CS7641 Spring 2022 Assignment 1

Jian-Yuan Lin
jasonlin0211@gatech.edu

1 INTRODUCTION

1.1 Problem Statement

In this assignment, 5 supervised learning method including decision tree (DT), boosting(I choose the popular adaboost, ADA), neural network (NN), support vector machine (SVM) and k-nearest neighbor (KNN) are investigate and compared. Two classification dataset, Heart Disease UCI [1] and Red Wine Quality Prediction [2], are selected for the implementation purpose of the above methods. After the observation and comparison, my conclusion about using the classic supervised learning methods are presented in the last chapter.

1.2 Dataset Introduction and Pre-processing

In the heart disease dataset, we have 303 row data, including 13 features about the patients like age, sex, resting blood pressure and several medical test results, and 1 target with only binary values 0 and 1. The goal is to classify if the patient has heart disease or not(0 means no, 1 means yes). This is a binary classification problem and numbers of the patients with and without heart disease are 165 and 138, respectively. In other words, this is a balanced dataset, and I hope to investigate how the aforementioned supervised learning methods perform on this typical classification problem. For the data pre-processing, since the cp (chest pain type), thal and slope(the slope of the peak exercise ST segment) are categorical type feature, I created dummy features to express them in 0 and 1 to potentially help the training process. Normalization is also done to ensure the feature values are within 0 - 1. The training/testing data ratio is set as 8:2, as the normal suggested way.

The second set of data is red wine quality prediction problem. In this dataset, there are 1599 rows of data with 11 features, including wine properties like PH value, density, residual sugar ... etc, and 1 target score which represents the wine quality. Figure 1 (a) shows the count of the different quality score. It is clear that although the quality score has a full range of 1 - 10, the data only cover score 3 - 8, which means we can't successfully predict extremely bad(< 3)

or extremely good(>8) wine. Therefore, I decided to change the label of the red wine quality to good or bad, by setting a standard at score 6. If the quality score is <= 6, the wine quality is marked as bad, otherwise the wine quality is marked as good. After changing the label, the classification problem is still a binary classification again, but the numbers of label 0 and 1 are very imbalanced(0: 1382, 1: 217), which can be used to investigate how the supervised learning methods behave on the imbalanced dataset. The same normalization and training/testing data ratio are implemented in the red wine dataset.

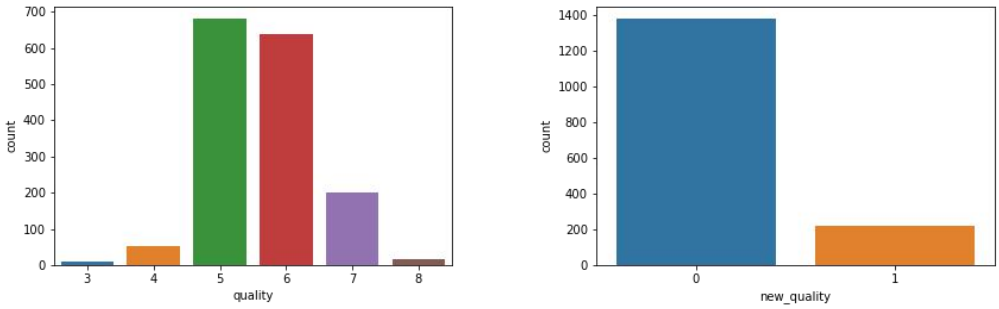

Figure 1—(a)Original wine quality score count (b) Modified new wine quality classification count

1.3 Algorithm Optimization and Evaluation

I use the same procedure to fine tune the parameters for the five supervised learning model in this assignment. First, I will just use the default parameter setting in scikit-learn for benchmark and initial understanding of the model behavior on the dataset. Then for the parameters that I am interested to investigate, I set several values in the parameters grid, and use grid search to combine all of the possible combinations and fit. After the fitting process, the optimal model can be found, and I use the optimal model to conduct the evaluation. k-fold cross validation (k set as 5) is implemented, and the learning curve as well as training time plot to interpret the model behavior and the model scalability. Finally, the f1 score is chosen to be the metrics to evaluate the model accuracy, since it considers both the recall and precision [3].

2 SUPERVISED LEARNING IMPLEMENTATION

2.1 Decision Tree

In my first implementation of decision tree (DT), I implemented it on the heart disease dataset. From the figure 2 below, I immediately found the DT is definitely overfitting the training data, since the training score is always 1, which means it fits the training data too hard and might not be performing well for the testing dataset. There are two variables I am interested in tuning: maximum depth and alpha(complexity parameter for pruning [4]). These two values can reduce the overfitting, while might sacrifice a little bit accuracy on training data. I set [0.05, 0.01, 0.005, 0.001] as options for alpha value and [3, 5, 10, 20] as options for maximum depth.

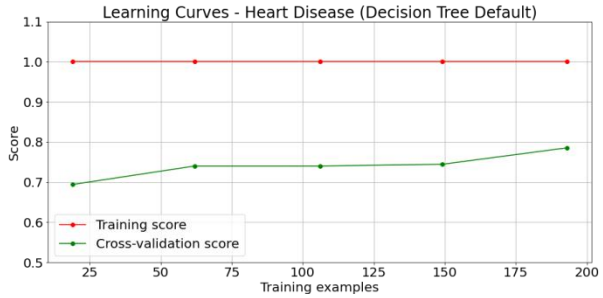

Figure 2— Learning curve of the preliminary test of DT on heart disease dataset

After the grid search, the best alpha = 0.01 and maximum depth = 5 in the heart disease dataset. The learning curves and scalability of the model plot are presented in Figure 3. It can be seen that the model behavior is what normally anticipated: the training score goes down and cross validation score goes up, which means the model is becoming more general while still maintain a good accuracy. But in this plot, the plateau for both line are not reached yet, so if we have more data, we might be able to generate a more robust model. For the red wine data set, the experiment result is very similar. The initial test of DT overfits the data, and after the optimization (found best alpha = 0.001 and maximum depth = 10), I think it still overfits the data, since the drop in the training score is very tiny. I think the reason is that due to the imbalance of the dataset, the model can lean to the majority first, and then just tweak a little bit to get the minority data correct. The scalability plot shows how the training time changes according to the training sample size, and from the both plots on the two datasets, no conclusion can be made at this point, but the training time is relatively fast.

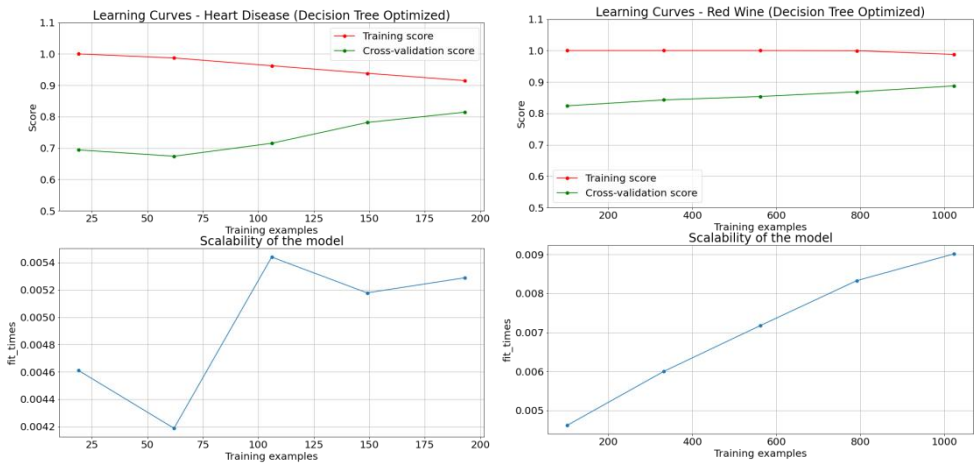

Figure 3—Learning curve and scalability plot of the optimized DT on heart disease and red wine data set

Table 1 — Optimal parameters for DT models on heart disease and red wine data set

Model Parameter	**Heart Disease**	**Red Wine**
Maximum Depth	5	20
Alpha	0.01	0.001

2.2 Boosting

For the boosting method, I choose to use the well known adaboost(ADA) algorithm, and the base estimator I just use the decision tree. Since the base is decision tree, I anticipate the default behavior will also be overfitting, and that is clearly shown in Figure 4. To optimize the model, I choose to search for the same two parameters in DT, the number of estimators and learning rate. Since the adaboost requires iterations to boost multiple weaker estimators, the training process will take longer time as shown in Figure 5. The training results for two datasets are very similar to the decision tree results. This is as expected, since the adaboost strategy is essentially set a number of weak decision tree as base case, then throughout the iterations, try to minimize the incorrect results. Our data is not very big, so it will easily lead to the similar result compared to the decision tree.

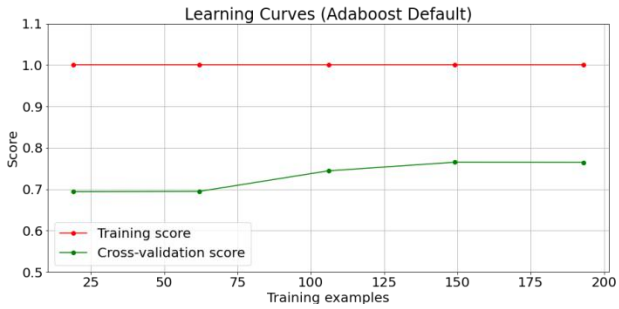

Figure 4— Learning curve of the preliminary test of adaboost on heart disease dataset

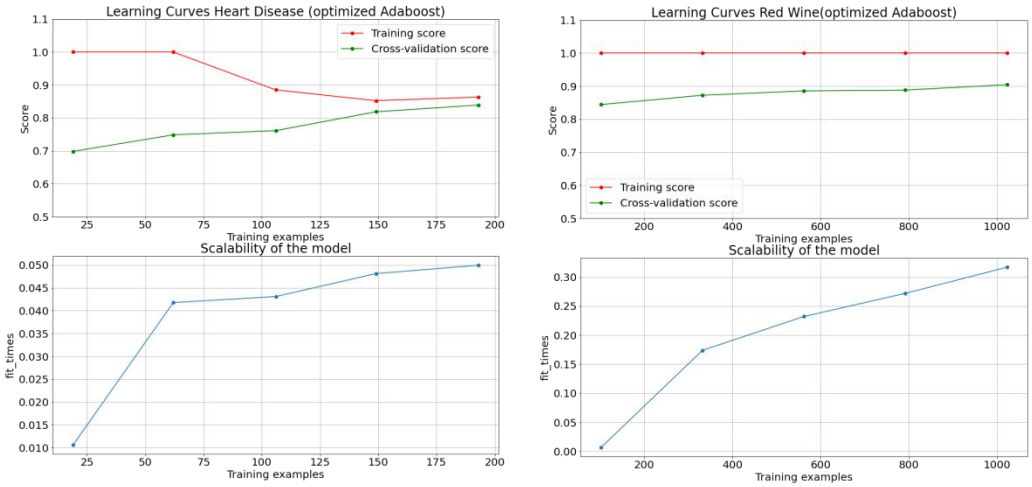

Figure 5— Learning curve and scalability plot of the optimized ADA on heart disease and red wine data set

Table 2 — Optimal parameters for ADA models on heart disease and red wine data set

Model Parameter	**Heart Disease**	**Red Wine**
Maximum Depth	5	5
Alpha	0.05	0.005
Number of Estimators	25	75
Learning Rate	0.1	0.1

2.3 Neural Network

For the Neural Network(NN), I choose the basic multilayer perception model, since compare to the convoluted neural network(CNN), there are less parameters to fine tune and network design is easier. In the initial attempt on the heart disease problem presented in Figure 6, I saw very good model training performance. The training score gradually goes down and validation score goes up, and even reach nearly plateau. In this method, I would like to investigate the tuning of hidden layer size, and the learning rate. I designed different layer number from 1 to 4, and for each layer number, I have different neuron numbers((100), (10,50), (10,50,80), (10, 50, 80, 120)). I didn't follow any particular design rules, but intuitively the deeper the NN model is, it should fit the data better due to more features are available to fit the data. However, throughout the process, I found it is not necessarily true.

In the heart disease problem, the best setup is actually layer = 1, which means just single layer is enough, and among the choices of different learning rate(1, 0.1, 0.01, 0.001), 0.001 is selected. Since larger learning rate could lead to suboptimal solution but lower learning rate could takes more epochs [5]. A epoch means the number of passes of the entire training dataset the machine learning algorithm has completed, and since our data is relatively small, more epochs here isn't an issue, which I guess it's the reason why smaller value was selected in grid search.

Interestingly, in the imbalanced red wine data, the behavior is totally different. Three layer NN structure is selected and the neuron numbers are (10, 50, 80), and the learning rate is selected as 0.001. The training score and cross validation score are basically flat, which I assume is due to the minority data is not sufficient to tune up the parameters in the three layer NN model, and more data is required to further improve the model. In the perspective of model training time, the NN model requires more time to train when the data is larger due to the computation time of the back-propagation process grows with the data size.

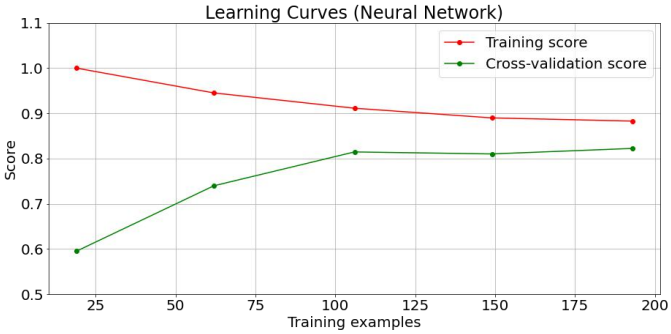

Figure 6—Learning curve of the preliminary test of NN on heart disease dataset

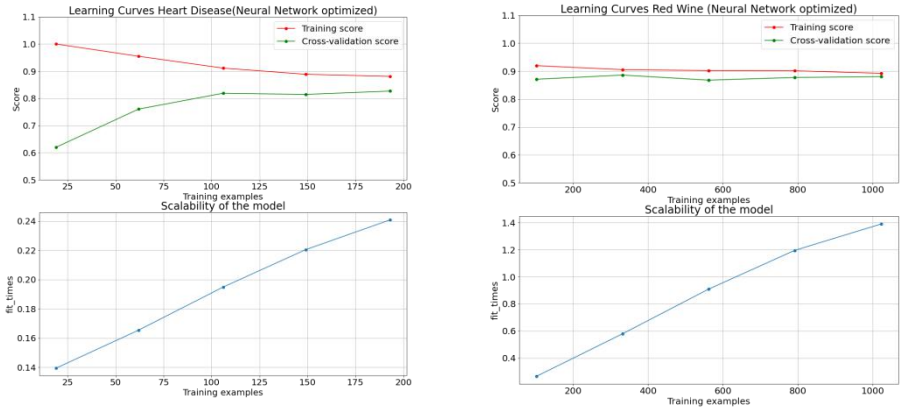

Figure 7—Learning curve and scalability plot of the optimized NN on heart disease and red wine data set

Table 3 —Optimal parameters for NN models on heart disease and red wine data set

Model Parameter	**Heart Disease**	**Red Wine**
Hidden Layer Size	(100)	(10,50,80)
Learning Rate	0.001	0.001

2.4 K-nearest Neighbor

For K-nearest neighbor(KNN) method, the most straight-forward tuning option will be the value of k. In the preliminary try on the heart disease data and red wine data, I use k = 5, which both leads to interesting results. In the heart disease learning curve, the initial point of the training score is actually lower than the following points, which is uncommon. I think the reason is the training examples are around 20, compare to k = 5, which means there are 5 centers in the model, the examples are not sufficient to be able to calibrate a good model. The rest of the mode behavior is more reasonable for the heart disease problem. But for the red wine data, both line are oscillating, and I can't even confirm that the model is actually learning the data pattern. I think this is also due to the data imbalance, guessing the red wine is bad directly is actually a good choice, since the percentage of the bad red wine of the overall red wine is 86.4 %. Therefore, k = 5 is not helpful at all. Therefore, I decided to try the k value from 1 - 30, and see how the individual k value will change the model performance.

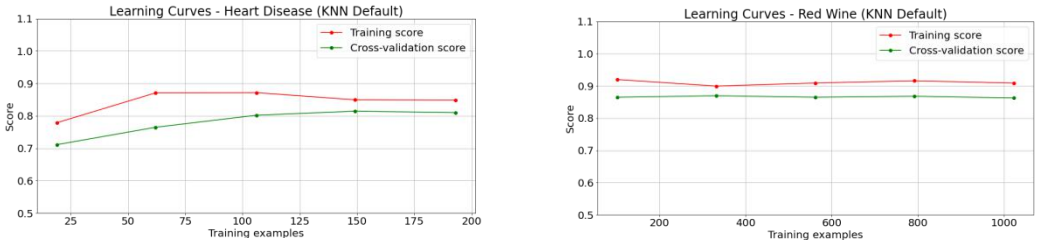

Figure 8—Learning curve of the preliminary test of KNN on both heart disease and red wine dataset

For the heart disease problem, the optimal k is found as 7, and therefore the result is similar to the initial try. And since the centers are even more in this model, it is reasonable that the initial training score is very bad, even lower than the cross validation score. For the red wine dataset, the optimal k is found as 1, and the training score is perfect. This interesting result essentially validates my prior hypothesis. In this imbalanced dataset, KNN actually mimics the behavior of directly guessing the red wine is bad, KNN just use the distance between the representative of bad wines, so if tuned properly, KNN can classify all of the training data correctly. Even for test dataset, it will still

have almost 87 % percentage to be correct, which is considered to be a good model in terms of just accuracy, but this model in fact is not informative nor useful in the real world. The relationship between the model training time and training examples is not apparent in this two figures, but the training process is relatively fast compared to NN or ADA.

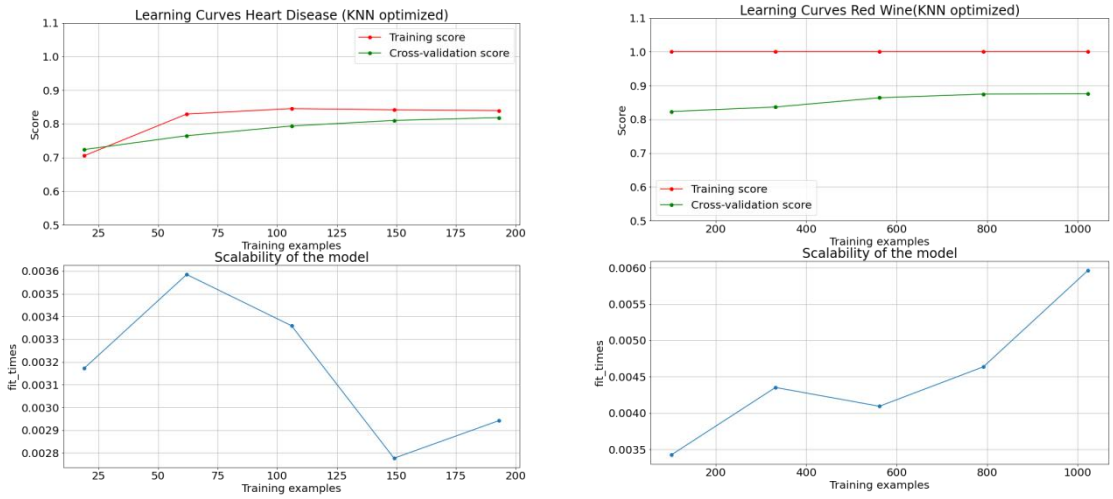

Figure 9— Learning curve and scalability plot of the optimized KNN on heart disease and red wine data set

Table 4 — Optimal parameters for KNN models on heart disease and red wine data set

Model Parameter	**Heart Disease**	**Red Wine**
k	7	1

2.5 Support Vector Machine

Support vector machine (SVM) seperates the features into different domain by utilizing kernel functions, and therefore the kernel method is the parameter that would be interesting to tune. There are four kernels to be chosen: linear, polynomial, sigmoid and radia basis function(RBF). I am also interested in the l2 regularization parameter C to see how it will change the model behavior. Using the same grid search method, the optimal parameters for SVM in both dataset can be found in Table 5.

Table 5 — Optimal parameters for SVM models on heart disease and red wine data set

Model Parameter	Heart Disease	Red Wine
kernel	poly	poly
C(Regularization Parameter)	0.1	2

Interestingly, the selected kernel in both dataset are both polynomial kernel, so we can directly compare the physical meaning in C. Higher value of C will lead the optimizer choose smaller-margin hyperplane, which is less likely to classify incorrectly but slower. On the other hand, lower value of C will lead the optimizer choose the larger-margin hyperplane, which might classify incorrectly but faster [6]. In the red wine dataset, since it's already imbalanced, if the classification result in the optimization process is wrong, there's not many data can re-direct the model back to the correct path, and therefore higher regularization is required. In contrast, in the heart disease problem, the data is more balanced, so faster calculation instead of absolute accuracy is preferred. For the training time, since the kernel function need to store the distances between the training points, if the training data size is larger, the training time will also grow up very fast, as show in Figure 10.

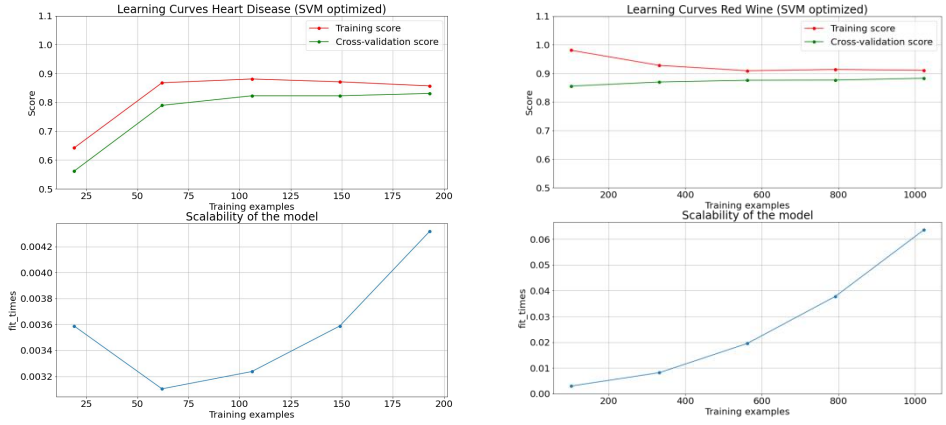

Figure 10—Learning curve and scalability plot of the optimized SVM on heart disease and red wine data set

3 CONCLUSION

To compare the above machine learning models, among many metrics, I used f1_score since it considers both the recall and precision. The f1_scores for every model in the previous chapter are presented in table 6.

Table 6 — f1_score for DT, ADA, NN, KNN, SVM

	DT	**ADA**	**NN**	**KNN**	**SVM**
Heart Disease	0.852	0.853	0.884	0.855	0.851
Red Wine	0.911	0.936	0.854	0.873	0.826

It is clear that in heart disease problem, the difference in the five models are not significant, which is reasonable. Since this is just a balanced binary classification problem with small amount of data, these five basic machine learning method can all reach acceptable performance. What's more important is the model behavior during the training process. For DT and ADA, without setting the a reasonable boundary for the parameters, overfitting happens very quickly, especially when the data is small. NN, ADA and SVM might have longer training time if the data size is bigger as discussed in the previous section, so this is one factor to be considered before implementation.

In the red wine problem, we can see that DT, ADA and KNN have relatively higher score. But this is due to the data imbalance, and in fact DT, ADA and KNN are obviously overfitted as discussed in previous section. KNN model with k = 1 indicates the machine learning model can be entirely mislead by the abnormal data. In other words, although these three models have good value in terms of metric, they are not applicable in the real world scenario. For the imbalanced data, proper resampling or some other techniques [7] should be used before applying the machine learning method.

In the future, if I have a supervised learning problem, I will do proper data preprocessing first, and see if the imbalanced condition can be avoided. For the preliminary trial, I would try DT and KNN first, since they are relatively simple and fast, and the model behavior can indicate a lot of information that is not obvious in the beginning. Then I can try the ADA, SVM to see if more

complicated machine learning method could get even better results. Finally, we can consider neural network based method, like the NN method used in this assignment, or even more advanced technique like convoluted neural network (CNN). The most important thing in solving machine learning problem is not how complicated the model structure is. It's correctly understanding the meaning behind the data and metrics, and correctly design the next step.

4 REFERENCES

1. Heart Disease UCI: https://www.kaggle.com/ronitf/heart-disease-uci
2. Red Wine Quality: https://www.kaggle.com/uciml/red-wine-quality-cortez-et-al-2009
3. Evaluation Metrics Machine Learning: https://www.analyticsvidhya.com/blog/2019/08/11-important-model-evaluation-error-metrics/
4. Minimal Cost-Complexity Pruning : https://scikit-learn.org/stable/modules/tree.html#minimal-cost-complexity-pruning
5. https://machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/
6. https://medium.com/machine-learning-101/chapter-2-svm-support-vector-machine-theory-f0812effc72
7. 5 Techniques to Handle Imbalanced Data For a Classification Problem: https://www.analyticsvidhya.com/blog/2021/06/5-techniques-to-handle-imbalanced-data-for-a-classification-problem/