Evaluation of the effect of Hyperparameters in Classification Problems.

Introduction

This report refers to the project that was conducted for the Machine Learning Scholarship Program for Microsoft Azure. In my project I tried to evaluate the effect that hyperparameters of several classification models have on the final output. For each case, we chose one hyperparameter to change, leaving the rest stable or in their default values.

For the purpose of the analysis we used the Higgs Dataset. It can be obtained for the following link: https://archive.ics.uci.edu/ml/datasets/HIGGS. It should be noted that it is in gz format, and the final csv file after decompression is above 7GB in size. It contains a total of 11000000 instances (rows), with each row to contain a total of 28 attributes (included the output class). It is a synthetic dataset, produced by Monte Carlo simulations and no pre-processing was performed in that. It tries to solve the case whereas a particle collision could produce a Higgs Boson or not. All the analysis was performed in my local PC (Windows 10 64bit / 8 GB RAM / python 3.7 / jupyter notebook in VSCode). In all cases was used the implementation of the algorithms as provided by the scikit python library.

There were chosen 3 classification algorithms to be studied. First we used the Decision Tree Classifier, in which we examined the effect of maximum depth of the tree in the final result. Then we used the Random Forest classifier, where we examined the effect of the number of estimators (trees) in the forest. Finally we examined the Gradient Boosted Tree Classifier, in which we examined two cases. In the first we examined the number of estimators, while the maximum depth of the tree was stable, and also we kept the number of estimators stable while we examined the effect of the maximum depth of each estimator.

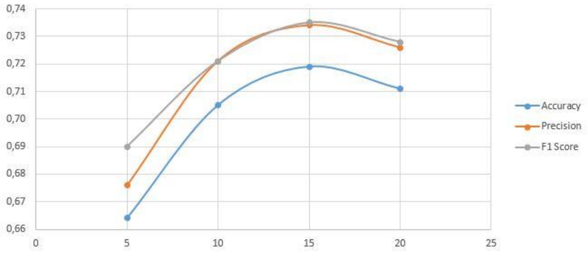
Decision Tree

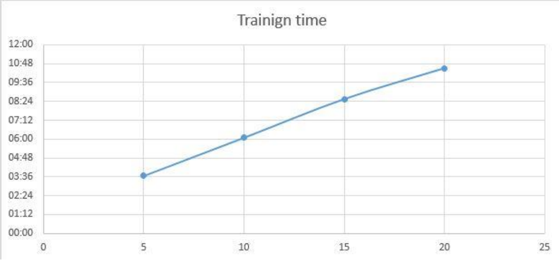
The Decision Tree is one of the most well-known classifiers that has also a regression variation. It outputs a tree-like structure that we start from the root, and depending on the value of each feature we move downwards to the leafs, which represent the output of the classification. It has the advantage to be able to be visualized easily and to be able to be converted in a set of rules.

As we discussed before, the hyperparameter we examined is max_depth, which represents the maximum depth of each tree. The results of the analysis are presented in the following table

Max depth	Accuracy	Precision	F1 Score	Training Time
5	0.664	0.676	0.690	00:03:39
10	0.705	0.721	0.721	00:06:05
15	0.719	0.734	0.735	00:08:33
20	0.711	0.726	0.728	00:10:31

As we can see from the presented results, in general as the number of max depth the metrics seem to improve, except for the final case where max_depth is equal to 20 and a drop in all metrics was shown. This could be an indication of overfitting, as the higher depth of the tree, hence a more complex one, tries to fit better in the training data. On the other hand, as it is expected, the time that is required to complete the training. It is worh noting that, as shown in the following picture, the required training time seems to have a linear relation in regard with the maximum depth.



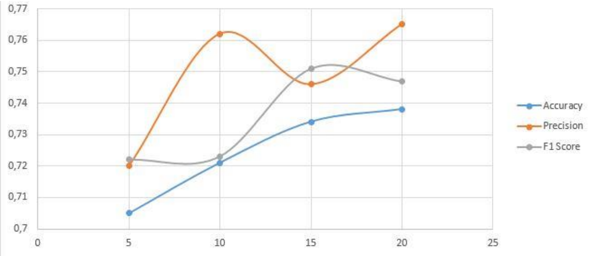


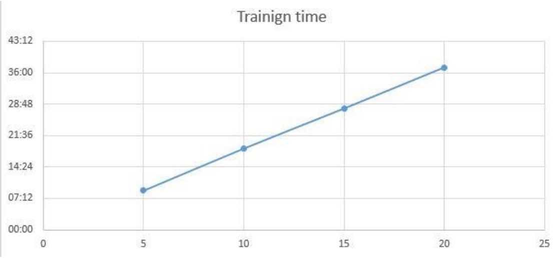
Random Forest

The Random Forest classifier, as its name implies consists of a large number of decision trees that operate as an ensemble. It is based in the wisdom of the crowd, a large group of classifiers outperform the single classifier. It is based on the idea of bagging, in which a random sample of the original dataset is used to train each tree. As it was stated, the parameter we chose to examine is the number of trees that consist the forest.

Number of trees	Accuracy	Precision	F1 Score	Training Time
5	0.705	0.720	0.722	00:09:02
10	0.721	0.762	0.723	00:18:40
15	0.734	0.746	0.751	00:27:50
20	0.738	0.765	0.747	00:37:14

As in the previous case, it seems that as the number estimators increase, although some fluctuation exists. Also, in case of Accuracy, it seems to tend to stabilize to a certain value. In the case of required training time, like in the decision tree, there is a linear relation with the number of estimators, as shown in the following picture. Also, it required more time to be trained, compared to the decision tree classifier.



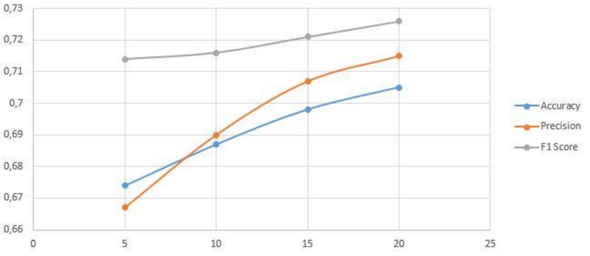


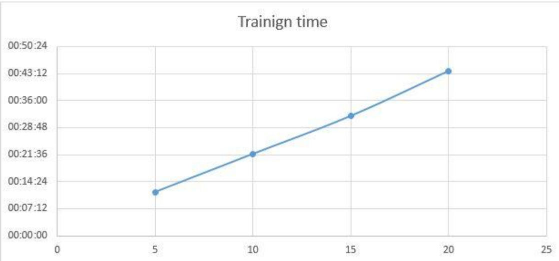
Gradient Boosted Trees

This is another case of an ensemble classifier, which like the random forest it uses several (simple) decision trees, but in this case it is based on the idea of boosting. Each new classifier focuses on the error of the previous tree. In this case we choose to investigate two hyperparameters, the number of the estimators (trees), while the maximum depth of each was set to 5, and the effect of the max depth of each tree, while the number of estimators was equal to 8. Let’s see the results

Depth = 5				
Number of trees	**Accuracy**	**Precision**	**F1 Score**	**Training Time**
5	0.674	0.667	0.714	00:11:33
10	0.687	0.690	0.716	00:21:44
15	0.698	0.707	0.721	00:31:52
20	0.705	0.715	0.726	00:43:54

As we can see from the results, and in this case the more the number of trees, the better the results, although it seems to perform worse compared to the previous classifiers presented. Also, it requires more time to be trained, although and in this case there is a linear relation of the required training time and the number of trees.

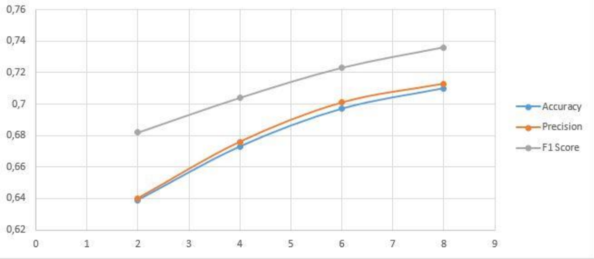


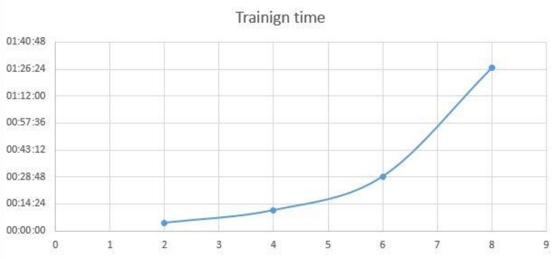


Now we are going to examine the opposite case. We have a fixed number of trees (8), while we examine the maximum depth of each tree. Like before, the results are presented below

Number of estimators = 0				
Max depth	**Accuracy**	**Precision**	**F1 Score**	**Training Time**
2	0.639	0.640	0.682	00:04:21
4	0.673	0.676	0.704	00:11:06
6	0.697	0.701	0.723	00:29:06
8	0.710	0.713	0.736	01:27:03

Like in the previous cases, the more depth, and so more complex, the tree, the better the metrics. And in this case the metrics seem to tend to stabilize to a certain value. In terms of time required and in this case the more complex the classifier, the more time it needs to be trained, but in this case there is not a linear relation, but rather squared, as it seems from the following picture.





Conclusion

As a general rule of thumb, the more complex the model, the better tend to be the results, but with a cost in terms of training time required. It is obvious, that although the results seem to be meaningful, and rather expected, it would be useful for extra research to take place. Examine more datasets, more cases and even run on different machines to see the how the results would vary.