Assignment 1: CS 7641

Gwénolé Hascoët

September 09, 2019

Data acquisition

The datasets used for this analysis come from Kaggle.com.

International football matches dataset

This first dataset is a census of the score and the result of each official football game since 1900 that has been combined with the FIFA ranking of international teams. The goal is to *predict whether the home team will win the game*. To improve the results of our classifiers over this dataset, I added 6 features to it: from the census of football games, I added the average result over the last 10 games of both teams, but also the average of goals conceded and scored over the last 10 games for the home team and the away team. Football is known to be very unpredictable since a football game result relies on a huge number of elements, such as the qualities of the players, their motivation to play a game (for a friendly game, the “best” teams regularly lose against weaker adversaries), luck, etc. Of course, these elements are not in our dataset, so their impact will be considered as noise since we do not have information on them, and they seem to appear randomly. I am interesting to see how well different Machine Learning algorithms can perform on this noisy dataset. (Of course, I don’t have high expectations, but I want to see how better it will be than throwing a coin). We can note that our dataset is balanced with 48.45% of positive elements (i.e. victory of the home team).

Phishing websites dataset

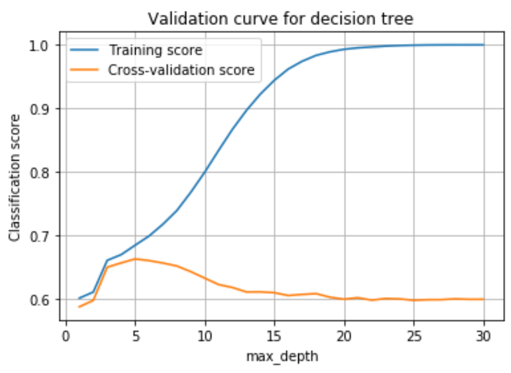
This dataset is made to train learners to identify phishing websites. To do so, this dataset contains 30 features correlated with the legitimacy of a website, including properties on its URL for example. The goal is to predict whether a website is the legitimate (label=1) or not (label = 0). This dataset is balanced (with 55.45% of positive instances), and the features used have been optimized to solve this classification problem. Consequently, we expect good results of the different algorithms we will use for this project.

Analysis

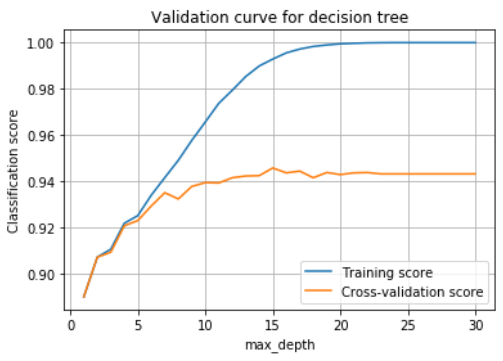
Decision Tree

I first tried a naïve decision tree that measures the Gini’s Index of each node as a criterion to split it into subsets. This index measures the probability that a randomly chosen element of a given node is wrongly classified. If this index worth 0, then all elements belong to the same class, and when it worth 1, all elements of the subset are randomly distributed across the different classes. Since we know the label of our variables, we choose the attribute with the lower Gini index as a root node. Yet, to build a tree we also need a criterion to stop it otherwise our classifier with overfit. To do so, we use pre-pruning to limit the tree depth to a hyperparameter “max_depth”. We use the validation curves to determine to deph of our tree that gives the best results. We also tuned the hyperparameter “min_samples_leaf” to ensure that each leaf of our tree has a minimum number of variables. Indeed, If the leaves of our tree contain very few elements, it is a sign of overfitting.

Validation curves



Validation curve of the football dataset



Validation curve of the phishing websites dataset

Both curves show a similar behavior of the evolution of the training score while increasing the depth of the decision tree. Indeed, the training score increases with the tree depth until reaching a perfect score. This prove that the tree aims to perfectly fit the training set when it is not constrained.

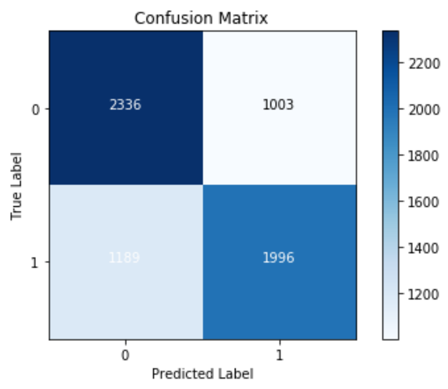
Also, after a given depth that differs for both trees, the cross-validation score stops increasing, and even starts to decrease for the Football dataset. After this given point, our decision tree overfits. Moreover, we can note that for both datasets, our classifier improves a lot during the firsts splits that suggests that some attributes are more relevant than other for the classification:

One advantage of decision trees is the information it provides on the impact of each feature during the decision process. In fact, the “feature_importance” informs us that the FIFA ranking of both teams have the most impact on classification for the football dataset.

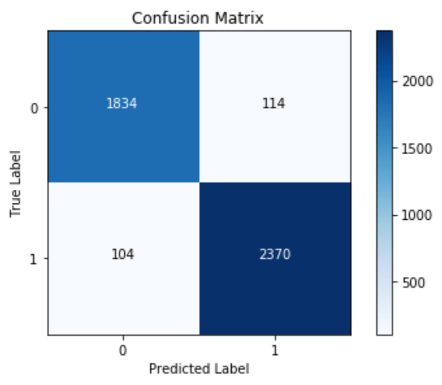
The hyperparameters tuning allows us to find the maximal depth and the minimum number of samples per leaf that yield the best accuracies. For the football dataset, a max depth of 7 nodes and a minimum of 142 elements per leaves provide **67.46%** of accuracy. For the phishing websites dataset, a maximal depth of 15 nodes gave the best results with **95.07%** of accuracy. For this second dataset, we did not set a minimum sample size per leaf because it reduced the accuracy of our classifier to 92.07%. This can be explained because the phishing websites dataset has a lot of features that are highly corelated with the desired output, so imposing a minimum number of elements per leaf reduces the depth of our decision tree, which tends to underfit.

Obviously, we note a huge difference of accuracy between our two datasets. Indeed, randomness and non-quantifiable attributes such as the motivation of players impact the result of a football game. This leads our first classifier to overfit for a short maximal depth, since information provided by additional splitting is biased by the inherent randomness of our dataset, which can explain the decrease of the validation curve while overfitting. On the other side, our second tree shows that numerous features of the second dataset are highly correlated with the output. In fact, we can use 3 times more attributes than the first dataset before overfitting.

Confusion Matrixes



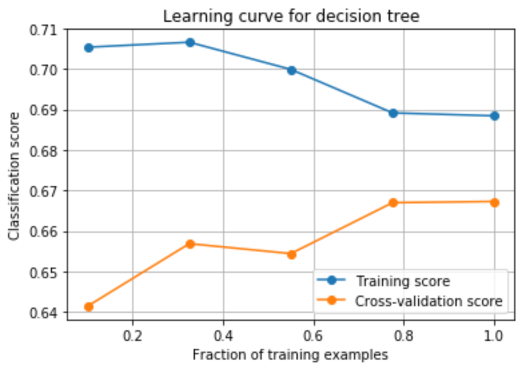
et Cc



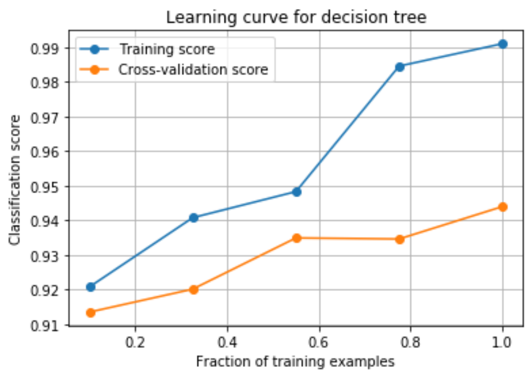
et

atrixed that accu : the “best” hyperparameters since for both dataset, the false positive rate is close from the false negative rate, so criterion such as the recall, the precision or the F-1 score won’t provide a lot of additional information, and it is harder to interpret them.

Learning curves



Learning curve of the football dataset



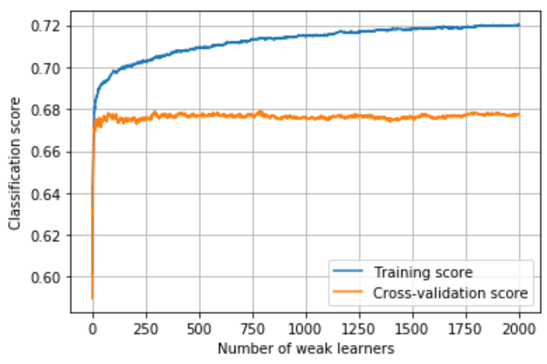
Learning curve of the phishing websites dataset

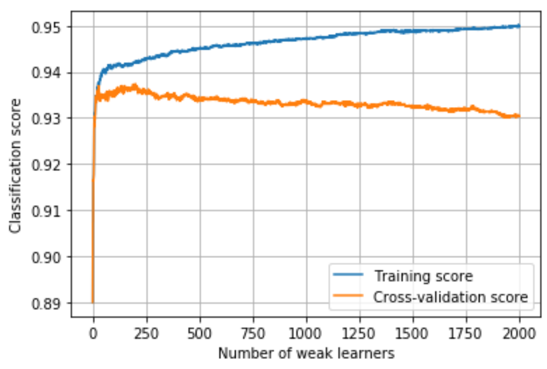
The behavior of the classifier over the football dataset suggests that increasing the size of our dataset wil not benefit much for our prediction. Indeed, the validation score and the training score converge toward 68% of accuracy, so we can espect at most 2 or 3% or increase of the accuracy by increasing the size of our dataset. The learning rate of the second classifier show a different behavior: the training score converges toward 1, and the cross-validation score keep increasing throuh all our dataset. So it suggest that increasing the size of our dataset might lead to an accuracy close to 1.

Boosting: Adaboost

Adaboost combines the output of weaks learners (decision stumps in this analysis), and weights them to obtain the best results possible. Since all decision stumps have the same structure, the hyperparameter that can be tunned is the number of weak learners.

Validation curves

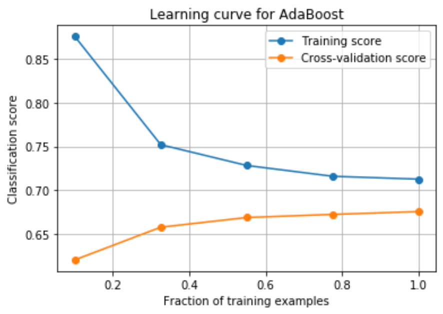


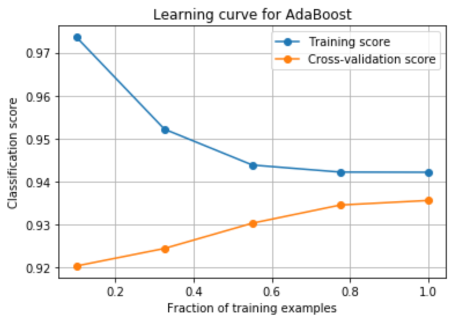


Validation curve of the football dataset *Validation curve of the phishing websites dataset*

For both datasets, the validation curves have a similar behavior: they first increase very fast when the number of decision stump increases. After a certain number of weak classifiers, Adaboost starts to overfit: after few dozens of decision stumps, the cross-validation score reaches its maximum and remains almost constant, independently of the number of weak classifiers added. In parallel, the training score keeps increasing with the number of weak learners, which is another sign of overfitting. If we have a closer look over the accuracy of this learner, we can note that Adaboost perform quite well on the first dataset, since will only few decision stumps, it performs better that the decision tree (detailed above). Moreover, with 314 decision stumps, it reaches 68.03% of accuracy, which is close to the asymptotic limit of the learning curves for our decision tree (see above). On the other hand, Adaboost is not as performant as the decision tree for the second dataset. Indeed, the maximum of accuracy on the testing set is reached for 201 learners, and worth 93.67%, which is less than the 95.07% of the decision tree. We may improve our learner of the second dataset by choosing decision trees which are deeper than 1.

Learning curves





Learning curve of the football dataset *Learning curve of the phishing websites dataset*

For both datasets, the cross-validation score increases with the number of instances used for training our model. In fact, when our dataset gets larger, it becomes more representative of the reality, and so the generalizing ability of our classifier increases. Of course, this is only true when our new elements added follows the actual distribution, otherwise, adding new elements to our dataset can bias it, and so it can reduce its ability to generalize. Yet, we do not face this specificity for tuned Adaboost, since the test accuracy keeps increasing with the number of instances used.

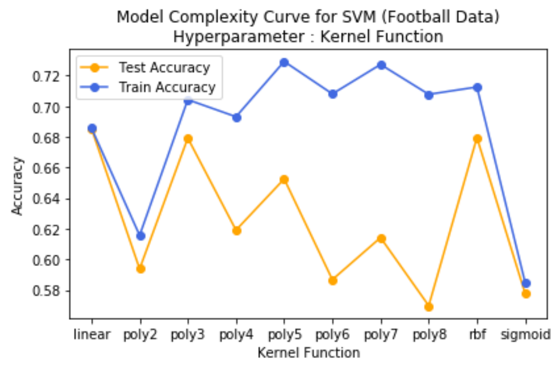
We can also note that both learning curves adopt the same behavior as the decision tree learning curve trained with the Football dataset: we observe that both the training score and the test score converge to the same limit (around 69% of accuracy for the first dataset, and 94% of accuracy for the phishing dataset).

We can deduce that our classifier almost reaches its limit in both cases; and increasing the size of the training set won't improve the performances greatly.

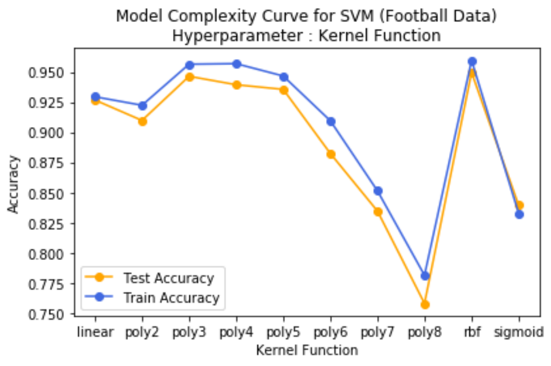
SVM

The first hyperparamet to choose while learning a SVM classifier is the kernel function it will use. I tried different kernel functions to choose the best hyperparameter in term of accuracy. If the best kernel function is a rbf, It is also relevant to find the best kernel coefficient γ to configure the sensitivity of the RBF kernel. Finaly, it is important to train the penalty constant C to get the best classification.

Validation curves



Validation curve of the football dataset

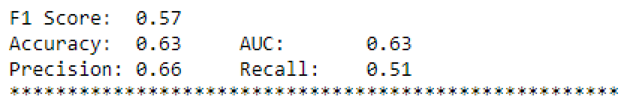


Validation curve of the phishing websites dataset

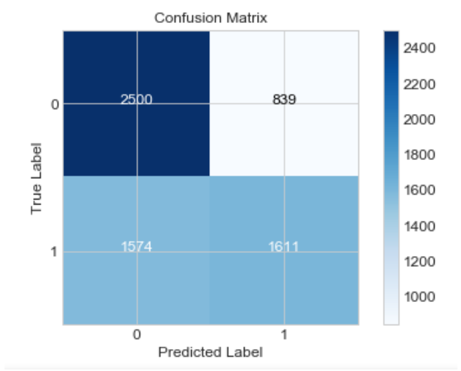
For the football dataset, three kernel functions gave similar results in term of accuracy: the linear, the rbf and the polynomial (with d=3) functions. For the phishing websites dataset, the rbf kernel function gets the best results.

Football dataset:

We chose to first try the RBF function for the football dataset. Indeed, there is a lot of noise in our data, so the linear kernel might separate the data by unbalancing the false positive rate and the false negative rate to maintain a good accuracy. We tuned the hyperparameter "C" (to C=1) to get the most accurate rbf learner, we got the following performances:

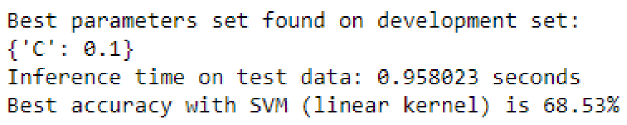


```
F1 Score:  0.57
Accuracy:  0.63     AUC:       0.63
Precision: 0.66     Recall:    0.51
******************************************************
```

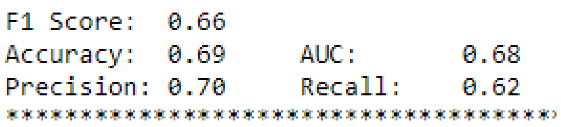


Scores of our classifier after hyperparameters tuning *Confusion Matrix of the football dataset*

In fact, even the rbf kernel unbalances the output: as we can see on the results below, the F1 score is significantly lower than the accuracy because our classifier labels most data as "0" (so non-victory for the home team). Therefore, the Recall is very bad. Consequently, I chose to try the linear kernel function for the first dataset, and it performed slightly better than the rbf one. After tuning the hyperparameter "C" of the linear learner to C=0.1, the accuracy of the linear kernel is 68.53%, which is until now the best performances we got for the football dataset.



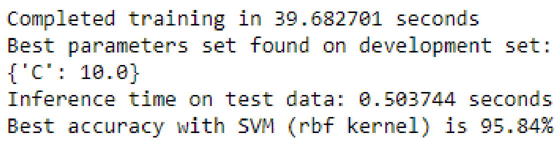
```
Best parameters set found on development set:
{'C': 0.1}
Inference time on test data: 0.958023 seconds
Best accuracy with SVM (linear kernel) is 68.53%
```



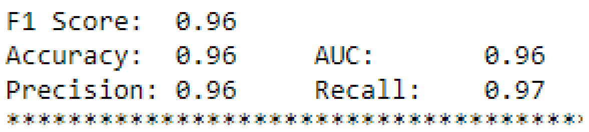
```
F1 Score:   0.66
Accuracy:   0.69      AUC:        0.68
Precision: 0.70       Recall:     0.62
*************************************
```

Phising websites dataset

For the second dataset, after tuning the hyperparameter C= 10 (gamma is tuned automatically by the learner), we got an accuracy of 95.85%:



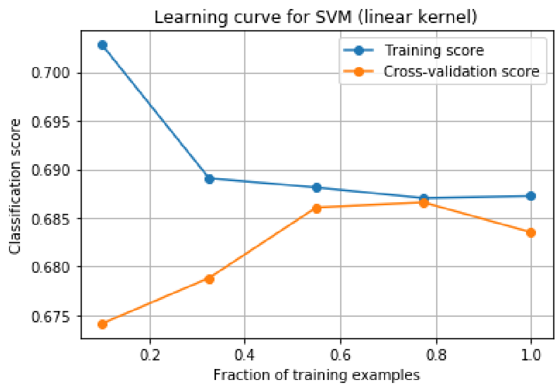
```
Completed training in 39.682701 seconds
Best parameters set found on development set:
{'C': 10.0}
Inference time on test data: 0.503744 seconds
Best accuracy with SVM (rbf kernel) is 95.84%
```



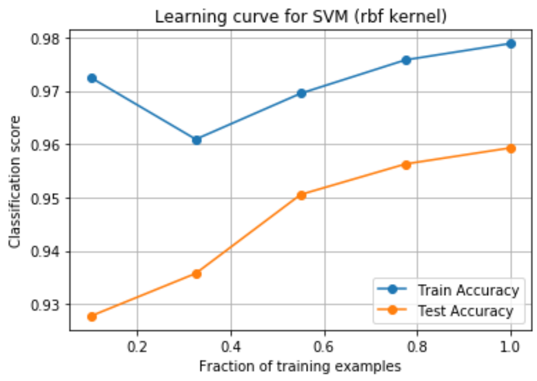
```
F1 Score:   0.96
Accuracy:   0.96      AUC:        0.96
Precision: 0.96       Recall:     0.97
*************************************
```

We note that the SVM classifier does not introduce a bias for this dataset since the F1-score is almost equal to the accuracy.

Learning curves



Learning curve of the football dataset



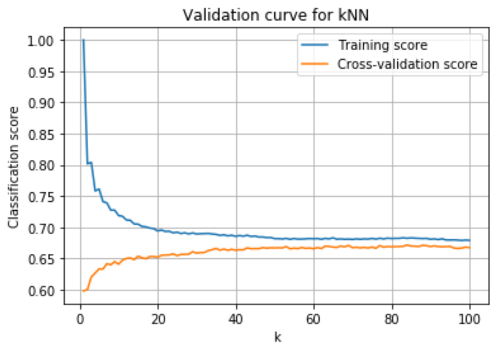
Learning curve of the phishing websites dataset

Football Dataset: If we look at the learning curves of the linear SVM, we note that once again, the classification score won't exceed 69%. We also note that with more data, the linear kernel accuracy might slightly decrease at it does for the last 20% of our dataset. Indeed, there is an important part randomness in the football dataset, so after a given number of training examples, our dataset become less and less linearly separable, and the linear SVM loses efficiency. Yet, there is no interest in increasing the dataset size because we almost reached the boundary of 69% of accuracy for this learner.

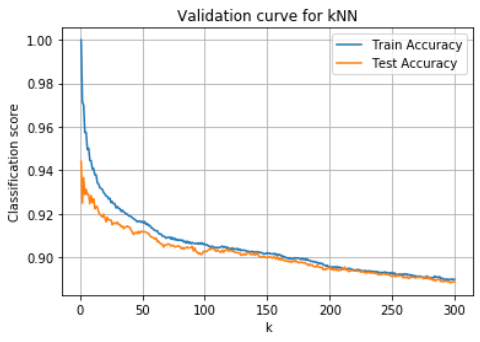
Phishing websites dataset: The learning curves do not show a decrease of the training score toward the cross-validation score. Therefore, since both scores keeps increasing as we increase the amount of data used by the learner, it seems that increasing the dataset size might increase the performances of our SVM.

K-NN

The principle of K-NN is to classify a given element by attributing it the label that appears the most in its neighborhood. Therefore, the hyperparameter to tune is the number of neighbors K that are used in the decision process.



Validation curve of the football dataset

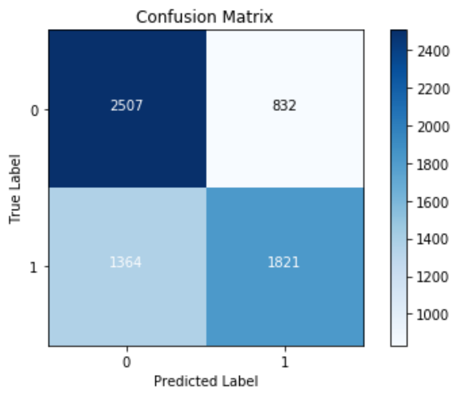


Validation curve of the phishing websites dataset

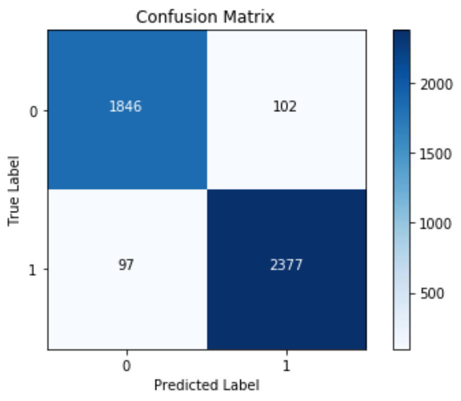
For the first dataset, with low values of K, our model suffers of overfitting since it predicts well the class over the training set, but not over the testing set, that indicates a high variance (which is coherent since the few neighbors used by classifier for the query might vary a lot from one dataset to another because of the noise). Then, for K between 50 and 100, our learner fits correctly the data, since the training and the classification score are very close. Then, when K keeps increasing, our classifier will start to underfit since it will not capture the singularities of the training set.

For the second dataset, low values of K give the best result, which indicates that our dataset is very representative of the reality. However, it starts underfitting very quickly when K increases.

The confusion matrixes are the followings, once again, the FNR and the FPR are close, so accuracy is a good criterion for the interpretation.



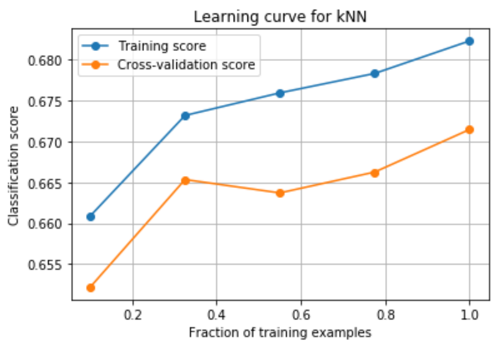
Confusion Matrix of the football dataset



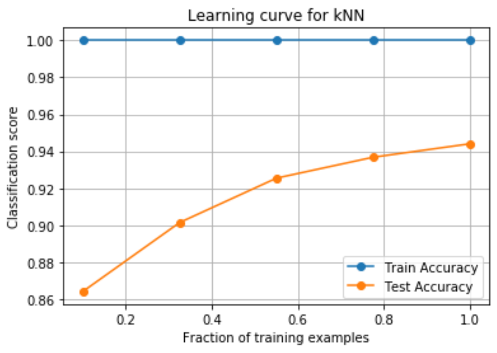
Confusion Matrix of the phishing websites dataset

From a grid search, we deduce that the best number of neighbors is 86 for the first dataset, and 1 neighbor for the second one. The classifiers trained with this hyperparameters has respectively 66.31% and 95.50% of accuracy.

Learning curves



Learning curve of the football dataset



Learning curve of the phishing websites dataset

For the first dataset, K-NN is the first algorithm for which the learning curve of the training and cross-validation scores does not converge: when the amount of training data increases, both the training score and the cross-validation score keeps growing. Moreover, the training score is much greater than the cross-validation one, so increasing the dataset size will very likely improve its predictions.

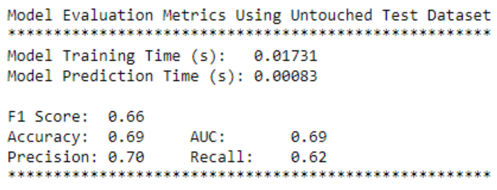
On the second curve, we note that the training score is always equal to 1 since we only took 1 neighbor for our classifier, so it labels an element of our training dataset with its actual output. (Remark: each element of the second dataset has appears only once). Also, the behavior of the curve indicates that increasing the dataset size might increase slightly the accuracy of the classifier.

Neural Network

For a neural network, several hyperparameters must be tuned to get the higher accuracy. Of course, the number of hidden nodes and hidden layers are essential in the design of a NN classifier. The choice of the solver and the activation function used are also determinant for the performances of our learner. Finally, we can tune the learning rate is determinant to control how fast the model will learn the problem, and the L2 penalty term alpha.

The best number of hidden units was computed for several numbers of hidden layers (1, 2, 5 and 10 hidden layers). For both datasets, we trained classifiers with different number of hidden layers, and then, we compared their scores after tuning of the number of hidden units, the learning rate and alpha. Then, we compared their scores after tuning of the number of hidden units, the learning rate and alpha.

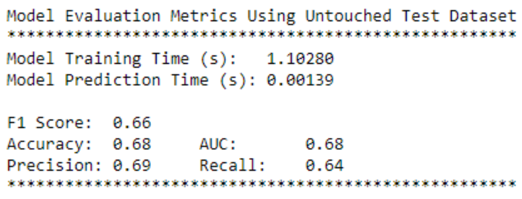
Football dataset:



```
Model Evaluation Metrics Using Untouched Test Dataset
*****************************************************
Model Training Time (s):   0.01731
Model Prediction Time (s): 0.00083

F1 Score:  0.66
Accuracy:  0.69     AUC:       0.69
Precision: 0.70     Recall:    0.62
*****************************************************
```

Scores of the NN with one hidden layer

after hyperparameters tuning



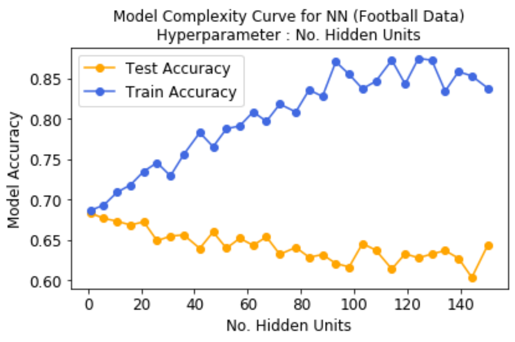
```
Model Evaluation Metrics Using Untouched Test Dataset
*****************************************************
Model Training Time (s):   1.10280
Model Prediction Time (s): 0.00139

F1 Score:  0.66
Accuracy:  0.68     AUC:       0.68
Precision: 0.69     Recall:    0.64
*****************************************************
```

Scores of the NN with 10 hidden layers

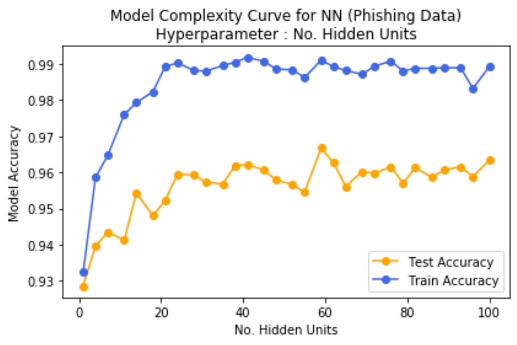
after hyperparameters tuning

During of several tests, the best number of hidden layers was one for both the first and the second dataset. (The scores are not plot for the second dataset by lack of space). Now let’s look more into details the tuning of the other hyperparameters:

With one hidden layer for both NN, we got the following validation curves for the number of hidden units:



Validation curve of the football dataset



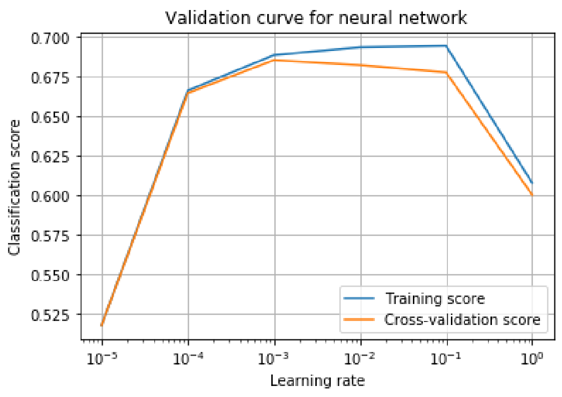
Validation curve of the phishing websites dataset

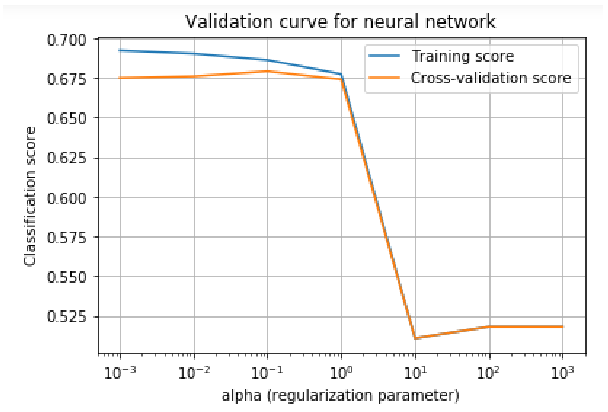
As we can see above, the best accuracy was obtained for a network of 1 hidden layer and 5 hidden units for the football dataset, and a network of 1 hidden layer and 60 hidden units for the phishing website dataset.

Now that we know the number of units and layers to incorporate in our two neural networks, we will tune the other hyperparameters. We note that our Neural Networks are small, so, logistic sigmoid activation function can be used without fear of the vanishing gradient problem. Now, we can tune the learning rate and L2 penalty term with a grid search:

Since the validation curves are very similar for both datasets, we only plot the curves of the first dataset for our analysis:

Validation curves of the first dataset:

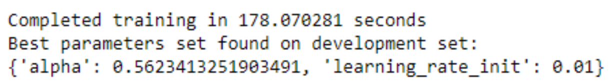




Learning rate: When the learning rate is too low, the Neural Network can't learn. Indeed, the gradient descent algorithm can't find a local minimum during the restricted number of iterations it has to converge, because the step is too small. On the contrary, when the learning rate become too large, the gradient descent doesn't find minimums, because a too large step makes it oscillate on both side of the minimum without reaching it.

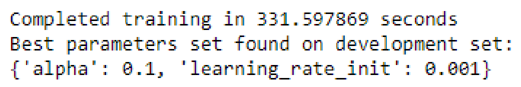
Alpha: When the L2 penalty exceeds a given point, in introduces a bias in our neural network, so both the train and test accuracy falls.

The tuning of alpha and the learning rate gave the following results:



```
Completed training in 178.070281 seconds
Best parameters set found on development set:
{'alpha': 0.5623413251903491, 'learning_rate_init': 0.01}
```

Hyperparameters tuning of the football dataset

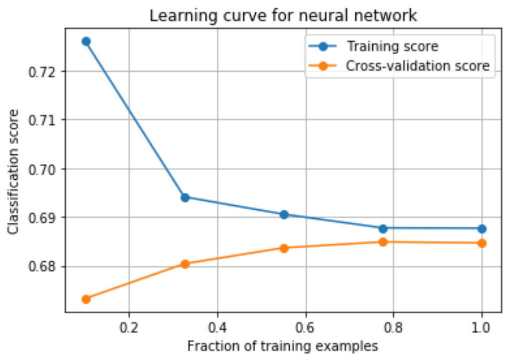


```
Completed training in 331.597869 seconds
Best parameters set found on development set:
{'alpha': 0.1, 'learning_rate_init': 0.001}
```

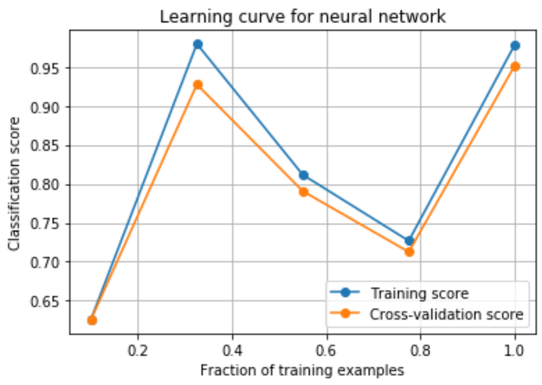
Hyperparameters tuning of the phishing websites dataset

After hyperparameters tuning, our classifier has an accuracy of 69.25% for the first dataset, and 96.26% for the phishing dataset.

Learning curves



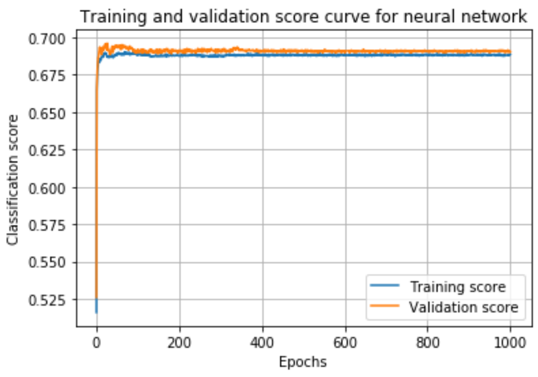
Learning curve of the football dataset



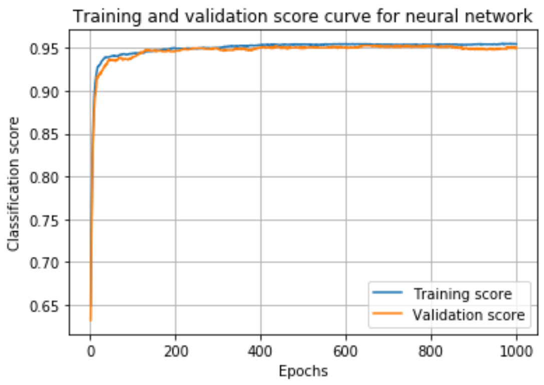
Learning curve of the phishing websites dataset

For the football dataset, the train score converges with the cross-validation score toward 69% of accuracy, so of classifier almost reach the maximum of its capacity (with 68.87% of accuracy), so it won't be useful to increase the dataset size. For the Phishing websites dataset, both the training score and the cross-validation scores are very high at the end of the learning step, so our classifier is fitting the data correctly, but it seems that increasing the dataset size will increase the Neural network accuracy.

Loss curves



Loss curve of the football dataset



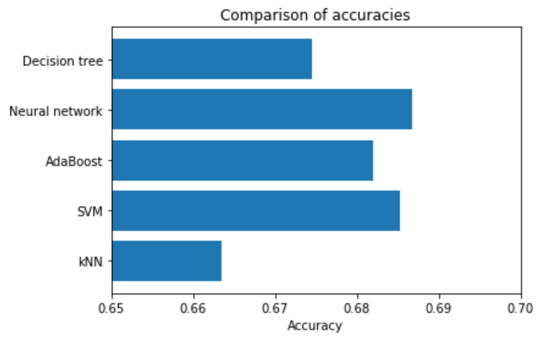
Loss curve of the phishing websites dataset

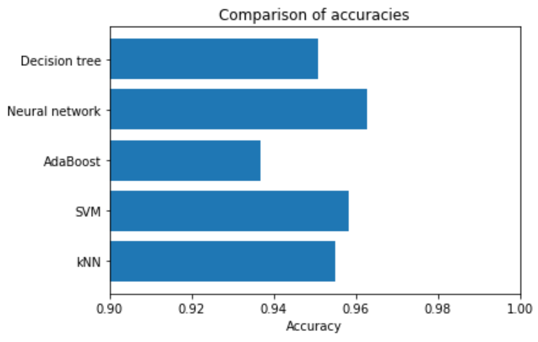
For both the football and the phishing website datasets, the training score and the validation score converge when the number of epoch increases. We note than when the number of epochs is to small, our learners underfit (they both have a low classification score). Indeed, a neural network need to be passed the full dataset several times to train its weight accurately (one passage is one epoch), and we observe that 50 epoch are needed to fit the first classifier, and almost 200 for the second. This can be explained because the first network is far smaller than the second one, so less weights must be trained. Moreover, our first dataset contains less features than our second one, so less information need to be captured by the classifier.

Comparison of the 5 classifiers

Accuracy

For both datasets, the criterion I used to compare their performances is the accuracy. Indeed, the False positive rate and the false negative rate are very close, so the recall, the F1 score and the precision are almost equal to accuracy. The only exceptions are the neural network and the SVM trained over the first dataset, since their precision is slightly better than their recall. Yet, I decided to keep the accuracy as our criterion because the difference is not tremendous, and a false negative prediction is not more severe than a false positive one for a football game.



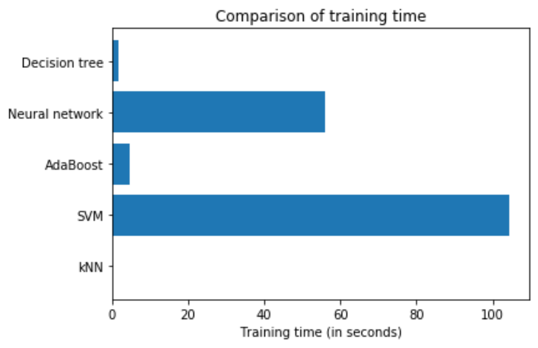


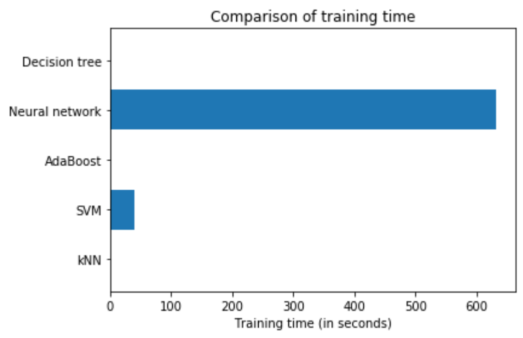
Comparison of accuracies for the football dataset *Comparison of accuracies for the phishing websites dataset*

For the football dataset, kNN and the decision tree are the learners with the worse accuracy. It is coherent that the decision tree does not perform well since its depth is very low ("max_depth" = 5), and our football dataset includes a lot of noise. Then, we saw on the learning curves that KNN is the only one of our 5 classifiers that might benefit from increasing the dataset size. Adaboost gives significantly better results, but it is surprisingly surpassed by the SVM with a linear kernel. Yet we saw that even if the SVM gets a better accuracy, it introduces a bias in our prediction since the F1 score is lower than the accuracy. We can make the same remark about our neural network, because even if it reaches the highest accuracy, it also introduces a prediction bias that increase the FNR and reduces the FPR, which is coherent since our dataset is composed of slightly more negative elements than positive ones.

For the second dataset, Adaboost is surprisingly the algorithm with the worse accuracy, behind decision tree. This comparation suggest that the choice of decision stumps as weak learner is not pertinent, and deeper decision trees or small Neural Network might have given better result if used as weak learners. The decision tree and the KNN classifier get very similar results. It is also interesting to note that when the adequate kernel function is chosen, the SVM got surprisingly almost as good results as a neural network of 100 hidden units. Then, the neural network gets the best accuracy for this dataset. We can also note that except for Adaboost that require to change of weak learner, increasing the size of the dataset might increase slightly the accuracy of our classifiers, as it is suggested by the learning curves.

Training time



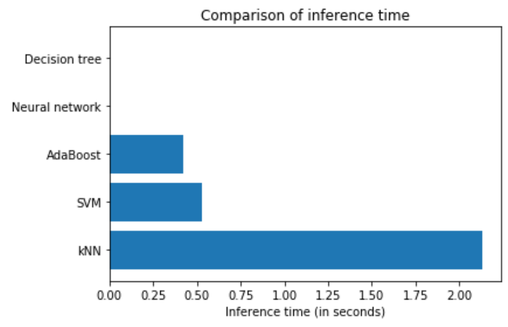


Comparison of training time for the football dataset *Comparison of training time for the phishing websites dataset*

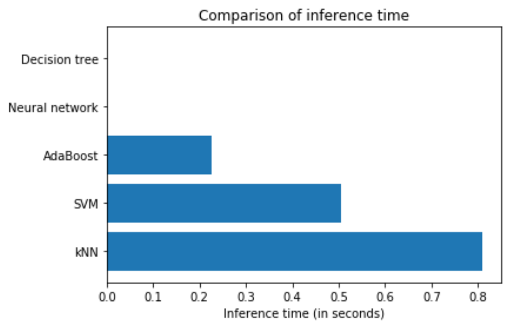
We can first note that the training time of KNN is the lower in both cases, because during the learning process, kNN does nothing more than storing the data, which takes a constant computational time. Then, our decision trees are also very fast to train since their dept is very low thanks to pre-pruning. The learning of Adaboost is also very fast because the weak learners are decision stumps, which are very fast to train. Moreover, number of boosting rounds is limited to few hundreds in both cases. Then, the training of our

SVM is significantly longer since it requires to solve a quadratic optimization problem, that generally takes a computational time between O(n^3) and O(n^5), depending on its hyperparameters. Finally, the training of our neural network is quite long: even for the first dataset where our classifier has only 1 hidden layer and 5 hidden units, the training is significantly longer than the one of Adaboost. This can be explained because the training of a neural network requires several epochs, which is not the case for the other classifiers.

Query Time



Comparison of query time for the football dataset



Comparison of query time for the phishing websites dataset

For both the football and the phishing dataset, the decision tree depths are low with respectively 5 and 15 nodes. Since querying an element with a decision tree involves passing through the tree, it is coherent that our query time is very quick for this classifier. It is almost the same idea for the Neural network which is made of only one hidden layer in both cases, so it is very fast to go through the network to label an element. The query time of Adaboost is a bit longer in both cases. Indeed, for both datasets, it requires to make a vote over few hundreds of decision stumps, so it is coherent that this query time is a bit longer than passing through only one decision tree. Yet, we might improve a lot this time by reducing the number of weak learners used, noting that the learning curves show similar results with less than 100 weak learners in both cases. Then, SVM require to compute a dot product, that takes 0.5 seconds in both cases. Finally, the K-nearest neighbors has the highest query time since it requires to look all over the data space to find the nearest neighbors. We note that for the second dataset it is a bit quicker since KNN takes only one neighbor, so it does not require a vote to label the query point.

Conclusion

The different learners tried showed very good result for the phishing website dataset. It is pertinent to use Machine Learning for such a problematic since there are millions of websites online that need to be checked, and it is easy to obtain the different features needed to make the classification (for example the length of the URL, or the redirections toward another website, etc.). Yet, the results on the football dataset show the limits of Machine Learning when the data is not pertinent enough to be used: almost all classifier trained upon this dataset reached an accuracy close to the maximum to might get by increasing the amount of data, yet accuracy does not exceed 70%. This suggests that getting better results would imply to increase the features size drastically (to get information on the players, for example), but consequently, the amount of data needed to train a classifier would increase exponentially. Yet, less than 50.000 official international football games were played until today, so it seems difficult to reach an accuracy of 80% will the existing data. Yet, some international selections such as the Qatar start to acquire daily data upon their player, and this practice is likely to spread toward the other selections, so in the future, we can hope that Machine Learning might have better results using these new data.