CS7641 Machine Learning: Assignment 1 Supervised Learning

Judy Jungeun Cha
jcha64@gatech.edu

1 INTRODUCTION

As I was a credit supervisor in risk management at my work, I have been always interested in how machine learning algorithms can be implemented in my area. As I was looking for insurance coverage for our customer's account receivable, our customer's credit is the key to determine insurance premium. However, the pricing method from insurance company was not revealed like Blackbox. With my background, the default credit card client's data set from UCI caught my eyes for analysis. My second data is wine data. As a wine lover, if we can tell which wine is white or red with only features information without tasting, that sounds very interesting to me. I want to explore how much the combination of alcohol, sugar level and pH can tell the correct answer which one is white or red wine.

1.1 Data selection

Table 1 — Logistic Regression Results

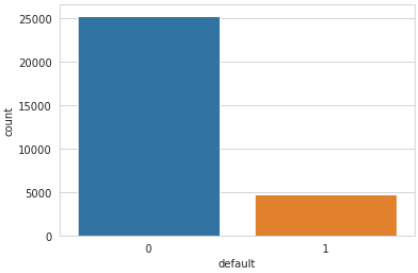
Name	# Of samples	Features / Class
Wine Quality (Wine)	6,497	Numerical (3): 'alcohol', 'sugar', 'pH'
Default of credit card clients (DOC)	30,000	Numerical (2): 'LIMIT_BAL', 'AGE' Categorical (9): 'EDUCATION', 'MARRIAGE', 'SEX', 'PAY_1' – 'Pay_6'

These datasets are originally from UCI. For the first dataset, I selected three numerical variables to determine white or red wine with 6,497 sample data. For DOC, I used 11 variables including 2 numerical and 9 categorical variables with 30,000 samples. Both datasets employed a binary variable, default payment (**Yes = 1**, No = 0) and wine type (**White wine = 1**, Red wine = 0) as response variables.

1.2 Preprocessing

We need to preprocess the data to better fit for our analysis. Some algorithms are sensitive to different scale and affects the model performance. Most of kernel-

based algorithms end towards the majority class. Also, gradient descending algorithms including neural network is sensitive to scale of data. Our DOC data shows imbalance data. We need a standardization of imbalance data.



1.2.1 *Cleaning*

Wine: 6,497 non-null with float64 (No need cleaning)

DOC: Fixed class name (PAY_0 -> PAY_1), Combining classification (education == 5 &6 -> 4, marriage==0 -> 3, all negative value of PAY-> 0)

1.2.2 *Check data set*

Based on describe () method, each variable has a different scale. Both datasets are imbalanced. The scales of variable can be standardized using StandardScarler. I checked with logistic regression to see more detail dataset characteristics.

Table 2 — Logistic Regression Results

Dataset	Train_scaled	Test_scaled	Model Parameter (Coefficient)
Wine	0.78083	0.77769	Sugar (1.67), Alcohol (0.51), pH (-0.68)
DOC	0.90775	0.90666	PAY_1(1.28), PAY_6 (0.44) … Education (-0.14)

Based on low scores of both training and testing scores for wine data, the logistic regression model seems underfitting with 77% accuracy. But we can get some idea about our variables that the higher Alcohol and/or higher Sugar level is likely to indicate white wine and the higher pH could be red wine. For DOC, the latest repayment status (Sep 2005) shows the highest coefficient of 1.28 with default payment status of credit card and logistic regression performs well with relatively high accuracy (90%).

The normalized data is used for neural network, boosting and SVM algorithms. The data is normalized using np.mean and np.std.

1.2.3 *Tools used*

To accomplish all of 5 algorithms analysis, I used scikit-learn (Sklearn) library in Python because it is most useful library for machine learning in Python and built upon familiar libraries like NumPy, pandas and Matplotlib.

Cross validation using 10 folds is used as a default, except for SVM algorithm of DOC data set. Since DOC has 11 features and not with linearly separable characteristics, 10-fold cross validation was not working in my local environment. So, I tried 5-fold cross validation instead of 10-fold.

2 SIMPLE DECISION TREE

Decision trees is relatively easy to understand with intuitive representation. It provides more detail information compared to logistic regression model.

2.1 Decision tree with Kfold (StratifiedKFold)

Cross validation is performed with KFold cross validation with StratifiedKFold. With KFold cross validation we can use 80-90% of data in training. Both data shows similar average test scores in between 5-fold and 10-fold as 0.84 (wine) and 0.91(DOC), respectively.

2.2 Pruning

Table 3 — Decision Tree scores with pruning

	No-prune		**Prune (Max-depth=5)**	
Dataset	Training	Testing	Training	Testing
Wine	0.9969	0.8584	0.8672	0.8607
DOC	0.9935	0.8768	0.9135	0.9133

For Wine data, training score is very high as 0.9969. It seems decision tree model can explain most of training data correctly through training process, while testing score is a little lower as 0.8584. We can say this model is overfitting. With pruning (limiting 5 max-depth), training score is lower but testing score remains the same level (0.8607), partly pruning could remove the overfitting issues. In other words, using pruning lowers the complexity of model and it leads to

reduce overfitting. For DOC data, it shows the same effects about overfitting that pruning improves the model with lower training score and higher testing score of 0.9133.

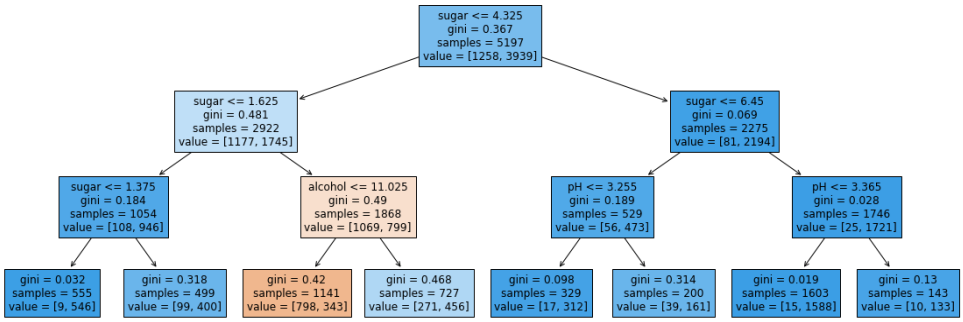


Figure 1 — [wine] Decision tree with pruning (max_death=3)

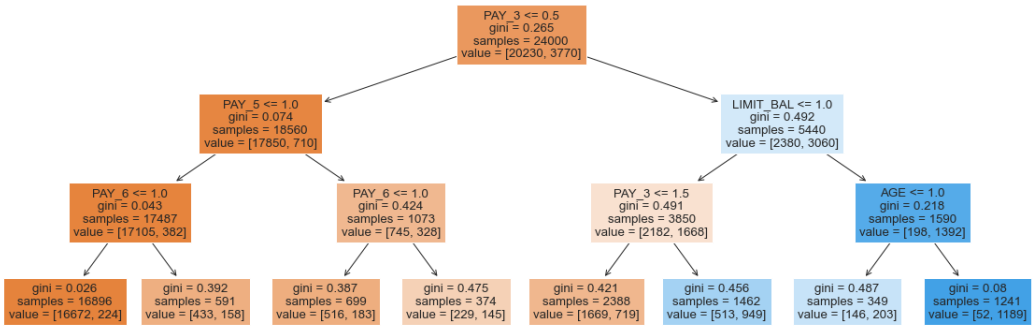


Figure 2 — [DOC] Decision tree with pruning (max_death=3)

In the wine decision tree (3 max_depth), only one orange leaf node (1.625 <= sugar <= 4.325 and alcohol <= 11.025) indicates red wine. The rest of leaf node falls into white wine. For DOC, PAY_3 is the root node. The repayment status in July 2005 shows more than 15 days delay is important factor the client becomes a credit card defaulter.

In the tree model, I used Gini impurity ('gini') which is a default decision factor to build the tree. Root node is the most important variable at the first split in the tree. For wine data, the gini of 0.367 at the root node is calculated like following: 1 – ((1258/5197)2 + (3939/5197)2) = 0.367. Note that 5,197 is # of training set (80% of total 6,497). Information gain is the difference of gini between parent and child nodes. The decision tree algorithm chooses the split value for each node to

maximize the information gain and builds the model accordingly. In this way, we don't need preprocess the different scale of data because scaling is not necessary to calculate information gain in building a decision tree.

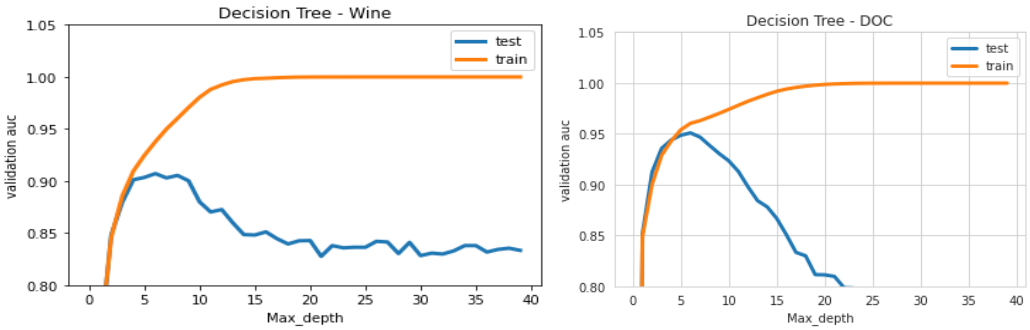


Figure 3 — Decision tree validation curve

Based on validation curve, both data sets show around 5 max_depth performs the best with the highest accuracy of testing data. The pruning with 5 max_depth is the most effective for decision tree algorithms of both wine and DOC.

3 NEURAL NETWORKS

Neural networks perform better for unstructured data like image through feed forward network or multilayer perceptron. Since two data sets of wine and DOC are both structure data, I expect less effective on wine and DOC data sets.

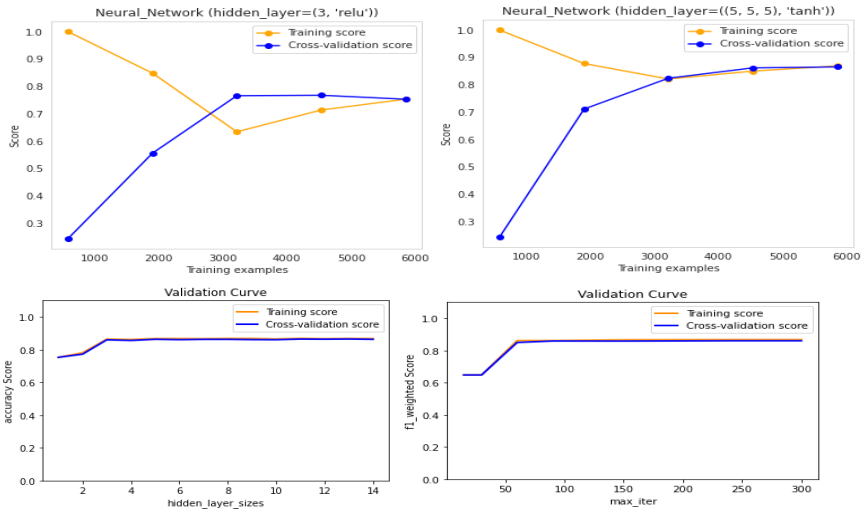


Figure 4 — [Wine] Neural network learning curve and validation curve

For wine data, I used networks of nodes with the different number of hidden layers with ‘tanh’ and ‘relu’ activation function including ((3), 'relu'), ((3), 'tanh'), ((3, 3), 'tanh'), ((5, 5, 5), 'tanh'). (5,5,5), ‘tanh’ means 5 of neurons and three hidden layer size with tanh activation function. Tanh is hyperbolic tangent, another type of ‘S’ shape function other than sigmoid function. Relu is popular because it is simple and fast.

Based on the results of wine data, the hidden layer of 3 with ‘tanh’ activation functions ((5,5,5), ‘tanh’) seems to have the best accuracy among tries for wine data. In the learning curve of hidden layer of 3 with ‘tanh’, training score and cross validation score have met around 3,100 training examples, and both of training and cross validation accuracy continue to increase slightly. This is the point that overfitting (training score > cross validation score) issues are resolved. In the validation curve, it shows accuracy score reaches the max at the hidden layer size of 3. After 3, it says the same accuracy. However, the hidden layer of 1 with ‘relu’ shows that after 3,100 training examples as cross validation score is higher than training score. The model can be underfitting from 3,100.

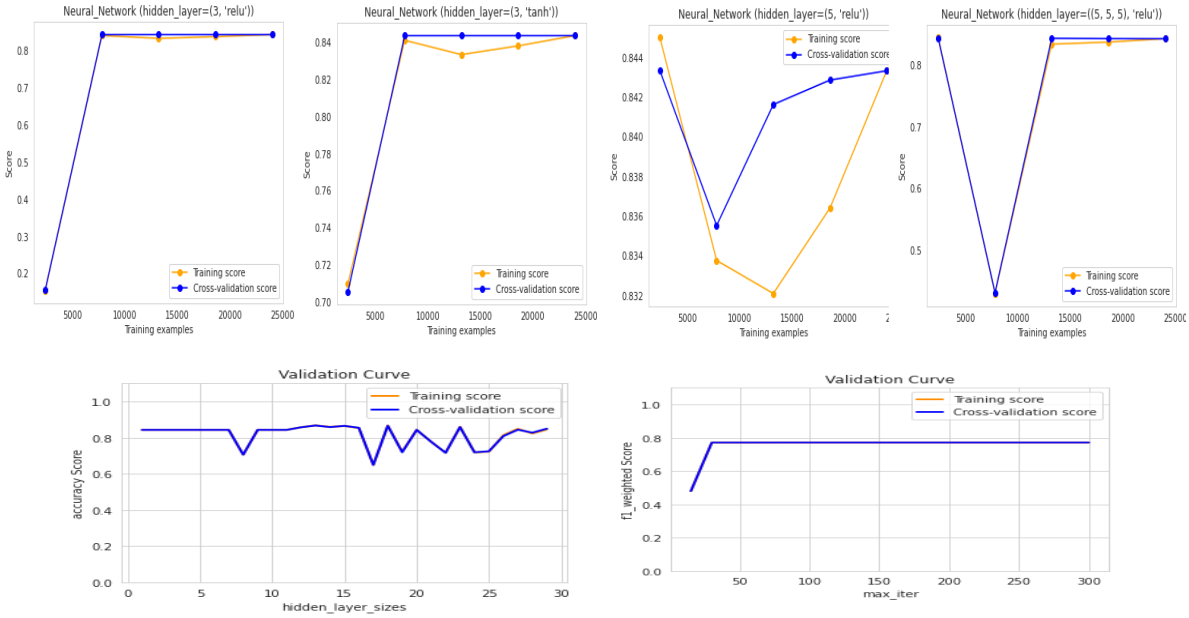


Figure 5— [DOC] Neural network learning curve and validation curve

For DOC dataset, the hidden layer of 1 with 'relu' activation functions shows the best accuracy score in my tries: ((3), 'relu'), ((3), 'tanh'), ((5), 'relu'), ((5, 5, 5), 'relu'). ((3), 'tanh') has very similar performance with ((3), 'relu').

Validation curve is based on the hidden layer of 1 with 'relu' activation function. It shows fluctuation over the hidden layer changes, but it is consistent with around 84% accuracy except for the hidden layer at some points including 8, 17, 19, 22, 24 and 25 hidden layer sizes. The best accuracy is already reached from the low hidden layer of sizes $(1 - 7)$. The increase of complexity with hidden layers seems not really help the model accuracy after the hidden layer size of 1.

4 BOOSTING

4.1 Random Forest

Ensemble learning performs great for structure data. Since wine and DOC datasets are structure data are structure data, I expected to have the good accuracy results from boosting algorithm.

First, I tried random forest for wine data. Random forest creates samples with bootstrap method and randomly select features to make a tree. For wine data, the train score is 0.9973 and test score is 0.8905 which seems a little overfitting to training score, but it is still good performance. The feature importance of random forest is [Alcohol: 0.2316, Sugar: 0.5003, pH: 0.2679]. Compared to feature importance of single decision tree [Alcohol: 0.1234, Sugar: 0.8686, pH: 0.0079], I can see pruning effects in random forest algorithm that it does not focus on one feature, but it gives more chances to the other features. It helps to decrease overfitting and increase generalization quality.

4.2 **Boostings**

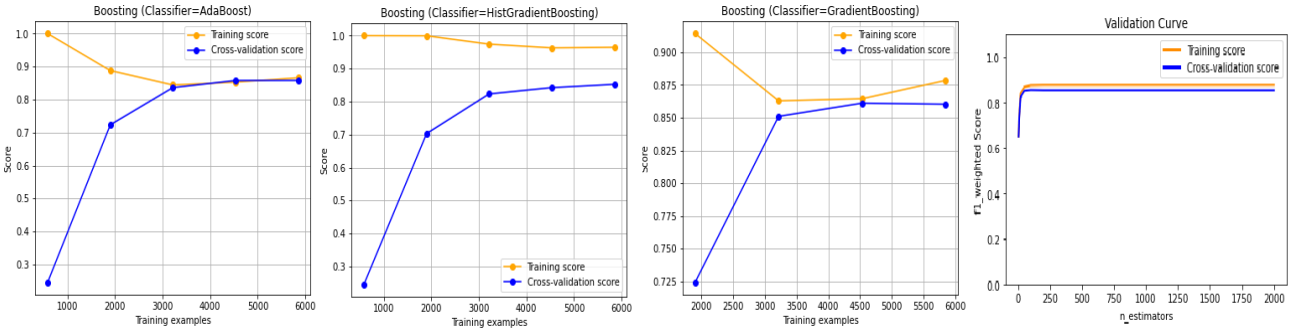


Figure 6— [Wine] Boosting learning curve and validation curve

AdaBoost trains the data with weak learners and give more weight next time to previous incorrect data to improve accuracy and reduce overfitting problems. Training accuracy is 0.8660 and testing accuracy is 0.8615 which means there is almost no overfitting in AdaBoost.

Gradient boosting uses 1,000 low depth trees (default value is 3 depth) to minimize loss function. It is effective to reduce overfitting issues. I noticed that gradient boosting takes longer than random forest because it adds decision tree sequentially. We can control complexity with learning rate, but high learning rate might lead overfitting model.

Histogram-based gradient boosting divided train data into 256 sections, and it is very fast to find the optimal split. Testing accuracy score is 0.8653 which is the highest accuracy performance in other 2 boosting algorithms (AdaBoost and Gradient boosting), but it is lower than random forest accuracy of 0.8905. But it is still higher than the testing accuracy of decision tree testing with no pruning (0.8584). We can say ensemble algorithm would perform better than single decision tree. 300 was used as max_itr to set the number of boosting.

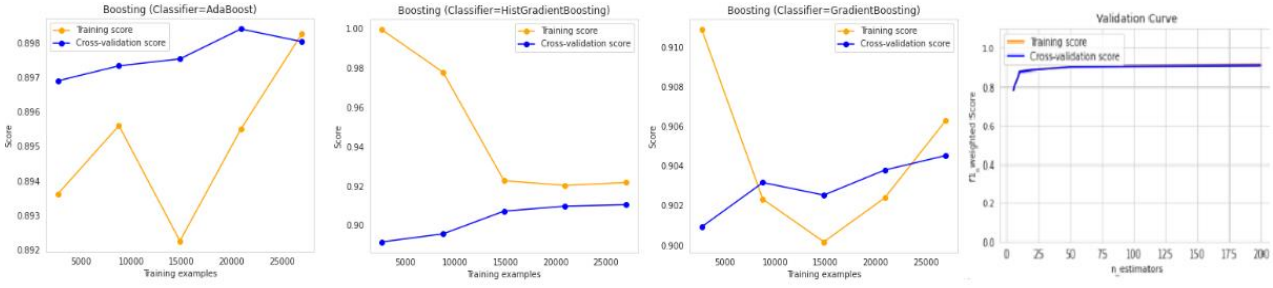


Figure 7— [DOC] Boosting learning curve and validation curve

For DOC, AdaBoost shows underfitting and Histogram-based gradient boosting performs the best among other boosting including random forest. It seems boosting algorithms perform better for DOC data than wine data. The learning curve of Histogram-based gradient boosting is stable with relatively high testing accuracy of 0.9076 as training examples increase. In validation curve of AdaBoost, the number of trees for ensemble algorithm reaches the max accuracy of 0.88 at around 10.

5 K-NEAREST NEIGHBORS

Table 4 — kNN validation curve

	Initial Try (k = 9)		**Best (1=< k =< 20)**				
Dataset	Training	Validation	k	Weights method	Distance	Training	Test
Wine	0.8835	0.8875	17	Distance	Euclidean	0.9971	0.8815
DOC	0.9039	0.8908	5	Uniform	Euclidean	0.9180	0.8941

First, data is split into train (80%) and testing (20%). And validation data set is created as 20% of training data for initial try. KNeighborsClassifier trains training data to find white wine and default payment. With k=9, the accuracy of validation dataset is 88.75% for wine data and 89.08% for DOC, respectively.

Next, I tried to find the best k to give better validation accuracy for each dataset within a reasonable rage (1=< k =< 20). The parameters for distance metrics have options of "Euclidean", "Manhattan" and "Minkowski". The weights parameters would be ‘uniform’ which is points are weighted equally and ‘distance’ weighting points by the inverse of their distance. The best option with distance metrics/weights to give better validation accuracy are euclidean/distance for wine data and euclidean/uniform for DOC.

With initial try (k=9), both wine and DOC data show good performance. Compared to k=9, with best k (wine: k=17, DOC: k=5), distance metrics and weights selection, we can see training score is improved a lot. Especially wine data (less sample) shows 99.7% accuracy of training score. However, both testing score with best k and metric combination has not improved much compared to the increase of training score. The tailored model might lead overfitting. Usually, the less complicated model with increasing k (k: 9 -> 17), it reduces overfitting but

based on below validation curve and learning curve for wine data, after k=9, k changes do not affect much on accuracy improvement anymore for wine data.

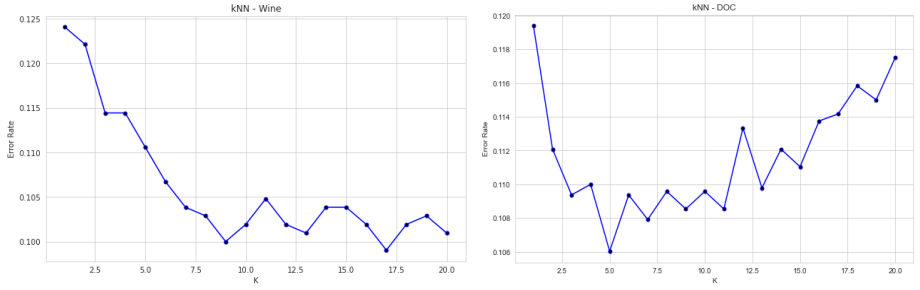


Figure 8— kNN Learning curve

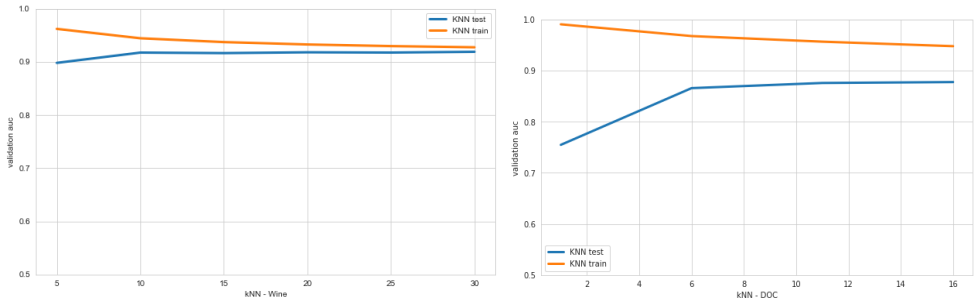


Figure 9— kNN validation curve

6 SUPPORT VECTOR MACHINES

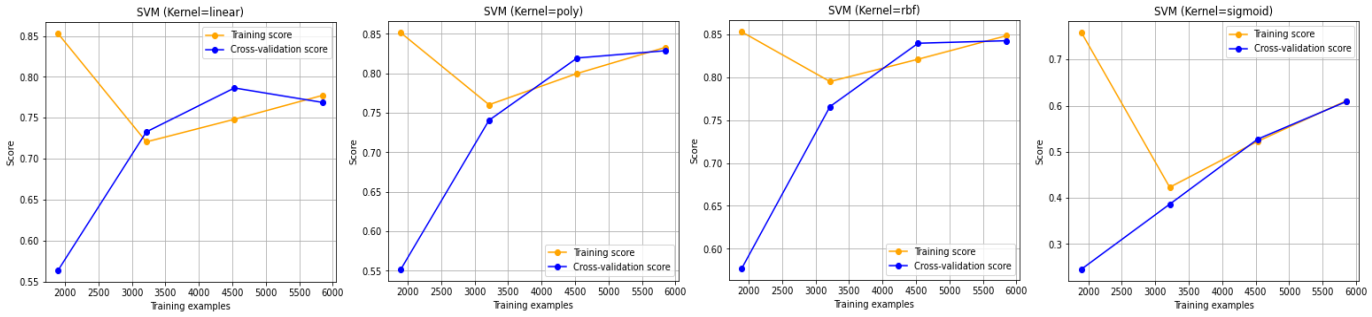


Figure 10— [wine] SVM learning curve

Based on performance of each kernel functions, rbf (Gaussian Radial Basis Function kernel) shows the best accuracy score. Rbf is one of the most popular kernel functions in SVM as it is similar with kNN algorithm. All of four kernel shows overfitting at the beginning as expected and cross validation score exceeds the training accuracy later. Especially rbf cross validation score has the highest score of 84% at 4,500 training examples compared to the other 3 kernel function.

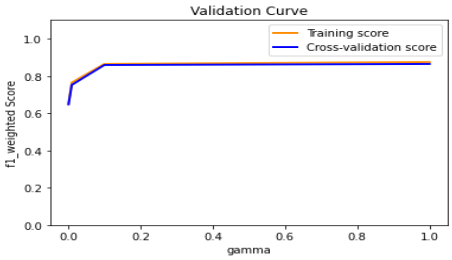


Figure 11 — [wine] SVM validation curve

C is a hyperparameter in SVM to control error. By grid search of SVM, the best C is 1 and the gamma is 0.1 above as shown in validation curve. Large C means the model accepts large errors.

Gamma is used in Gaussian RBF kernel. Gamma is needed to decide how curvature we want in our decision boundary. High Gamma means more curvature. The larger gamma means the closer other points. It affects the model. The high curvature boundary seems work well in wine data set. Both hyperparameters needs to be set before the training model.

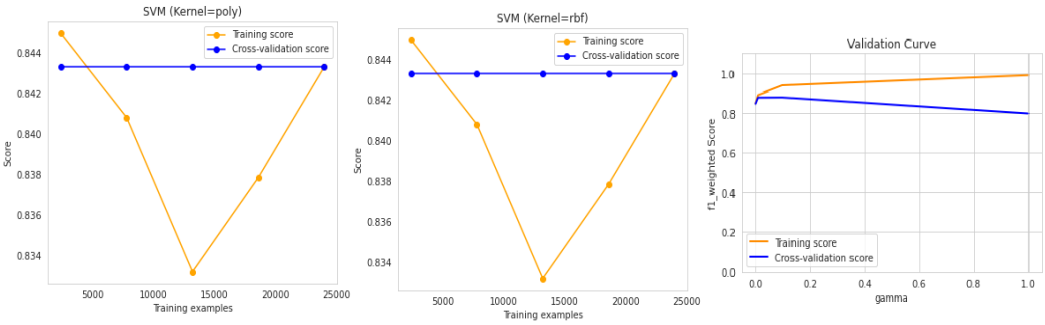


Figure 12 — [DOC] SVM learning and validation curve

SVM works well with a clear margin of separation. However, SVM becomes computationally expensive on large scale data sets. It does not work well when it is sensitive to the type of kernel as well. In the case of DOC dataset, DOC is relatively large 30,000 samples with 11 features compared to wine dataset (6,497 samples with 3 features). It takes me very long time to run under the same condition with wine data. I tried with lower folding condition of c=5 instead of c=10 and got Poly and rbf like above. However, in classification report, both Poly and rbf have zero value on '1' which is white wine classification. In addition, the linear and sigmoid activation function was not working even without folding condition. DOC with 11 features seems not separable well using SVM algorithm.

7 CONCLUSION

Table 5 — Algorithm's results summary

	Wine		DOC	
Algorithms	Test acc.	Conditions	Test acc.	Conditions
Decision tree	0.8584	no prune	0.8768	no prune
DT (Prune)	0.8607	max depth=5	0.9133	max depth=5
Neural Networks	0.8615	(5,5,5), tanh	0.8433	(3), relu
Boosting	0.8905	Random forest	0.9076	HistGradientBoosting
kNN	0.8815	n=17, distance, euclidean	0.8941	n=5, uniform, euclidean
SVC	0.8400	rbf	0.8450	rbf

All details including learning curves and validation curves have been discussed above for each 5 algorithms. We need to consider all details together for analysis, but I want to summarize each algorithm and compare their performance based on their testing accuracy results.

For wine data, random forest boosting algorithm works well the most among other algorithms. But SVC is not working well with wine data. 'rbf' was faster than any other kernel methods but it still came relative low accuracy and took me long time to run as well. Overall, the wine data set seems not working well to classify white or red wine using the supervised learning methods since it shows relative low testing accuracy results (0.84 – 0.89). In addition, each algorithm does not provide the distinct test results difference between them.

The decision tree with 5 max_depth pruning works the best for DOC data. Also, Histogram based boosting model works well with DOC. As I expected, neural networks and SVC do not perform well for DOC. DOC seems not to be separable with clear margin. Ensemble learning works well on both data set.