Supervised Learning

Ting-Yu Kang 903443289

Datasets and Problem Description

1. Letter Recognition Dataset (UCI Machine Learning Repository):
   This dataset is generated from black-and-write rectangular images of 26 English characters with 20 different fonts. The images are converted into 16 numerical attributes for machine learning. The goal is to identify the character of each image.
   https://archive.ics.uci.edu/ml/datasets/Letter+recognition

2. Spambase (OpenML.org):
   This dataset is collected from real-world emails identified “spam” or “non-spam” by individuals. For each email, there are 57 attributes each describes one feature or characteristic of it. The purpose is to identify an email is Spam or not from these attributes.
   https://www.openml.org/d/44

3. Dataset Comparison

	Letter Recognition	Spambase
Instances	20000	4601
Attributes	16	57
Classes	26	2

4. Interesting points
   The two datasets are both classification problems. They are interesting for me, because these datasets are close-related with our real life. To illustrate, letter recognition technique can be used to identify texts, and the algorithm for Spambase can be implemented as a general-purpose spam email filter.
   Another reason I choose these datasets is that they are almost the opposite. The Letter Recognition dataset has **more classes** than Spambase, while Spambase has three-times **more attributes** than Letter Recognition dataset. I am looking forward to finding some interesting results given these differences by running on the five underlying supervised machine algorithms.

Tools

1. In this assignment, I implemented the underlying machine learning algorithms with **Scikit-learn** machine learning library and data structures from **Pandas** and **NumPy**. The programming language I use is **Python**.
2. The programs can be executed both on **MacOS** environment and on **Google Colaboratory**.

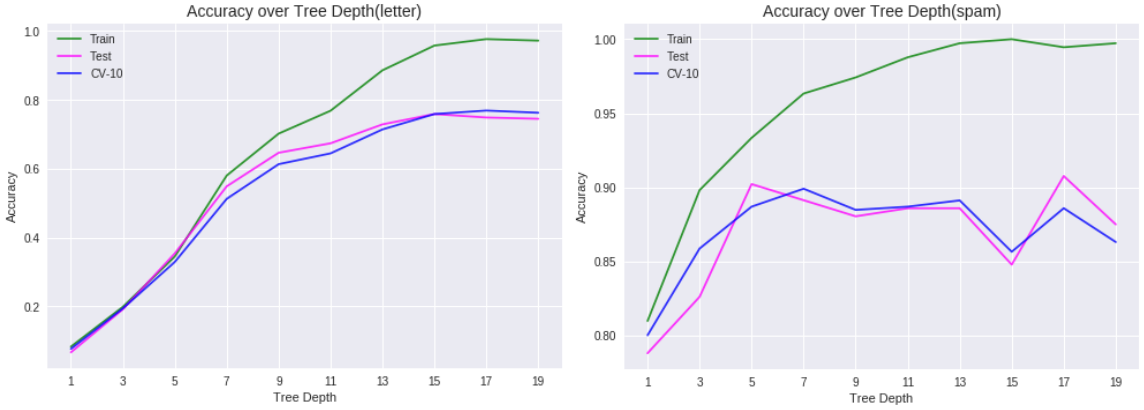
Methodology

1. The datasets are separated into **80%** training set and **20%** testing set.
2. For the five machine learning algorithms, first I look for the hyperparameters with the best accuracy by trying different values. To be more efficient without losing generosity, I only use **20%** of data (randomly sampled) to find the hyperparameters.
3. Next, I use the optimized hyperparameters to calculate accuracy over different data size. Finally, I will combine the results from these 5 algorithms to make comparison.
4. Before starting machine learning, I would **standardize** the training data and because it makes the prediction more accurate. It might because the effects of noise and outliers are decreased.
5. Although **10-fold cross-validation** takes about 10 times more time than traditional training and testing, it indeed help get more general and smoother accuracy line.
6. In the following graphs, the green line is **training set** accuracy; the pink line is **testing set** accuracy; and the blue line is **10-fold cross-validation** accuracy.

Decision Trees

Decision Tree is the most straightforward supervised machine learning algorithm. The intuition behind the algorithm is simple, yet also very powerful. In this algorithm, I am going to implement a **CART decision tree** algorithm and explore the effect of **pruning** and the difference of **entropy** and **Gini index**.

1. Accuracy over maximum tree depth (pruning)

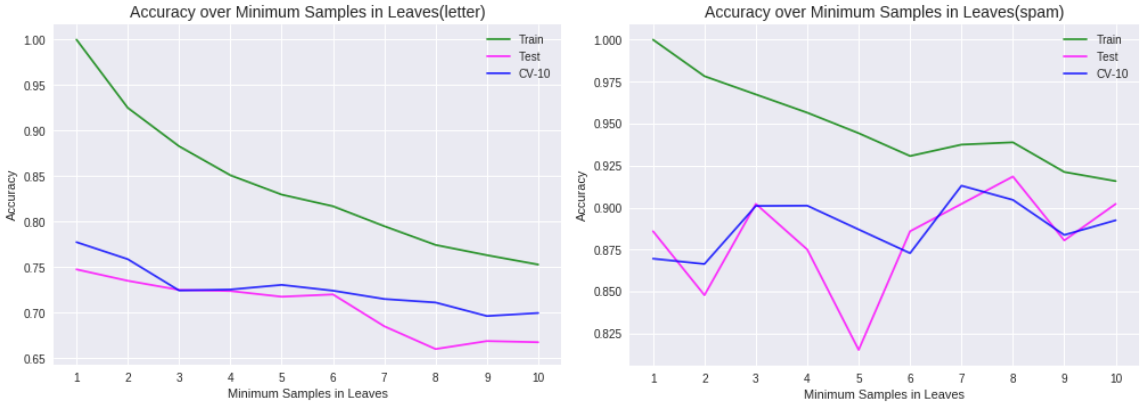


(a) For the Letter dataset, the cross-validation accuracy increases as the depth grows and gradually flattens when the depth is between 15 and 17. If the tree keeps growing, it will be at risk **overfitting**.

(b) For the Spam dataset, the cross-validation accuracy drops after depth reaches around 6 and 7. Beyond that point, we can clearly see that the model starts to **overfit** since the cross-validation accuracy drops while the training accuracy keeps arising.
The occurrence of overfitting may because there are a lot of attributes in the Spam dataset, so it is more possible for the model to learn from the **noise** instead of the **information**.

(c) Therefore, I choose depth **15** for letter recognition and **6** for spam email identification.

2. Accuracy over minimum samples in leaves (pruning)

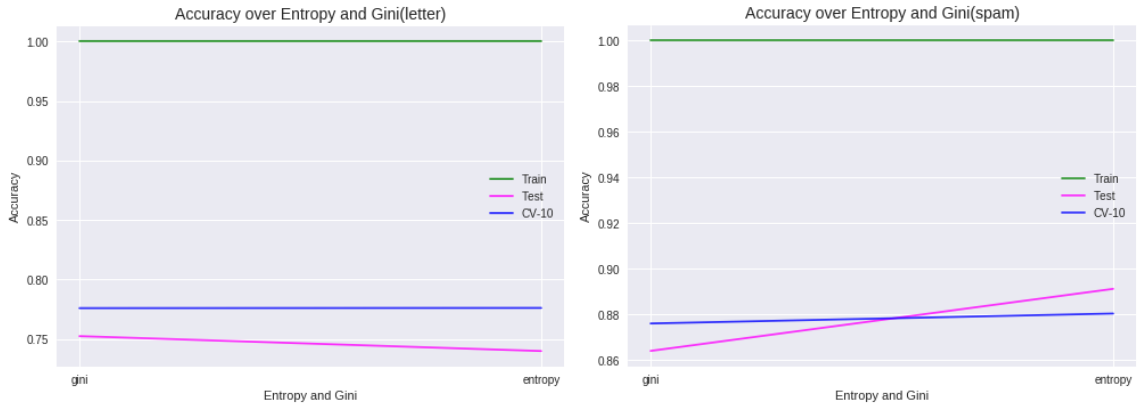


(a) For the Letter dataset, the training accuracy decreases as the minimum samples in leaves increases, showing that pruning in this case will cause **underfit**.

(b) For the Spam dataset, the maximum accuracy is between 7 and 8. I infer that because this dataset has more attributes (57) compared to the Letter dataset (16), the tree-split needs to be limited to prevent **overfit**.

(c) Therefore, I choose **1** for Letter Recognition and **7** for Spam identification.

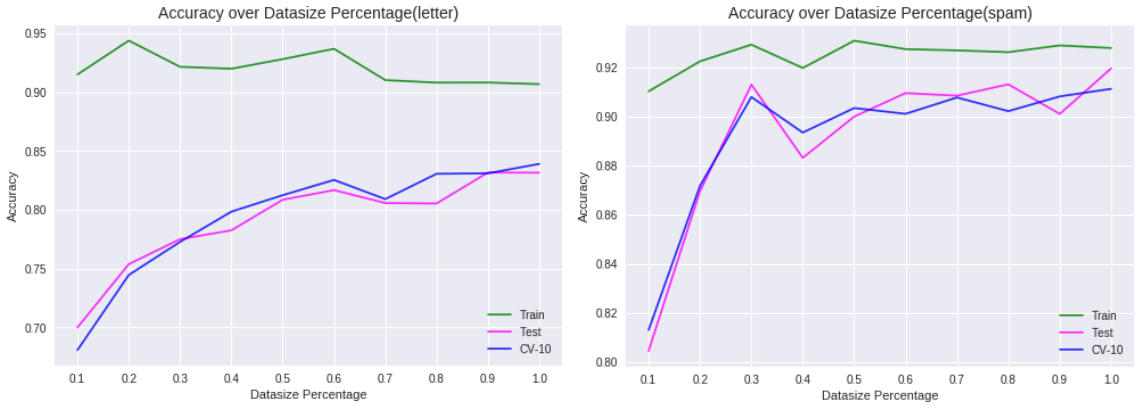
3. Accuracy over Gini and Entropy



(a) There are pretty much the same between Information Gain Entropy and Gini index according to the cross-validation accuracy. I decide to choose **Gini index** as the criterion.

4. Accuracy over data size percentage

   Combined the hyperparameters chosen from the previous experiment, the results are as follows:



   (a) For both datasets, the accuracy increases as the dataset size grows, showing that the hyper parameters are well-chosen.

   (b) For the Spam dataset, only 30% of dataset is enough for the decision tree to perform well.

Neural networks

The most important hyperparameters in neural network are **number of iterations** and **layers of perceptrons**. An iteration is a round of **feed forward** and **back propagation**. Unlimited iterations or overnumbered of layers will not only slow down the performance but also cause overfitting.

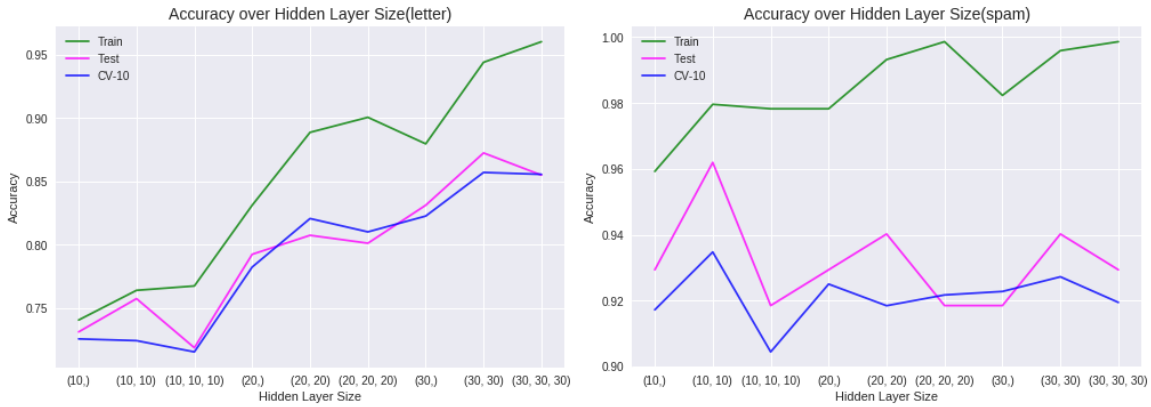
1. Accuracy over maximum Iteration

   (a) For the Letter dataset, the cross-validation accuracy increases significantly slower than the training error at iteration 200, showing that the model starts to **overfit** after this point.

   (b) For the Spam dataset, the cross-validation accuracy remains, and the testing set accuracy drop at iteration 200, showing that over 200 iteration, it will be at risk of **overfitting**.

   (c) Therefore, I choose **200** as the maximum iteration limit for both datasets.

2. Accuracy over hidden layer size



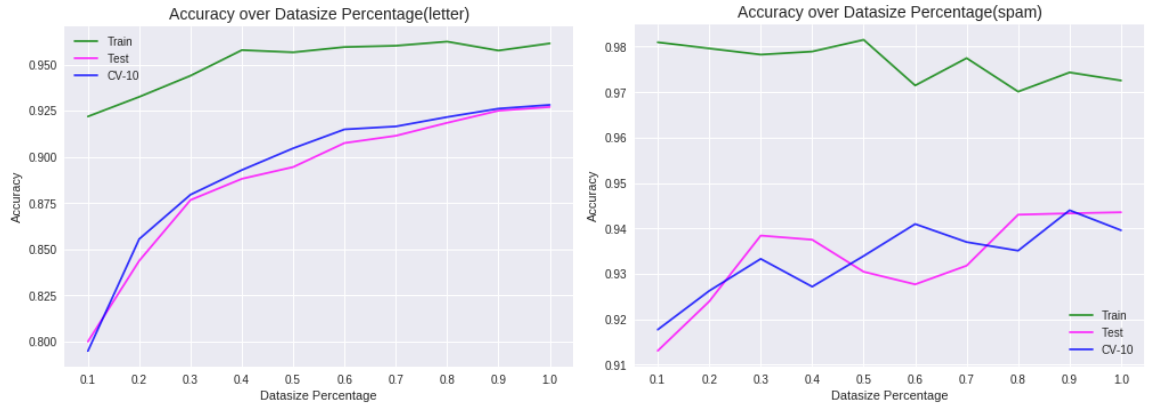
(a) For the Letter dataset, the accuracy increases as the number of layers and perceptrons increases. Layer (30, 30) maximizes the accuracy.

(b) For the Spam dataset, the cross-validation accuracy and testing set accuracy are maximized at (10, 10), after that, the testing set accuracy drops while training set accuracy still increases (**overfitting**).

(c) There are only two classes for Spam dataset, but 26 classes for Letter Recognition dataset, so the Letter dataset might need more perceptrons to make more precisely classification. Therefore, I choose **(30, 30)** from Letter dataset and **(10,1 0)** for Spam dataset.

3. Accuracy over data size percentage

Combined the hyperparameters chosen from the previous experiments, the results are as follows:



(a) For the Letter dataset, the cross-validation and testing accuracy go up smoothly, showing that the hyperparameters are **well-chosen**.

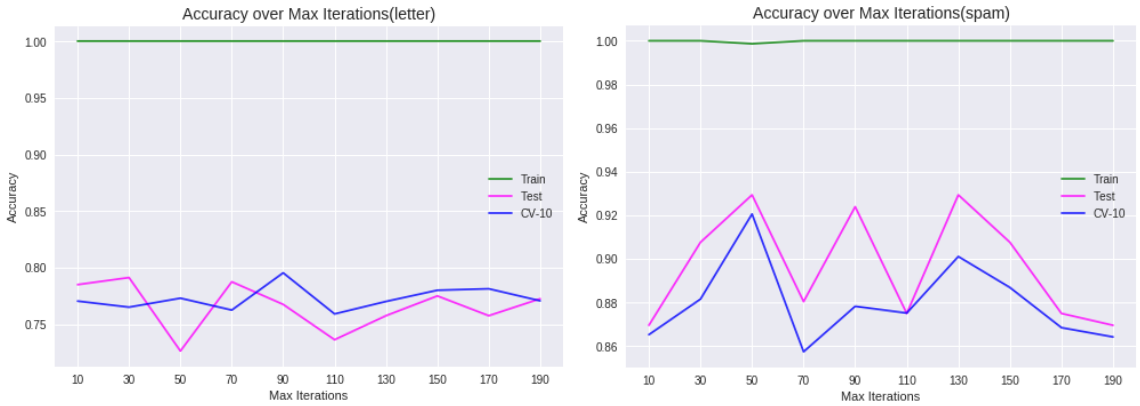
(b) For the Spam dataset, the cross-validation accuracy is maximized when 90% of dataset is used, showing that it might be overfit when 100% of dataset is used. Also, the decrease training accuracy might result from the effect of the **noise data** from Spam emails.

(c) Both datasets perform better when the data size percentage increases, especially for Letter Recognition dataset.

Boosting

Boosting is one of the three important techniques (the others are bagging and stacking) of ensemble learning. It combines a lot of **weak learners** and iteratively improving the performance of prediction. It is a powerful tool and will not overfit unless the base learners already do so. In this part, I will implement **Adaboost** algorithm with **CART decision tree** as base learners.

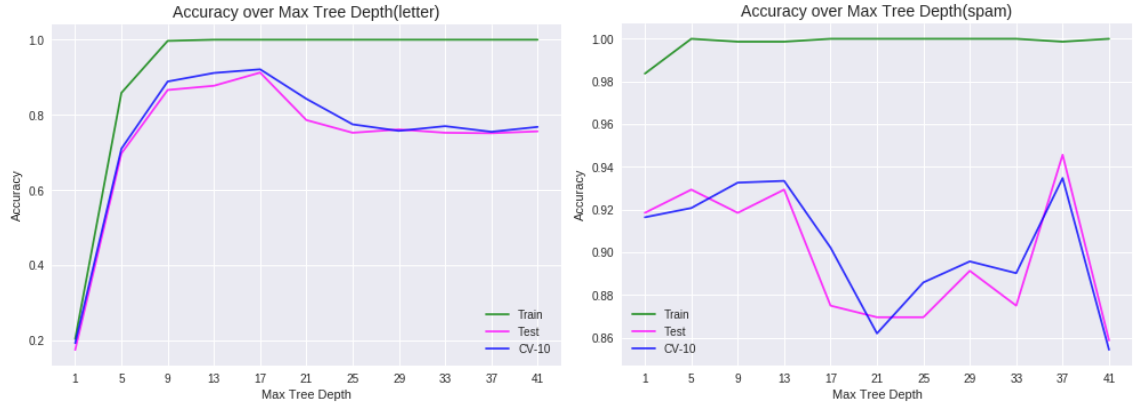
1. Accuracy over numbers of iterations



(a) For both datasets, the relation between iterations and accuracies is **not significant**. I infer that in the first few iterations, the models already perform well. Therefore, it is no need to iterate too many times before generating a good model.

(b) According to the cross-validation accuracy from the graph, I choose **90** for the Letter dataset and **50** for the Spam dataset.

2. Accuracy over max depth of decision trees



(a) For the Letter dataset, we can see the accuracy peak occurs when max tree depth is 17. After that point, the accuracy drops. The reason might be the base learner, in this case, the **decision tree is overfitting**. We can also observe the occurrence of overfitting by the 100% training set accuracy.

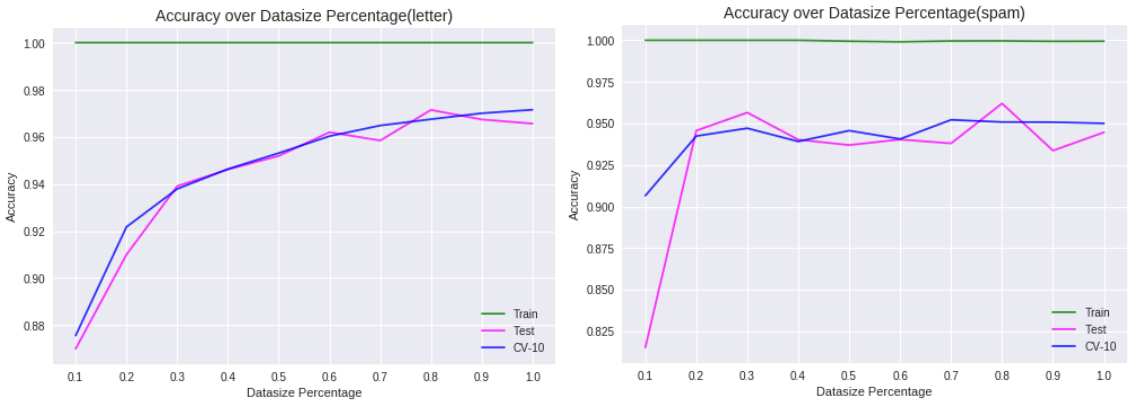
(b) For the Spam dataset, the same thing happens, and the max accuracy occurs at depth 13. Interestingly, there is also a peak occurs at depth 37. I supposed that it might be **a local maximum** because for decision tree, the shorter one is the better (rule of thumb).

Also, the accuracy in depth 13 and 37 are nearly the same but latter one takes much more time to train. Therefore, I will choose 13 instead of 37.

(c) To conclude, to prevent overfit, I choose depth **17** for the Letter dataset and **13** for the Spam dataset.

3. Accuracy over data size percentage

Combined the hyperparameters chosen from the previous experiment, the results are as follows:



(a) For the Letter dataset, the cross-validation accuracy goes up smoothly, showing that the hyperparameters are **well-chosen**. The accuracy is very high, nearly 97%

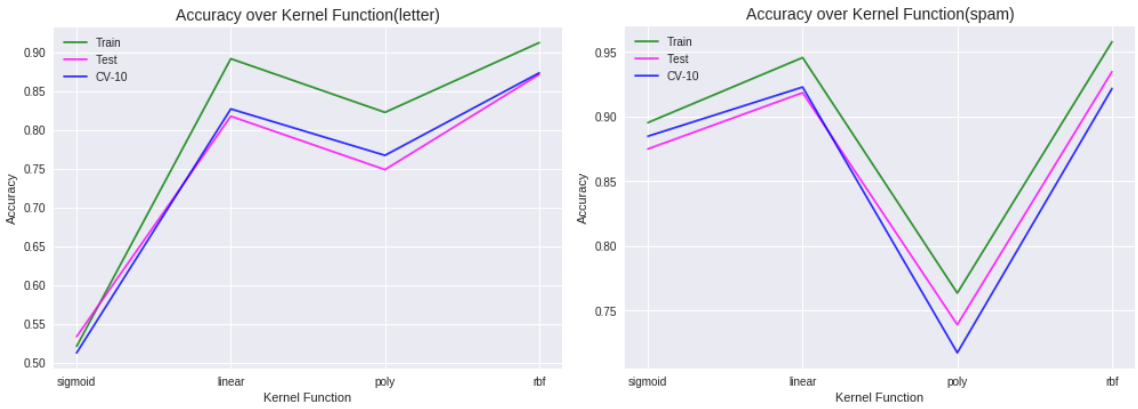
(b) For the Spam dataset, it only needs 20% of data to reach nearly 95% accuracy. It might because spam emails have some **significant characteristics** that can be classified easily.

(c) The results not only show the importance of finding appropriate hyperparameters, but also tell us that **the union of weak learners** can be really powerful.

Support Vector Machines (SVM)

In SVM algorithm, the choose of **kernel function** matters a lot, because the decision boundary highly depends on the distribution of values in datasets. In this algorithm, I am going to explore the effect of applying different kinds of kernel functions on the two datasets.

1. Accuracy over kernel function (linear, polynomial, sigmoid, Gaussian (RBF))



(a) For the **sigmoid** function, we can see that it performs really bad on Letter Recognition but good on identifying spam mails. It is because sigmoid function returns two values, **0 and 1**, therefore it is more suitable for **binary classification problems**, in this case, spam mail identification.

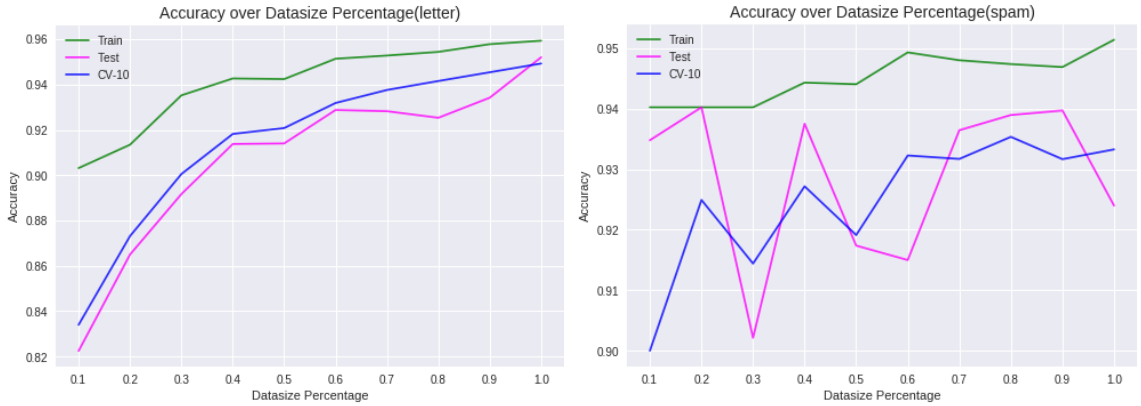
(b) For the **polynomial** function, the accuracies are lower than the linear function in both datasets. What we learn is that the complex functions are **not always better** than the simple ones. I think it might because these two datasets are relatively trivial to be classified so the linear function performs better.

(c) The Gaussian Radial Basis Function (RBF) is the best kernel function for the two datasets and performs slightly better than the linear kernel. It shows that the datasets are separated well by nonlinear feature mapping from hyperplane.

(d) In conclusion, **RBF** kernel function is chosen for both datasets.

2. Accuracy over data size percentage

   Use the RBF function chosen from the previous experiment, the results are as follows:



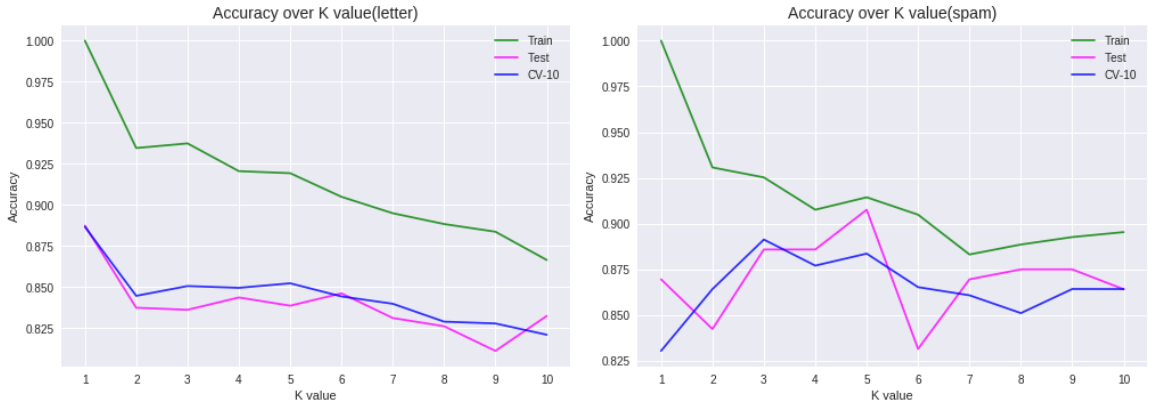
(a) For the Letter dataset, the cross-validation and testing accuracy go up smoothly, showing that the kernel function is **well-chosen**.

(b) For the Spam dataset, although the testing and cross-validation accuracy seems fluctuated, they are in fact in the range of **0.90 to 0.94**, which means that just a small amount of dataset is enough for SVM to perform well.

K-nearest neighbors (KNN)

KNN algorithm is a **lazy learning** algorithm, which means the training phase is very fast and most of computation is conducting in **testing phase**. The algorithm uses the **whole dataset** for prediction instead of trained models. Also, there are only **two** hyperparameter: **K** and **distance function** to be determined, making it is easy to fine-tune the model.

1. Accuracy over K value



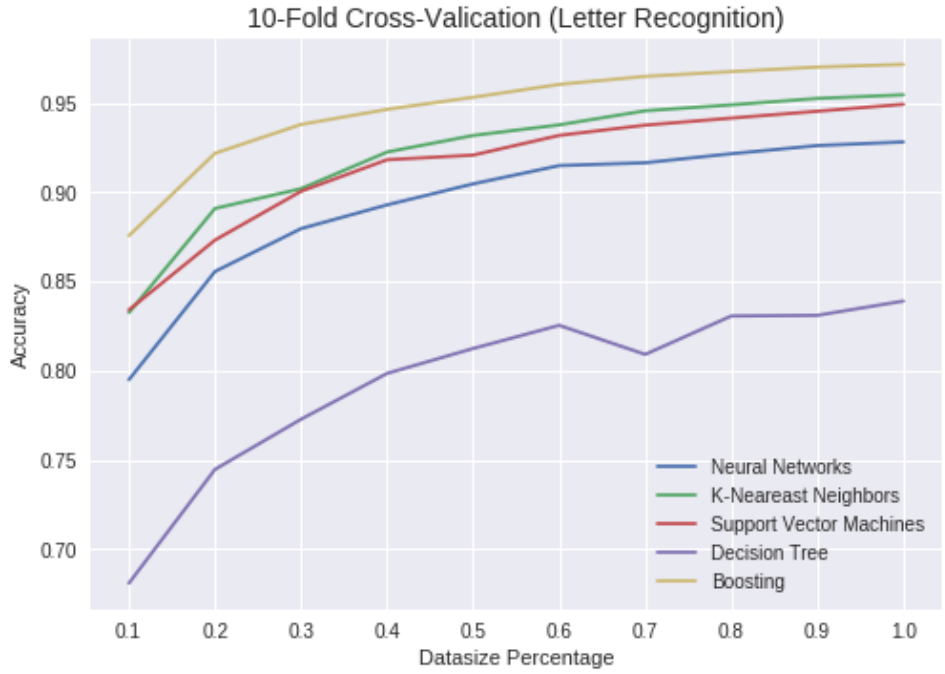
(a) For the Letter dataset, the training accuracy decreases as K increases. Thus, K = 1 is the best.
(b) For the Spam dataset, the cross-validation accuracy is maximized when K = 3. After that, it performs worse.
(c) Because there are only 16 attributes for Letter Recognition dataset, but 57 for Spam dataset, it might need more neighbors for Spam to vote for more precise classification.
(d) Noted that the training accuracy is decreased as the K increased, showing that making K too large is indeed harmful. Therefore, I choose **K = 1** from Letter Recognition dataset and **K = 3** for Spam dataset.

2. Accuracy over data size percentage
   Use the K value chosen from the previous experiment, the results are as follows:

(a) For both datasets, KNN algorithm performs well with the accuracy increases as the dataset size increases.
(b) From the trend of the graphs, it can still do better by providing more data for KNN to learn.

Summary

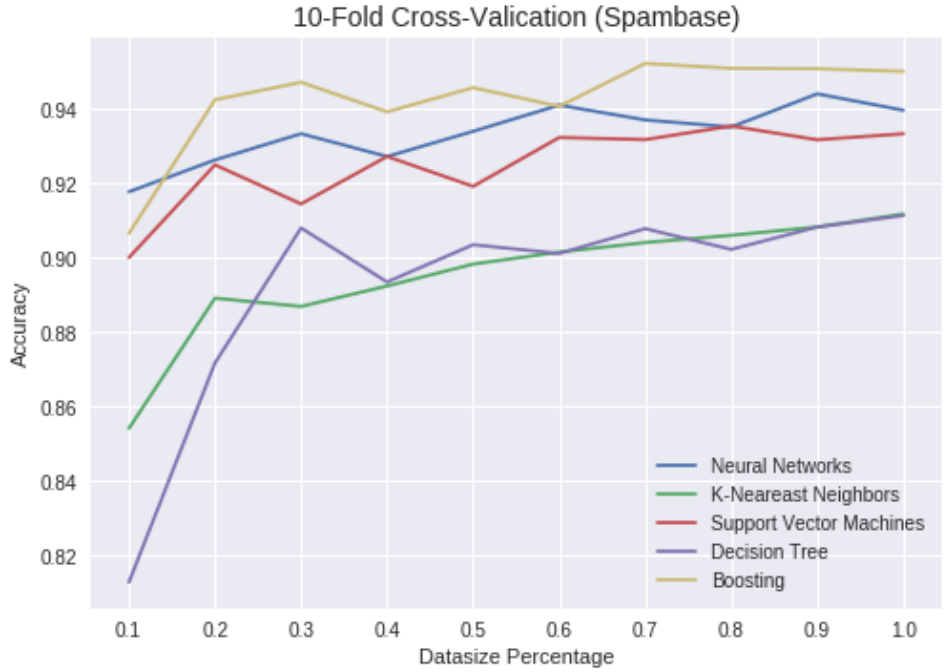
1. Follows are the combination of cross-validation accuracies from the five algorithms on the **Letter Recognition** dataset:



(a) The five algorithms all perform better when the size of data increases.

(b) Boosting is the best algorithm among the five algorithms for its accuracy reaches **97.16%**

2. Follows are the combination of cross-validation accuracies from the five algorithms on the **Spambase** dataset:



(a) The five algorithms all perform better when the size of data increases, but not as significant as the previous graph.

(b) For the Spambase dataset, only **20% - 40%** of dataset (920-1840 entries) is enough to build a well-performed model.

(c) **Boosting** is the best algorithm among the five algorithms for its accuracy reaches **95.00%**

3. The last part is the comparisons between algorithms

Algorithm	Training Speed	Testing Speed	Prediction Accuracy
Decision trees	Fast	Fast	Lower
Neural networks	Slow	Fast	Lower
Boosting	Slow	Fast	**Highest**
Support Vector Machines	Fast	Fast	Higher
k-nearest neighbors	Fast	Depends on K	Higher

Reference

[1] Scott Robinson. (2018). K-Nearest Neighbors Algorithm in Python and Scikit-Learn. Retrieved from https://stackabuse.com/k-nearest-neighbors-algorithm-in-python-and-scikit-learn/

[2] Scott Robinson. (2018). Introduction to Neural Networks with Scikit-Learn. Retrieved from https://stackabuse.com/introduction-to-neural-networks-with-scikit-learn/

[3] Usman Malik. (2018). Implementing SVM and Kernel SVM with Python's Scikit-Learn. Retrieved from https://stackabuse.com/implementing-svm-and-kernel-svm-with-pythons-scikit-learn/

[4] Scott Robinson. (2018). Decision Trees in Python with Scikit-Learn. Retrieved from https://stackabuse.com/decision-trees-in-python-with-scikit-learn/

[5] Robert R.F. (2018). DeFilippiBoosting, Bagging, and Stacking — Ensemble Methods with sklearn and mlens. Retrieved from https://medium.com/@rrfd/boosting-bagging-and-stacking-ensemble-methods-with-sklearn-and-mlens-a455c0c982de

[6] Ritchie Ng. Ensemble Learning, Adaboost, Retrieved from https://www.ritchieng.com/machine-learning-ensemble-of-learners-adaboost/

[7] Avinash Navlani. (2018). AdaBoost Classifier in Python, Retrieved from https://www.datacamp.com/community/tutorials/adaboost-classifier-python