**CS7641: Supervised Learning-Assignment 1**

**GTID:** Yzhang3006  **Name:** Yiying Zhang

In this assignment, I use python based scikit-learn, pandas, and numpy libraries to explore different supervised learning algorithms, including Decision Tree, Neural networks, Boosting, Support Vector Machines, K-nearest neighbors (KNN). To understand concepts like learning curve, performance, and to play with different key parameters, I draw different with graphs of the accuracy score under different constraints against 2 interesting datasets with the help of matplotlib library.

**Description of Problems - Why They Are Interesting**

The first classification question I am interested in is Online Shoppers Purchasing Intention. The dataset: “Online Shoppers Purchasing Intention Dataset Data Set” is available in UCI Machine Learning Repository. [put in footer(<https://archive.ics.uci.edu/ml/datasets/Online+Shoppers+Purchasing+Intention+Dataset>)] Reason why this data is interesting is obvious: with millions of online business blooming every where nowadays, the business owner would like to land more sales when competing with thousands of competitor. By understanding the customer behavior, the business can make the product be more attractive to the user and earn more money. Who doesn’t like money? From business perspective, finding the pattern among the online shoppers is just like finding the key to a golden treasury. From a software develop engineer (or product manager) perspective, learning the pattern can help us modify/upgrade the functionality of the product I build.

The second interesting classification question is also in Business area: default of credit card clients. The Dataset is available in UCI Machine Learning Repository.

[put in footer(https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients#)]. The reason this topic is interesting is that credit card has become so common in our daily life. With an SSN it is so easy for an adult to get a credit card. However, credit card over-issuing to unqualified applicant has become a significant problem [citation: Butaru, F., Chen, Q., Clark, B., Das, S., Lo, A. W., & Siddique, A. (2016). Risk and risk management in the credit card industry. *Journal of Banking & Finance*, *72*, 218-239.]. It’s not only impacting the card-holder but also banks and even macro-economic: consumer finance confidence. The result of the machine learning this dataset can be used to increase the accuracy of the risk prediction using some easily access data, such as business financial statement, marriage status, age, customer transaction and repayment records etc.

**Description of Datasets – Data type, Attributes, Observations**

Both datasets are nontrivial: both dataset are larger than 10k record.

For **Dataset 1:** “Online Shoppers Purchasing Intention Dataset Data Set” Attribute Characteristics: Integer, Real. It has total 12,330 record, with 18 attributes, 10 numerical and 8 categorical attributes. The last attribute 'Revenue' is a binary attribute which is used as the class label(Y): ‘False’ refers to not landing a final sale, ‘True’ refer to make a sale. The rest attributes(X) include different types of pages visited by the visitor in that session and total time spent in each of these page categories as well as metrics measured by "Google Analytics" for each page in the e-commerce site, like Bounce Rate", "Exit Rate" and "Page Value".

For **Dataset 2**: “default of credit card clients Data Set” Attribute Characteristics: Integer. It has total 30,000 record, with 24 attributes. The last attribute “default payment next month” is a binary attribute which is used as the class label(Y): default payment (Yes = 1, No = 0). The rest attributes(X) include Amount of the given credit, Gender, Education, Marital Status, Age, History of past payment, Amount of bill statement, Amount of previous payment etc.

**Description of Data Preprocessing - Cleaning, Encoding Categorical Data, Fill NA etc.**

Both datasets, the data have no ‘NA’ file. First, rename dataset2 file name from “default of credit card clients.xls” to “default\_of\_credit\_card\_clients.xls”. For data cleaning, for encoding the categorical data to numerical data, the method is replacing with a dictionary mapping. For example, dataset 1 attribute 'VisitorType' set[[1]](#endnote-1)"Returning\_Visitor" to 1, "Other" to 2, "New\_Visitor" to 0; attribute “Month”, set “Jan”~”Dec” to “1”~”12”. Since Month is chronological, I choose mapping to define the number from 1 to 12 instead of One Hot Encoding, which might be random relationship on the mapping.

Dataset 2 has 2 headers, so ignore 1 row by using the pandas “skiprows” parameter in “read\_excel” method.

**Evaluation Mothed Used in Machine Learning Algorithms and Why**

**Learning curve(using accuracy score)** is a good tool to diagnose whether an algorithm is suffering from **bias or variance** problem or a bit of both. Learning curve can visualize the performance by plotting a graph as a function of training size vs accuracy score. Usually when training error and cross validation(test) error is converge to a relatively large error, it is when we consider there is bias. When there is a big gap between training error and cross validation error, we consider there might be a high variance of the data.

**Confusion matrix** is a straightforward, descriptive table used to describe the performance of the classification problem by counting the “True” or “False” in terms of actual vs predicted. Since the 2 problem here are Boolean classification problem, so confusion matric can be well used here.

**Decision Tree** - Confusion Matrix, Learning and complexity curve, Model Selection, Cross Validaiton. Explain accuracy, variance, bias and analysis of these plots.

From the standard sklearn library instantiate a classifier using DecisionTreeClassifier method, the splitting attribute is based on GINI index default in the library.

Use the accuracy score against the training size, the training size is from 1% of the entire dataset to 99%, we can explore the learning curve as below figure1. Here are some findings:

1. For both datasets, the training accuracy score are all 100% no matter which what dataset training size is. The reason is more about the dataset itself: 1) first the decision tree just works very well in training dataset. For example, dataset 1 has binary outcome with most of the attribute is a binary value, this makes it naturally fits a decision tree model. One or two factors in the data can play a very heavy role in the final result. 2) the dataset is “non-trivial”, even with 1% as training, there is still over 100 record. It is enough for the classifier to predict.
2. dataset 1 the best accuracy score is about **85%,** when training size between 1% to 95%. However for dataset 2 the accuracy score is only about **73%,** when training size between 1% to 95%. The accuracy score of Dataset2 is worse than dataset1, the reason might be dataset2 is more complex to predict because it has more attribute (24 vs 18 in dataset1) the relationship of the variable is not that strong with the result. Even 1% training size, the test accuracy is as good as 85%, and 73%. The common practice as of **75%** is good enough for model.
3. Overfitting occurs around training size = **93%** of the entire dataset because the test accuracy starts to go down with the size increase.
4. The gap between train dataset and test data set can reveal the variance, in below chart, dataset2 has higher variance than dataset1.

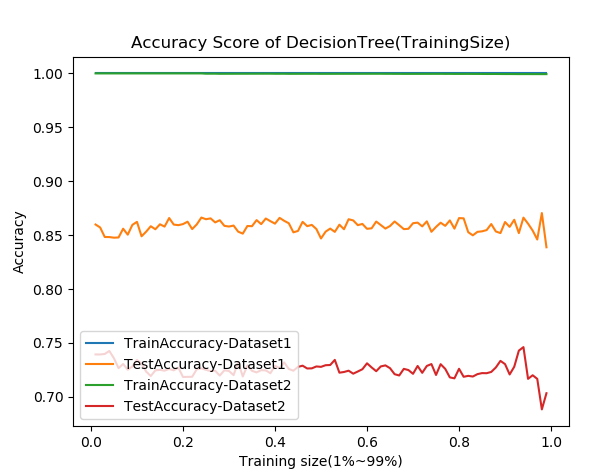
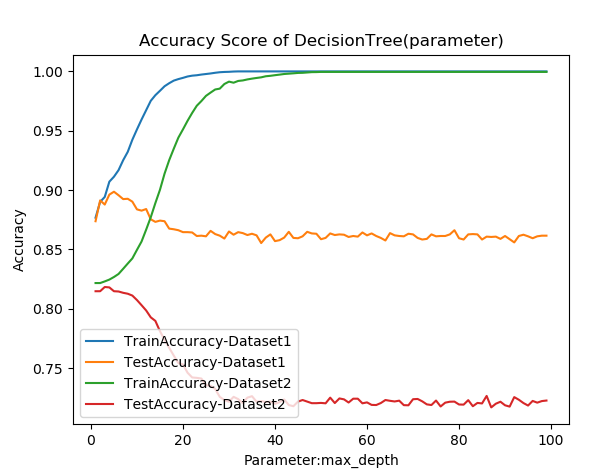


Figure 1: Learning Curve

Use the accuracy score against the tree depth, the training size is used the common practice: 75%. we can explore the accuracy score against the tree depth. As below figure2, here are some findings:

  
Figure 2: Max\_Depth

1. The **overfitting** happened around max\_depth = 5 for dataset1. While max\_depth = 10 for dataset2. Since the max\_depth increase, the accuracy score of test data is dropping. When the depth is small (like than 5), the accuracy can improve for testing dataset.
2. Both datasets are doing pretty good even the depth is 1. This align with the conclusion from figure1, which mean the data is very well performed under decision tree. So the optimal max\_depth for both set can set to **5**.
3. dataset2 **has higher variance** than dataset1 by looking at the huge gap between train/test accuracy score.

**Neural Network**

From the standard sklearn library instantiate a classifier using MLPClassifier method, activation function use logistic, solver use sigmoid.

Use the accuracy score against the training size, the training size is from 1% of the entire dataset to 99%, we can explore the learning curve as below figure3. Here are some findings:

1. Dataset 1 the best accuracy score is about **85%,** when training size between 1% to 95%. However, for dataset 2 the accuracy score is only about **77%,** when training size between 1% to 80%.
2. Overfitting occurs around training size = **95%** for dataset 1, but dataset2 has overfitting when training size larger than **85%** because the test accuracy starts to go down with the size increase. So the best training size for neural network is still 75%
3. There is no gap between train dataset and test data set so neural network can overcome high variance problem.

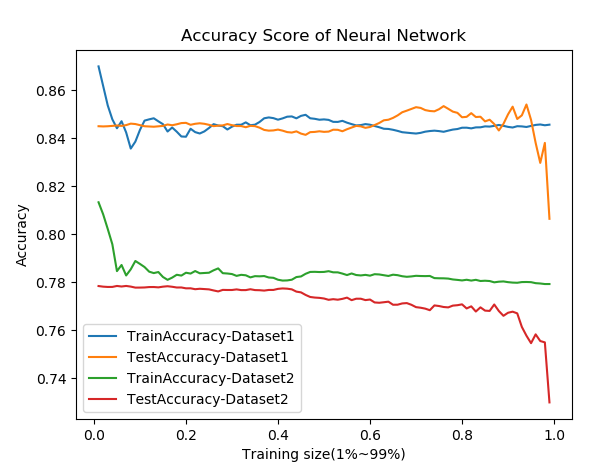


Figure 3: Paramter-Learning Curve

Use the accuracy score against the tree depth, the training size is used the common practice: 75%. we can explore the accuracy score against the hidden layer. As below figure 4, here are some findings:

1. Increasing the number of hidden layers may or may not improve the accuracy. The accuracy depends on the complexity of the problem. In here dataset2, add more hidden layer didn’t make the accuracy change.
2. looking at dataset1, Increasing the number of hidden layers much more than the enough layers (like here the sufficient layer is 1) will cause accuracy in the test set to decrease. This will make your network over-adapt to the training set, it will learn the training data, but it cannot be extended to new invisible data.

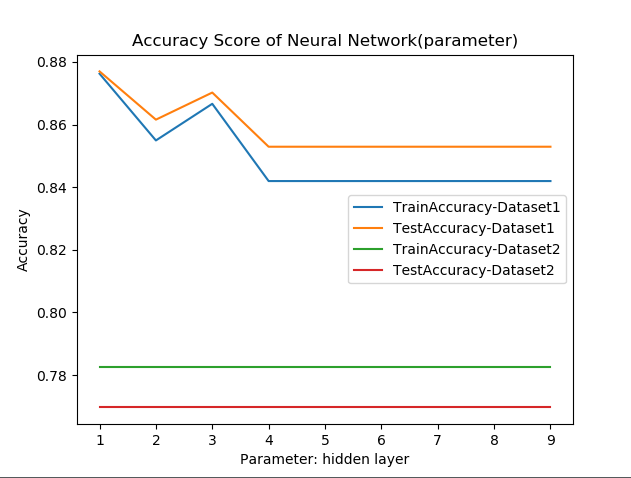


Figure 4: Parameter--hidden layer

**Boosting**

Boosting ensemble, a series of classifiers, each one compensating for the other’s weakness. From the standard sklearn library instantiate a classifier using AdaBoostClassifier.

Use the accuracy score against the training size, the training size is from 1% of the entire dataset to 99%, we can explore the learning curve as below figure5. Here are some findings:

1. Dataset 1 the best accuracy score is about **89%,** when training size between 30% to 92%. However, for dataset 2 the accuracy score is only about **82.5%,** when training size between 20% to 80%. **(This is so far the best! Boosting did improve weak learner.)** Boosting can help us start with low variance classifiers and continue to improve those until we got low bias.
2. Overfitting occurs around training size = **93%** for dataset 1 because the test accuracy starts to go down with the size increase. So the common practice of choosing 75% as of training dataset also works here.
3. There is no gap between train dataset and test data set so neural network can overcome high variance problem. the bias problem is also relative lower than other method.

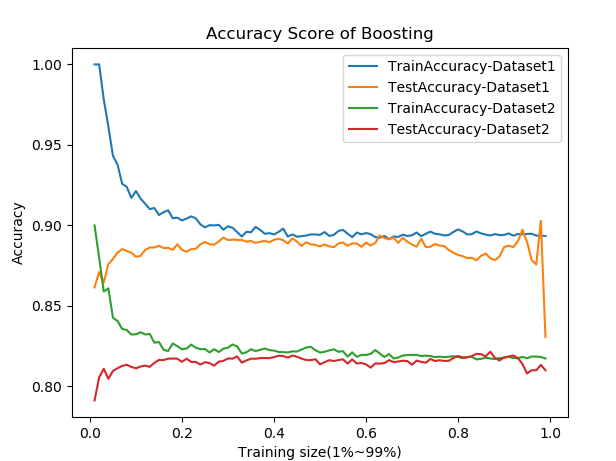


Figure 5: learning curve

Use the accuracy score against the tree depth, the training size is used the common practice: 75%. we can explore the accuracy score against the n\_estimators. As below figure 6, here are some findings:

1. Dataset 1 the best accuracy score is about **84%,** when estimators >10. However, for dataset 2 the accuracy score is only about **81%,** no matter what estimators is. **Better than decision tree. But the default n-estimator is good enough.**
2. When the n\_estimators = 1, this is same as decision tree. So the dataset1 accuracy rate is 85% which is the same as decision tree.
3. looking at dataset 1&2, Increasing the number of estimator may or may not improve the performance. Because the base classifier we used here is decision tree. For Dataset2, Decision tree doesn’t good at prediction. So boosting cannot really improve the performance for dataset2 when repeating a bad classifier.

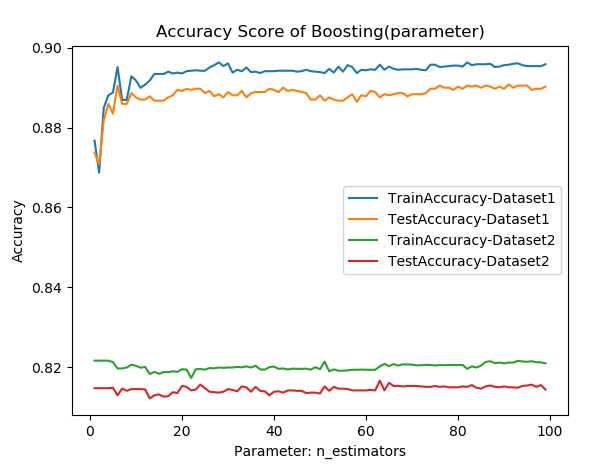


Figure 6: Parameter—n\_ estimators

**SWM**

Support Vector Machine is a frontier which best segregates the two classes. From the standard sklearn library instantiate a classifier using SVC

Use the accuracy score against the training size, the training size is from 1% of the entire dataset to 99%, we can explore the learning curve as below figure 7. Here are some findings:

1. Dataset 1 the best accuracy score is about **89%,** when training size between 10% to 90%. However, for dataset 2 the accuracy score is only about **81%,** when training size between 18% to 90%.
2. Overfitting occurs around training size = **90%** for dataset 1 as the test accuracy starts to go down with the size increase. Common practice of choosing 75% of training dataset also works here.
3. A little variance problem since there is gap between training and testing accuracy score. the bias problem is also relative lower than other method.

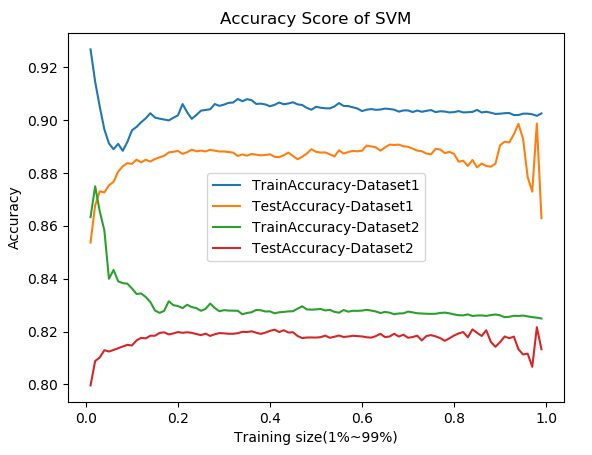


Figure 7: learning curve

Use the accuracy score against the kernel: 'rbf', 'linear', 'poly', 'sigmoid', the training size is used the common practice: 75%. we can explore the accuracy score against the kernel in the scatter plot. As below figure 8, here is some finding:

1. For Dataset 1, the best kernel function is the default **“rbf**”( the best accuracy score is about **89%**), the worst is the “sigmoid”( the worst accuracy score is about **84%** ), similar for dataset2: the best kernel function is the default “rbf”( the best accuracy score is about **82%** the worst is the “sigmoid”( the worst accuracy score is about **70%** ),

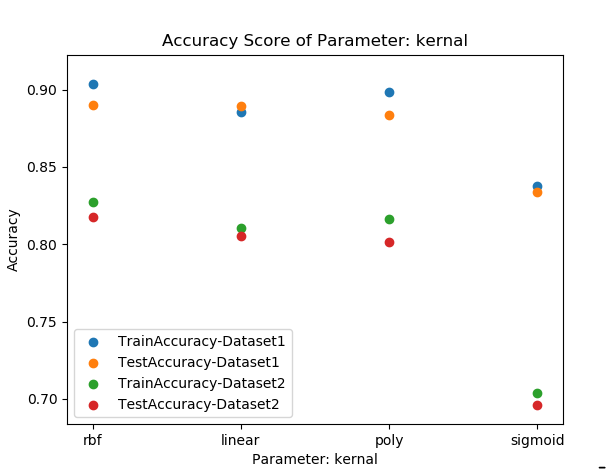


Figure 8: Parameter—kernel function

**KNN**

KNN algorithm is a simple, easy-to-implement mthod is a frontier which best segregates the two classes. From the standard sklearn library instantiate a classifier using SVC

Use the accuracy score against the training size, the training size is from 1% of the entire dataset to 99%, we can explore the learning curve as below figure 9. Here are some findings:

1. Dataset 1 the best accuracy score is about **83%,** when training size between 15% to 90%. However, for dataset 2 the accuracy score is only about **72.5%,** when training size between 18% to 90%.  **(Not doing good comparing to boosting)**
2. Overfitting occurs around training size = **95%** for dataset 1 because the test accuracy starts to go down with the size increase. So the common practice of choosing 75% as of training dataset also works here.
3. **High variance problem** occurs since the gap between training and testing accuracy score is the biggest among all other algorithm, especially for dataset2. the bias problem is also relative higher than other method.

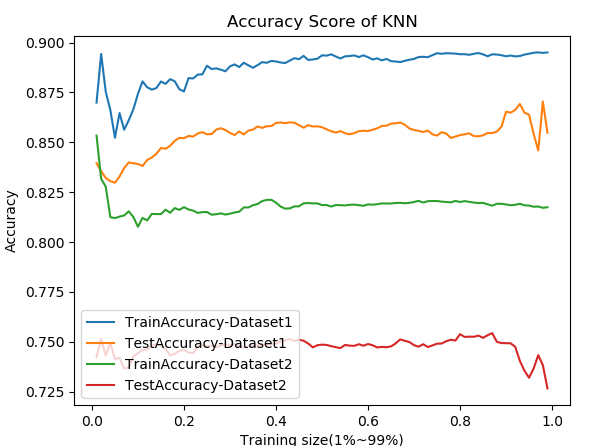


Figure 9: Learning curve (K = 5, default)

Use the accuracy score against the “k”(k means get average from the nearest k points), the training size is used the common practice: 75%. we can explore the accuracy score against the k. As below figure 10, here is some finding:

1. When k =1, it is overfitting, because the test accuracy score is very bad but the train set is 100%, when k increases, the Train accuracy will decrease but the test accuracy increase. When k around k>20, the accuracy rate stop increasing.
2. Dataset 1 the best accuracy score is about **86%,** when k >20. Dataset 2 the accuracy score is only about **77%,** when k >20.

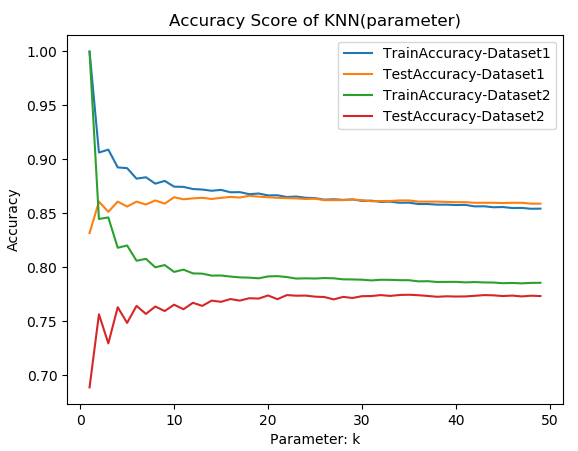


Figure 10: Parameter—k

10. KNN - Confusion Matrix, Learning and complexity curve, Model Selection, Cross Validaiton. Explain accuracy, variance, bias and analysis of these plots.

11. GridSerach CV improvement on all these aglorithms.

12. Need Learning curve and model complexity curves for every model and both datasets

13. Need analysis and discussion of cross validation

14. Need Comparison (accuracy and runtime) charts for all algorithms. Which performs better? Why? Performance metrics needed.

1. [↑](#endnote-ref-1)