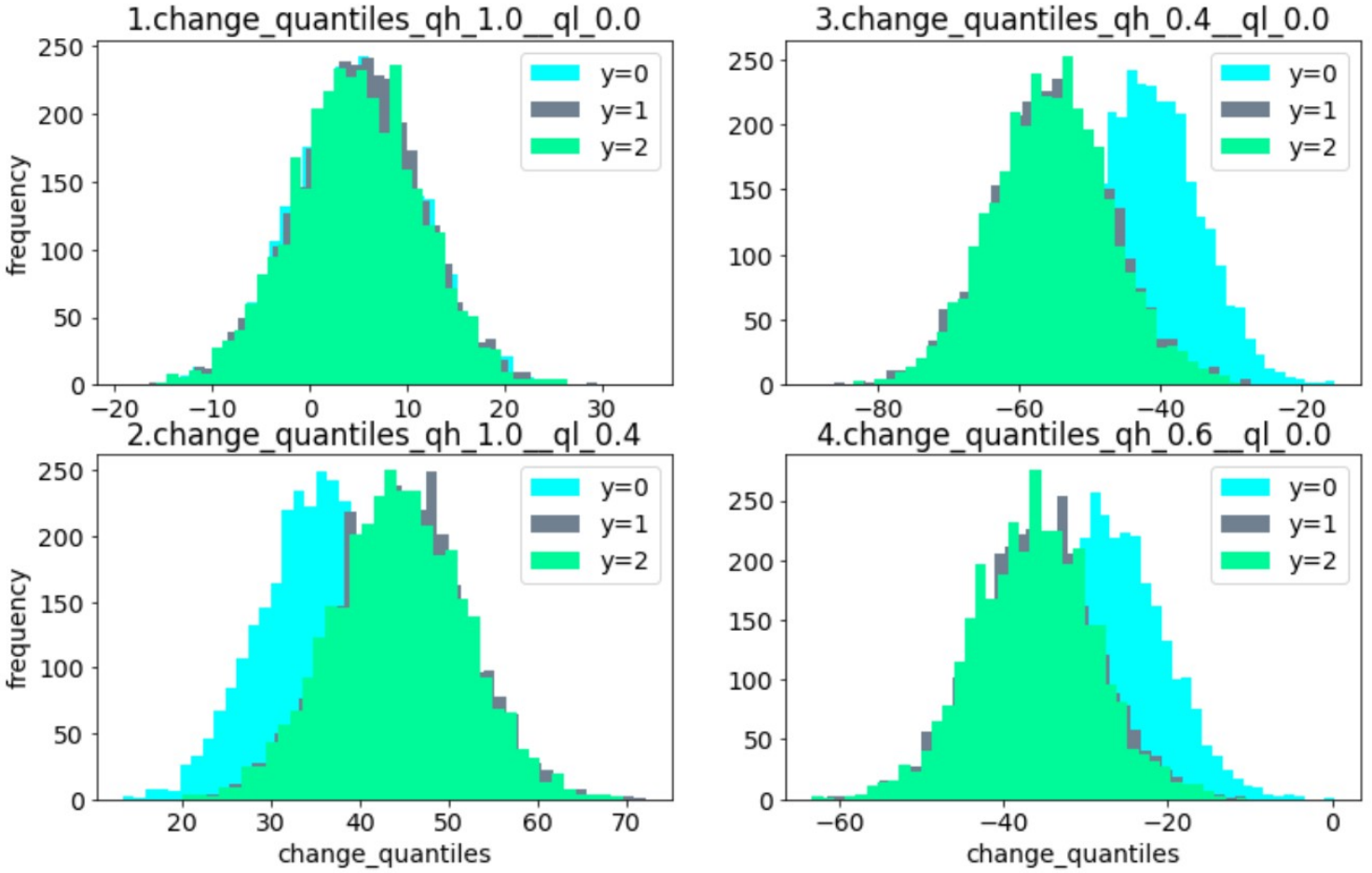
Feature relevance





Feature relevance

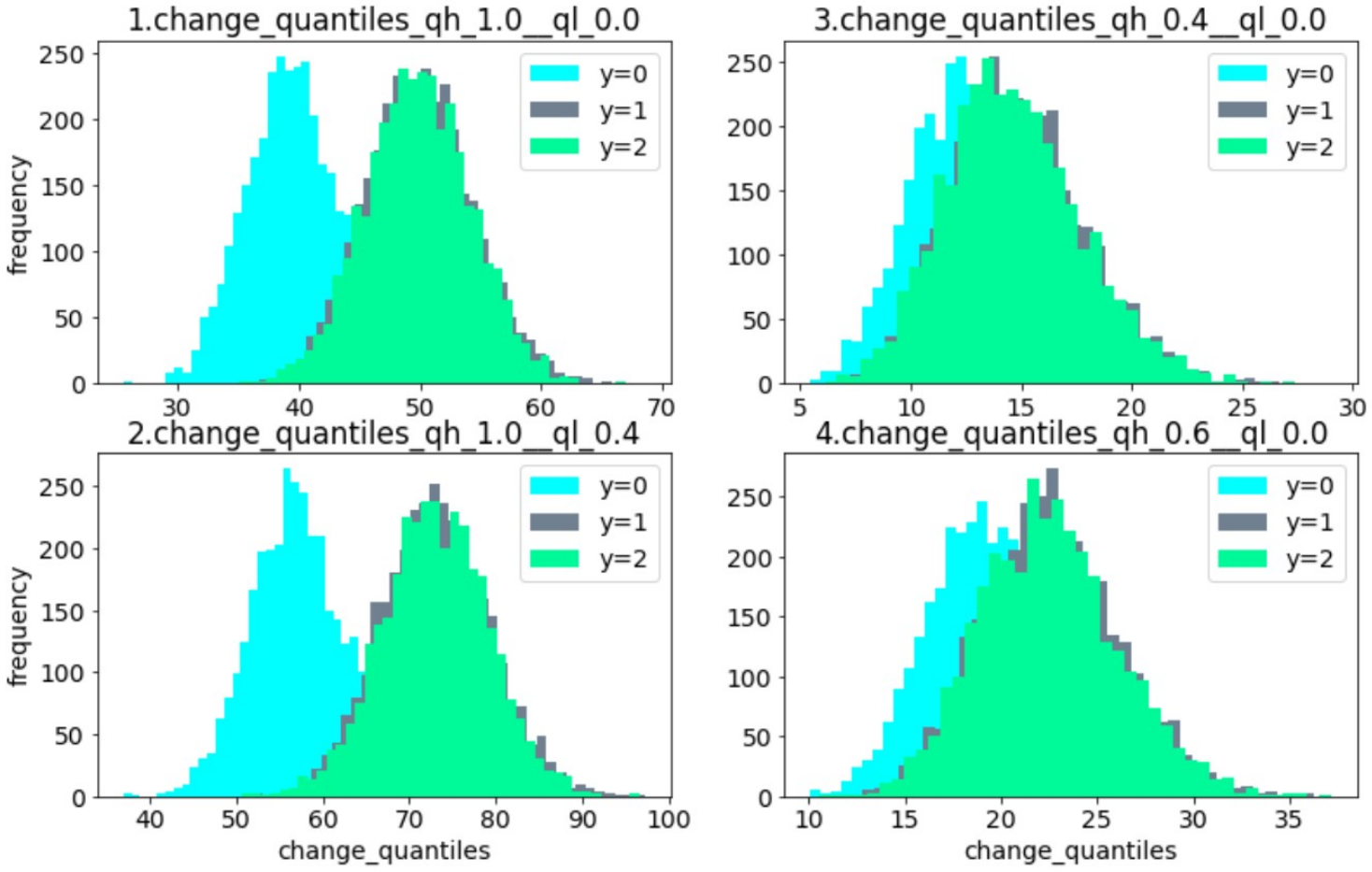
2211



We have plotted the distribution of change_quantile for its for differnet versions. We can see that the second, third and fourth features clearly are founnd seperate for label 0 and label 2. It is a little odd that feautre one, the most important feature according to tsfresh is found similar for all three labels! The reason why this change_quantile is the proper feature for this kind is that we have different overall amount of fluctuations for each label.

Feature relevance

2211



As one can see, the most important feature, when taking the absolute value of changes perfectly seperates label 0 from label 2. It should be mentioned that with or without the taking the absolute value, label 1 and label 2 are always attached together.

change_quantiles__f_agg_"var"_*isabs_False__qh_0.6__ql_0.0*

change_quantiles__f_agg"var"_*isabs_False__qh_1.0__ql_0.4*

change_quantiles__f_agg"var"_*isabs_True__qh_1.0__ql_0.0*

change_quantiles__f_agg"var"__isabs_False__qh_1.0__ql_0.0

2202

Feature relevance

This family of feature dominates the top of our most important features because it tries to get information about the variance of the dataset using a qh and ql as corridor variables to pick a corridor on the dataset and then it applies the function “var” shown in the description of the variable. Also in one of them isabs(True) is found, meaning that it is taking the absolute value of the differences in the dataset. This is crucial in our case because we need to see what patterns emerge in our dataset and how to recognize them.

cwt_coefficients__coeff_1__w_10__widths_(2, 5, 10, 20)

cwt_coefficients__coeff_2__w_5__widths_(2, 5, 10, 20)

These functions calculate a Continuous wavelet transform for the Ricker wavelet, also known as the "Mexican hat wavelet". This feature calculator takes three different parameters: widths, coeff and “w”. The feature calculator takes all the different widths arrays and then calculates the cwt one time for each different width array. Then the values for the different coefficient for coeff and width “w” are returned. This is also useful for our case because the Ricker wavelet has a high resistance to noise and it can successfully recognize the principal features of our dataset.

Feature relevance

change_quantiles__f_agg_"mean"__isabs_True__qh_0.8__ql_0.4

Also, we have a “mean” corridor that extracts the mean of the absolute values of the dataset. This works like the “var” corridor but the average value of a section of the dataset is not so crucial to the training of the model as the variance and that’s why this parameter does not rank higher or appear multiple times.

fft_aggregated__aggtype_"variance"
fft_coefficient__attr_"angle"__coeff_6

Finally two parameters related to the fast fourier transform of the dataset are found. Aggregated returns the variance of the fourier transform meaning the difference between the many frequency components in the dataset. Coefficient returns the angle component of the fourier transform, meaning the phase of the signal in the frequency domain. These together bring valuable information about the frequency components of the datatset. Combined with the variance parameters it can describe the behaviour of the samples in a more accurate way.

```
value__c3__lag_1                                                0.12287123
value__agg_linear_trend__attr_"stderr"__chunk_len_5__f_agg_"var"    0.07211669
value__cid_ce__normalize_False                                  0.06484082
value__change_quantiles__f_agg_"var"__isabs_True__qh_1.0__ql_0.0    0.03075107
value__absolute_maximum                                         0.02146667
value__abs_energy                                               0.01945943
value__skewness                                                 0.01857250
value__c3__lag_2                                                0.01620622
value__change_quantiles__f_agg_"var"__isabs_False__qh_0.4__ql_0.0    0.01572730
value__change_quantiles__f_agg_"var"__isabs_False__qh_0.2__ql_0.0    0.01537133
dtype: float64
Total importance of the first 20 features: 0.51789
```

Feature relevance

2213

- `c3__lag_1` : c3 statistics is used to measure non linearity in the time series; lag it is a parameter used to calculate the value of the feature c3. It might have a high importance because the time series are represented by a sin function, that has a marked non-linearity. The latter should vary between time serieses with different labels being the sine inserted differrently.
- `linear_trend_chunk_len_5` : Calculates a linear least-squares regression for values of the time series that were aggregated over chunks versus the sequence from 0 up to the number of chunks minus one. The chunksize is regulated by “chunk_len”, in this case 5. It specifies how many time series values are in each chunk.

Its importance might derive from the fact that, although sin is not a linear function, for small values of x (and periodically) it behaves like x. So when we divide it in chuncks this behaviour could arise. Adding once the peak of the sine and the once the valley on different spots, the linear function of the chunks shifts from x to -x.

- `cid_ce__normalize_False` : This function calculator is an estimate for a time series complexity (A more complex time series has more peaks, valleys etc.). Normalize equal to False represents that the signal has discrete values and during the analysis it is not trasformed towards a complex frequency-domain representation.

As the description says, this parameters highly carachterizes a function like the sine function that has multiple maxima and minima, positioned on multiple spots for different time serieses.

- `change_quantiles__agg_"var"__isabs_True__qh_1.0__ql_0.0` : First fixes a corridor given by the quantiles ql and qh of the distribution of the data x. Then calculates the average, absolute value of consecutive changes of the time series x inside this corridor. Think about selecting a corridor on the y-Axis and only calculating the mean of the absolute change of the time series inside this corridor.

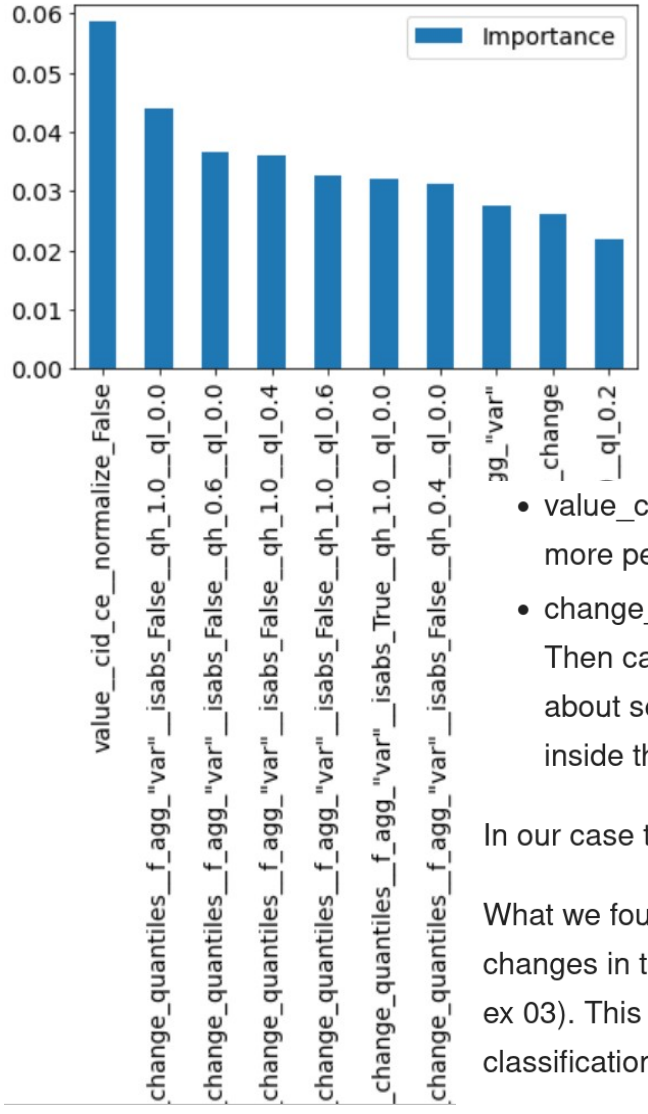
Being the function periodic, this may imply a higher importance of this feature given that in the corridors chosen the mean of the absolute change does not vary too much troughout the series.

- `absolute_maximum` : This feature calculates the highest absolute value of the time series x.

Being the data generated as a sine with noise, the maximum of the function should fall around similar values for each time series.

- `abs_energy` : It returns the absolute energy of the time series which is the sum over the squared values.

- `skewness` : Returns the sample skewness of x (calculated with the adjusted Fisher-Pearson standardized moment



Feature relevance

2215

- value_cid : This function calculator is an estimate for a time series complexity (A more complex time series has more peaks, valleys etc.)
- change_quantiles (agg_function): First fixes a corridor given by the quantiles ql and qh of the distribution of x. Then calculates the average, absolute value of consecutive changes of the series x inside this corridor. (Think about selecting a corridor on the y-Axis and only calculating the mean of the absolute change of the time series inside this corridor.)

In our case the aggreagation function used by change_quantiles is the variance.

What we found is in line with our expectations: for our problem it makes a lot of sense that looking at local and global changes in the variance is the most effective way to determine if there is a positive or negative trend (as discussed in ex 03). This in combination with the values_cid features is enough to obtain good results for our time series classification problem.

```
value__change_quantiles__f_agg_"var"__isabs_False__qh_1.0__ql_0.0
value__change_quantiles__f_agg_"var"__isabs_False__qh_0.6__ql_0.0
value__change_quantiles__f_agg_"var"__isabs_False__qh_0.4__ql_0.0
value__change_quantiles__f_agg_"var"__isabs_False__qh_1.0__ql_0.4
value__change_quantiles__f_agg_"var"__isabs_False__qh_1.0__ql_0.6
```

Feature relevance

2220

From looking at the doccumentation I have made my conclusions. Consider 1 sample, which is a time series. The first order differences are calculated. To know what this is please see the example below.

time ----------1----2----3----4-----5

values ----------1----3----4----7----10

first order ----------2----1----3----3

second order ----------1----2----0

Some of these first order differences are used and the rest are discarded. They are kept if both the values used to calculate the difference are in the corridore. The corridore is a subset of the sample between the quantiles. To find the corridore first the sample is ordered. The position of each value in the order is called the rank. The lower and higher quantiles ql and qh are fractions. They are fractions of the rank that decide the bounds of the corridore. This is more easily understood by looking at the example below. Where the red dots show the quantile fraction of the rank and the green line is the data that represents the corridore. If the sample values used to calculate the difference reside in this corridore then the difference is used for the rest of the calculation. The variance of the surviving differences is calculated. This variance is the sample's feature value. This is what is fed into xgboost to represent the sample.

All of the 5 most important features are calculated in this way but with different quantiles.

Feature relevance

2220

Why is this feature useful?

The 5 most important features use the quantile ranges

qh --- ql
1.0 -- 0
0.4 -- 0
1.0 -- 0.4
1.0 -- 0.6

The first one from 1 to 0 means the entire sample is used. The others could have issues because the signals are randomly placed in the samples. This means that the points of the sample that represent the signal might be completly ignored if they by chance dont fall within the corridore.

Using the fist order difference will flatten any gradual change in the values whilst any steep change will have a big peak. When you look at the plot below you can see that the values of the origional sample gradually move up to the hight of the injected sine peak. This means a high value does not constitute a signal. When the first order difference are plotted the peak of the signal rises above the rest of the samples. Thus a signal is distinguishable simply because there is a value much higher than the others. The samples with a signal will have this dominating peak in the first order difference and thus the variance will be high. The variance will be higher for every sample with a signal and lower for every sample with out a signal. This make the samples easy to distinguish.

I am not sure how this would help distinguish between a positive or negative sine signal. As the variance would have a similar magnitude and can only be positive. I am not sure how changing the quartiles could help either.

Feature relevance

2207

1	value__c3__lag_1	0.14067159593105316
2	value__cid_ce__normalize_False	0.13021141290664673
3	value__c3__lag_2	0.0715719535946846
4	value__agg_linear_trend__attr_"intercept"__chunk_len_50__f_agg_"min"	0.07060516625642776
5	value__cwt_coefficients__coeff_11__w_2__widths_(2, 5, 10, 20)	0.0411791168153286

1. `tsfresh.feature_extraction.feature_calculators.c3` uses c3 statistics to measure non linearity in the time series → makes sense that it is the most significant feature as it focuses on non-linearities such as the injected signals that we try to classify. [1]

2. `tsfresh.feature_extraction.feature_calculators.cid_ce` aims to estimate the time series complexity. A more complex time series has more peaks, valleys etc. → again this makes sense as the non-linearities we are injecting in the signals are actually peaks and valleys. [2]

3. `tsfresh.feature_extraction.feature_calculators.c3`, this time with a different lag, so we are changing the time window on which this features operates to detect non-linearities.

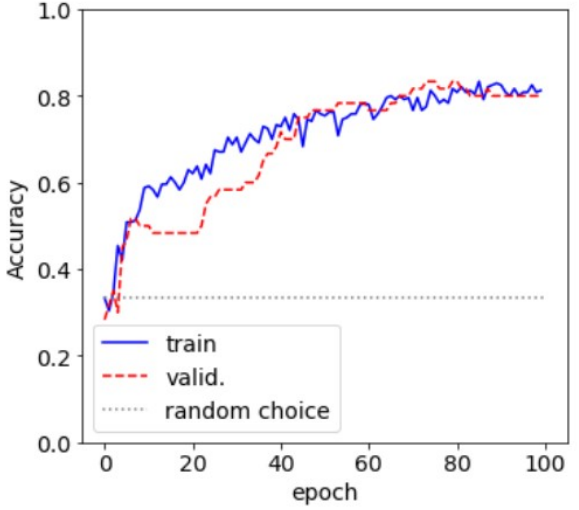
4. `tsfresh.feature_extraction.feature_calculators.agg_linear_trend` calculates a linear least-squares regression for values of the time series that were aggregated over chunks versus the sequence from 0 up to the number of chunks minus one. This feature assumes the signal to be uniformly sampled. It will not use the time stamps to fit the model. The parameters attr controls which of the characteristics are returned. Possible extracted attributes are “pvalue”, “rvalue”, “intercept”, “slope”, “stderr”. The fourth most significant feature is characterized by the "intercept" attribute. [3]

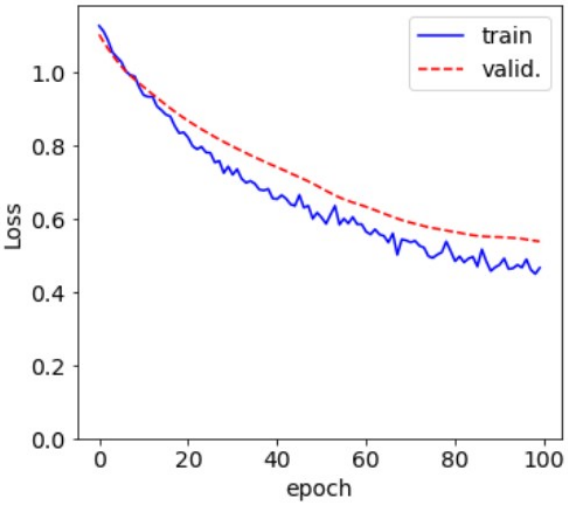
5. `tsfresh.feature_extraction.feature_calculators.cwt_coefficients` calculates a Continuous wavelet transform for the Ricker wavelet, also known as the “Mexican hat wavelet”. This feature calculator takes three different parameter: widths, coeff and w. The feature calculator takes all the different widths arrays and then calculates the cwt one time for each different width array. Then the values for the different coefficient for coeff and width w are returned. [4]

FFNN with reduced XGB features

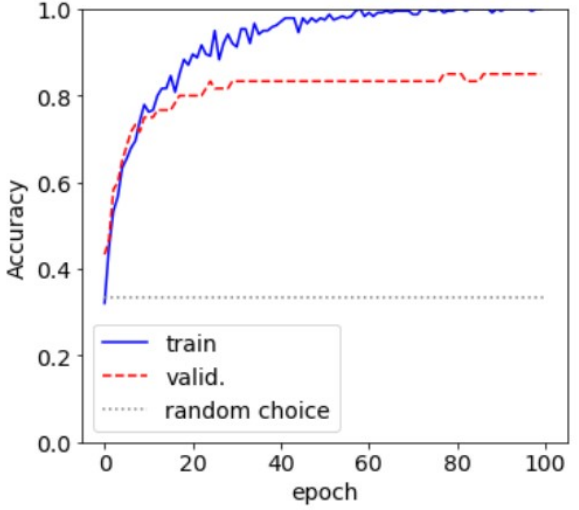
2215

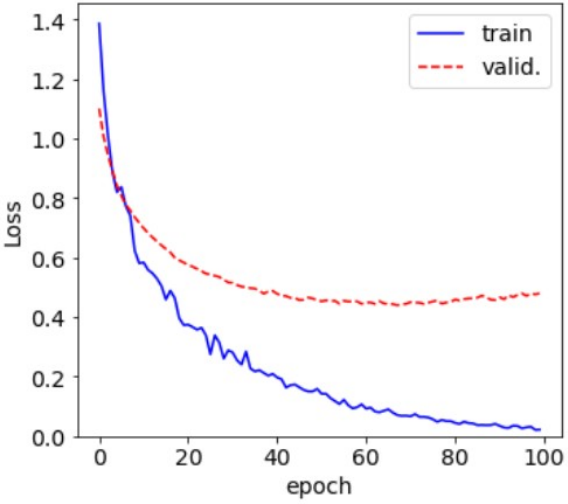
```
Testing with N = 300
with reduced features
```

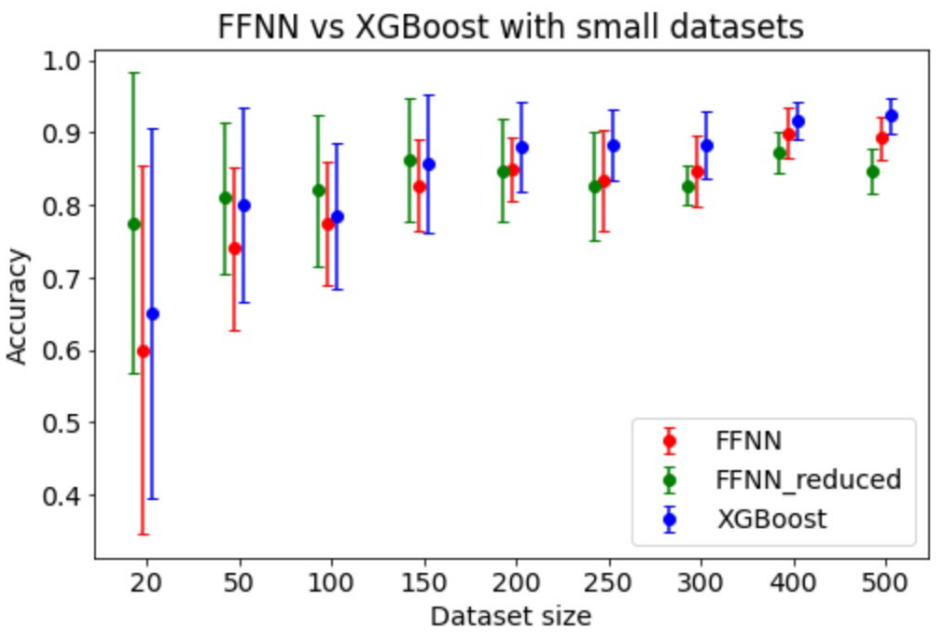





```
with all features
```





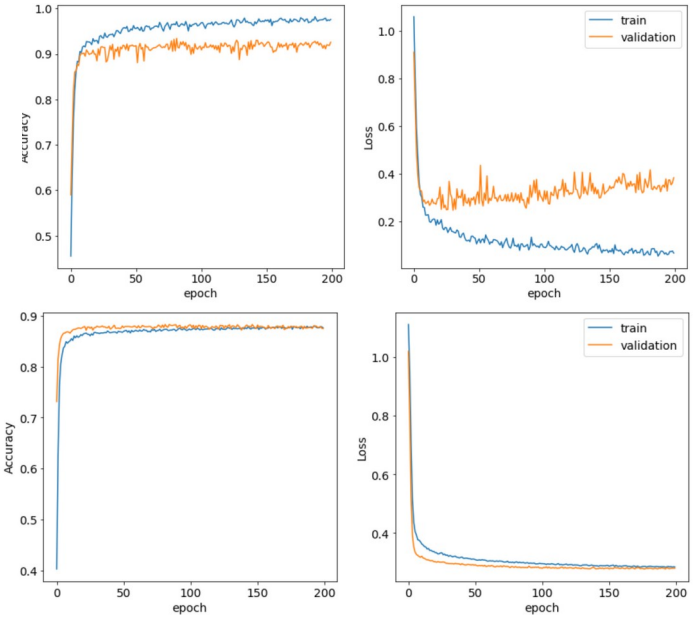


FFNN with reduced XGB features

deduce some conclusions:

2215

- XGBoost is only slightly better than the FFNN that uses all the features, and in some instances it is outperformed by the FFNN that uses only the most important feature. First one should consider that the FFNNs has been tuned while we didn't optimize the XGBoost hyperparameters. Second we can't underestimante the high standard devition for $N < 100$: one should increase the number of iteration to obtain more precise results (especially for the smaller datasets, the ones more susceptible to the sampling).
- For the lower dataset sizes the FFNN trained only with the most important features outperforms the one trained on all the features: this can be explained by the fact that with few data the model has a change to obtain better results if it can learn the most significant aspects that are useful to distinguish between the different cases. If it gets too many features with few data it does not have the change to discriminate which ones are the more important before it runn out of training sample and so the performace is lower. Viceversa with more data ($N > 300$) more features (even if less important) can still improve a litte the ability of the model to discriminate the different labels.



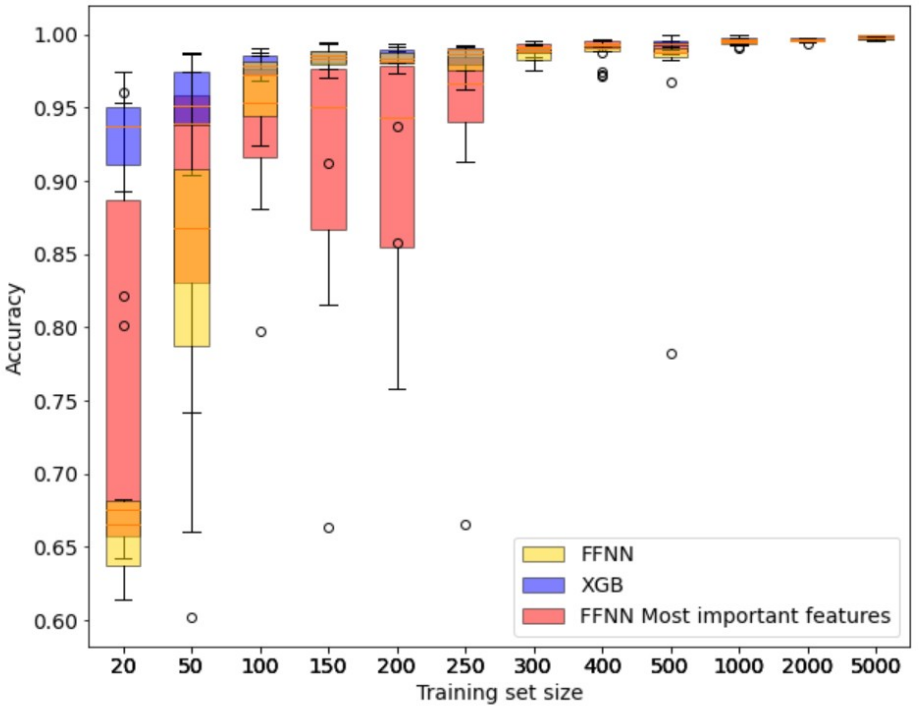
FFNN with reduced XGB features

2202

The FFNN trained on the whole dataset of features reaches an overall validation accuracy of 92%. However, such model requires a large amount of parameters (two layers of 200 & 20 neurons). If we look at the training process, we can see that the train accuracy improves, while the validation accuracy is stationary around 90%. This fact, along with the observation that the loss function on the validation set does not improve, prompts a situation of overfitting.

Reducing the dataset to the 19 features which account for 99% of variance in the data, instead, allows to crucially reduce the size of the layers (20 + 10 neurons), reaching at the same time an accuracy of $\sim 93\%$.

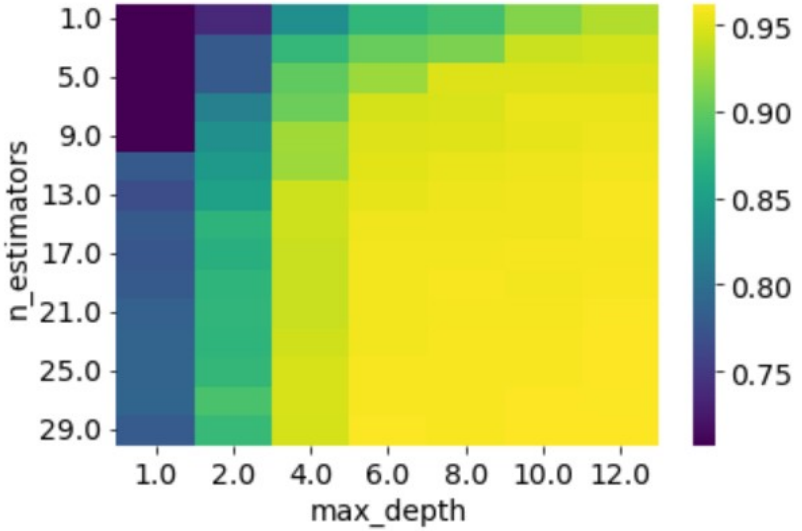
FFNN with reduced XGB features

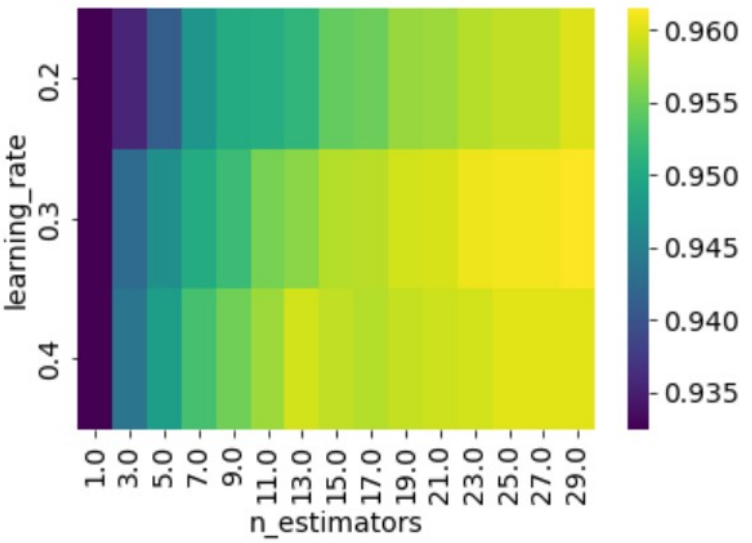


2216

The plots show that if we train the FFNN model using only the ten must impostant features, it provides similar results.

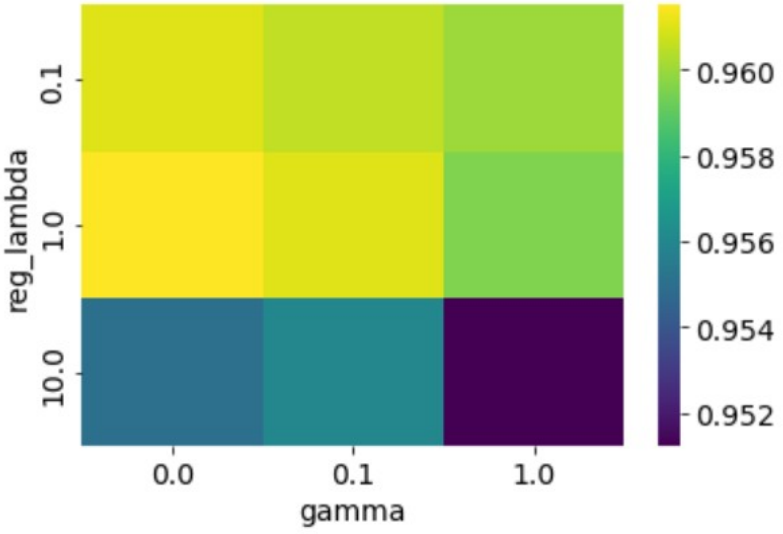
Tuning XGB

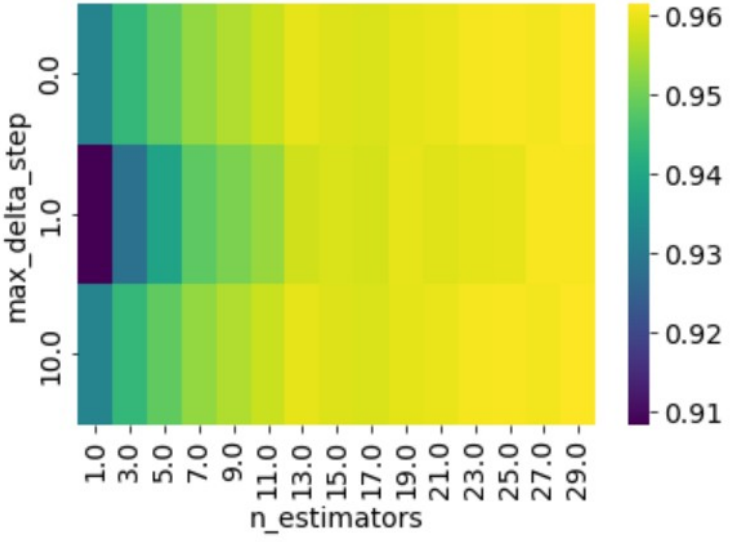




2216

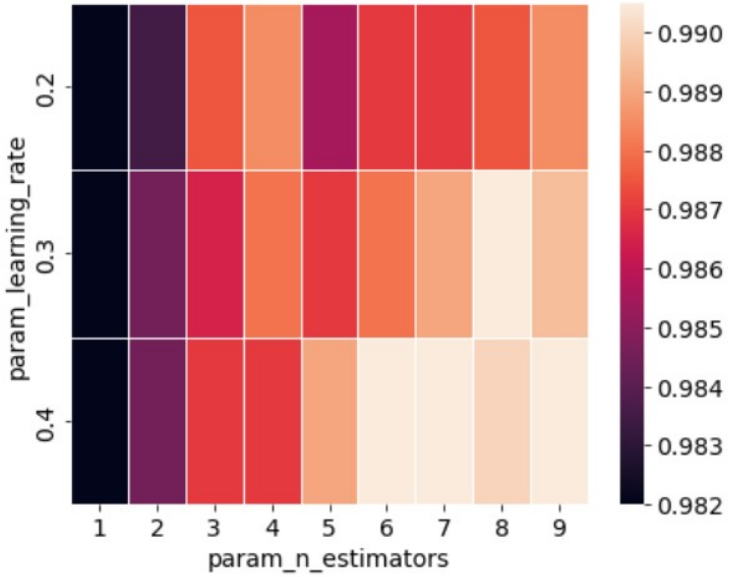
Animation?

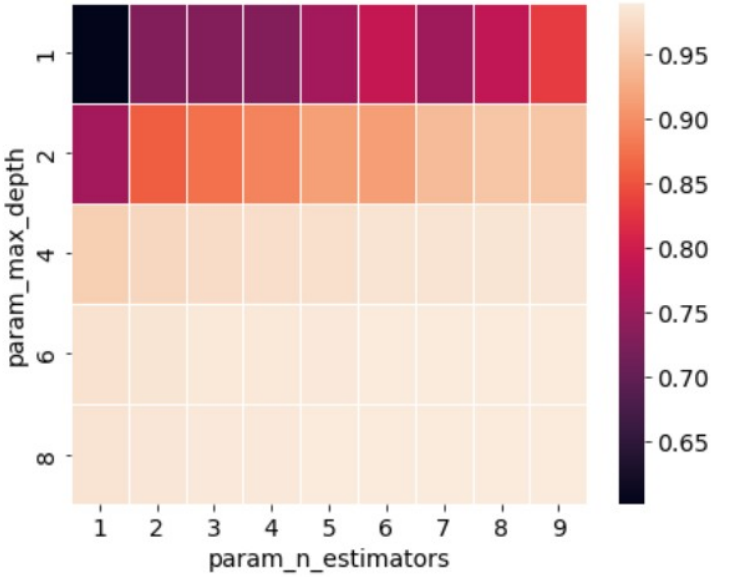


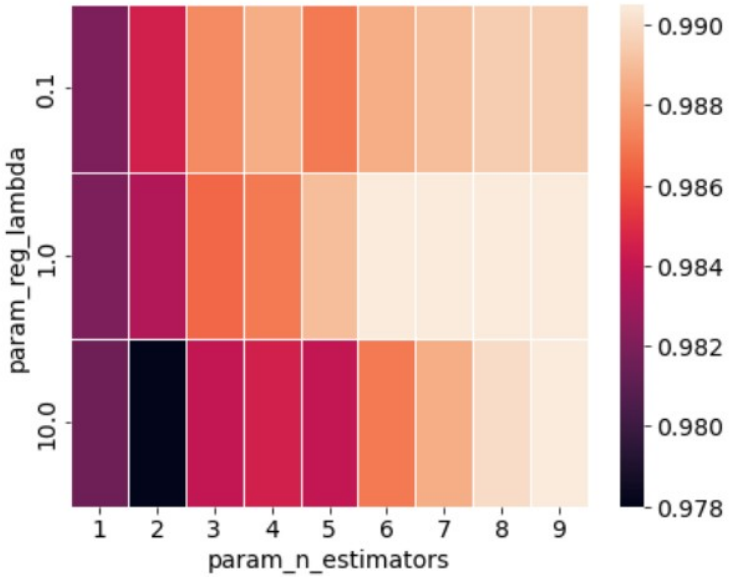


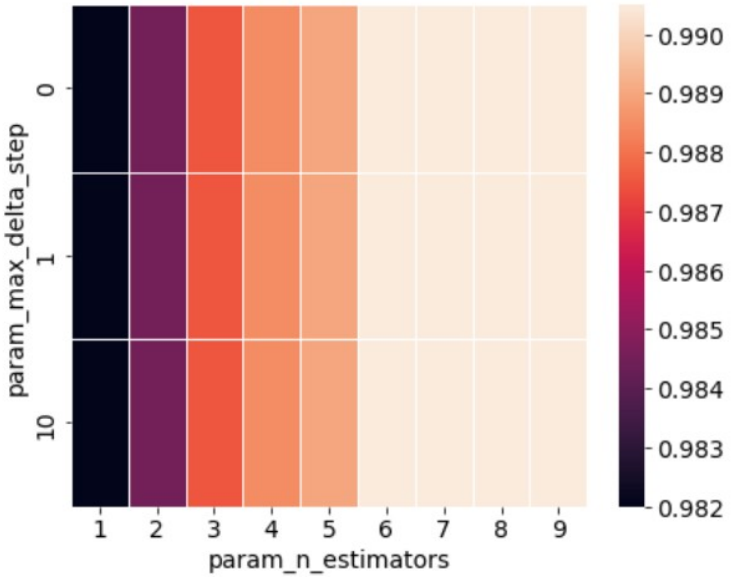
Tuning XGB

2211









Tuning XGB

2211