CSE514 Data Mining
Programming Assignment 2

Xingjian Ding 502558

April 19, 2022

1 Introduction

1.1 description

In this project assignment, the data set used is a Letter Recognition Data Set from https://archive.ics.uci.edu /ml/datasets/letter+recognition. This data set contain 20,000 records and each record has 16 attributes. This project will compared the result of 5 different classification models with cross-validation and dimension reduction. After compared performance of those models with dimension reduction, choose a best approach to predict. To compared the performance, validation accuracy score is used to show the performance.

1.2 choose and prediction

For Pair 3, I am going to choose U and V, because I think these two letter is similar to each other and can show the difference between models. I predict Pair Two: M and Y is the the easiest to classify and Pair Three: U and V is the hardest to classify.

1.3 dimension reduction

The dimension reduction is important for this problem. Because original data set has 16 attributes. The compared pair of letter is similar, some of the attributes for two letter is almost same which can not help the model to train. For example, it can ignore the Low variance features. However, reduction from 16 to 4 may caused some important features be ignored. Therefore, I think dimension reduction is useful to this problem, but we may need attributes more than four.

2 Result

2.1 k-nearest neighbors

The k-nearest neighbors (KNN) algorithm is a data classification method for estimating the likelihood that a data point will become a member of one group or another based on what group the data points nearest to it belong to.
It's mainly used for classification problems. KNN is a lazy learning and non-parametric algorithm.
It's called a lazy learning algorithm or lazy learner because it doesn't perform any training when you supply the training data. Instead, it just stores the data during the training time and doesn't perform any calculations. It doesn't build a model until a query is performed on the dataset. This makes KNN ideal for data mining.
It decide a number k which is the nearest Neighbor to that data point that is to be classified. If the value of k is 5 it will look for 5 nearest Neighbors to that data point.

Advantages of using the k-nearest neighbors algorithm:
- It's easy to understand and simple to implement
- It can be used for both classification and regression problems
- It's ideal for non-linear data since there's no assumption about underlying data

- It can naturally handle multi-class cases
- It can perform well with enough representative data

Disadvantages of using the k-nearest neighbors algorithm:

- Associated computation cost is high as it stores all the training data
- Requires high memory storage
- Need to determine the value of K
- Prediction is slow if the value of N is high
- Sensitive to irrelevant features

([1] What Is K-Nearest Neighbor? An ML Algorithm to Classify Data, https://learn.g2.com/k-nearest-neighbor)

For additional hyperparameters tuned, the neighbors number was changed from 3 to 7 and used several different algorithm from auto, ball tree, kd tree and brute.

2.1.1 H and K before Dimension Reduction

k = 3: 0.952453 (std: 0.006579)
k = 4: 0.935849 (std: 0.004134)
k = 5: 0.950943 (std: 0.009840)
k = 6: 0.938868 (std: 0.011041)
k = 7: 0.941132 (std: 0.013416)
auto: 0.950943 (std: 0.009840)
ball tree: 0.951698 (std: 0.010512)
kd tree: 0.950943 (std: 0.009840)
brute: 0.950189 (std: 0.014596)

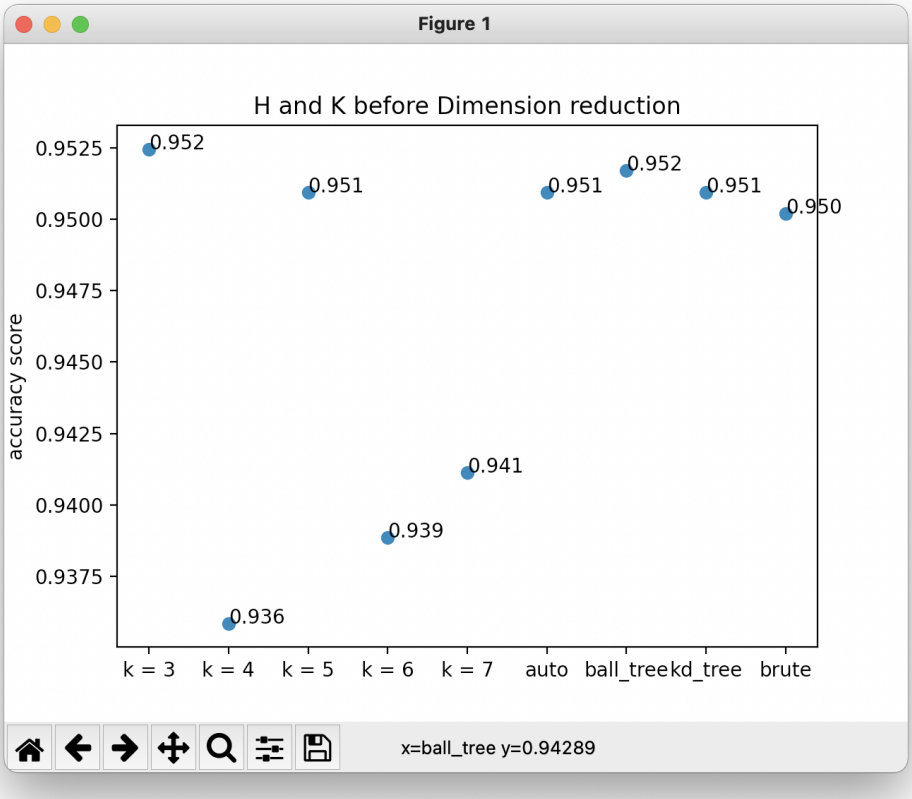

Figure 1: k-nearest neighbors: H and K before Dimension reduction

2.1.2 M and Y before Dimension Reduction

k = 3: 0.997887 (std: 0.002817)
k = 4: 0.997887 (std: 0.002817)
k = 5: 0.997887 (std: 0.002817)
k = 6: 0.997183 (std: 0.004106)
k = 7: 0.996479 (std: 0.004454)
auto: 0.997887 (std: 0.002817)
ball tree: 0.997887 (std: 0.002817)
kd tree: 0.997887 (std: 0.002817)
brute: 0.997887 (std: 0.002817)

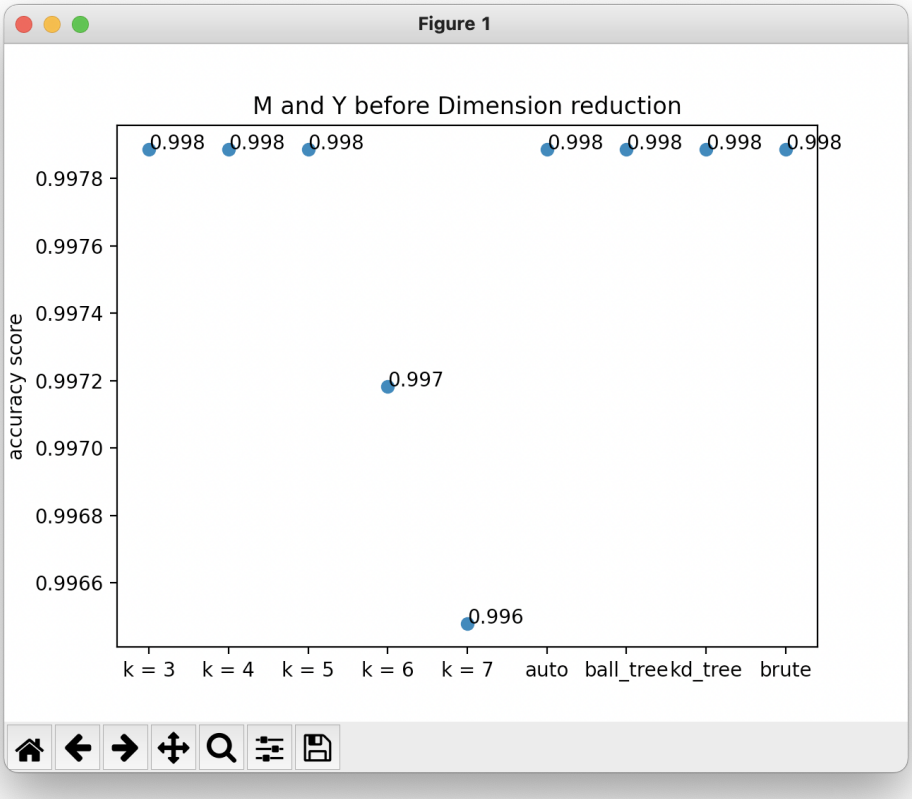

Figure 2: k-nearest neighbors: M and Y before Dimension reduction

2.1.3 U and V before Dimension Reduction

k = 3: 0.997183 (std: 0.002635)
k = 4: 0.997183 (std: 0.002635)
k = 5: 0.997887 (std: 0.002817)
k = 6: 0.998592 (std: 0.001725)
k = 7: 0.997887 (std: 0.002817)
auto: 0.997887 (std: 0.002817)
ball tree: 0.997887 (std: 0.002817)
kd tree: 0.997887 (std: 0.002817)
brute: 0.998592 (std: 0.002817)

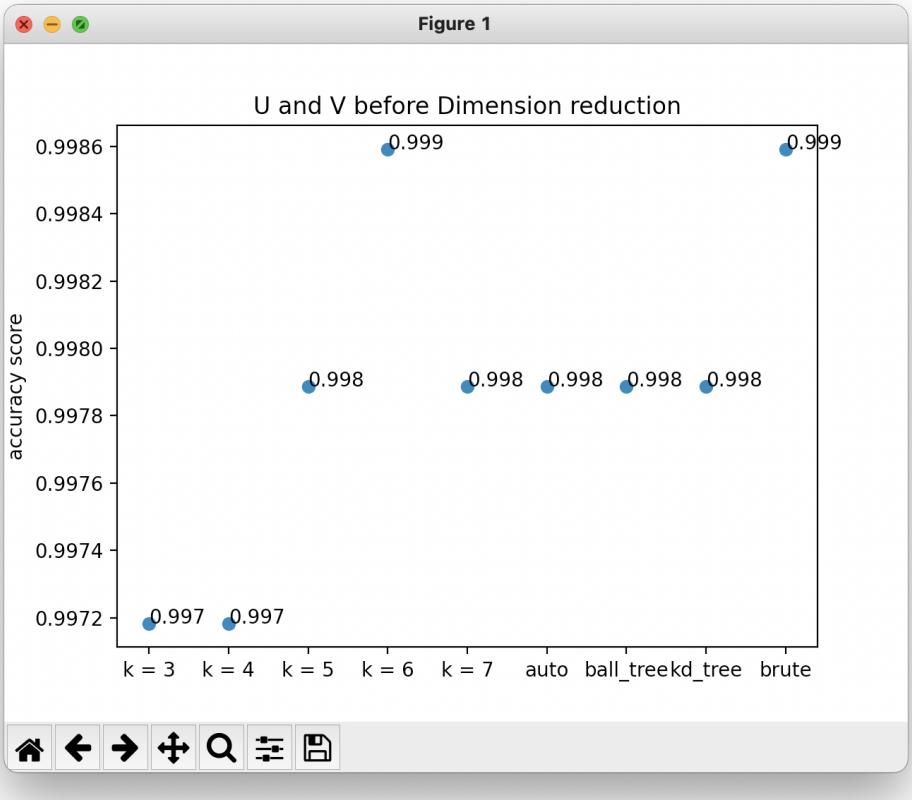

Figure 3: k-nearest neighbors: U and V before Dimension reduction

2.1.4 Dimension Reduction

For each classifier, several method of dimension reduction are used. The first method used is **Simple Quality Filtering**, algorithm will calculate the low variance for each feature then drop the low variance features.
For **Filter Methods**, the Pearson's Correlation will be used to measures the linear correlation between two variables, which are both continuous. It varies from -1 to +1, where +1 corresponds to positive linear correlation, 0 to no linear correlation, and 1 to negative linear correlation. Then drop the strongly correlate features.
The third method used to dimension reduction is **Embedded Methods**. I used the SelectKBest function from python package feature selection in sklearn.

2.1.5 H and K after Dimension Reduction

k = 3: 0.895094 (std: 0.026938)
k = 4: 0.896604 (std: 0.022966)
k = 5: 0.895849 (std: 0.028097)
k = 6: 0.896604 (std: 0.020475)
k = 7: 0.892830 (std: 0.026101)
auto: 0.895849 (std: 0.028097)
ball tree: 0.894340 (std: 0.025925)
kd tree: 0.895849 (std: 0.028097)
brute: 0.888302 (std: 0.022338)

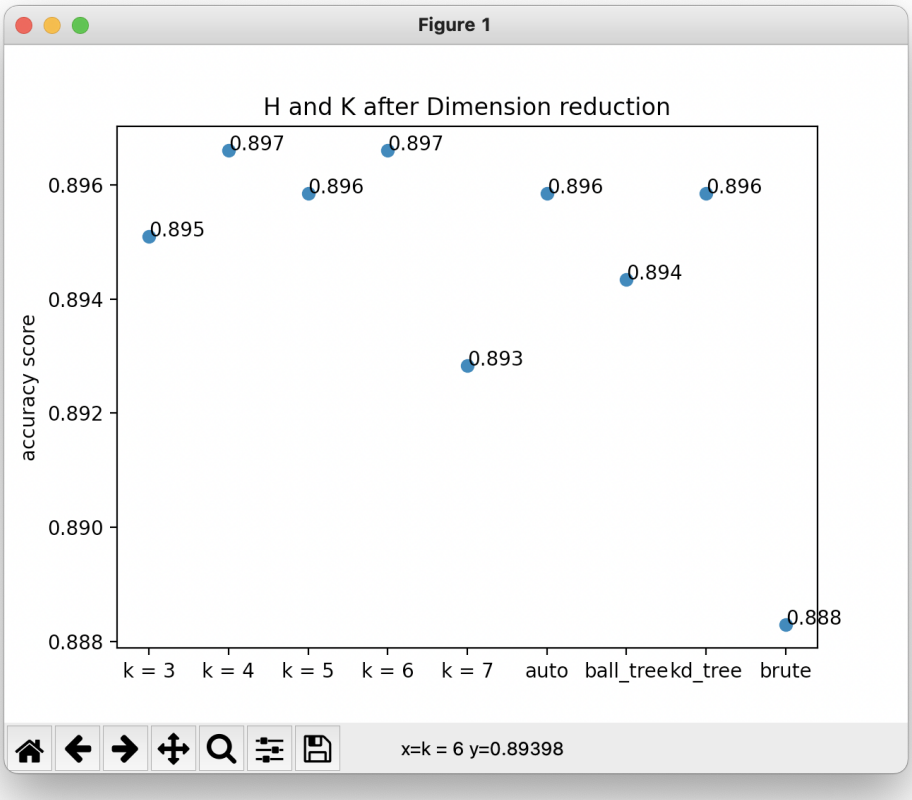

Figure 4: k-nearest neighbors: H and K before Dimension reduction

2.1.6 M and Y after Dimension Reduction

k = 3: 0.988732 (std: 0.005634)
k = 4: 0.990141 (std: 0.003450)
k = 5: 0.990141 (std: 0.003450)
k = 6: 0.989437 (std: 0.003857)
k = 7: 0.988732 (std: 0.006058)
auto: 0.990141 (std: 0.003450)
ball tree: 0.990845 (std: 0.004776)
kd tree: 0.990141 (std: 0.003450)
brute: 0.990845 (std: 0.004776)

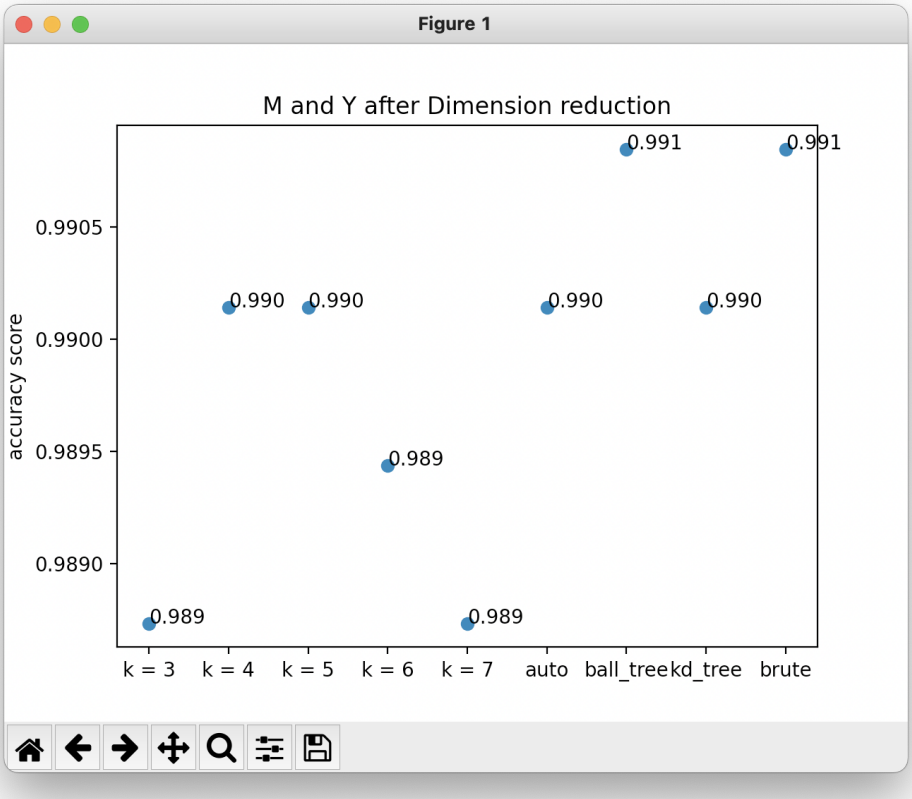

Figure 5: k-nearest neighbors: M and Y after Dimension reduction

2.1.7 U and V after Dimension Reduction

k = 3: 0.979565 (std: 0.010299)
k = 4: 0.978159 (std: 0.008143)
k = 5: 0.977452 (std: 0.007577)
k = 6: 0.973924 (std: 0.008514)
k = 7: 0.974630 (std: 0.007840)
auto: 0.977452 (std: 0.007577)
ball tree: 0.977452 (std: 0.007577)
kd tree: 0.977452 (std: 0.007577)
brute: 0.978162 (std: 0.009544)

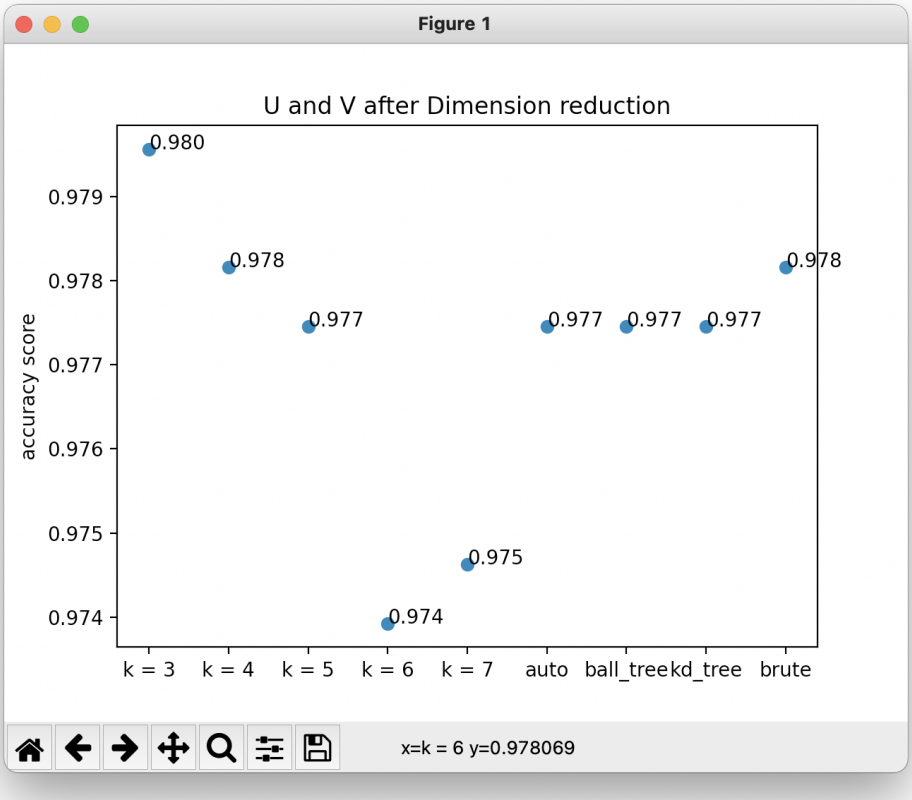

Figure 6: k-nearest neighbors: U and V after Dimension reduction

2.2 Decision tree

A decision tree is a decision support tool that uses a tree-like model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility.
It is one way to display an algorithm that only contains conditional control statements.

Advantages of the Decision trees:
- Are simple to understand and interpret. People are able to understand decision tree models after a brief explanation.
- Have value even with little hard data. Important insights can be generated based on experts describing a situation (its alternatives, probabilities, and costs) and their preferences for outcomes.
- Help determine worst, best, and expected values for different scenarios.
- Use a white box model. If a given result is provided by a model.
- Can be combined with other decision techniques.

Disadvantages of decision trees:
- They are unstable, meaning that a small change in the data can lead to a large change in the structure of the optimal decision tree.
- They are often relatively inaccurate. Many other predictors perform better with similar data. This can be remedied by replacing a single decision tree with a random forest of decision trees, but a random forest is not as easy to interpret as a single decision tree.
- For data including categorical variables with different numbers of levels, information gain in decision trees is biased in favor of those attributes with more levels.
- Calculations can get very complex, particularly if many values are uncertain and/or if many outcomes are linked.
(Decision tree, https://en.wikipedia.org/wiki)

For additional hyperparameters tuned, the max depth was changed from 10 to 40 and min samples leaf was changed from 1 to 5.

2.2.1 H and K before Dimension Reduction

MD=10: 0.932830 (std: 0.007697)
MD=20: 0.944151 (std: 0.006917)
MD=30: 0.942642 (std: 0.006917)
MD=40: 0.941887 (std: 0.005119)
ML=1: 0.941132 (std: 0.007770)
ML=2: 0.935094 (std: 0.004401)
ML=3: 0.935094 (std: 0.005006)
ML=4: 0.927547 (std: 0.006492)
ML=5: 0.933585 (std: 0.008472)

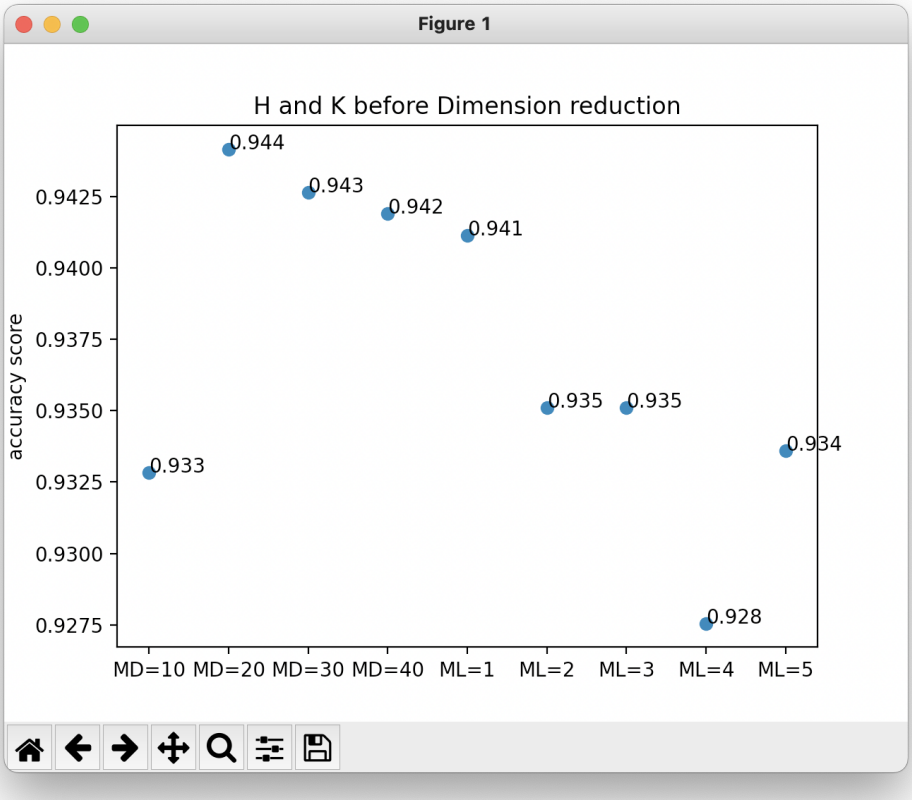

Figure 7: decision tree: H and K before Dimension reduction

2.2.2 M and Y before Dimension Reduction

MD=10: 0.990845 (std: 0.003591)
MD=20: 0.990845 (std: 0.003591)
MD=30: 0.992958 (std: 0.002227)
MD=40: 0.990845 (std: 0.002817)
ML=1: 0.990845 (std: 0.005270)
ML=2: 0.991549 (std: 0.004776)
ML=3: 0.988732 (std: 0.001408)
ML=4: 0.986620 (std: 0.005634)
ML=5: 0.985211 (std: 0.008152)

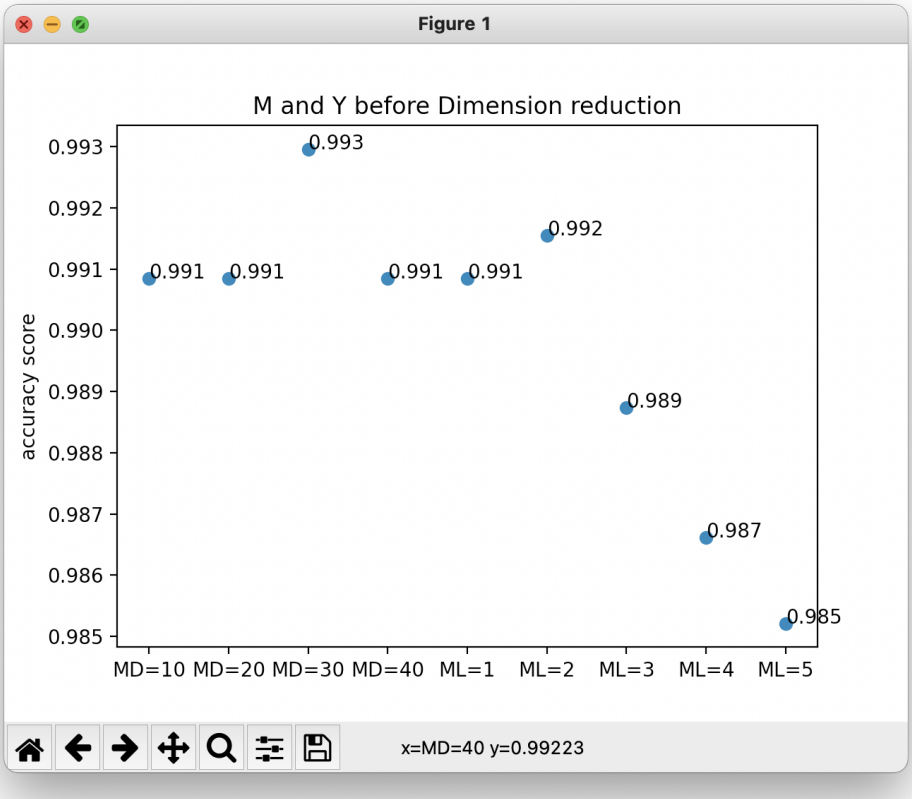

Figure 8: decision tree: M and Y before Dimension reduction

2.2.3 U and V before Dimension Reduction

MD=10: 0.978866 (std: 0.011777)
MD=20: 0.980277 (std: 0.017616)
MD=30: 0.979570 (std: 0.014496)
MD=40: 0.982387 (std: 0.008619)
ML=1: 0.978167 (std: 0.015002)
ML=2: 0.980277 (std: 0.010581)
ML=3: 0.973232 (std: 0.018837)
ML=4: 0.977455 (std: 0.010342)
ML=5: 0.977460 (std: 0.010100)

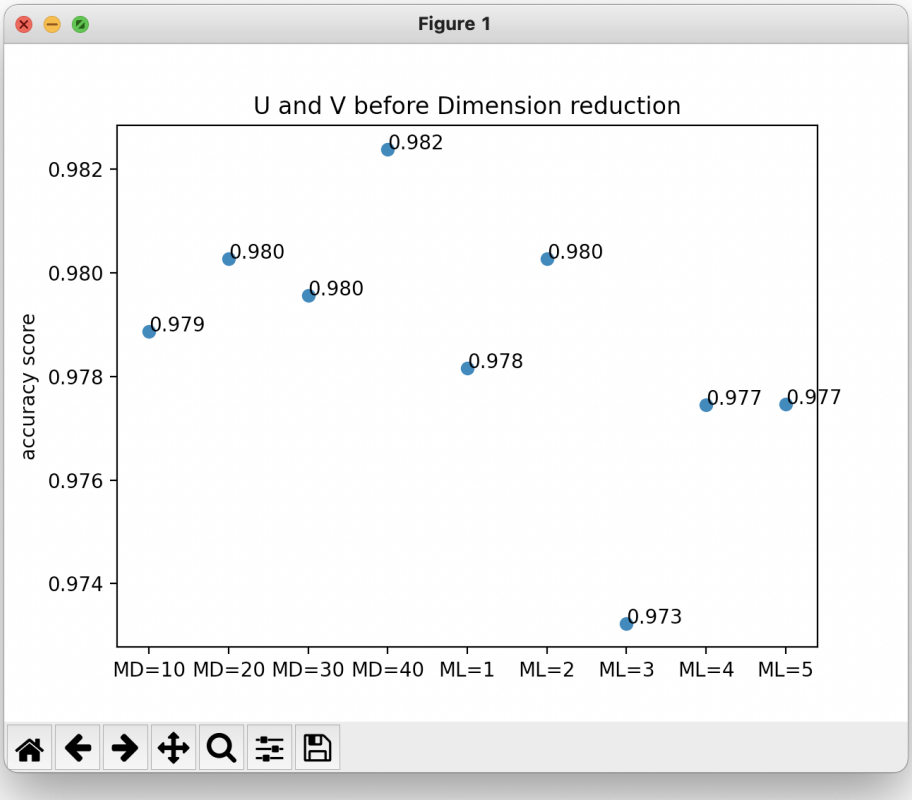

Figure 9: decision tree: U and V before Dimension reduction

2.2.4 Dimension Reduction

For each classifier, several method of dimension reduction are used. The first method used is **Simple Quality Filtering**, algorithm will calculate the low variance for each feature then drop the low variance features.
For **Filter Methods**, the Pearson's Correlation will be used to measures the linear correlation between two variables, which are both continuous. It varies from -1 to +1, where +1 corresponds to positive linear correlation, 0 to no linear correlation, and 1 to negative linear correlation. Then drop the strongly correlate features.
The third method used to dimension reduction is **Embedded Methods**. I used the SelectKBest function from python package feature selection in sklearn. (Same as before)

2.2.5 H and K after Dimension Reduction

MD=10: 0.898868 (std: 0.016081)
MD=20: 0.896604 (std: 0.015939)
MD=30: 0.898868 (std: 0.017111)
MD=40: 0.897358 (std: 0.017276)
ML=1: 0.896604 (std: 0.016977)
ML=2: 0.888302 (std: 0.016466)
ML=3: 0.890566 (std: 0.020391)
ML=4: 0.889811 (std: 0.020167)
ML=5: 0.893585 (std: 0.020025)

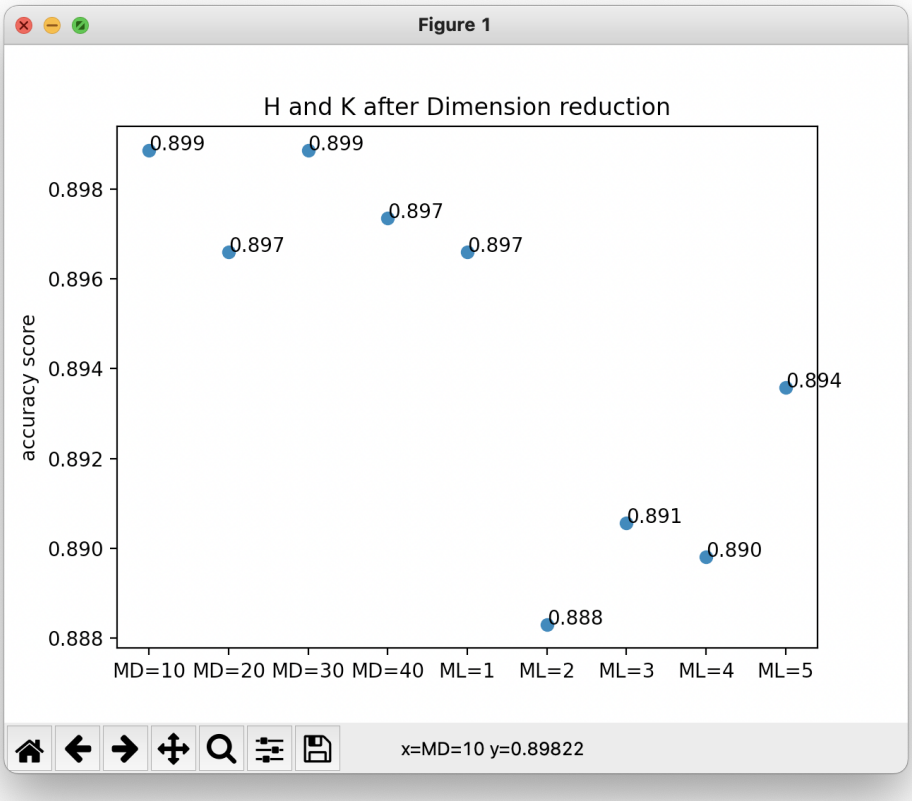

Figure 10: decision tree: H and K after Dimension reduction

2.2.6 M and Y after Dimension Reduction

MD=10: 0.988732 (std: 0.004106)
MD=20: 0.988028 (std: 0.004776)
MD=30: 0.988028 (std: 0.003591)
MD=40: 0.987324 (std: 0.004776)
ML=1: 0.988732 (std: 0.005634)
ML=2: 0.983099 (std: 0.008152)
ML=3: 0.983803 (std: 0.006531)
ML=4: 0.982394 (std: 0.006299)
ML=5: 0.982394 (std: 0.007386)

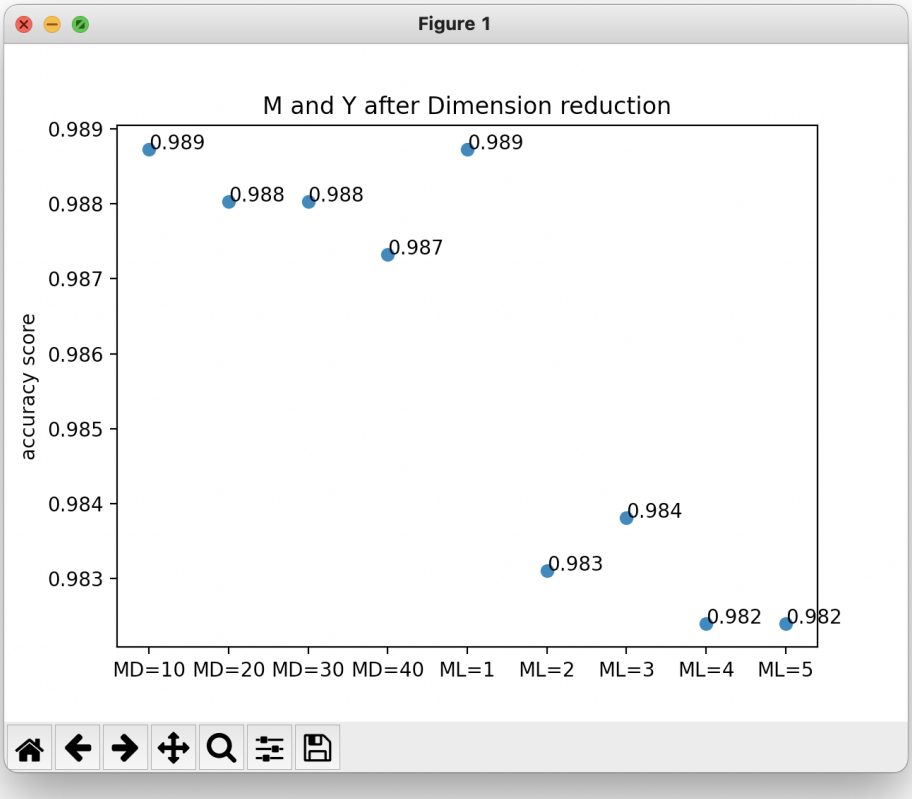

Figure 11: decision tree: M and Y after Dimension reduction

2.2.7 U and V after Dimension Reduction

MD=10: 0.977445 (std: 0.008522)
MD=20: 0.976741 (std: 0.008225)
MD=30: 0.976741 (std: 0.009615)
MD=40: 0.976741 (std: 0.008225)
ML=1: 0.976741 (std: 0.008225)
ML=2: 0.971809 (std: 0.007725)
ML=3: 0.971099 (std: 0.010566)
ML=4: 0.973212 (std: 0.009110)
ML=5: 0.971799 (std: 0.010733)

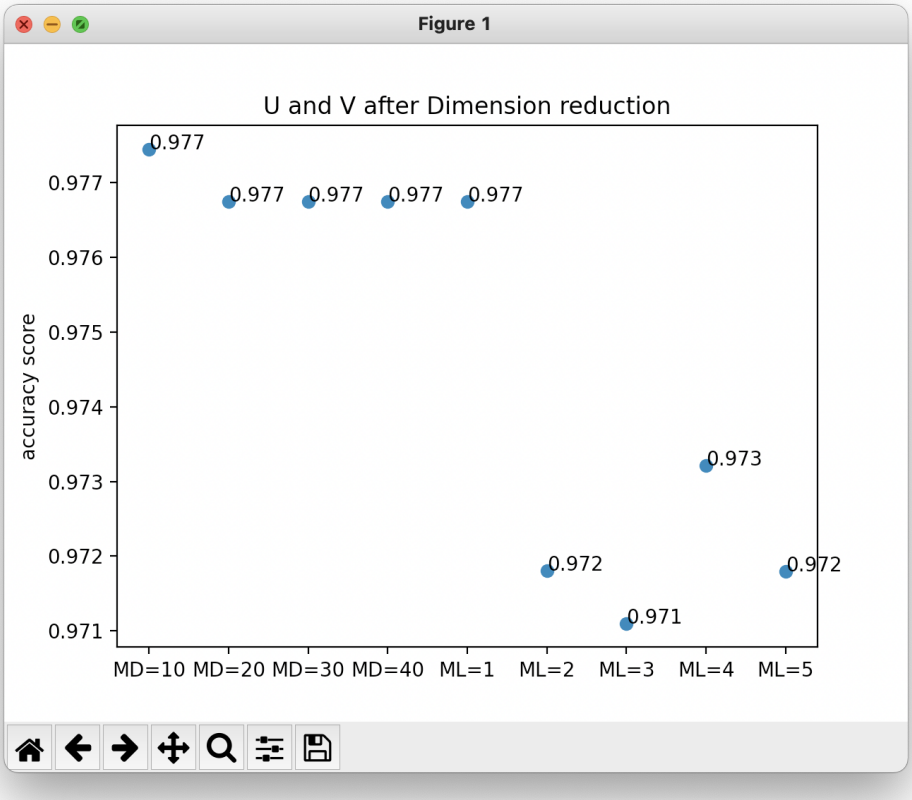

Figure 12: decision tree: U and V after Dimension reduction

2.3 Random Forest

Random forest is a commonly-used machine learning algorithm trademarked by Leo Breiman and Adele Cutler, which combines the output of multiple decision trees to reach a single result. Its ease of use and flexibility have fueled its adoption, as it handles both classification and regression problems.

Advantages of the Decision trees:
- Reduced risk of overfitting
- Provides flexibility
- Easy to determine feature importance

Disadvantages of decision trees:
- Time-consuming process
- Requires more resources
- More complex

(Random Forest, https://www.ibm.com/cloud/learn/random-forest)

For additional hyperparameters tuned, the number of trees = was changed from 50 to 150, the max depth was changed from 10 to 50.

2.3.1 H and K before Dimension Reduction

number of trees = 50 : 0.972830 (std: 0.008058)
number of trees = 75 : 0.969811 (std: 0.007160)
number of trees = 100 : 0.972075 (std: 0.010566)
number of trees = 125 : 0.974340 (std: 0.008058)
number of trees = 150 : 0.970566 (std: 0.011789)
MD=10: 0.971321 (std: 0.013202)
MD=20: 0.970566 (std: 0.012028)
MD=30: 0.974340 (std: 0.011545)
MD=40: 0.975094 (std: 0.010012)
MD=50: 0.971321 (std: 0.010832)

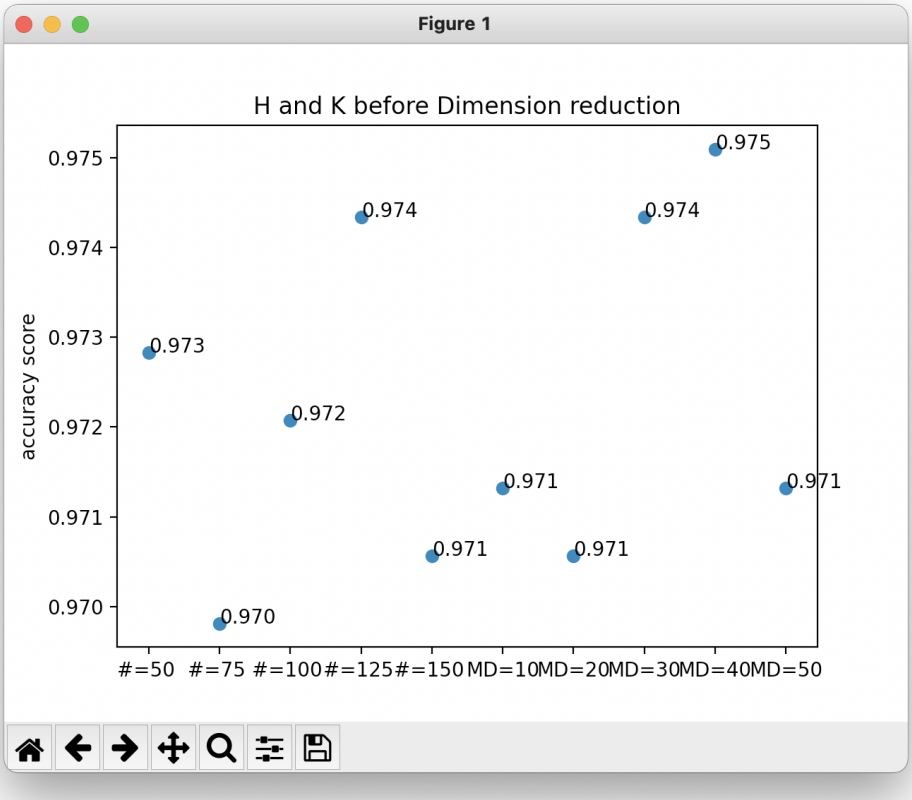

Figure 13: Random Forest: H and K before Dimension reduction

2.3.2 M and Y before Dimension Reduction

number of trees = 50 : 0.996479 (std: 0.003857)
number of trees = 75 : 0.996479 (std: 0.003857)
number of trees = 100 : 0.996479 (std: 0.003857)
number of trees = 125 : 0.995775 (std: 0.005175)
number of trees = 150 : 0.996479 (std: 0.003857)
MD=10: 0.996479 (std: 0.003857)
MD=20: 0.996479 (std: 0.003857)
MD=30: 0.997183 (std: 0.002635)
MD=40: 0.996479 (std: 0.003857)
MD=50: 0.995775 (std: 0.005175)

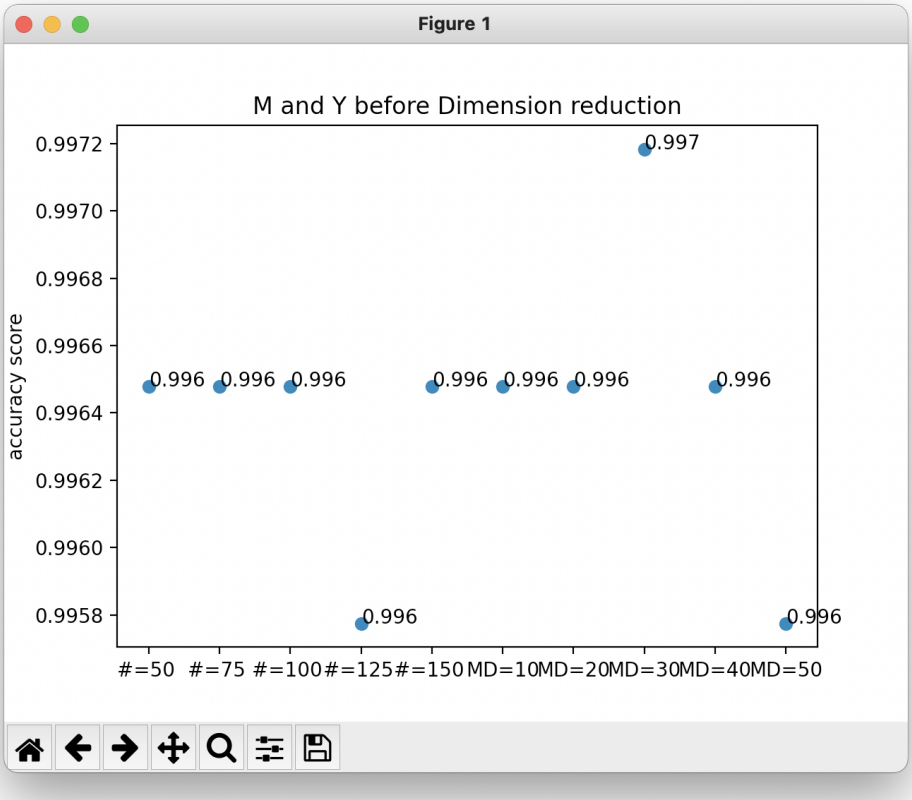

Figure 14: Random Forest: M and Y before Dimension reduction

2.3.3 U and V before Dimension Reduction

number of trees = 50 : 0.996474 (std: 0.002235)
number of trees = 75 : 0.997178 (std: 0.002643)
number of trees = 100 : 0.997882 (std: 0.002826)
number of trees = 125 : 0.996474 (std: 0.003155)
number of trees = 150 : 0.997882 (std: 0.002826)
MD=10: 0.996474 (std: 0.003155)
MD=20: 0.997178 (std: 0.002643)
MD=30: 0.996474 (std: 0.002235)
MD=40: 0.996474 (std: 0.002235)
MD=50: 0.996474 (std: 0.003155)

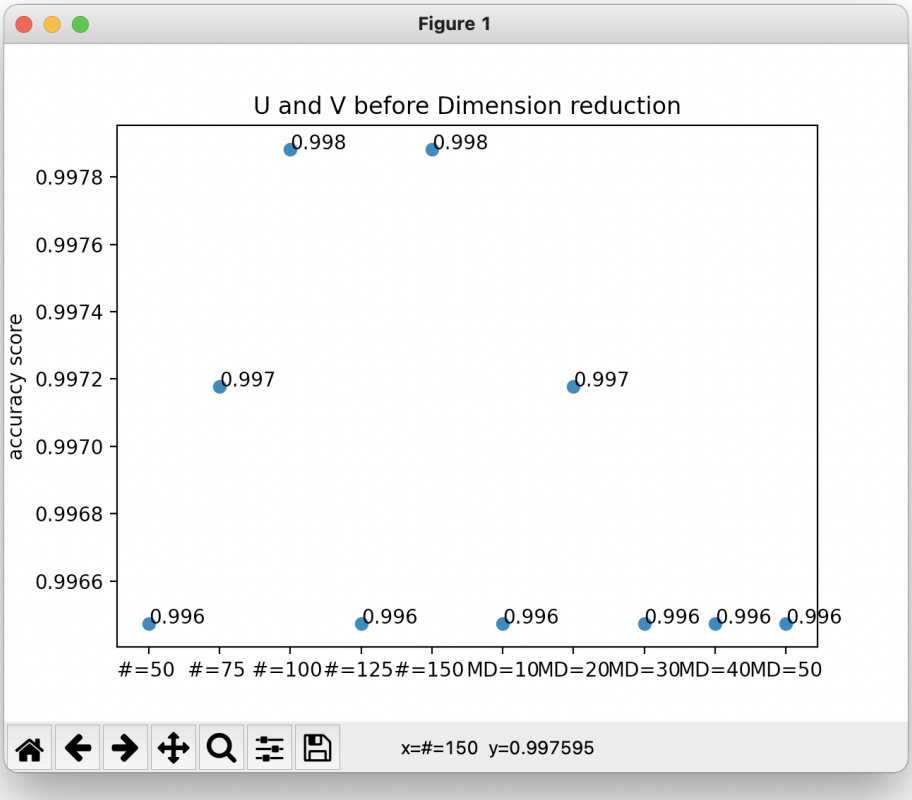

Figure 15: Random Forest: U and V before Dimension reduction

2.3.4 Dimension Reduction

For each classifier, several method of dimension reduction are used. The first method used is **Simple Quality Filtering**, algorithm will calculate the low variance for each feature then drop the low variance features.
For **Filter Methods**, the Pearson's Correlation will be used to measures the linear correlation between two variables, which are both continuous. It varies from -1 to +1, where +1 corresponds to positive linear correlation, 0 to no linear correlation, and 1 to negative linear correlation. Then drop the strongly correlate features.
The third method used to dimension reduction is **Embedded Methods**. I used the SelectKBest function from python package feature selection in sklearn.(Same as before)

2.3.5 H and K after Dimension Reduction

number of trees = 50 : 0.907170 (std: 0.008801)
number of trees = 75 : 0.910189 (std: 0.008404)
number of trees = 100 : 0.910943 (std: 0.007770)
number of trees = 125 : 0.907925 (std: 0.009119)
number of trees = 150 : 0.907925 (std: 0.012309)
MD=10: 0.910943 (std: 0.012763)
MD=20: 0.910943 (std: 0.012075)
MD=30: 0.910943 (std: 0.014240)
MD=40: 0.907925 (std: 0.010832)
MD=50: 0.910943 (std: 0.014828)

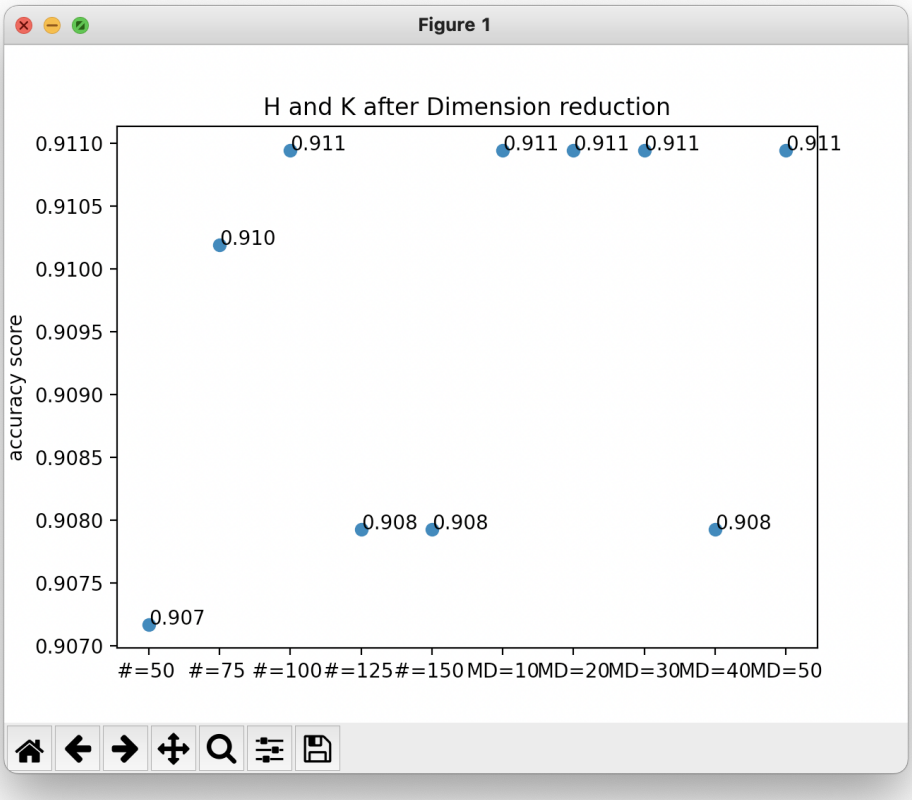

Figure 16: Random Forest: H and K after Dimension reduction

2.3.6 M and Y after Dimension Reduction

number of trees = 50 : 0.989437 (std: 0.009448)
number of trees = 75 : 0.987324 (std: 0.009604)
number of trees = 100 : 0.987324 (std: 0.008509)
number of trees = 125 : 0.986620 (std: 0.010773)
number of trees = 150 : 0.988028 (std: 0.009859)
MD=10: 0.986620 (std: 0.008739)
MD=20: 0.987324 (std: 0.008509)
MD=30: 0.986620 (std: 0.009553)
MD=40: 0.988732 (std: 0.008152)
MD=50: 0.988028 (std: 0.009859)

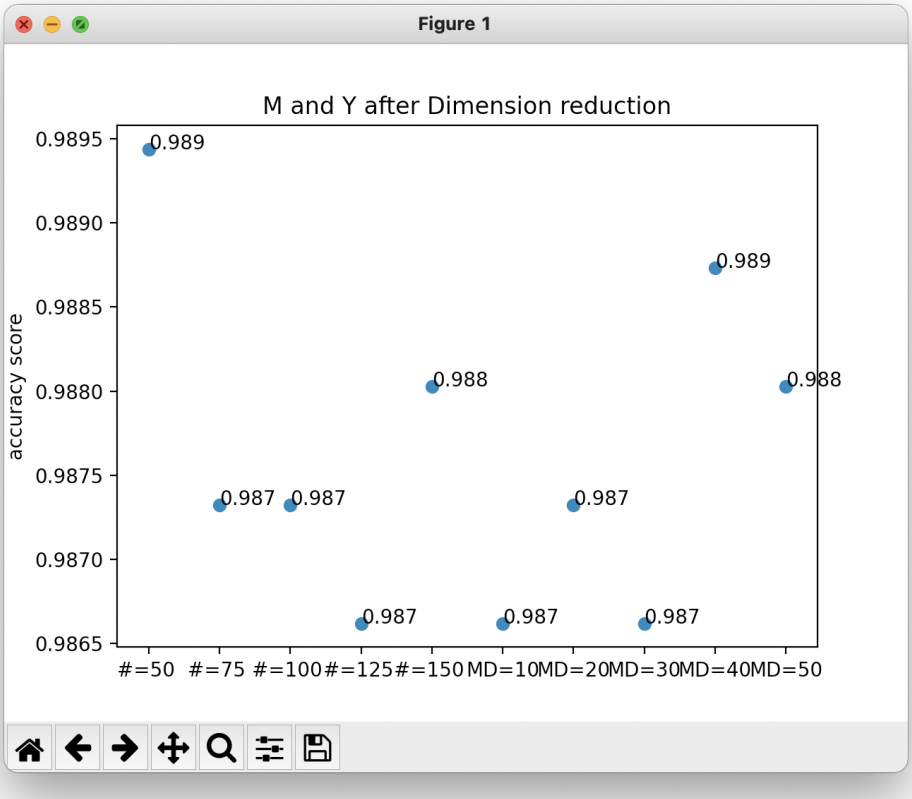

Figure 17: Random Forest: M and Y after Dimension reduction

2.3.7 U and V after Dimension Reduction

number of trees = 50 : 0.978147 (std: 0.005673)
number of trees = 75 : 0.976738 (std: 0.004270)
number of trees = 100 : 0.978149 (std: 0.003487)
number of trees = 125 : 0.973220 (std: 0.006535)
number of trees = 150 : 0.977442 (std: 0.005757)
MD=10: 0.974628 (std: 0.005644)
MD=20: 0.974625 (std: 0.006849)
MD=30: 0.976738 (std: 0.004816)
MD=40: 0.977442 (std: 0.005757)
MD=50: 0.974625 (std: 0.006849)

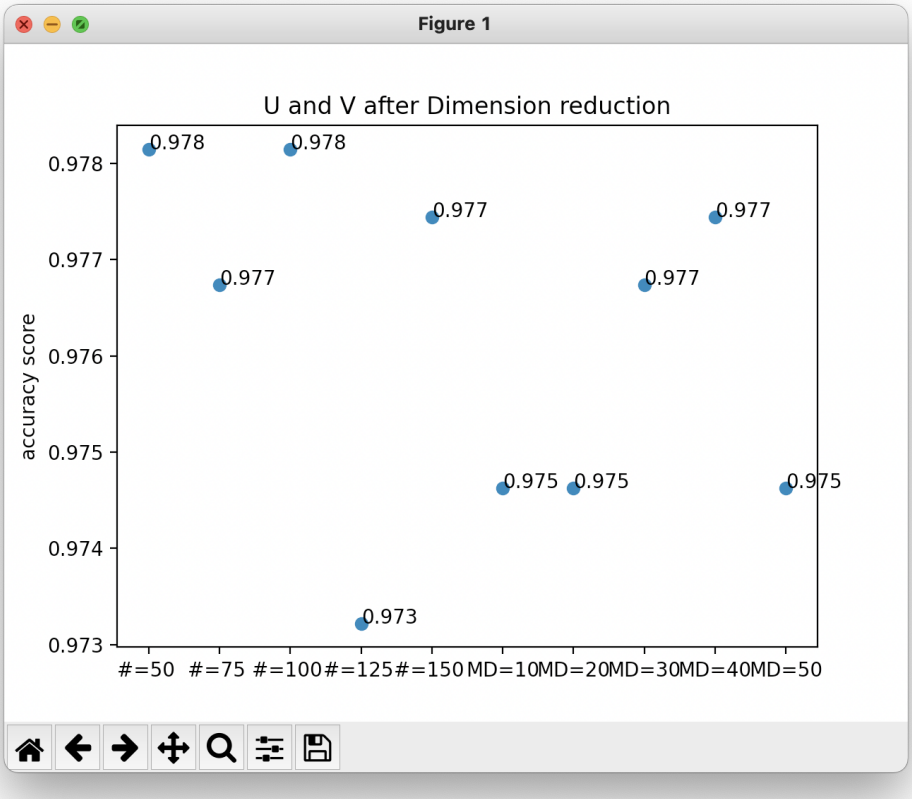

Figure 18: Random Forest: U and V after Dimension reduction

2.4 SVM

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

The advantages of support vector machines are:
- Effective in high dimensional spaces.
- Still effective in cases where number of dimensions is greater than the number of samples.
- Uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.
- Versatile: different Kernel functions can be specified for the decision function. Common - kernels are provided, but it is also possible to specify custom kernels.

The disadvantages of support vector machines include:
- If the number of features is much greater than the number of samples, avoid over-fitting in choosing Kernel functions and regularization term is crucial.
- SVMs do not directly provide probability estimates, these are calculated using an expensive five-fold cross-validation (see Scores and probabilities, below).
(Support Vector Machines, https://scikit-learn.org/)

For additional hyperparameters tuned, the C was changed from 0.6 to 1.4 and model used linear, poly, rbf and sigmoid in kernel.

2.4.1 H and K before Dimension Reduction

C=0.6 : 0.970566 (std: 0.006492)
C=0.8 : 0.972075 (std: 0.006999)
C=1.0 : 0.974340 (std: 0.006492)
C=1.2 : 0.974340 (std: 0.005546)
C=1.4 : 0.975094 (std: 0.006131)
linear: 0.927547 (std: 0.016604)
poly: 0.960000 (std: 0.012763)
rbf: 0.974340 (std: 0.006492)
sigmoid: 0.501132 (std: 0.001509)

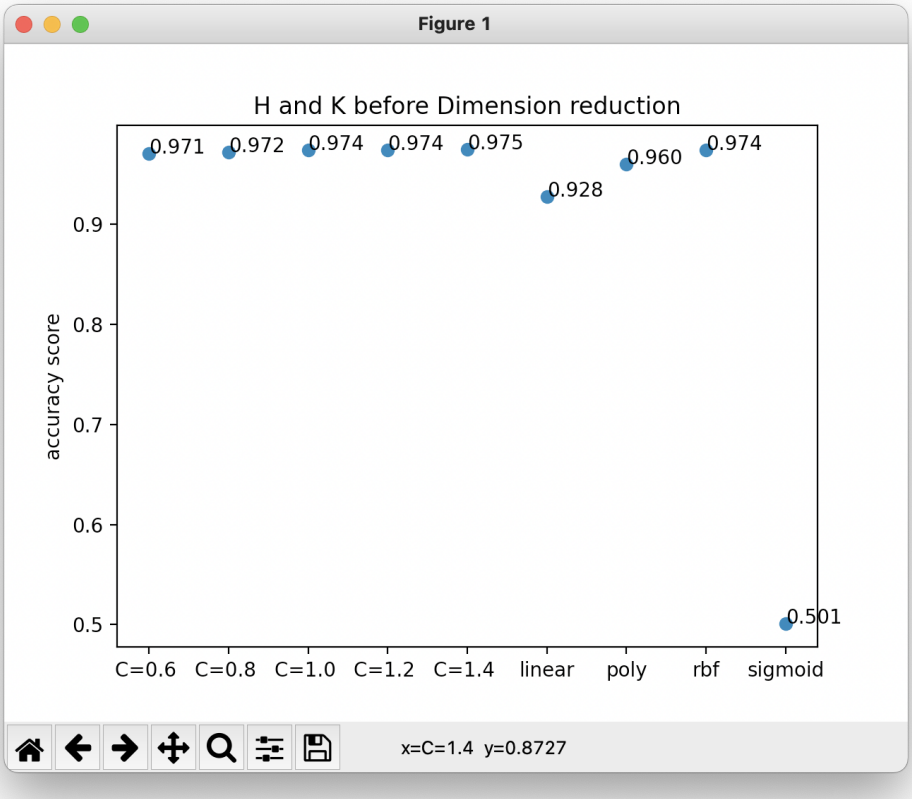

Figure 19: SVM: H and K before Dimension reduction

2.4.2 M and Y before Dimension Reduction

C=0.6 : 0.997887 (std: 0.002817)
C=0.8 : 0.998592 (std: 0.001725)
C=1.0 : 0.999296 (std: 0.001408)
C=1.2 : 0.999296 (std: 0.001408)
C=1.4 : 0.999296 (std: 0.001408)
linear: 0.996479 (std: 0.005455)
poly: 0.997887 (std: 0.002817)
rbf: 0.999296 (std: 0.001408)
sigmoid: 0.502817 (std: 0.001408)

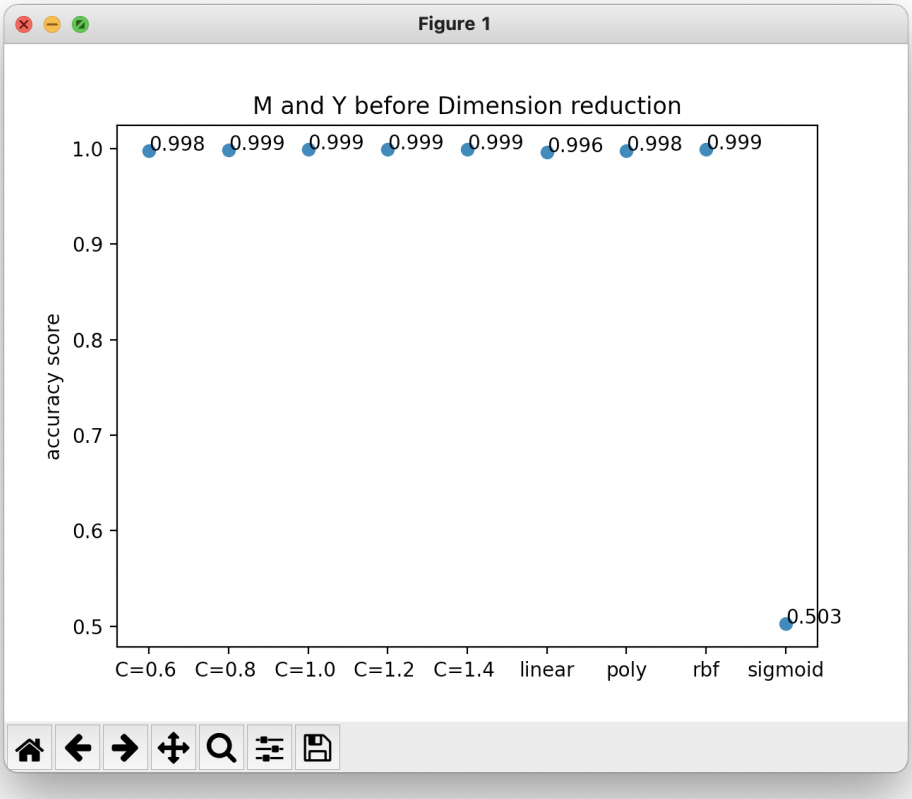

Figure 20: SVM: M and Y before Dimension reduction

2.4.3 U and V before Dimension Reduction

C=0.6 : 0.997885 (std: 0.001727)
C=0.8 : 0.999293 (std: 0.001413)
C=1.0 : 0.999293 (std: 0.001413)
C=1.2 : 0.999293 (std: 0.001413)
C=1.4 : 0.999293 (std: 0.001413)
linear: 0.995068 (std: 0.001723)
poly: 0.998589 (std: 0.001728)
rbf: 0.999293 (std: 0.001413)
sigmoid: 0.514448 (std: 0.000727)

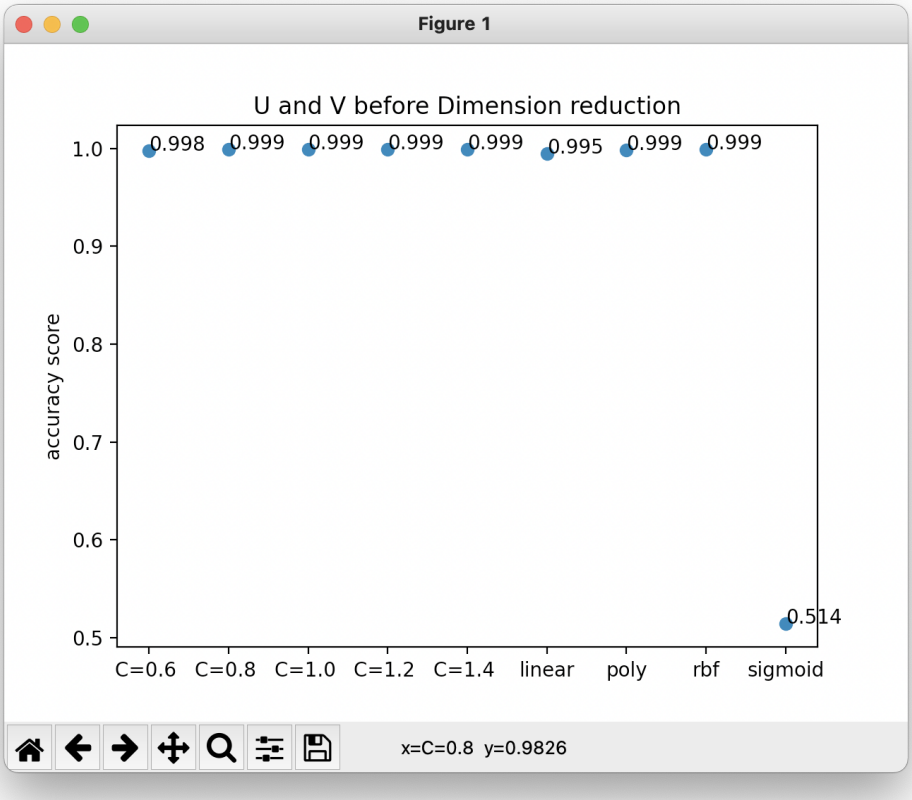

Figure 21: SVM: U and V before Dimension reduction

2.4.4 Dimension Reduction

For each classifier, several method of dimension reduction are used. The first method used is **Simple Quality Filtering**, algorithm will calculate the low variance for each feature then drop the low variance features.
For **Filter Methods**, the Pearson's Correlation will be used to measures the linear correlation between two variables, which are both continuous. It varies from -1 to +1, where +1 corresponds to positive linear correlation, 0 to no linear correlation, and 1 to negative linear correlation. Then drop the strongly correlate features.
The third method used to dimension reduction is **Embedded Methods**. I used the SelectKBest function from python package feature selection in sklearn. (Same as before)

2.4.5 H and K after Dimension Reduction

C=0.6 : 0.901887 (std: 0.019681)
C=0.8 : 0.901887 (std: 0.020806)
C=1.0 : 0.901132 (std: 0.022184)
C=1.2 : 0.900377 (std: 0.021427)
C=1.4 : 0.900377 (std: 0.015939)
linear: 0.855094 (std: 0.020195)
poly: 0.882264 (std: 0.017111)
rbf: 0.901132 (std: 0.022184)
sigmoid: 0.503396 (std: 0.001849)

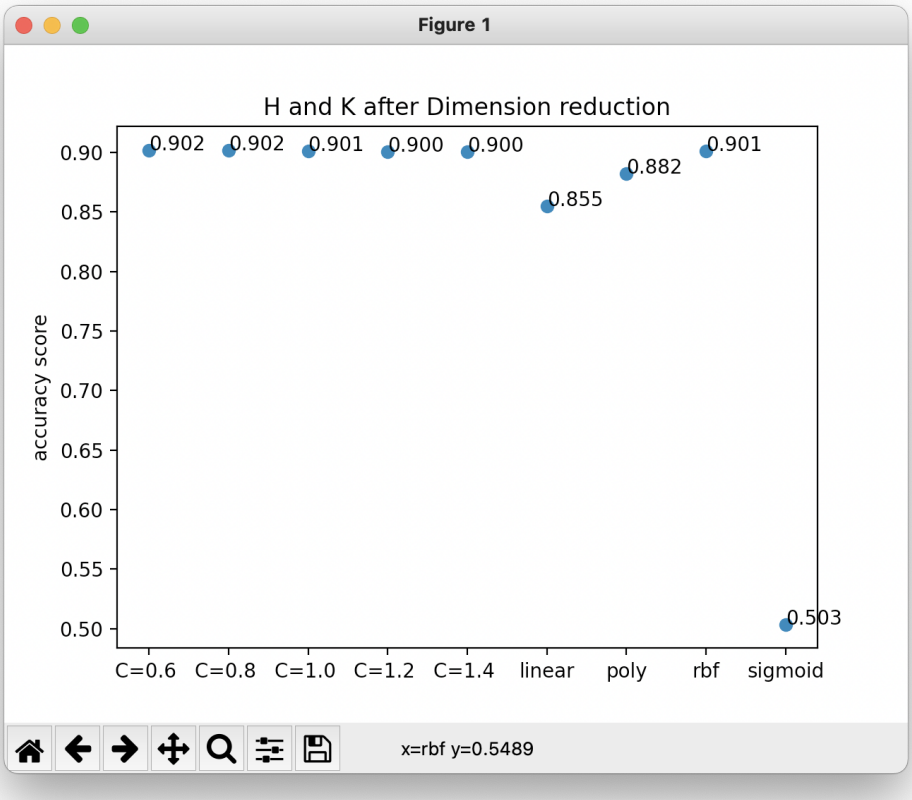

Figure 22: SVM: H and K after Dimension reduction

2.4.6 M and Y after Dimension Reduction

C=0.6 : 0.988732 (std: 0.004106)
C=0.8 : 0.988732 (std: 0.004106)
C=1.0 : 0.989437 (std: 0.003857)
C=1.2 : 0.990141 (std: 0.004106)
C=1.4 : 0.990141 (std: 0.004106)
linear: 0.979577 (std: 0.006454)
poly: 0.988028 (std: 0.003591)
rbf: 0.989437 (std: 0.003857)
sigmoid: 0.501408 (std: 0.002817)

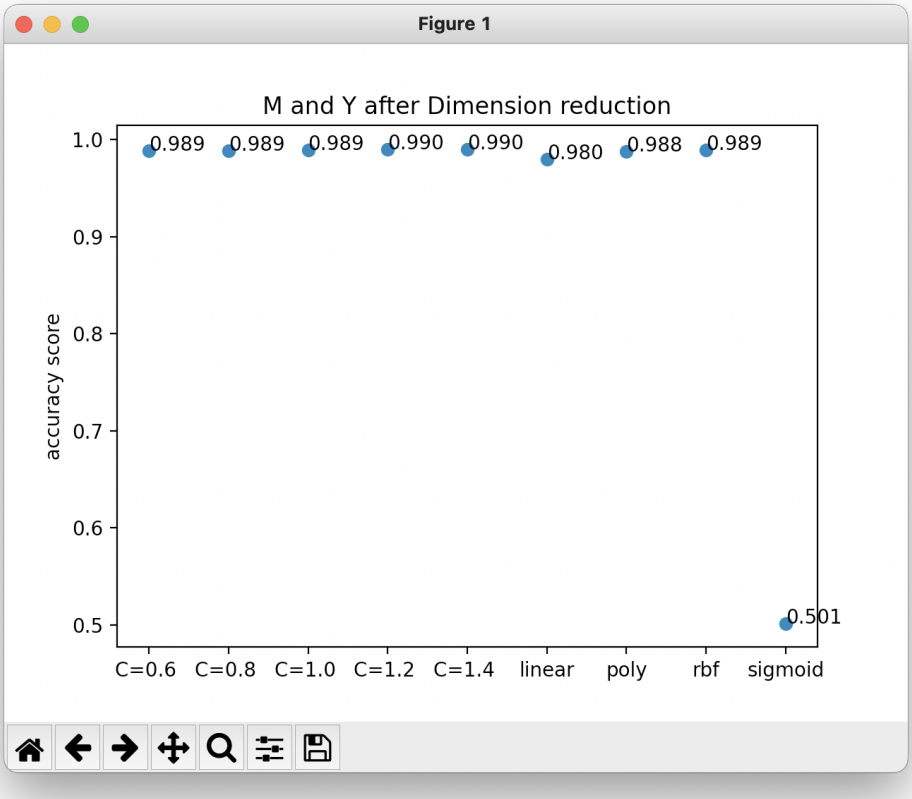

Figure 23: SVM: M and Y after Dimension reduction

2.4.7 U and V after Dimension Reduction

C=0.6 : 0.975332 (std: 0.003876)
C=0.8 : 0.974625 (std: 0.004702)
C=1.0 : 0.979568 (std: 0.005163)
C=1.2 : 0.978863 (std: 0.004438)
C=1.4 : 0.978159 (std: 0.005160)
linear: 0.954183 (std: 0.011188)
poly: 0.966162 (std: 0.009400)
rbf: 0.979568 (std: 0.005163)
sigmoid: 0.516560 (std: 0.001403)

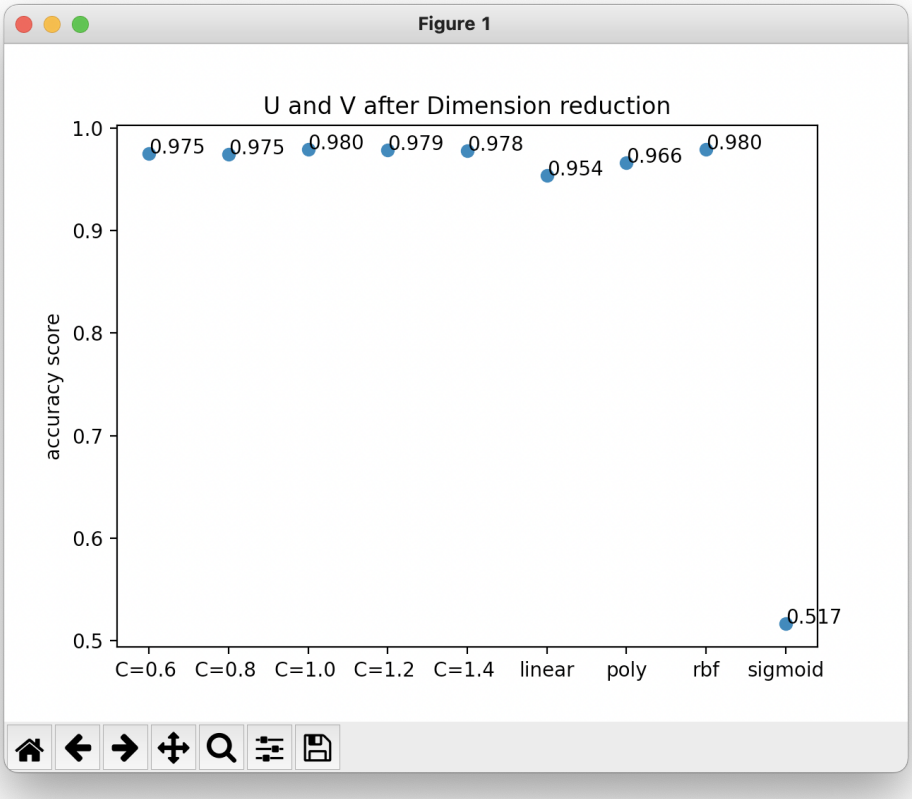

Figure 24: SVM: U and V after Dimension reduction

2.5 Artificial Neural Network

Artificial neural networks are the modeling of the human brain with the simplest definition and building blocks are neurons.

Advantages of Artificial Neural Network
- Storing information on the entire network
- Ability to work with incomplete knowledge
- Having fault tolerance
- Having a distributed memory
- Gradual corruption
- Ability to make machine learning
- Parallel processing capability

Disadvantages of Artificial Neural Network
- Hardware dependence
- Unexplained behavior of the network
- Determination of proper network structure
- Difficulty of showing the problem to the network
- The duration of the network is unknown

For additional hyperparameters tuned, the hidden layer sizes was changed from 100 to 500 and changed activation to identity, logistic, logistic and relu.

2.5.1 H and K before Dimension Reduction

HL=100 : 0.972830 (std: 0.007317)
HL=200 : 0.970566 (std: 0.016257)
HL=300 : 0.971321 (std: 0.008801)
HL=400 : 0.974340 (std: 0.009057)
HL=500 : 0.972075 (std: 0.008801)
identity: 0.920755 (std: 0.022767)
logistic: 0.973585 (std: 0.006750)
tanh: 0.978868 (std: 0.008129)
relu: 0.975849 (std: 0.009724)

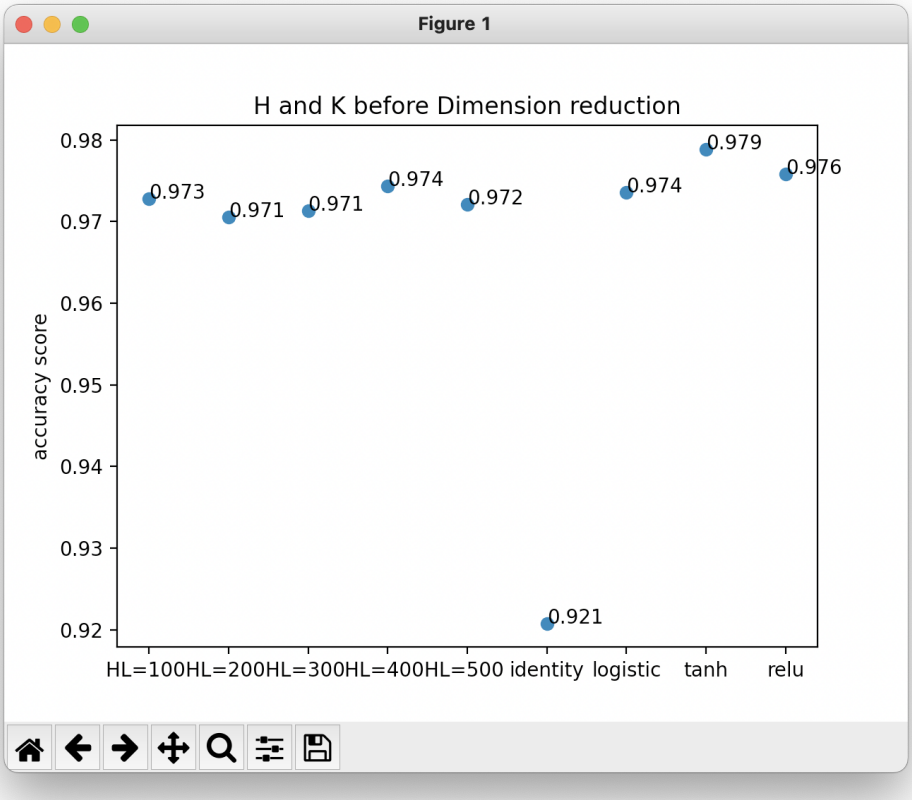

Figure 25: NN: H and K before Dimension reduction

2.5.2 M and Y before Dimension Reduction

HL=100 : 0.998592 (std: 0.001725)
HL=200 : 0.997887 (std: 0.002817)
HL=300 : 0.998592 (std: 0.001725)
HL=400 : 0.998592 (std: 0.001725)
HL=500 : 0.997887 (std: 0.001725)
identity: 0.997183 (std: 0.001408)
logistic: 0.996479 (std: 0.002227)
tanh: 0.997887 (std: 0.001725)
relu: 0.997183 (std: 0.002635)

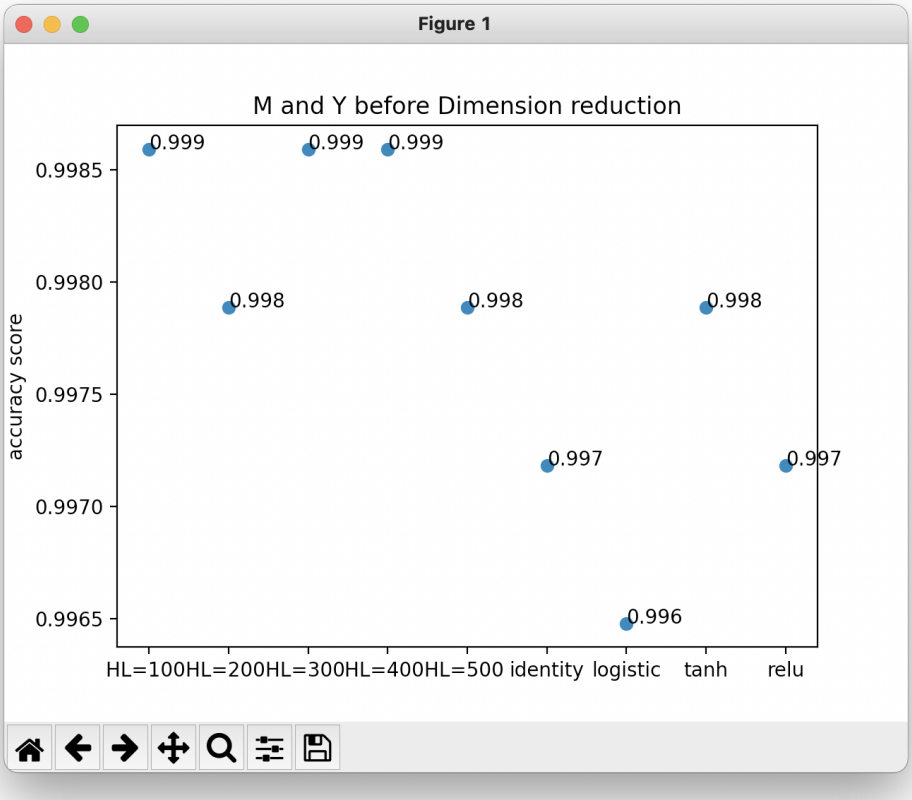

Figure 26: NN: M and Y before Dimension reduction

2.5.3 U and V before Dimension Reduction

HL=100 : 0.997183 (std: 0.004106)
HL=200 : 0.997183 (std: 0.002635)
HL=300 : 0.997887 (std: 0.002817)
HL=400 : 0.997183 (std: 0.004106)
HL=500 : 0.997887 (std: 0.002817)
identity: 0.990141 (std: 0.006058)
logistic: 0.994366 (std: 0.004225)
tanh: 0.997887 (std: 0.002817)
relu: 0.997183 (std: 0.004106)

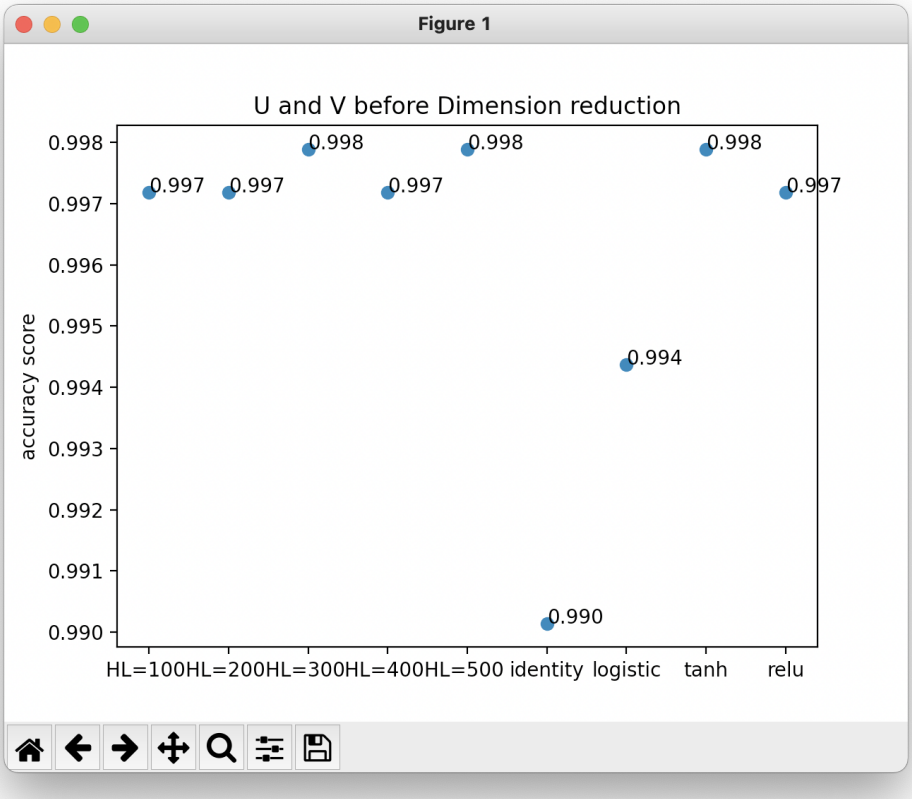

Figure 27: NN: U and V before Dimension reduction

2.5.4 Dimension Reduction

For each classifier, several method of dimension reduction are used. The first method used is **Simple Quality Filtering**, algorithm will calculate the low variance for each feature then drop the low variance features.
For **Filter Methods**, the Pearson's Correlation will be used to measures the linear correlation between two variables, which are both continuous. It varies from -1 to +1, where +1 corresponds to positive linear correlation, 0 to no linear correlation, and 1 to negative linear correlation. Then drop the strongly correlate features.
The third method used to dimension reduction is **Embedded Methods**. I used the SelectKBest function from python package feature selection in sklearn. (Same as before)

2.5.5 H and K after Dimension Reduction

HL=100 : 0.895849 (std: 0.027379)
HL=200 : 0.888302 (std: 0.021952)
HL=300 : 0.891321 (std: 0.025303)
HL=400 : 0.890566 (std: 0.025925)
HL=500 : 0.888302 (std: 0.022966)
identity: 0.849057 (std: 0.016187)
logistic: 0.899623 (std: 0.020335)
tanh: 0.900377 (std: 0.024292)
relu: 0.893585 (std: 0.021664)

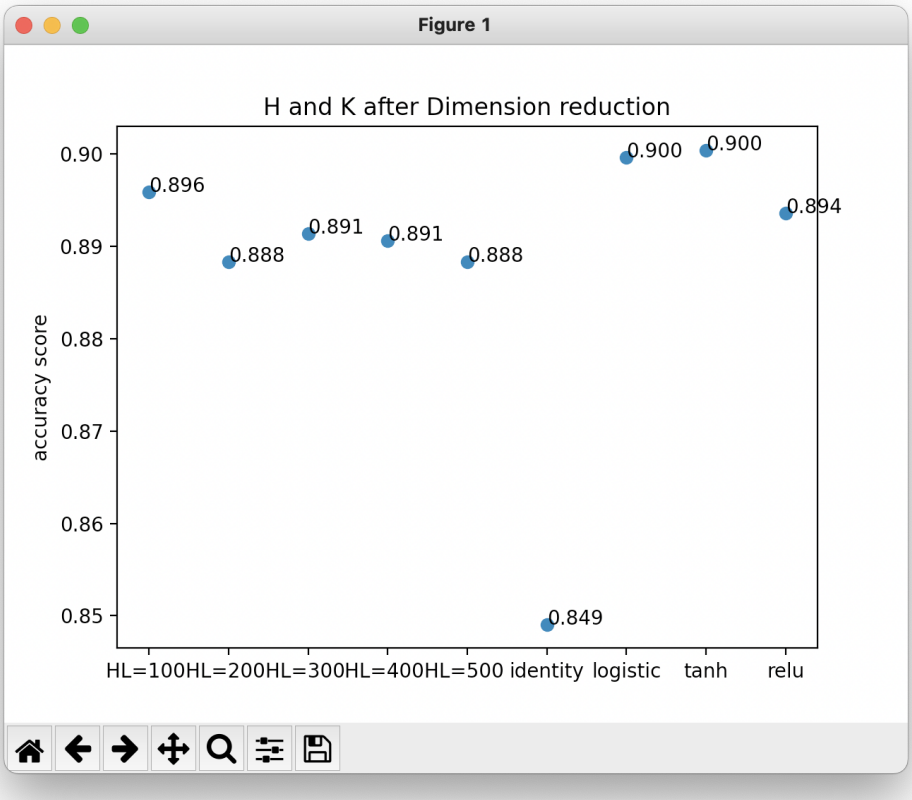

Figure 28: NN: H and K after Dimension reduction

2.5.6 M and Y after Dimension Reduction

HL=100 : 0.988732 (std: 0.004106)
HL=200 : 0.988732 (std: 0.004106)
HL=300 : 0.989437 (std: 0.003149)
HL=400 : 0.990845 (std: 0.002817)
HL=500 : 0.987324 (std: 0.004776)
identity: 0.974648 (std: 0.009553)
logistic: 0.975352 (std: 0.010205)
tanh: 0.990845 (std: 0.004225)
relu: 0.988732 (std: 0.004106)

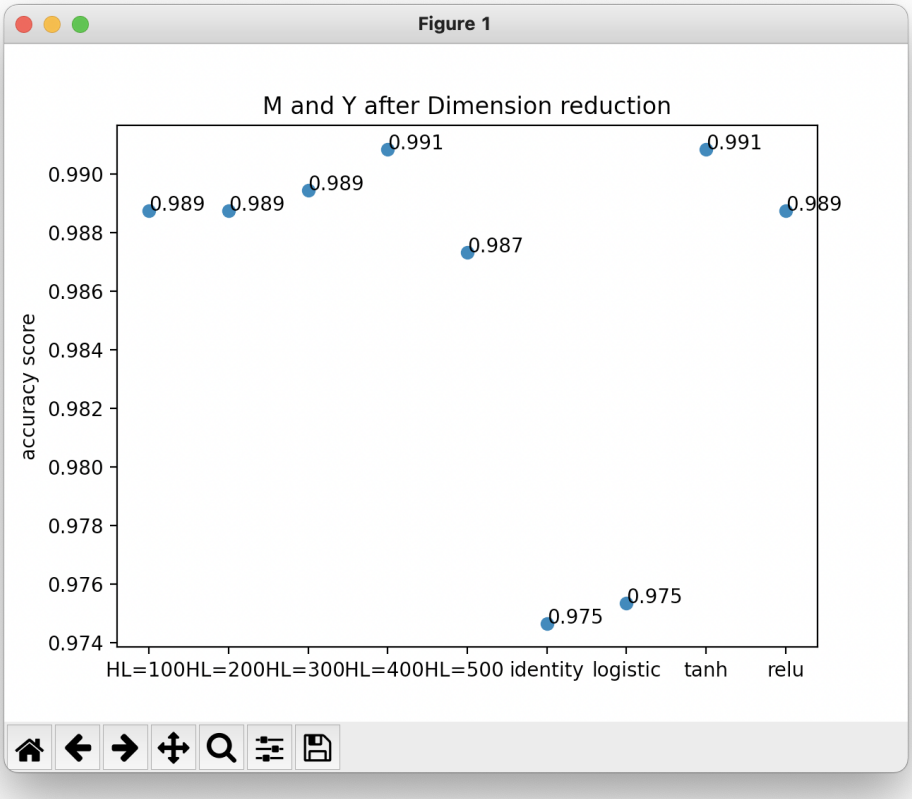

Figure 29: NN: M and Y after Dimension reduction

2.5.7 U and V after Dimension Reduction

HL=100 : 0.959140 (std: 0.019219)
HL=200 : 0.959844 (std: 0.019476)
HL=300 : 0.957030 (std: 0.019828)
HL=400 : 0.960551 (std: 0.023387)
HL=500 : 0.957729 (std: 0.019011)
identity: 0.952088 (std: 0.017318)
logistic: 0.952088 (std: 0.017318)
tanh: 0.950679 (std: 0.017373)
relu: 0.953501 (std: 0.017429)

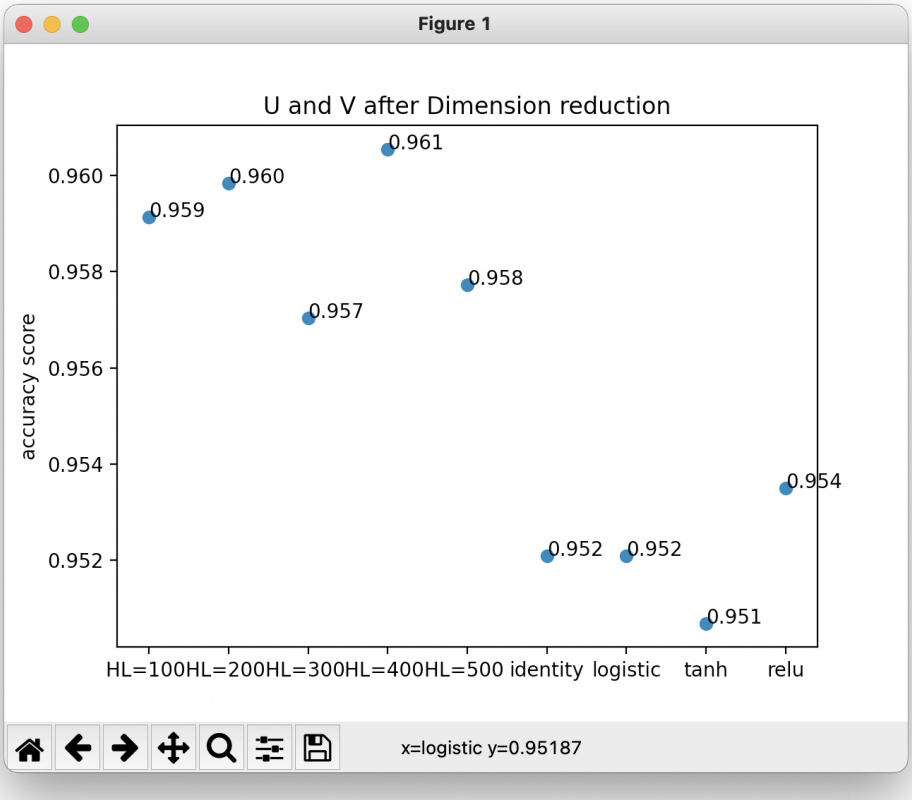

Figure 30: NN: U and V after Dimension reduction

2.6 Extra Trees Classifier

Extremely Randomized Trees Classifier(Extra Trees Classifier) is a type of ensemble learning technique which aggregates the results of multiple de-correlated decision trees collected in a "forest" to output it's classification result. In concept, it is very similar to a Random Forest Classifier and only differs from it in the manner of construction of the decision trees in the forest.

The advantage and disadvantage of Extra Tree Classifier is similar to Random Forest algorithms. The result of this two model is same, however the Extra Trees Classifier is much faster.

For additional hyperparameters tuned, the number of tree was changed from 50 to 150.

2.6.1 H and K before Dimension reduction:

number of trees=50: 0.978113 (std: 0.005546)
number of trees=75: 0.979623 (std: 0.006579)
number of trees=100: 0.981132 (std: 0.007916)
number of trees=125: 0.976604 (std: 0.004401)
number of trees=150: 0.976604 (std: 0.002824)

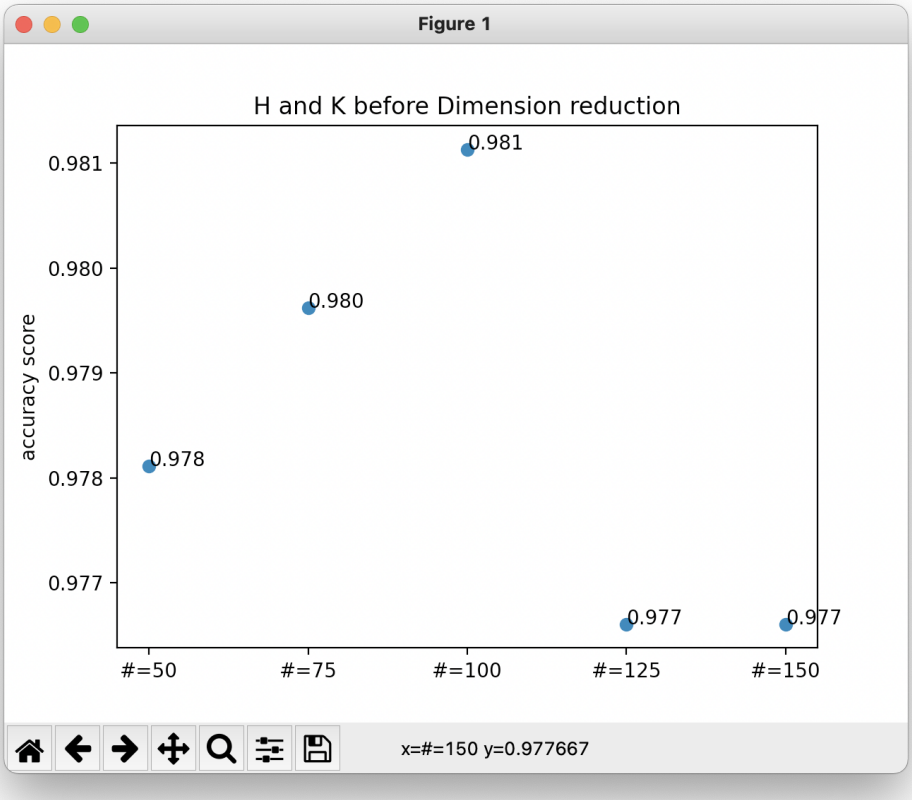

Figure 31: Extra Trees Classifier: H and K before Dimension reduction

2.6.2 M and Y before Dimension reduction:

number of trees=50: 0.997183 (std: 0.003450)
number of trees=75: 0.997183 (std: 0.003450)
number of trees=100: 0.996479 (std: 0.003149)
number of trees=125: 0.997183 (std: 0.003450)
number of trees=150: 0.997183 (std: 0.003450)

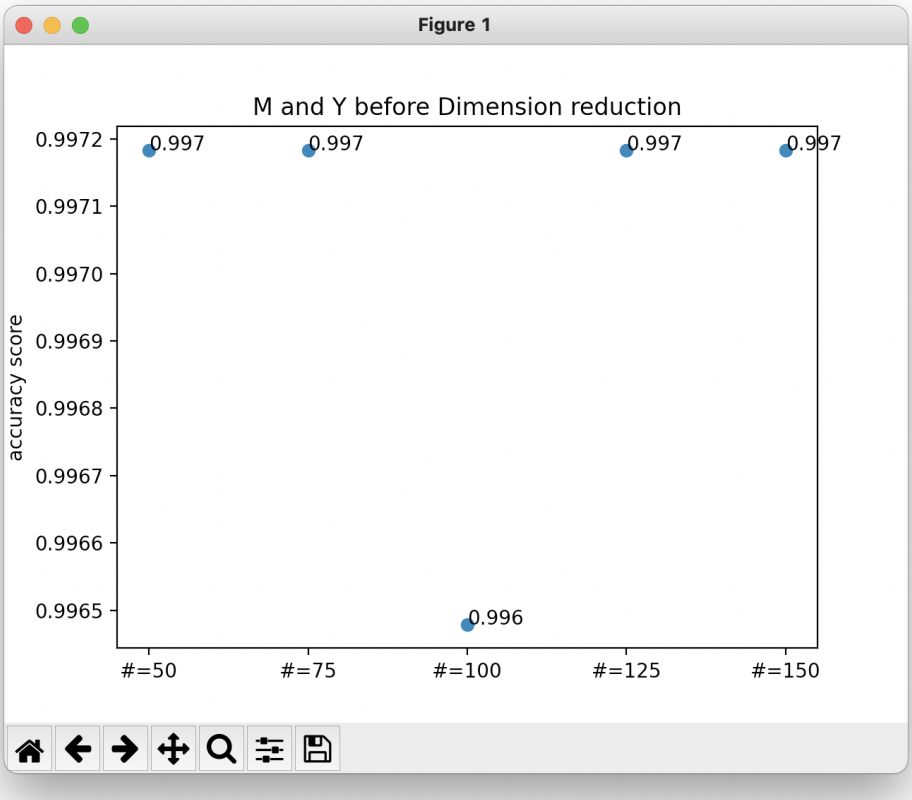

Figure 32: Extra Trees Classifier: M and Y before Dimension reduction

2.6.3 U and V before Dimension reduction:

number of trees=50: 0.998592 (std: 0.001725)
number of trees=75: 0.999296 (std: 0.001408)
number of trees=100: 1.000000 (std: 0.000000)
number of trees=125: 1.000000 (std: 0.000000)
number of trees=150: 0.999296 (std: 0.001408)

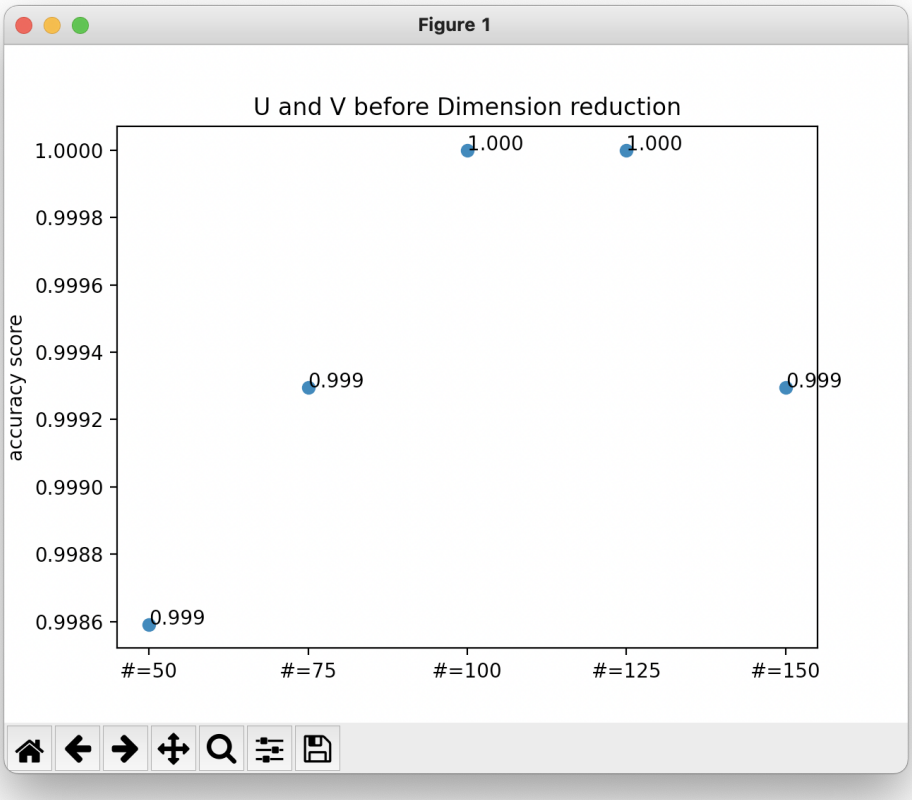

Figure 33: Extra Trees Classifier: U and V before Dimension reduction

2.6.4 Dimension Reduction

For each classifier, several method of dimension reduction are used. The first method used is **Simple Quality Filtering**, algorithm will calculate the low variances for each feature then drop the low variance features.
For **Filter Methods**, the Pearson's Correlation will be used to measures the linear correlation between two variables, which are both continuous. It varies from -1 to +1, where +1 corresponds to positive linear correlation, 0 to no linear correlation, and 1 to negative linear correlation. Then drop the strongly correlate features.
The third method used to dimension reduction is **Embedded Methods**. I used the SelectKBest function from python package feature selection in sklearn. (Same as before)

2.6.5 H and K after Dimension reduction:

number of trees=50: 0.907170 (std: 0.014828)
number of trees=75: 0.907925 (std: 0.012763)
number of trees=100: 0.908679 (std: 0.015170)
number of trees=125: 0.910943 (std: 0.014828)
number of trees=150: 0.908679 (std: 0.013159)

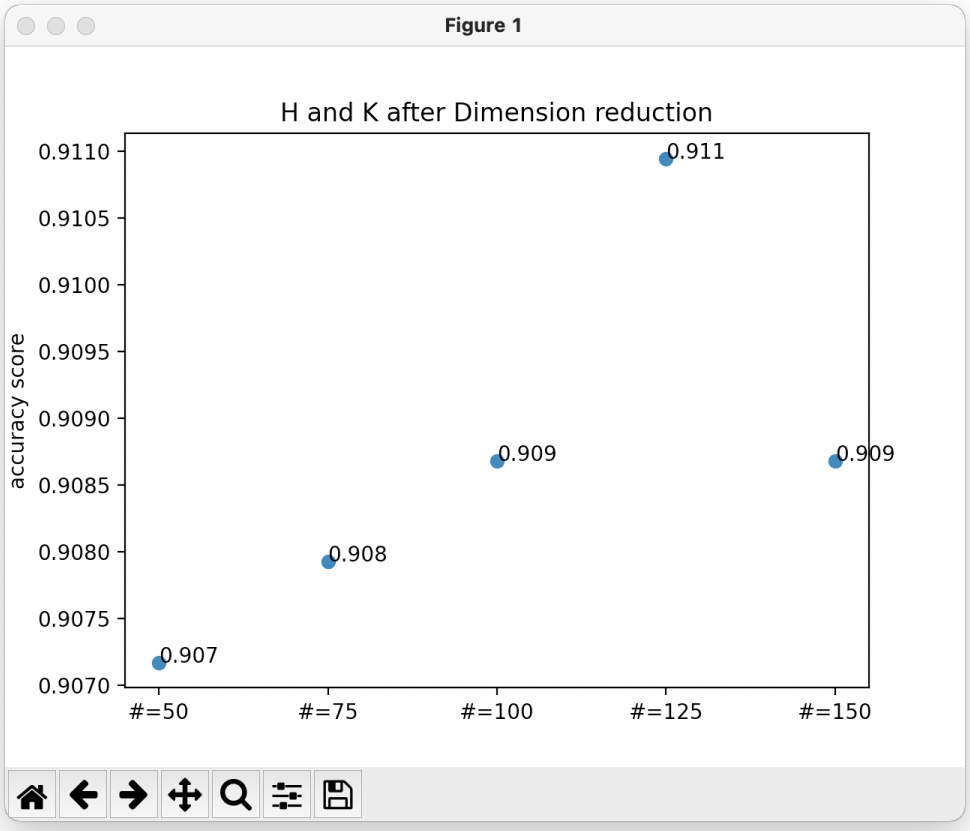

Figure 34: Extra Trees Classifier: H and K after Dimension reduction

2.6.6 M and Y after Dimension reduction:

number of trees=50: 0.990845 (std: 0.007585)
number of trees=75: 0.992254 (std: 0.005175)
number of trees=100: 0.992254 (std: 0.004106)
number of trees=125: 0.990845 (std: 0.006531)
number of trees=150: 0.990845 (std: 0.003591)

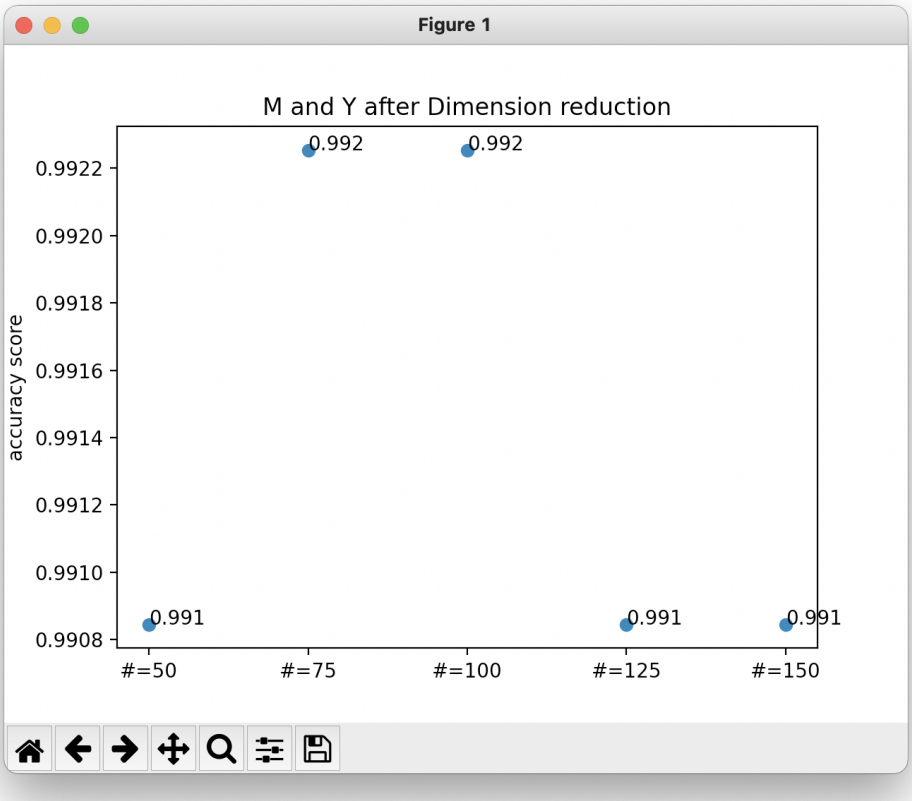

Figure 35: Extra Trees Classifier: M and Y after Dimension reduction

2.6.7 U and V after Dimension reduction:

number of trees=50: 0.978856 (std: 0.008340)
number of trees=75: 0.978856 (std: 0.008340)
number of trees=100: 0.980264 (std: 0.007262)
number of trees=125: 0.978856 (std: 0.008340)
number of trees=150: 0.978856 (std: 0.008340)

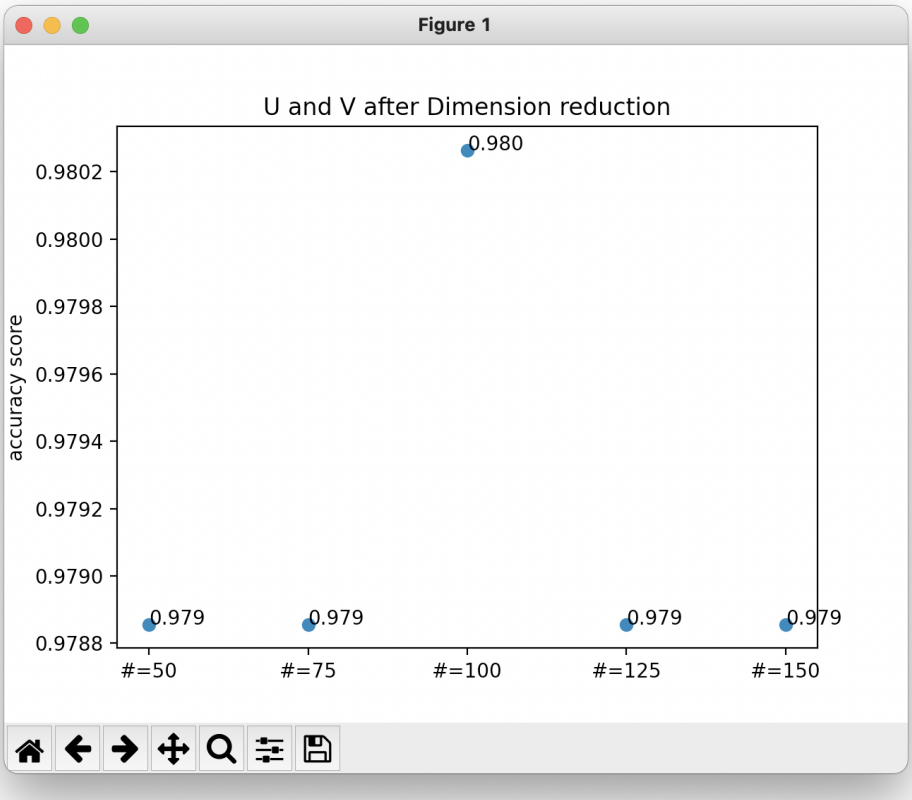

Figure 36: Extra Trees Classifier: U and V after Dimension reduction

2.7 Quadratic Discriminant Analysis

QDA is a variant of LDA in which an individual covariance matrix is estimated for every class of observations. QDA is particularly useful if there is prior knowledge that individual classes exhibit distinct covariances. A disadvantage of QDA is that it cannot be used as a dimensionality reduction technique.

For additional hyperparameters tuned, the regularizes the per-class covariance estimates changed from 0.0 to 0.8.

2.7.1 H and K before Dimension reduction:

RP=0.0: 0.922264 (std: 0.009119)
RP=0.2: 0.916226 (std: 0.006492)
RP=0.4: 0.904151 (std: 0.003019)
RP=0.6: 0.896604 (std: 0.008472)
RP=0.8: 0.882264 (std: 0.009955)

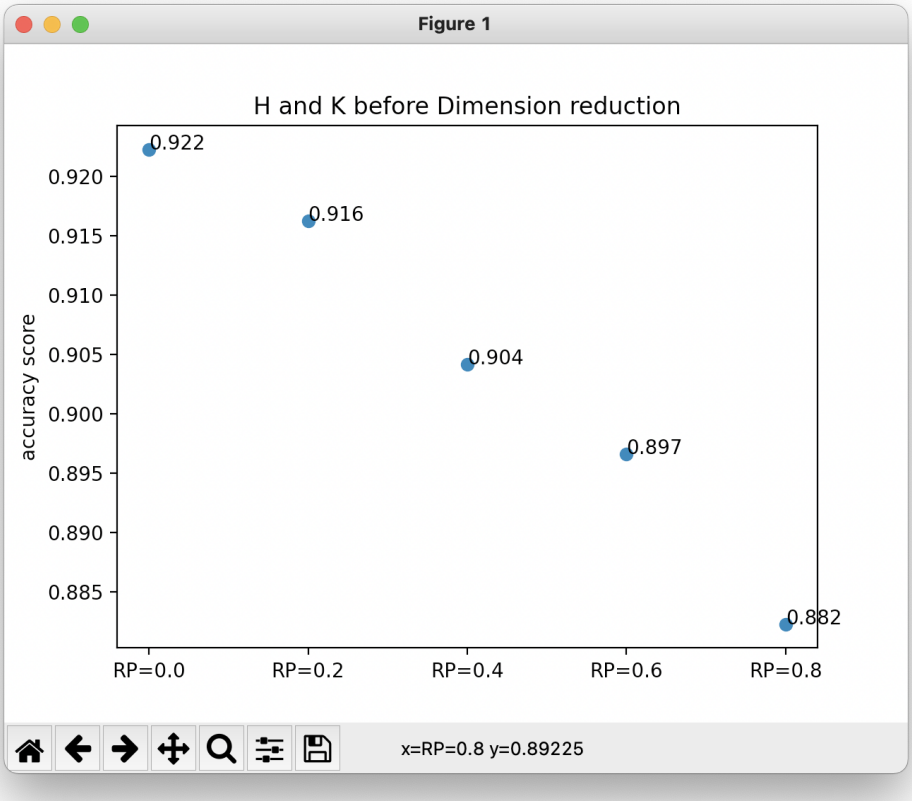

Figure 37: Quadratic Discriminant Analysis: H and K before Dimension reduction

2.7.2 M and Y before Dimension reduction:

RP=0.0: 0.996479 (std: 0.003149)
RP=0.2: 0.996479 (std: 0.003149)
RP=0.4: 0.995775 (std: 0.002635)
RP=0.6: 0.992254 (std: 0.004671)
RP=0.8: 0.989437 (std: 0.003149)

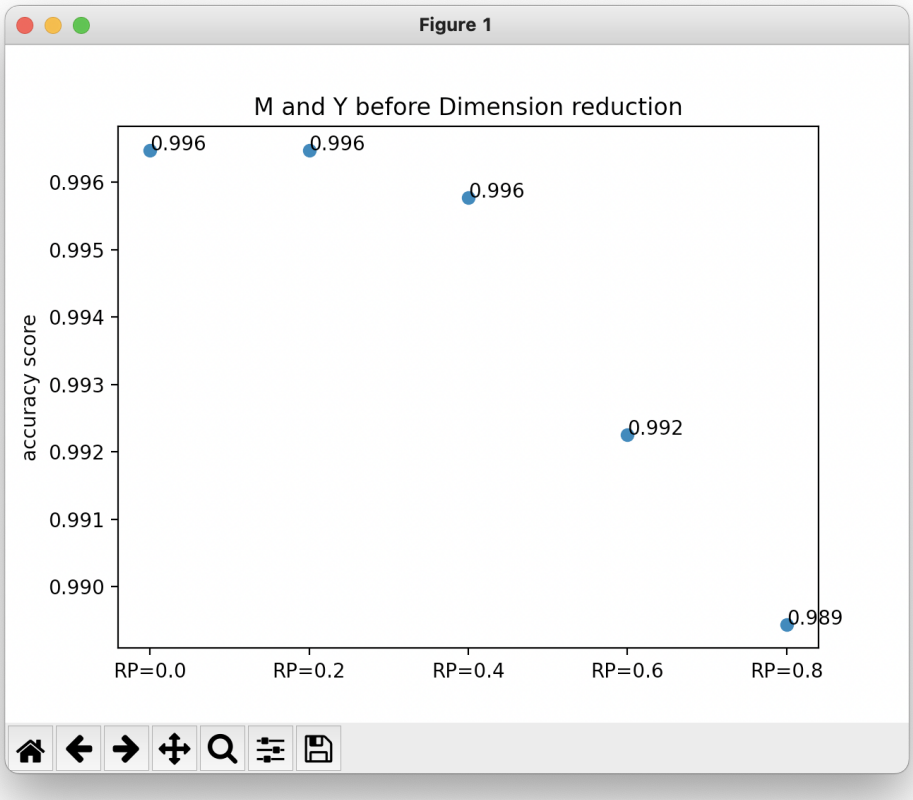

Figure 38: Quadratic Discriminant Analysis: M and Y before Dimension reduction

2.7.3 U and V before Dimension reduction:

RP=0.0: 0.992950 (std: 0.003864)
RP=0.2: 0.990131 (std: 0.004117)
RP=0.4: 0.986602 (std: 0.006485)
RP=0.6: 0.985898 (std: 0.005923)
RP=0.8: 0.978853 (std: 0.005916)

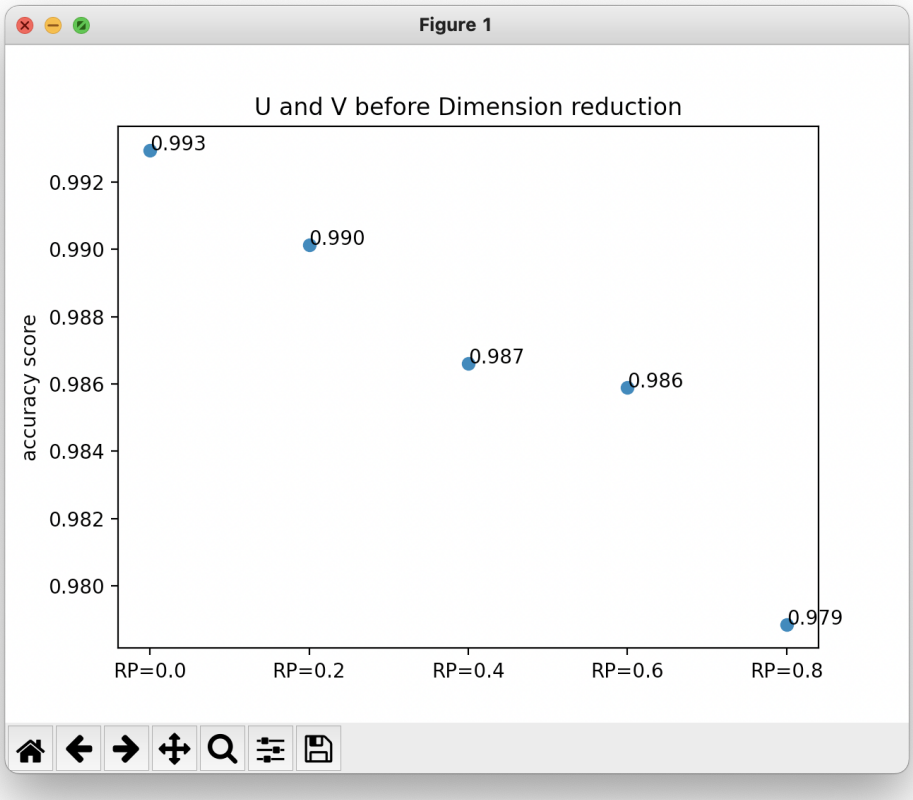

Figure 39: Quadratic Discriminant Analysis: U and V before Dimension reduction

2.7.4 Dimension Reduction

For each classifier, several method of dimension reduction are used. The first method used is **Simple Quality Filtering**, algorithm will calculate the low variance for each feature then drop the low variance features.
For **Filter Methods**, the Pearson's Correlation will be used to measures the linear correlation between two variables, which are both continuous. It varies from -1 to +1, where +1 corresponds to positive linear correlation, 0 to no linear correlation, and 1 to negative linear correlation. Then drop the strongly correlate features.
The third method used to dimension reduction is **Embedded Methods**. I used the SelectKBest function from python package feature selection in sklearn. (Same as before)

2.7.5 H and K after Dimension reduction:

RP=0.0: 0.852075 (std: 0.021533)
RP=0.2: 0.852830 (std: 0.016876)
RP=0.4: 0.849811 (std: 0.015723)
RP=0.6: 0.841509 (std: 0.022767)
RP=0.8: 0.828679 (std: 0.019767)

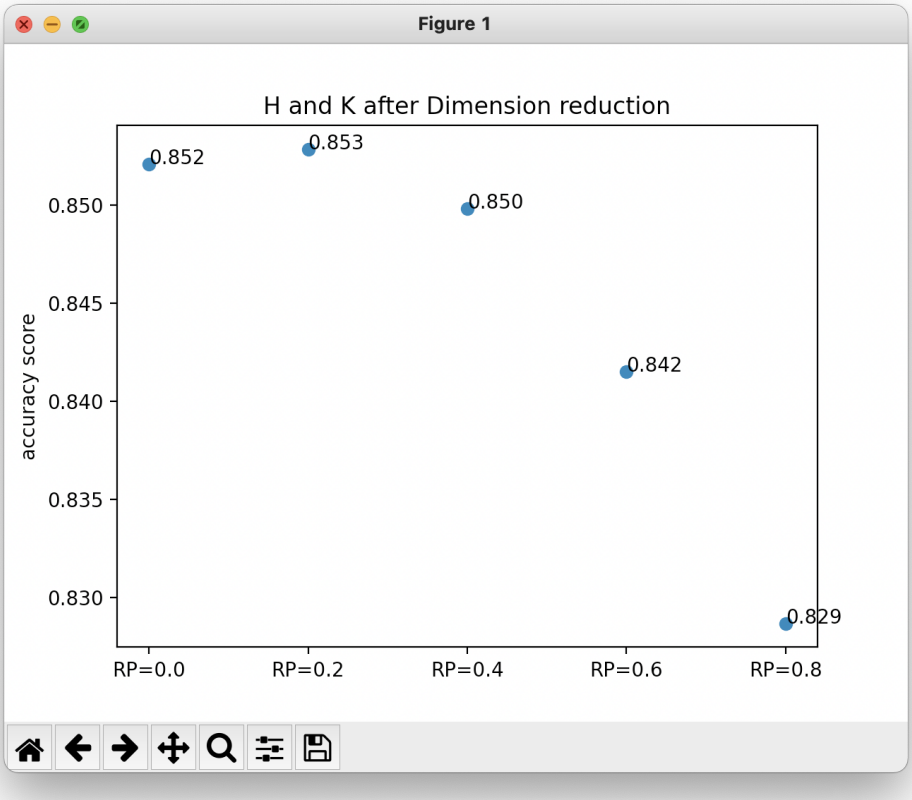

Figure 40: Quadratic Discriminant Analysis: H and K after Dimension reduction

2.7.6 M and Y after Dimension reduction:

RP=0.0: 0.979577 (std: 0.007182)
RP=0.2: 0.981690 (std: 0.010302)
RP=0.4: 0.984507 (std: 0.005721)
RP=0.6: 0.983099 (std: 0.004106)
RP=0.8: 0.978873 (std: 0.005892)

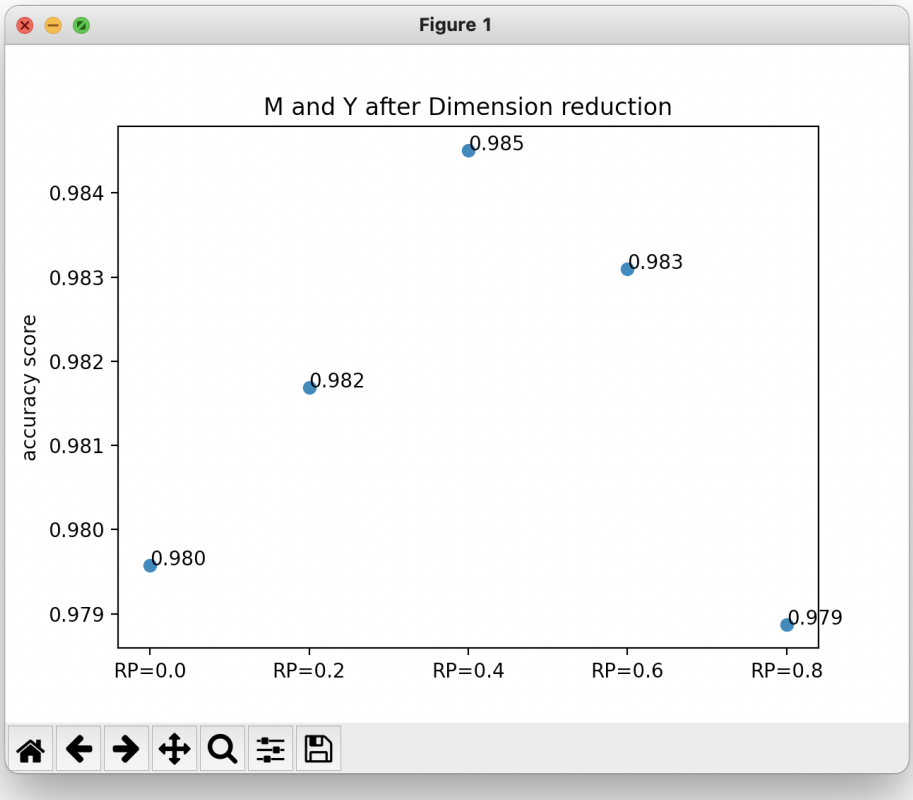

Figure 41: Quadratic Discriminant Analysis: M and Y after Dimension reduction

2.7.7 U and V after Dimension reduction:

RP=0.0: 0.953494 (std: 0.008426)
RP=0.2: 0.945742 (std: 0.011021)
RP=0.4: 0.949263 (std: 0.010804)
RP=0.6: 0.951384 (std: 0.011194)
RP=0.8: 0.949975 (std: 0.013408)

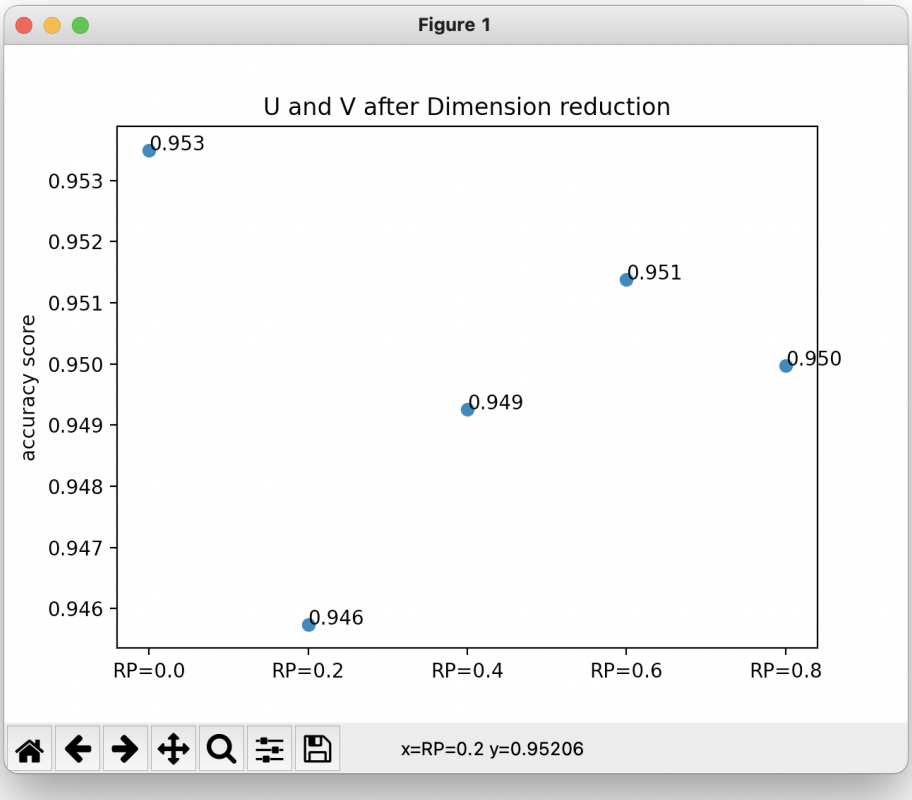

Figure 42: Quadratic Discriminant Analysis: U and V after Dimension reduction

3 Discussion

The performance and run time of the different classifiers table:

Performance of the Different Classifiers							
	KNN	DT	RF	SVM	ANN	ETC	QDA
before Dimension reduction							
H and K	0.94348	0.935765	0.968	0.921677	0.970566	0.980981	0.904151
M and Y	0.99867	0.989515	0.996408	0.943818	0.9982	0.996901	0.99507
U and V	0.997416	0.980033	0.995982	0.944875	0.995928	0.998591	0.986194
after Dimension reduction							
H and K	0.900964	0.891153	0.893283	0.85501	0.888386	0.906264	0.852075
M and Y	0.990923	0.98529	0.987394	0.934351	0.987637	0.991831	0.980563
U and V	0.97769	0.972043	0.977671	0.921384	0.958578	0.979975	0.952359
total	0.968191	0.958966	0.96979	0.920186	0.966549	0.975757	0.945069

Figure 43: The performance of the different classifiers table

Running Time of the Different Classifiers							
	KNN	DT	RF	SVM	ANN	ETC	QDA
before Dimension reduction							
H and K	0.865339	0.32554	9.012527	1.50378	94.20784	4.108	0.168606
M and Y	0.702767	0.20448	8.305461	1.130137	53.47563	3.005477	0.101043
U and V	0.638704	0.192475	9.92830682	0.981691	74.01555	3.222661	0.100642
after Dimension reduction							
H and K	0.527324	0.133277	8.343147	19.26781	75.02508	3.043898	0.06275
M and Y	0.515768	0.131866	7.753551	1.438104	74.28079	2.757397	0.079831
U and V	0.546382	0.12691	7.842014	8.847804	77.88339	2.80509	0.069147
total	3.870824	1.196691	51.28141	33.24773	448.9679	19.00742	0.645883

Figure 44: The run time of the different classifiers table

From the performance table of different classifiers and compared between different classifiers, we can find the order of performance is Extra Tree Classifier, Random Forest, k-nearest neighbors, Artificial Neural Network, Decision tree, Quadratic Discriminant Analysis, SVM. The running time of Extra Tree Classifier is 19 second and Random Forest is 51 second which is much more than the 3 second from k-nearest neighbors. After consider the running time, i will choose the **k-nearest neighbors** for this problem.

From the running time table of different classifiers, the running time of most samples in the table is improved after dimension reduction. However, the performance was decreased. I think the reason is the method of dimension reduction is not a good choice. In this data set, most of the features are important, reducing features from 16 to 4 may lose some information causing a decrease in performance. If a new data set coming with the same problem, I will consider the size of the data set. If the size is too large, the kNN model is the first choose. Decision Tree and Quadratic Discriminant Analysis also have a low running time which fit the large size problem well. In the case of small sample, the random forest and Extra Tree Classifier will become a better choice. For the dimension reduction used for this project, the decreasing performance show they are not the best choice. New dimension reduction method should implement for the new data set.