Supervised Learning

Spring 2021 CS7641 Assignment 1

Hung-Hsi Lin   hlin83@gatech.edu

ABSTRCT

This paper aims to apply different machine learning algorithms, including decision trees, boosting, k-nearest neighbors, support vector machines and neural networks on two different datasets. The prediction performance, optimization steps and efficiency of each algorithms are compared and analyzed in this paper.

1 INTRODUCTION

Before diving into the results, the information of two selected interesting datasets and the preprocess applied on them will be introduced first. Then the steps of implementation and optimization on all five algorithms will be illustrated. Finally, the prediction results and the required time in training and prediction for each algorithm are summarized, comparing which algorithm offers the best performance in terms of speed or accuracy in these two datasets.

1.1 Dataset selection and preprocess

The first dataset I selected is phishing website [1], checking if the browsing webpage is suspicious or not. There are 11054 instances, 32 features and 2 classes in this dataset. The features include requestURL, redirecting, abnormalURL, popwindows,..etc. The output class is a binary result, 1 means the instance is a phishing website, 0 means not. For each feature, they have only 3 results, 0, +1 or -1. Therefore, this dataset is a typical classification problem and relatively easy to solve by some algorithms. For the second dataset, I tried to find the features which have continuous numbers so that I could tell if the algorithm works well on both discrete and continuous data. Besides, I also looked for the dataset which has more instances but fewer features, so that I could tell if the time complexity for training and prediction depends more on feature or instance size. Since a lot of my family members are suffering from the same disease, I decided to study the diabetes [2]. All the features, including age, times of pregnancy or BMI are continuous numbers, which is different from the first phishing dataset. The output is 0 or 1, showing if the patient have diabetes. The instance size is 15000 and have only 10 features. By analyzing the running time for each algorithm, I expect to tell if the training/query time depends more on feature/instance size.

For data preprocess, I first dropped all the instance having NA, and meaningless column (e.g., patient ID, index). Each column was normalized, the numbers are rescaled to be ranging from 0 and 1. Then I split 80% dataset into training set and saved 20% for testing. The training set is used for classifier building and validation, and the testing set is only used for verifying the scores obtained by each optimized classifier in the end.

1.2 Algorithm optimization process

For each algorithm, I first used the default arguments for initial tests. Then the gridSearchCV method in sklearn with cross-validation of 5 is used for optimizing multiple parameters simultaneously. The cross-validation(cv) is to make sure classifiers do not overfit into specific datasets. If the cv is not high enough, then we might end up "preferring" training on some sets, causing the variance to be higher. After the "nearly-optimized" classifier is built from gridSearchCV method, then the validation curve method is applied for further optimizing on one or two arguments. The f1_weighted and cv = 5 were selected in validation curve method, the parameters which have the highest score in validation will be chosen to construct the best classifier.

1.3 Score and execution time comparison between algorithms

After the models with optimized parameters are built, then the prediction accuracy in validation and testing sets is calculated using f1_score method with weighted average. Compare with the case using recall or precision, f1 score is a better way to determine the model prediction accuracy for imbalanced datasets [3]. The time for training/testing is also recorded, so that we could finally compare the performance for each algorithm in both accuracy and efficiency.

2 IMPLEMENTATION AND OPTIMIZATION ON ALGORITHMS

2.1 Decision Trees

For Decision Trees (DT), the pruning action could reduce the size of tree and mitigate the overfitting issue [4]. Therefore, ccp_alpa, the complexity parameter used for Minimal Cost-Complexity Pruning is the parameter could strongly affect the performance of the classifier [5]. Before fine tune on the cpp_alpha parameter, the gridSearchCV is applied to optimize the other parameters including criterion and splitter for both datasets. The returned best estimator showed that gini index with best splitter offers the best scores in both datasets, and the ccp_alpha plays an important role in the final f1 scores. While the ccp_alpha value is 0, no pruning is performed, means the model is too specific for training sets. In the figure 1.a and 1.c, we could clearly see that the accuracy on training is 100% as the ccp_alpha is 0, however, the performance for the validation is really bad, resulting int high variance of the model. As the alpha increases, pruning starts, the training accuracy becomes lower (red) because the overfitting is released, and the validation scores increase because the model becomes more generalized. As the alpha becomes too high, meaning pruning is too aggressive, then both training and validation score become be lower and lower, and the model will be having high bias.

After using gridSearchCV and finely search on alpha, the optimized parameters for Decision Trees classification for both datasets are shown in below table, and the learning curves for training/validation sets are plotted in Figure 1.b and 1.d.

Diabetes				Phishing			
criterion	splitter	ccp_alpha	F1 score	criterion	splitter	ccp_alpha	F1 score
gini	best	0.0003	0.92	gini	best	0.0001	0.963

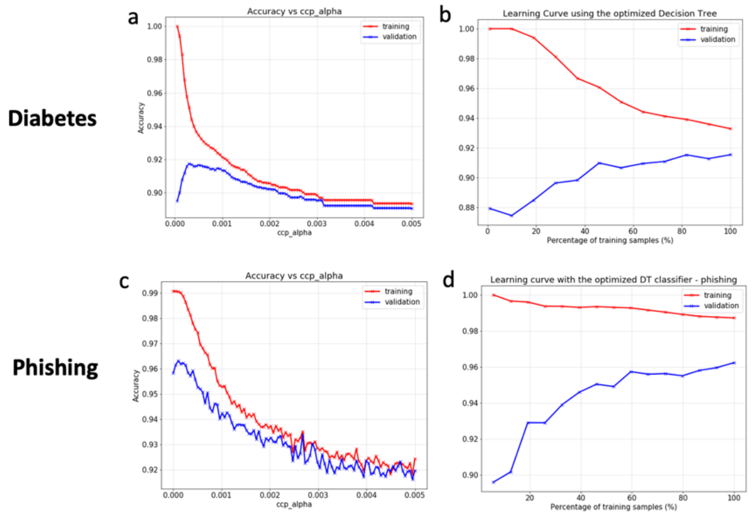

Figure 1— The accuracy under variant ccp_alpha value for (a) diabetes and (c) phishing datasets. The calculated accuracy using the optimized parameters for Decision Trees classifier are shown in (b) and (d) for diabetes and phishing, respectively.

2.2 Boosting

The idea of boosting is an ensemble technique in creating a strong classifier from a mount of weak classifiers [6]. In this assignment, AdaBoost was implemented [6]. By following the same procedure mentioned in decision tree, gridSearchCV was first applied to find the best parameters for weak classifier. The best parameters of the decision trees are with splitter of best and max_depth of 1, act as the stumps instead of huge trees. One of the important parameters for AdaBoost classifier is the n_estimator, which means the numbers of weak learner to iteratively train. In Figure 2.a and 2.d, the validation scores are really low for both datasets as the n_estimators are lower than 10, means the number of estimators is insufficient for making right decision from voting. As n_estimators get higher, the accuracy increases and then get saturated, demonstrates the fact that after a specific number of n_estimator, accuracy will be converged. Another important parameter is learning rate, which shrinks the contribution of each classifier [6]. The performance from default (=1) works pretty well, and the performance decreases a lot while learning rate is larger than 2, where the model got trained too fast for worse accuracy. The final performances of the optimized AdaBoost classifier are shown in Figure 2.c and 2.f. As we could see, from the learning curve, the scores increase dramatically as the size is larger, means that the AdaBoost classifier needs sufficient data size for making good training. The high variance in small sample size proves this concept. The final scores of validation sets for both datasets are high with low biases and perform better than Decision Tree shown before. This observation is expected because we know that the AdaBoost is not prone to overfit compared with DT in theory [7], therefore, better scores in "testing" could be expected as well which will be discussed in section 3.

Diabetes				Phishing			
criterion	n_estimator	Learning rate	F1 score	criterion	n_estimator	Learning rate	F1 score
entropy	225	1.1	0.96	gini	75	1.0	0.971

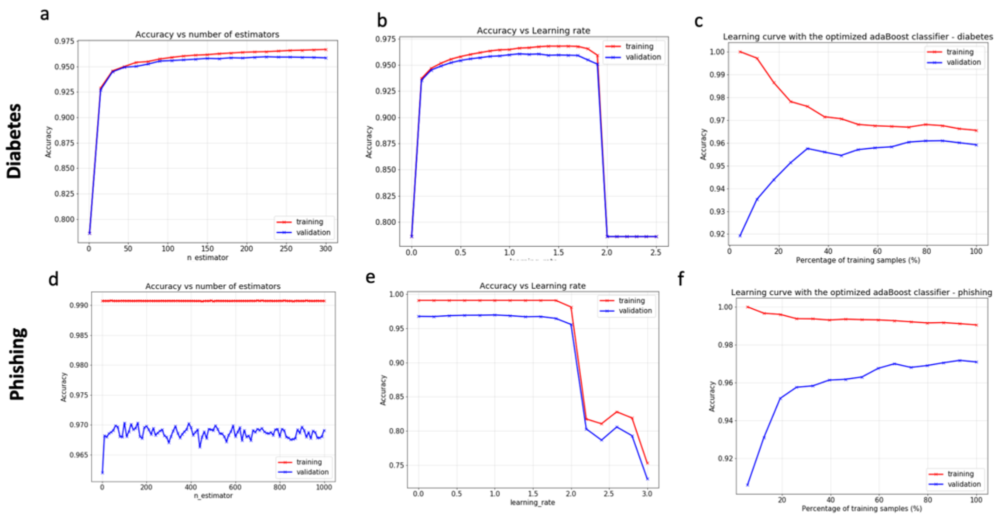

Figure 2 - The accuracy under variant n_estimator for (a) diabetes and (d) phishing datasets. The accuracy under variant learning_rate for (b) diabetes and (e) phishing datasets. The accuracy using the optimized parameters AdaBoost classifier are shown in (c) and (f) for diabetes and phishing, respectively.

2.3 k-Nearest Neighbors

For k-Nearest Neighbors (KNN), the most important parameter is for sure the number of neighbors. Besides, I also have interest on the weight function used in prediction, either uniform, meaning all points in each neighborhood are weighted equally, or distance, while the weight points by the inverse of their distance. In figure 3.a and 3.d, the plots show the influence of neighbor numbers using uniform weight. For diabetes, the training score is high and validation score is low as the k = 1, showing the fact that the model has high variance due to overfitting issue. As the k increase, the scores from training and validation start converging, variance is getting reduced, however the bias becomes higher. For the case of phishing the validation scores is the highest for k = 1, indicating that this dataset might have a lot of repeated instance making the weights strongly biased. As the k increases, the same trend seen in diabetes occurs, the variance decreases while the bias increases.

As taking the *distance* was used in weights shown in Figure 3.b and 3.e, the training sets are almost independent of neighbor numbers, both datasets show similar results, that the models are overfitted because the centered points has more power on making decision than its neighbors, making the model exhibiting high variance, which could be observed from the difference between training and validation. Therefore, in these two datasets, unform weight might be a better selection to avoid high variance. The learning curves using the optimized parameters for KNN are plotted in Figure 3.c and 3.f for both datasets, and we could find that comparing with other classifier, KNN tends to have larger bias and larger variance. The details of comparison will be demonstrated in following sections.

Diabetes				Phishing			
weights	leaf_size	neighbors	F1 score	weights	leaf_size	neighbors	F1 score
uniform	30	5	0.877	uniform	30	1	0.951
distance	30	10	0.878	distance	30	10	0.954

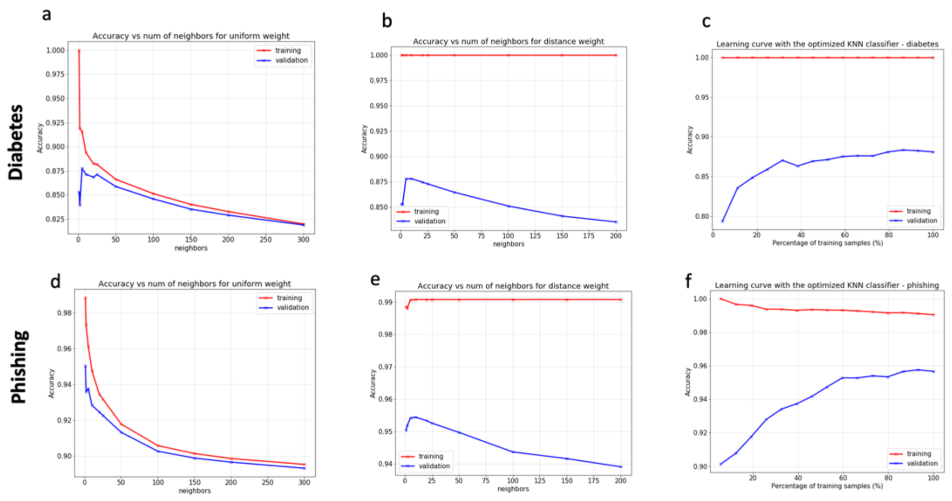

Figure 3 - The scores under different number of neighbors using *uniform* for (a) diabetes and (d) phishing datasets. The scores under different number of neighbors using *distance* for (b) diabetes and (e) phishing datasets are also shown. The learning curve of KNN classifier with optimized parameters are shown in (c) and (f) for diabetes and phishing.

2.4 Support Vector Machines

SVM creates virtual planes to separate classes using different Kernel functions [8], therefore, the most interest parameter will be the Kernel selected in algorithm.

Linear, polynomial (poly), Radia Basis Fuction (rbf) and sigmoid are the Kernel used for testing. Besides, the regularization parameter (C) and Kernel coefficient (gamma) control the regulation also affect strongly on the performance of the classifier. I used the same strategy for narrowing down the searching range using gridSearchCV first, found rbf and poly Kernal works the best in my datasets. Then optimize C and gamma for rbf and poly Kernel respectively, also searched for the best degree. In Figure 4.a and 4.d, the performance using rbf Kernel with optimized C and gamma performed pretty well in both datasets. As the size increases, both variance and bias decreases. For poly Kernel, the degree of polynomial was optimized in Figure 4.b and 4.e for both datasets. In both cases, the biases are high and variances are low at low degree. And as the degree increases, scores of training set keep increasing, but the validation goes down after the minimum degree, means the model becomes overfitting to training set and resulting in high variance and high bias.

Compared with the results obtained from poly and rbf Kernel, we could see in both cases, rbf Kernel works better than the poly Kernels especially for diabetes cases. It implies that the polynomial Kernel might work well for discrete data type, but not the continuous numbers like BMI, age features included in the diabetes dataset. The learning curves are plotted in Figure 4.c and 4.f with the optimized degree, Kernel, gamma and C in SVM classifier. The accuracy is not that different compared to the results analyzed in previous section, but the required time for optimizing the parameters do take longer than the other algorithm listed before. The details about the required time for training/query will be discussed in details in later sections.

Diabetes				Phishing			
Kernel	C	gamma	F1 score	Kernel	C	gamma	F1 score
rbf	17.5	1.875	0.901	rbf	10	0.3333	0.965
Poly – 4 degree	4	1.875	0.865	Poly – 5 degree	1	0.3333	0.962

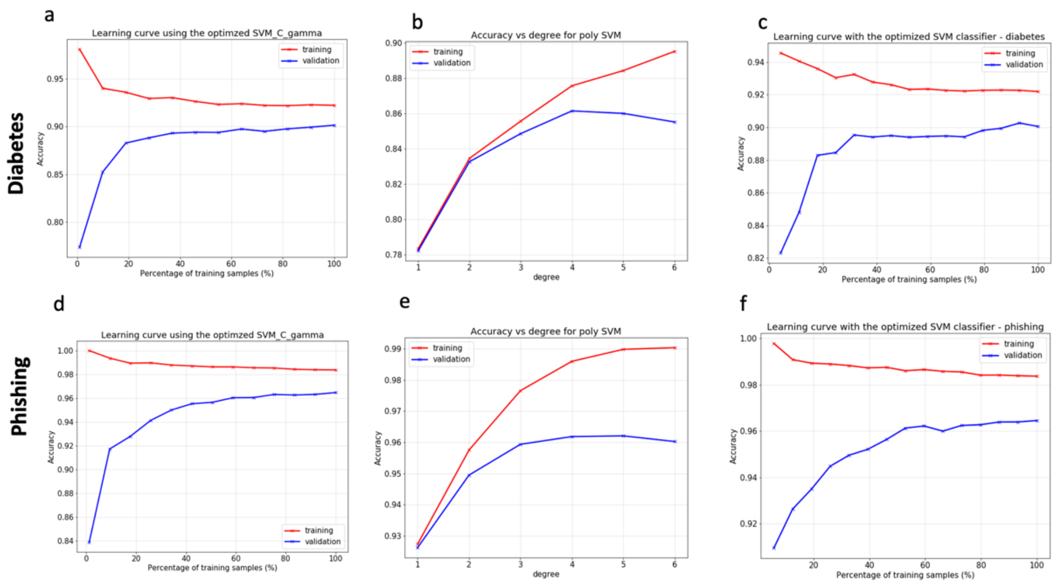

Figure 4 - The accuracy using the optimized *rbf* Kernel for (a) diabetes and (d) phishing datasets. The accuracy under variant degree of *poly* Kernel for (b) diabetes and (e) phishing datasets are also shown. The accuracy using the optimized parameters of SVM classifier are shown in (c) and (f) for diabetes and phishing, respectively.

2.5 Neural Networks

Neural Networks performs backpropagation method for optimizing the weight of neuron and the bias [9]. The most interesting parameters to be tuned in Neural Network are the hidden layer size, representing the number of neurons in ith layer, the penalty parameter alpha, activation function and learning rate for weight updates. Again, there are too many parameters to be optimized, therefore, the gridSearchCV again used first for narrowing down the range of each parameters. The activation function as relu and the learning_rate as constant are found to be the best in both cases, also the hidden layer size of (250, 500). During the gridSearchCV scan, I found that the larger the neuron size in hidden layer, more time is required for training for each weight. Another interesting thing is that a larger size of (300, 600) was not selected, demonstrating the fact that more hidden layer size will cause overfitting and increase the bias in model.

The finer scan of penalty parameter alpha is shown in Figure 5.a and 5.c. The small number of alpha could help lowering both bias and variance for validation sets, however, it increase the bias for both training and validation sets dramatically as the number goes higher. So the number of alpha has to be carefully tuned. The final results of the optimized Neural Network classifier are shown in Figure 5.b and 5.d, showing low bias and low variance for both datasets. And we could clearly see the validation scores is highly depends on the percentage of instance used, more data helps achieving better performance in learning curve.

Diabetes				Phishing			
Hidden layer	Activation function	Alpha	F1 score	Hidden layer	Activation function	Alpha	F1 score
(250, 500)	relu	0.5	0.93	(250, 500)	relu	0.005	0.963

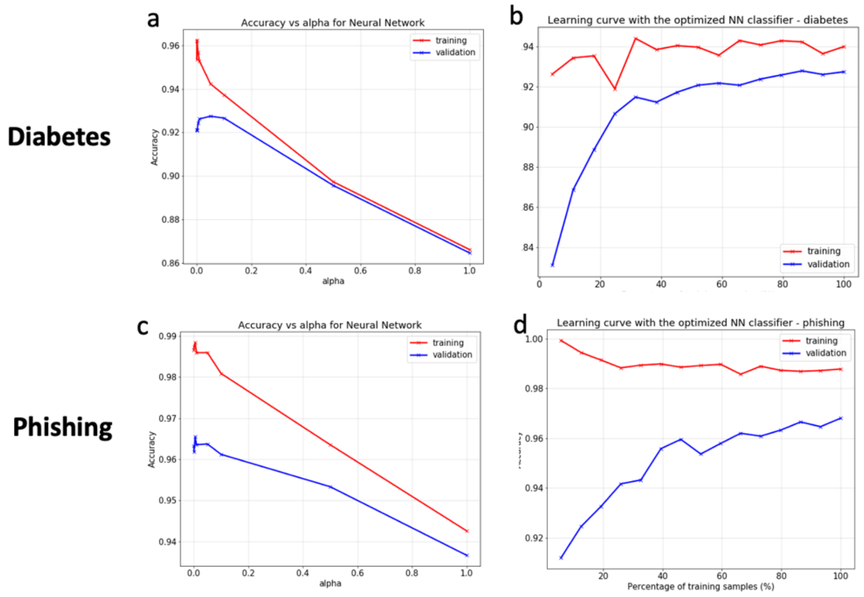

Figure 5 - The accuracy under variant alpha for (a) diabetes and (c) phishing datasets. The accuracy using the optimized parameters of Neural Network classifier are shown in (c) and (f) for diabetes and phishing, respectively.

3 COMPARISON

3.1 Accuracy

Figure 6 show the calculated accuracy from training set and testing sets for each algorithm in both data sets. The difference of scores between training/validation and testing are close in all algorithm, this low variance means the optimized models are generalized in both data sets. And the prediction accuracies on testing sets are mostly above 90%, demonstrates the fact that the biases for all models are also low. From Figure 6.a, most of classifiers work well for diabetes datasets besides KNN, the only one which has scores in both training and testing sets lower than 90%. In phishing datasets, KNN works pretty well and comparable with other algorithms. The main difference between two datasets is the type of features, which is continuous for diabetes and discrete for phishing. While the feature type is continuous, KNN may perform worse than other algorithms. DT also works well for phishing dataset but not that good diabetes, demonstrating the fact that continuous feature type might degrade the performance of DT, too. On the other hand, AdaBoost works better than Decision Trees and KNN in both datasets because Boosting tends to avoid overfitting, so it might be more independent from the type of dataset. SVM works well also on phishing data but not diabetes, implies that the Kernel for separating classes might need to be further optimized for continuous type of data. Neural Networks works pretty well in both datasets in terms of accuracy, along with its nature of expensive calculation time. The performance is still a little bit worse than AdaBoost which is not expected. The reason might be the hidden layer and other parameters for Neural Network was not optimized or explored enough, due to the lack of time.

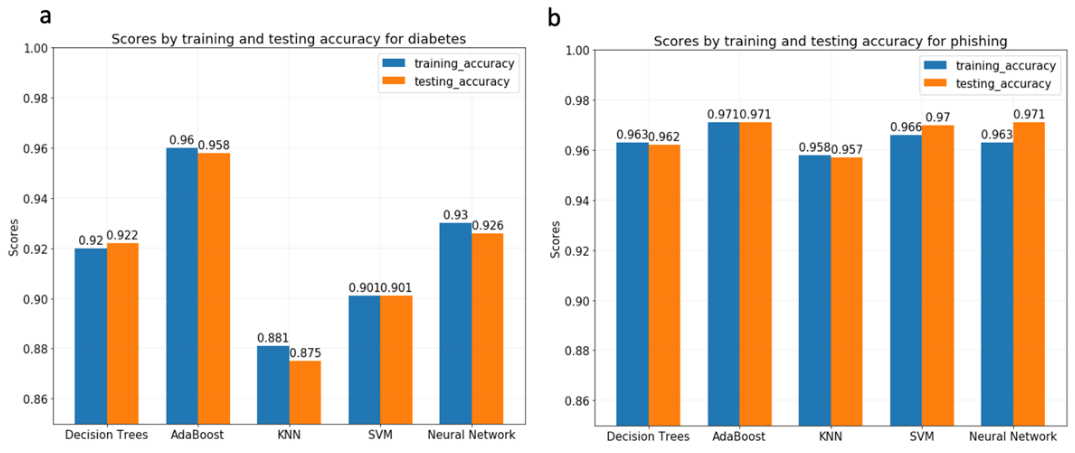

Figure 6 - The scores calculated from training (blue) and testing sets (orange) for (a) diabetes and (b) phishing datasets.

3.2 Time

The required time for training and testing for each classifier are summarized in Figure 7.a (diabetes) and 7.b (phishing). KNN is the only algorithm which needs longer testing time than training time, because there's no need for training in KNN model, and the testing time is highly depends on the number of selected neighbors. The most time-consuming model to train from my experiments is the Neural Networks, and I think the reason is that (250, 500) hidden layers were chosen for better accuracy. If the number of layers could be reduced, for sure the time could be reduced a lot, however, the calculated biases and weights might not be that accurate if the numbers of "neurons" are not enough. Good news for Neural Networks is, testing time is way faster than training time and also faster compared with other algorithms because the calculation is relatively straightforward which uses the existing biases and weights in neurons. Decision trees is the algorithm works really efficient in both training and testing because there's only one tree to be built and just one traversal along the tree for each query request. The accuracy is still above average from the previous accuracy tests, but the performance of decision tree might not be that good for more complexity datasets. The required time of training and testing for AdaBoost algorithm are highly depends on the n_estimator settings. I found that the time required for training takes really long while the n_estimator parameter was set too high. Therefore, same optimization work for AdaBoost is necessary to find a balance between accuracy and efficiency. For SVM, the requested time for training and testing are both above average, which are highly depending on the C parameter and the kernel selected (rbf is classified as a time-consuming method).

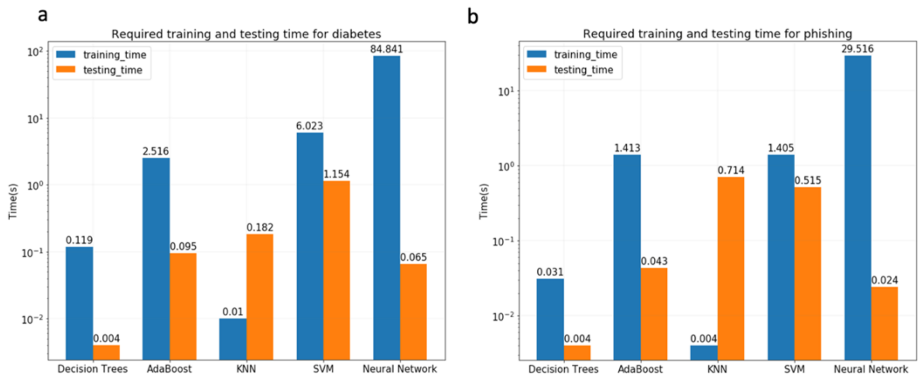

Figure 7 - The time required for performing training (blue) and testing (orange) for (a) diabetes and (b) phishing datasets, the time in y axis is shown in log scale.

4 SUMMARY

The below table summarizes the *relative* properties of 5 optimized classifiers from these experiments and the dependencies of each characteristics.

Classifiers	Testing accuracy	Training Time	Query time
Decision Tree	Medium – depends on pruning and data type	Fast – just need to build a tree, pruning could speed it up	Fast – just need one traversal along the built big tree
AdaBoost	High – overcomes overfitting	Slow – depends on the # of estimators	Medium – depends on the numbers of estimators
KNN	Low – depends on datatype and # of neighbors	Fast – no need for training, just saving the data	Slow – depends on the K and data size
SVM	High – depends on the selected Kernel and parameters	Slow – depends on the selected Kernel and parameters	Medium – depends on the feature size since the distance need to be calculated
Neural Network	High – depends on hidden layer and parameters	Extremely slow – could be pretty bad if large numbers of hidden layer are required	Fast – just need to perform calculations using the existing weights and bias in NN.

5 REFERENCES

1. https://www.kaggle.com/eswarchandt/phishing-website-detector
2. https://www.kaggle.com/fmendes/diabetes-from-dat263x-lab01
3. https://blog.exsilio.com/all/accuracy-precision-recall-f1-score-interpretation-of-performance-measures/
4. https://www.displayr.com/machine-learning-pruning-decision-trees/
5. https://scikit-learn.org/stable/modules/tree.html#minimal-cost-complexity-pruning
6. https://towardsdatascience.com/boosting-algorithms-explained-d38f56ef3f30
7. https://www.datacamp.com/community/tutorials/adaboost-classifier-python
8. https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47
9. https://www.codementor.io/@james_aka_yale/a-gentle-introduction-to-neural-networks-for-machine-learning-hkijvz7lp