**Introduction:**

1. **Description**

* In this assignment, I’m trying to build a model that can classify the different characters in UCI letter-recognition dataset. The prediction is important because it can helps us to classify different letters from ancient books, which is important to save culture. In classification model, I think accuracy, the running time and whether the classifier is easy to use are important factors to determine whether a model is a good one or not.
* Binary classification is the task of classifying the elements of a set into two groups on the basis of a classification rule. I chose ‘A’ and ‘C’ for the third problem. I think ‘H and ‘K’ may the hardest pair to classify, because as my results, after using KNN to classify these two characters, the accuracy of KNN continues to decrease. And many properties of ‘H’ and ‘K’ have little difference, which is hard for classifier to classify.
* I think dimension reduction is not useful in this classification problem. Because after using methods like PCA, as the result shows, the accuracy of KNN decreases, and the running time will increase. NMF performs best in all of the five algorithms, its accuracy is the highest, and there is little variance in accuracy after using NMF. I think the accuracy, running time, space complexity are all important factors to judge a dimension methos.

1. **Results**

2.1 KNN

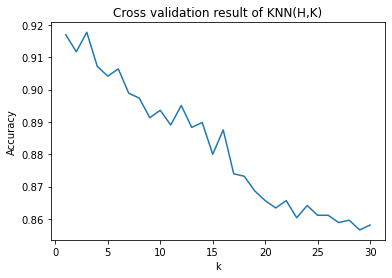
k-nearest neighbors is a machine learning algorithm that based on the similarity defined by distance between data points to make prediction. When encounter an unseen observation, kNN will perform majority vote (or get the average for regression problem) on k seen observations that is closest to the unseen one to determine the class value of it.

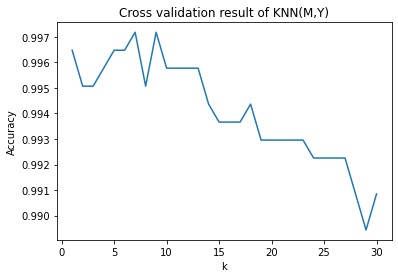
Advantage: training is very fast, and how the result gets predicted is very intuitive.

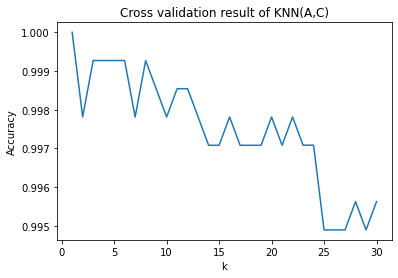
Disadvantage: finding k nearest points is very time consuming

I tested KNeighborsClassifier(n\_neighbors=n\_neighbors,metric='chebyshev',algorithm='ball\_tree',leaf\_size=30,weights='distance',p=4)

The results are as follows:

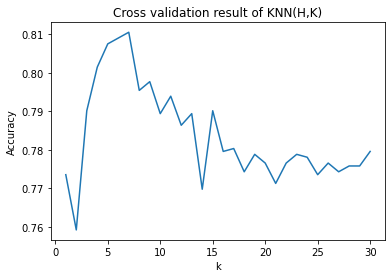


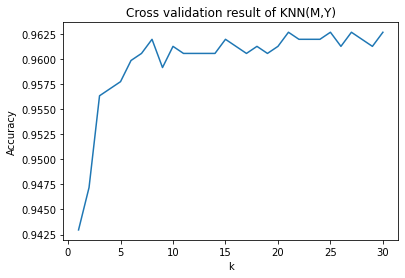


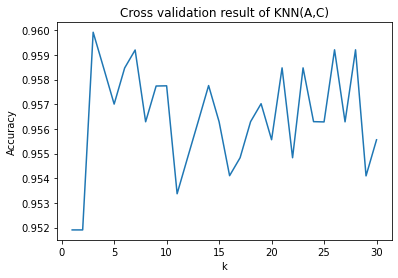


PCA is the process of computing the principal components and using them to perform a change of basis on the data, sometimes using only the first few principal components and ignoring the rest.

After PCA:







2.2 Decision tree

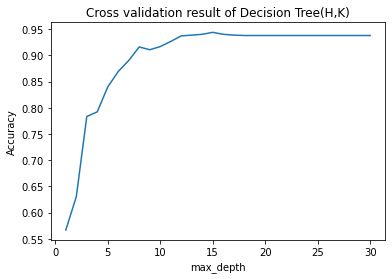
Decision tree is a machine learning algorithm that has a root node followed by a series of decision nodes that try to split the data and maximize the homogeneity of each node (each node will be as pure as possible). The leaf node will contain the prediction.

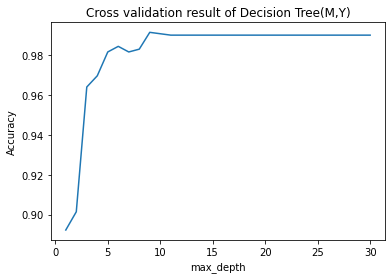
Advantage: work for both numerical and categorical data, easy to explain and produce more intuitive result

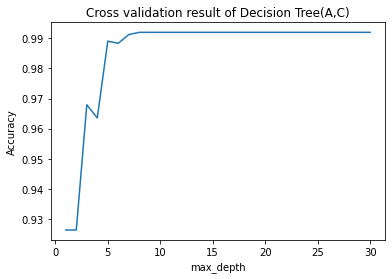
Disadvantage: Easy to get overfitted as it is affected by noise in the data.

I tested random\_state, max\_depth, splitter, criterion min\_samples\_leaf and min\_samples\_split

The results are as follows:

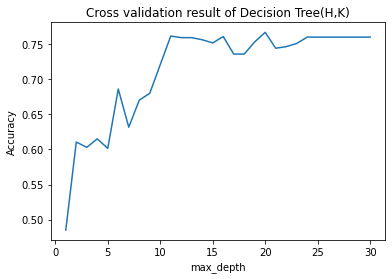


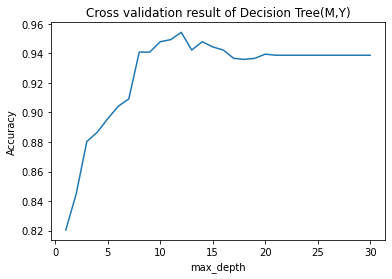


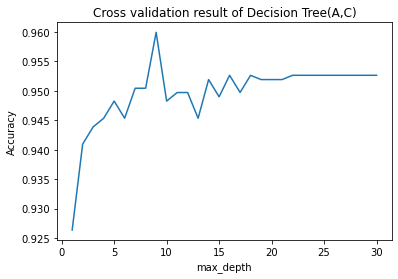


PCA is the process of computing the principal components and using them to perform a change of basis on the data, sometimes using only the first few principal components and ignoring the rest.

After PCA:







2.3 Random Forest

Random Forest is a bagging machine learning algorithm that contains large number of individual decision trees, where each tree will predict a specific class label and then major voting will decide the vote for final prediction. The idea is for all the individual tree with low performance to work together to output a strong result

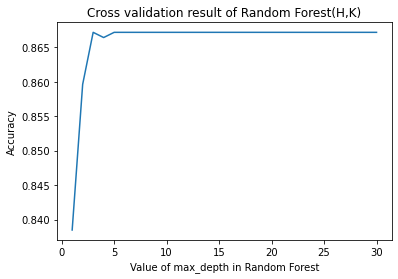
Advantage: avoid overfitting (the problem of single decision tree), being able to combine result of different decision trees to have a stronger model. It also can solve both classification and regression problem.

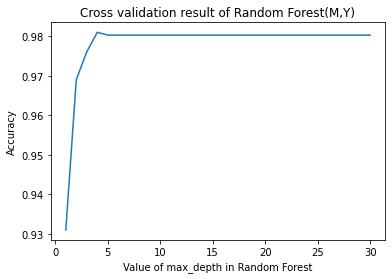
Disadvantage: The model created is too complicated that there is no intuitive reasoning behind. The training time is also huge since there is a lot of trees.

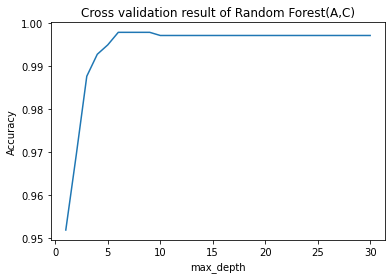
I tested

rf = RandomForestClassifier(random\_state=0,max\_depth=k,max\_features='sqrt',criterion='gini',min\_impurity\_decrease=0.02,bootstrap=True)

The results are as follows:

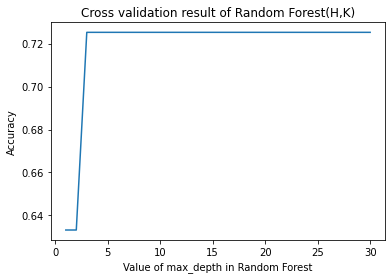


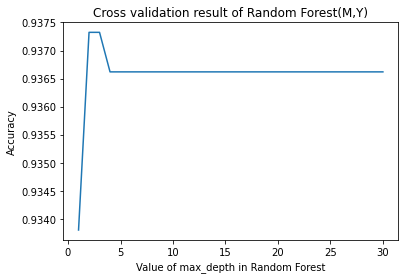


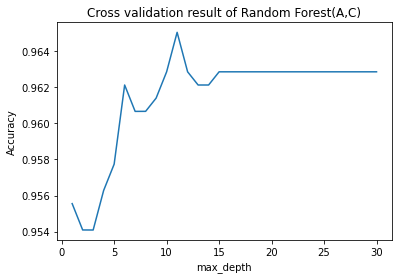


NMF decomposes (or factorizes) high-dimensional vectors into a lower-dimensional representation.

After NMF:







2.4 SVM

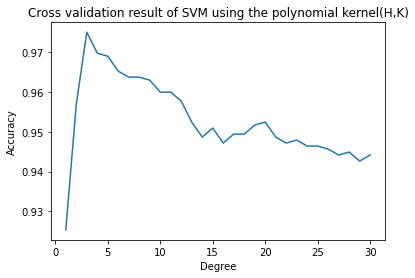
Support vector machines (SVMs) are a set of supervised learning methods used for classification, regression and outliers detection

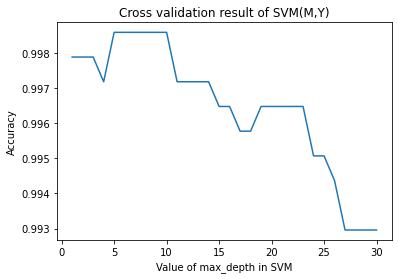
Advantage: SVM works relatively well when there is a clear margin of separation between classes. SVM is more effective in high dimensional spaces. SVM is effective in cases where the number of dimensions is greater than the number of samples.

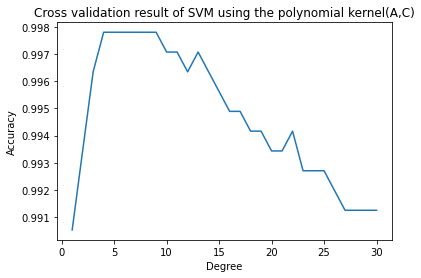
Disadvantage: SVM algorithm is not suitable for large data sets. SVM does not perform very well when the data set has more noise target classes are overlapping. In cases where the number of features for each data point exceeds the number of training data samples, the SVM will underperform.

I tested max\_iter, kernel, degree, cache\_size and shrinking

The results are as follows:

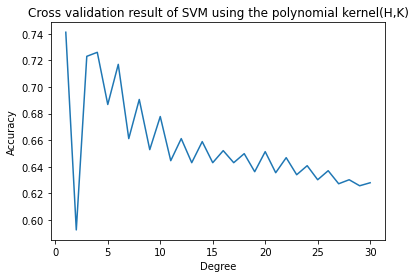


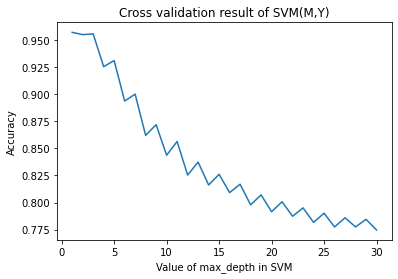


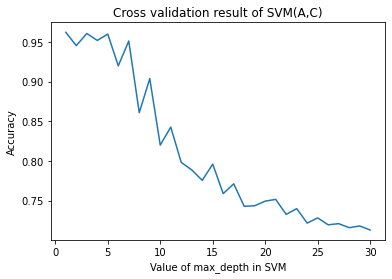


FastICA seeks an orthogonal rotation of prewhitened data, through a fixed-point iteration scheme, that maximizes a measure of non-Gaussianity of the rotated components.

After FastICA:







2.5 Artificial Neural Network

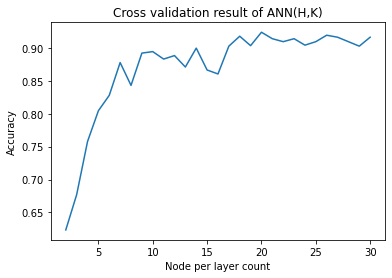
Artificial Neural networks (ANN) or neural networks are computational algorithms. It intended to simulate the behavior of biological systems composed of “neurons”. ANNs are computational models inspired by an animal's central nervous systems. It is capable of machine learning as well as pattern recognition.

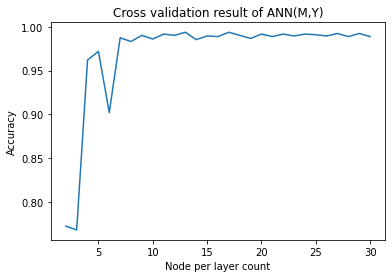
Advantage: Problems in ANN are represented by attribute-value pairs. ANNs are used for problems having the target function, the output may be discrete-valued, real-valued, or a vector of several real or discrete-valued attributes. ANN learning methods are quite robust to noise in the training data. The training examples may contain errors, which do not affect the final output.

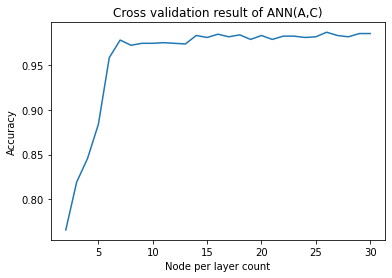
Disadvantage: Hardware dependence: Artificial neural networks require processors with parallel processing power, in accordance with their structure. For this reason, the realization of the equipment is dependent. Unexplained behavior of the network: This is the most important problem of ANN. When ANN produces a probing solution, it does not give a clue as to why and how. This reduces trust in the network.

I tested optimizer, loss and metrics

The results are as follows:

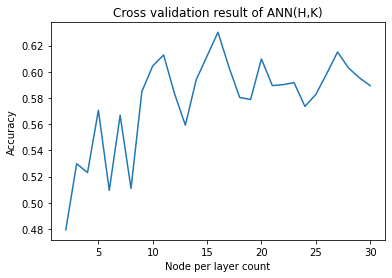


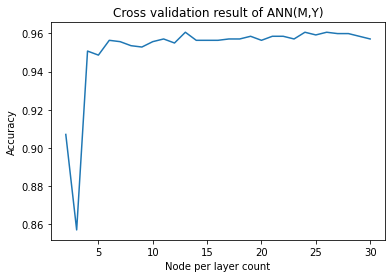


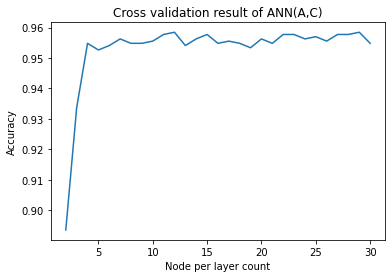


Sparse principal component analysis (sparse PCA) is a specialised technique used in statistical analysis and, in particular, in the analysis of multivariate data sets. It extends the classic method of principal component analysis (PCA) for the reduction of dimensionality of data by introducing sparsity structures to the input variables.

After Sparse PCA:







2.6 AdaBoost

AdaBoost is an ensemble learning method (also known as “meta-learning”) which was initially created to increase the efficiency of binary classifiers. AdaBoost uses an iterative approach to learn from the mistakes of weak classifiers, and turn them into strong ones.

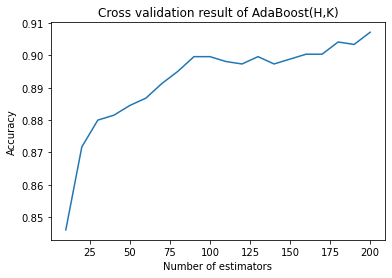
Advantage: being able to aggregate multiple week models to output a robust model, and it is robust against overfitting.

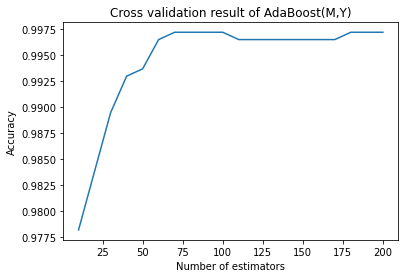
Disadvantage: it needs good dataset to be able to perform well – not good against noise.

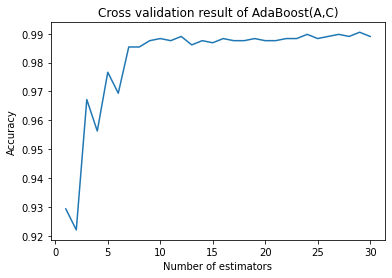
I tested

ada = AdaBoostClassifier(n\_estimators=k,learning\_rate=0.5,algorithm='SAMME',random\_state=20,base\_estimator=None)

The results are as follows:

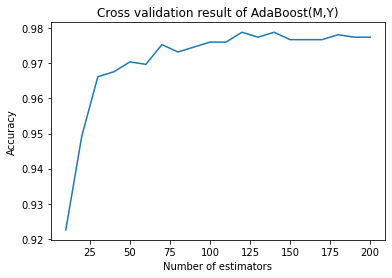


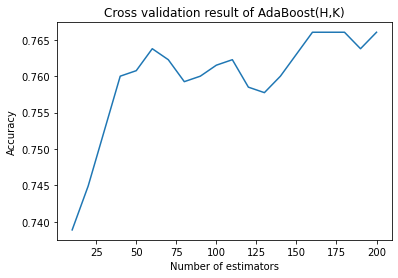


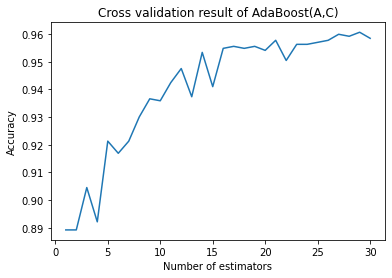


kernel principal component analysis (kernel PCA) is an extension of principal component analysis (PCA) using techniques of kernel methods. Using a kernel, the originally linear operations of PCA are performed in a reproducing kernel Hilbert space.

After kernel PCA:







2.6 Ridge regression

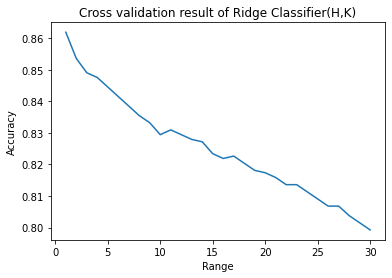
Ridge regression is a method of estimating the coefficients of multiple-regression models in scenarios where linearly independent variables are highly correlated

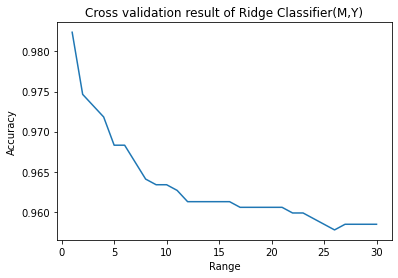
Advantage: prevent the weights from blowing (and also prevent overfitting). Also, the training is fast.

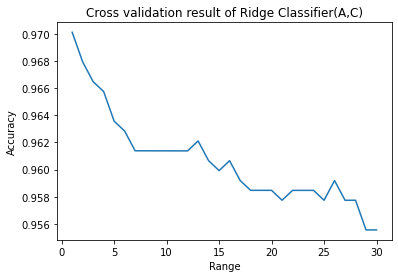
Disadvantage: Since we treat the problem as a regression, it may not be the best classifier.

I tested alpha, solver, random\_state, fit\_intercept and normalize

The results are as follows

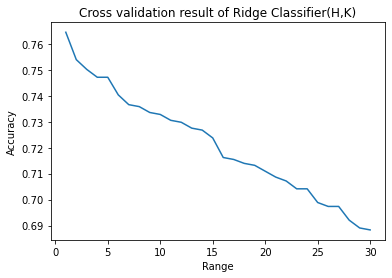


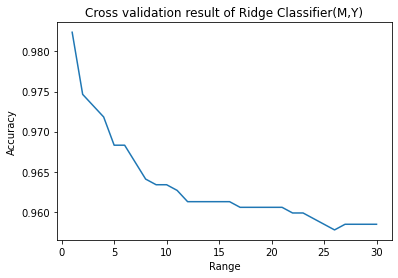


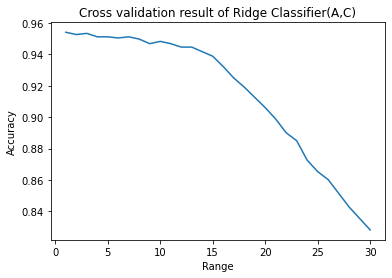


MDS is a method to visualize the di-similarity similarity between samples. MDS returns an optimal solution to represent the data in a smaller dimension space, in case the number of dimensions k is predefined.

After MDS:







**3.Discussion:**



In the table, it can be easily found that KNN model has the best accuracy up to 99% on validation set. And in (M,Y) and (A,C) pairs the accuracy is higher than 97% for all models. But it pair (H,K), the accuracy of decision tree is about 90%, and for ANN the accuracy is about 94%.

The running of ANN is highest which takes about 1000ms, and its validation time is about 90ms.

For random forest the running time takes about 88ms on training set, and other models takes about 1-4ms

On validation set, random forest takes about 6ms, and others takes about 0-4ms.

For extra models AdaBoost and Ridge, the running time of AdaBoost is about 163ms on training set and 9ms on validation set, and for Ridge it takes less than 2ms both on training and validation set.

After dimension reduction:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A\_C | TRAIN\_TIME | VAL\_TIME | TRAIN\_ACCURACY | VAL\_ACCURACY |
| KNN | 0.999212265 | 3.000259399 | 0.972303207 | 0.947712418 |
| Decision Tree | 1.000404358 | 0.5323 | 0.973760933 | 0.941176471 |
| Random Forest | 100.0015736 | 7.998466492 | 0.976676385 | 0.967320261 |
| SVM | 59.00883675 | 3.992080688 | 0.556851312 | 0.549019608 |
| Ridge Classifier | 0.999689102 | 0.7889 | 0.955539359 | 0.941176471 |
| AdaBoost Classifier | 133.9924335 | 6.999731064 | 0.987609329 | 0.947712418 |
| ANN | 968.0013657 | 88.01174164 | 0.958454788 | 0.947712421 |
| H\_K |  |  |  |  |
| KNN | 0.999212265 | 4.001379013 | 0.854339623 | 0.756756757 |
| Decision Tree | 1.998186111 | 0.9667 | 0.64 | 0.628378378 |
| Random Forest | 112.9922867 | 5.999565125 | 0.772075472 | 0.709459459 |
| SVM | 46.99993134 | 4.008769989 | 0.631698113 | 0.614864865 |
| Ridge Classifier | 2.000808716 | 0.6895 | 0.609056604 | 0.554054054 |
| AdaBoost Classifier | 129.991293 | 9.999752045 | 0.714716981 | 0.608108108 |
| ANN | 1023.013115 | 87.98742294 | 0.644528329 | 0.547297299 |
| M\_Y |  |  |  |  |
| KNN | 0.999927521 | 3.001213074 | 0.965492958 | 0.955696203 |
| Decision Tree | 2.00009346 | 0.53734 | 0.968309859 | 0.949367089 |
| Random Forest | 108.001709 | 5.999088287 | 0.966197183 | 0.962025316 |
| SVM | 54.00300026 | 4.006147385 | 0.609859155 | 0.588607595 |
| Ridge Classifier | 0.998735428 | 0.6893 | 0.945070423 | 0.949367089 |
| AdaBoost Classifier | 155.9984684 | 9.000301361 | 0.977464789 | 0.949367089 |
| ANN | 1092.993259 | 92.00167656 | 0.957746506 | 0.955696225 |

In the table, it can be easily found that KNN model has the best accuracy up to 97% on validation set in all pairs. And SVM has lowest accuracy which is about 55%. All models don’t perform well in pair (H, K), the accuracy is below 90%.

The running of ANN is highest which takes about 1000ms, and its validation time is about 90ms. And KNN has the lowest running in all models.

For random forest the running time takes about 88ms on training set, SVM is 54ms on training set and other models takes about 1-4ms

On validation set, the models take time less than 10ms.

For extra models AdaBoost and Ridge, the running time of AdaBoost is about 140ms on training set and 9ms on validation set, and for Ridge it takes less than 1ms both on training and validation set.

**Lessons:**

It takes lots of time for a neural network to train

The training and predicting time on validation set is generally low for models that are not neural network

I would choose KNN, because KNN is easy to explain, requires little training time. And KNN’s output is good even with dimension reduction. Using dimension reduction decreases the accuracy and also increases run times. If I was given the same task, I would choose several models to train the data, and choose the model with highest accuracy with lower running time.