CSE514 – Fall 2022 Programming Assignment

Letter Recognition Classification

> ***Hanqi Chen*** ***505891***

Introduction

Description and Practical Impacts of The Problem

In this project, I used the Letter Recognition dataset from the UCI repository. This dataset contains 20,000 records and each record consist of 16 features and 1 class type. The problem is to **train the models to classify the capital letter based on their features**. Dividing to several binary and multi-class classification problems (**Choosing U and V as Pair 1, M and Y as Pair 2, A and B as Pair 3, and combining all these 6 classes as multi-class data**). Before I start training models, I think Pair 1 will be the hardest to predict since it has less training data than other pairs (Pair 1: 1474, Pair 2: 1579, Pair 3: 1556), according to my previous experience, the models trained from more data can fit original data than others models(equals to predict easier).

In addition to binary classification, this project also included multi-data as a **multi-class classification**. **The advantages of the multi-class classifier are they are more useful to solve real problems**(most time needed to classify data into different classes not only two) and **for data with high variance, classifying them into several classes than two can be more informative to understand the whole original data**. The multi-class classifiers also have disadvantages. The most obvious one is **the cost of training models**. Take the decision tree classifier as an instance, for the binary classifier, only consist of one root and two nodes, for the multi-class classifier, needs many levels with more than two nodes at each level, which leads to more computing cost. And another disadvantage is due to more factors that need to consider, **the accuracy of mult-class classifiers always worse than binary classifiers**.

From practical impact respect, solving this problem can improve the **extraction technologies** of letters from files such as pdf files. Furthermore, it can be used in **recommended systems**, the system can use these classification models to know what class of results that people want to search for based on their input content, which can decrease the search response time.

Motivation for Multiple Classifiers

When considering whether a classifier is "best" or not, the **mean accuracy** and **running time** are all important factors that need to be considered. From my perspective, I will first set a threshold of mean accuracy such as 0.95 since it's the most important factor for choosing a classifier. And then I will compare the running time of classifiers whose mean accuracy passes the accuracy threshold and choose the fastest classifier as the best. In addition, I will also consider **the difference between mean and best accuracy** if several classifiers all pass the accuracy threshold and have close running times since the difference can reflect the stability of classifiers' predictions.

Motivation for Dimension Reduction

The main motivation for dimension reduction is **efficiency** which means dropping the useless or less informative data to reduce the cost spent on training and validating classifier models. **When considering whether a dimension reduction method is "good" or "bad", first it's after dimension reduction, whether the cost such as running time on training and validating classifier models decreased and another concern is if the accuracy of new models' mean accuracy still can pass the required accuracy threshold**. For this problem, original data have 16 features and many of them have some less informative data such as low variance data, which has less help to train the models. Drop these data may reduce the training cost and also keep it still can pass the mean accuracy threshold.

Description of the Dimension Reduction Methods

Simple Quality Filtering - Low variance

Simple quality filtering is to drop the less informative data (Missing values, low Variance, and domain expert judgment). For this problem, there is no missing value, and hard to determine whether the feature makes sense at the beginning, therefore, I only set a variance threshold of 2 to drop the data with low variance.

Embedded Methods - Univariate feature selection¶

As one of the embedded methods, univariate feature selection works by selecting the best features based on univariate statistical tests. For this problem, I choose *chi2* as a statistical test and set `k = 4` which means new data will remove all other features except the 4 highest-scoring features.

Feature Extraction - PCA

Principal component analysis (PCA) is a technique that transforms high-dimensions data into lower dimensions while retaining as much information as possible. For this problem, I assigned `n_components = 4` which means the new data will only keep the 4 features with the most information and drop others.

Resulte

There is total of 7 models in this section and each with 2 hyperparameters tuning which implement with `StratifiedKFold` with `k = 5`.

K-Nearest Neighbors

Description

K-nearest neighbors algorithm (KNN) is a supervised learning method, which takes an input and finds the k nearest neighbors to the input, and votes the class based on these neighbors' labels.

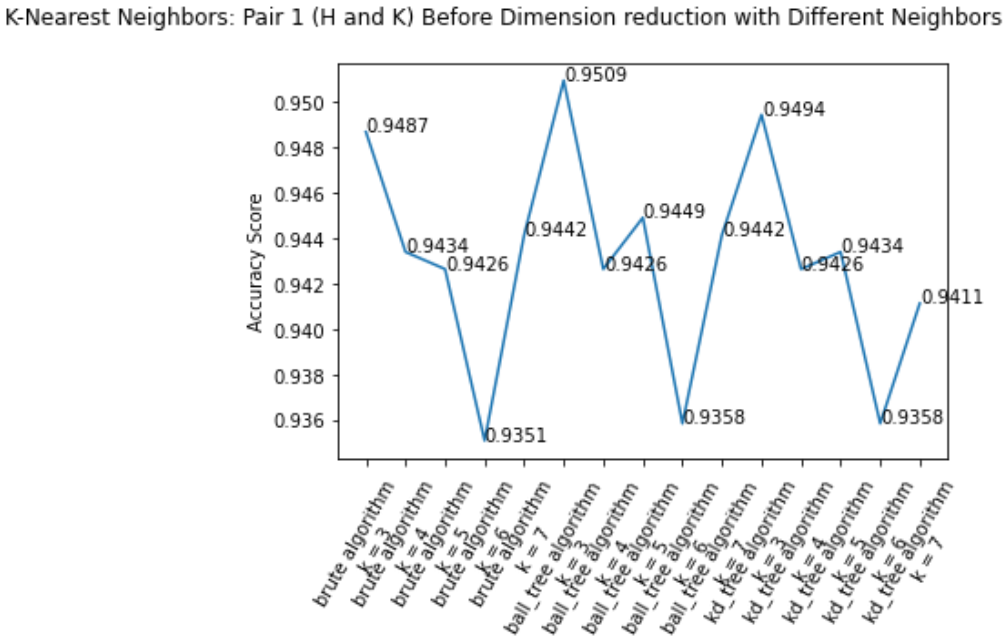
- Advantages：
  - **No Training Period**: KNN does not need to do any training on the model itself(different from ANN), the only thing that needs to train is to train data itself. Because of this, KNN is very time efficient in terms of improvising for random modeling on the available data.
  - **Easy Implementation**: KNN is easy to implement since the only thing that needs to do is to calculate the distance between different data points on the basis of data of different features and this distance can easily be calculated using distance formulas such as Euclidian or Manhattan. It also makes it easy to add new data to the models.
- Disadvantages:
  - **Does not work well with large datasets** as calculating distances between each data instance would be very costly.
  - **Does not work well with high dimensionality** as this will complicate the distance calculating process to calculate the distance for each dimension.
  - **Sensitive to noisy and missing data**

This project used algorithm from `sklearn.neighbors.KNeighborsClassifier`, and tuned 2 hyperparameters, which are `n_neighbors = [3, 4, 5, 6, 7]` and `algorithms = ['brute', 'ball_tree', 'kd_tree']`.

Cross Validation Results**

Pair 1 H and K

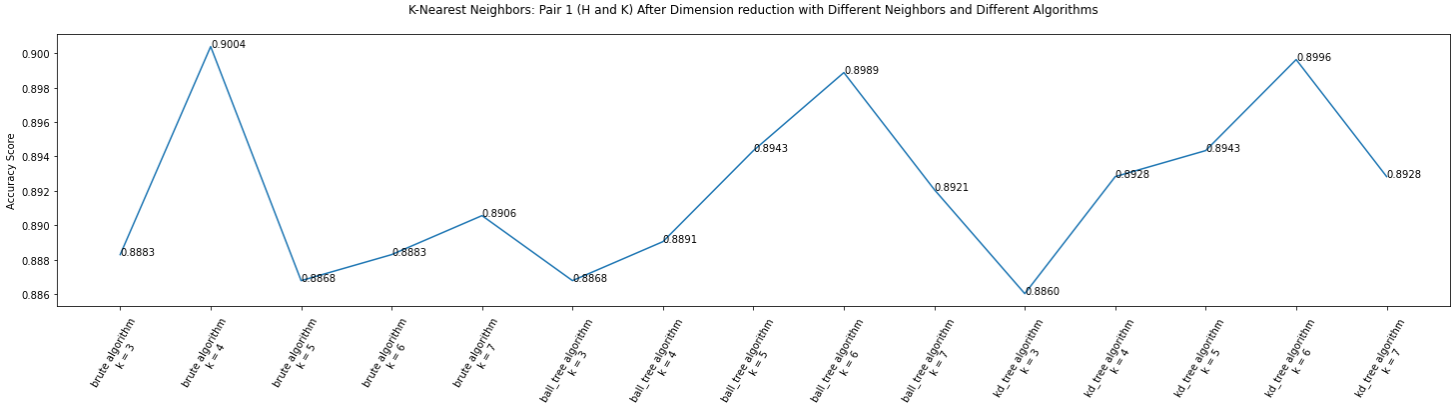
Without Dimension Reduction



Mean Accuracy: 0.943 Running Time: 0.8646

Best Accuracy: 0.9509 (K = 7, Algorithm = ball_tree)

With Dimension Reduction

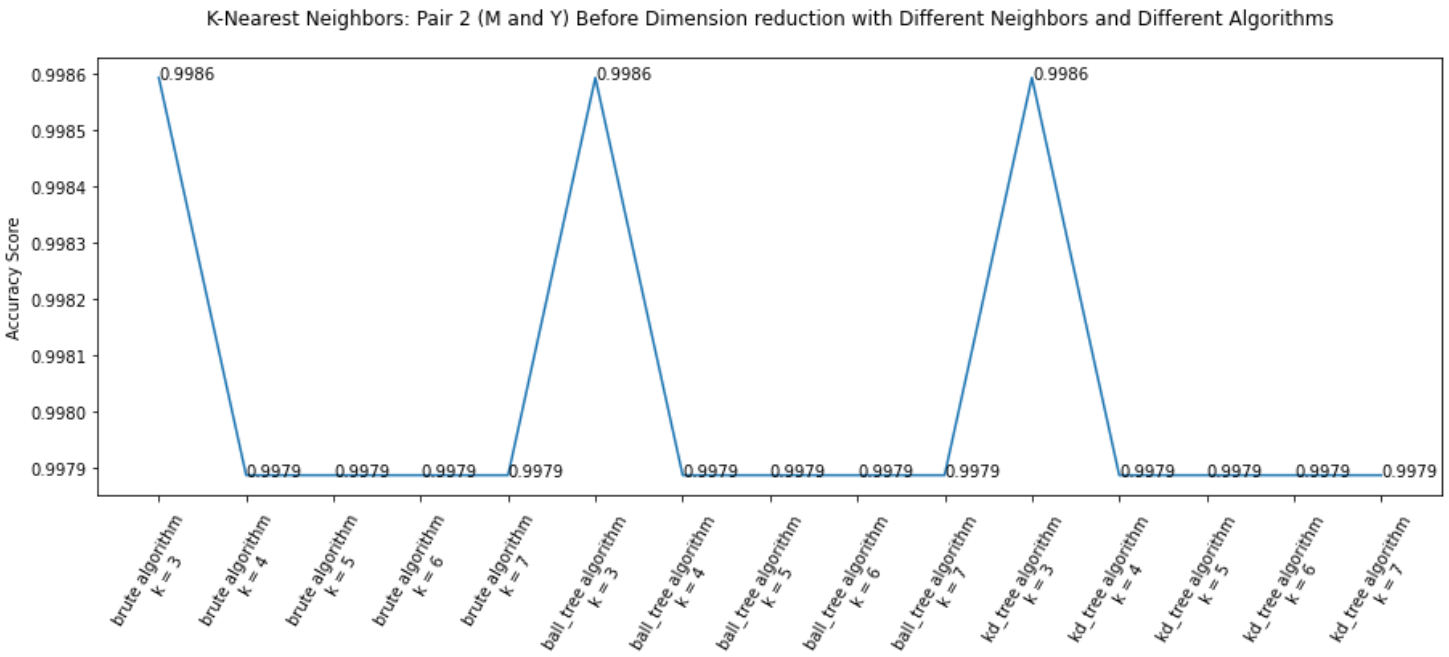


Mean Accuracy: 0.98921 Running Time: 0.7735

Best Accuracy: 0.9004 (K = 4, Algorithm = brute)

Pair 2 M and Y

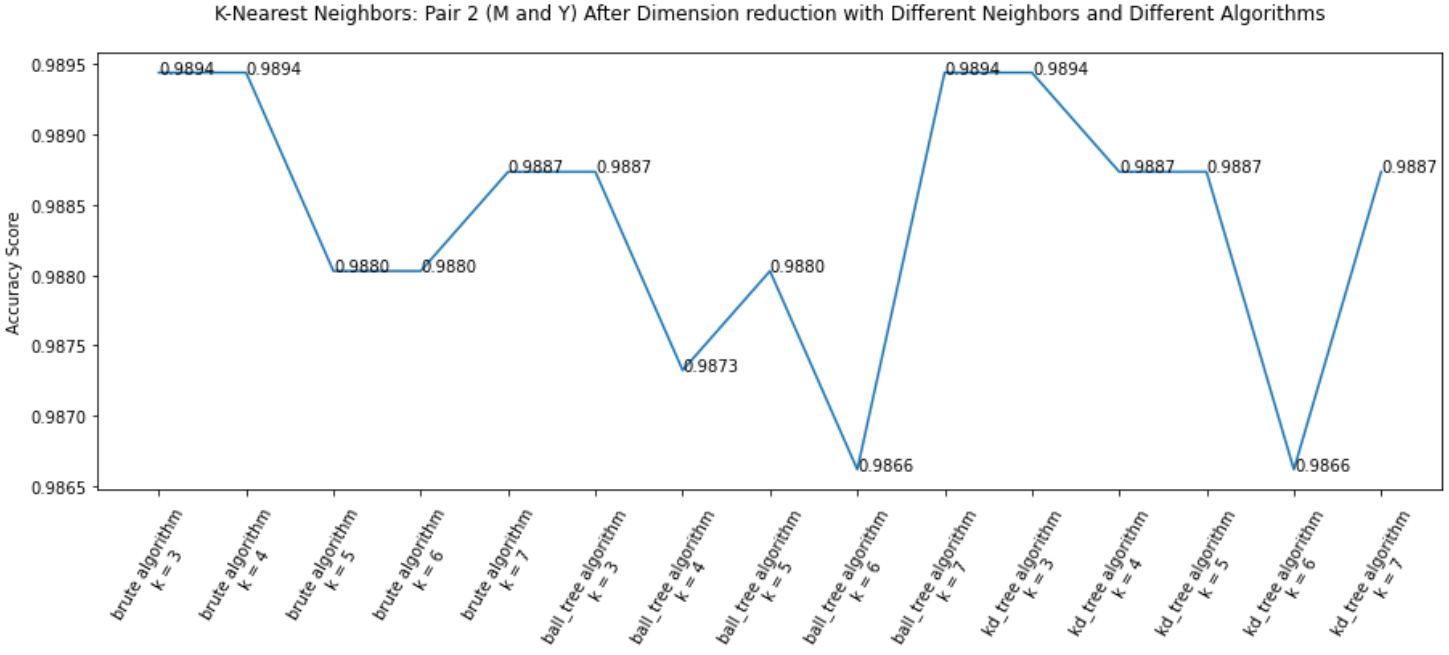
Without Dimension Reduction



Mean Accuracy: 0.998 Running Time: 0.9804

Best Accuracy: 0.9986 (K = 3, Algorithm = brute/ball_tree/kd_tree)

With Dimension Reduction

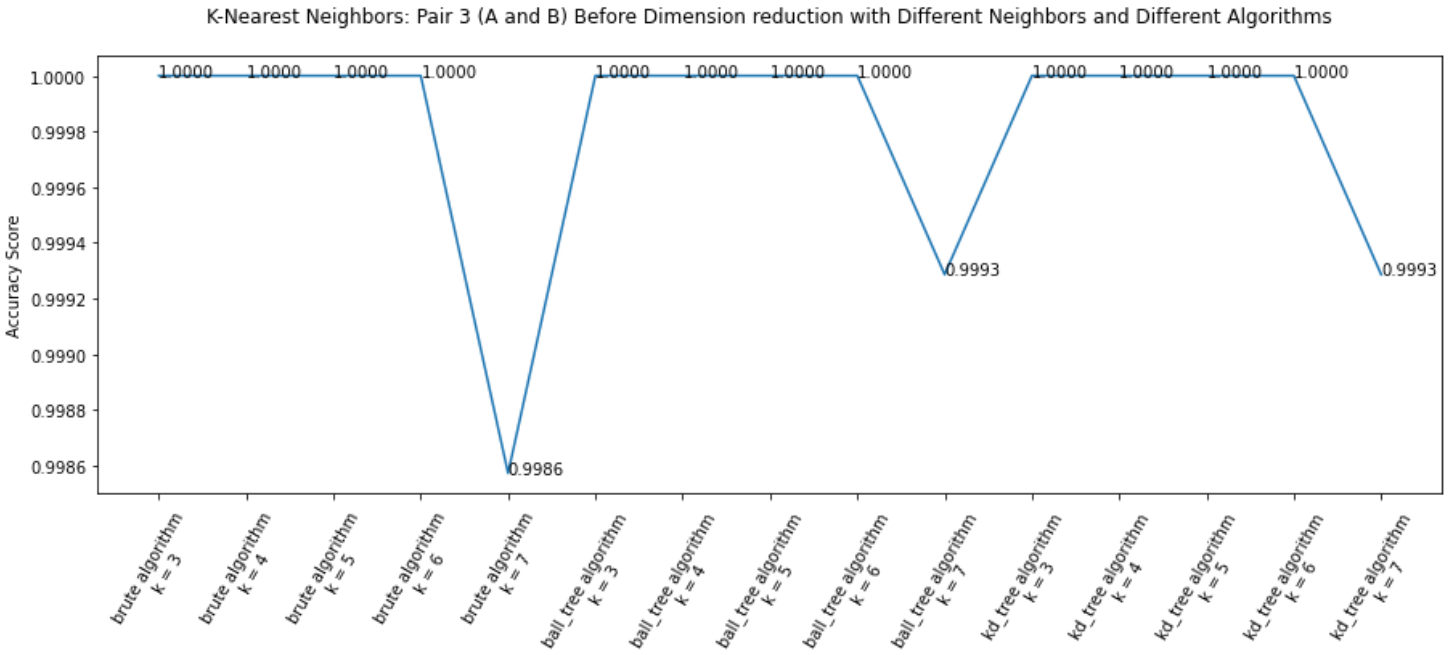


Mean Accuracy: 0.9884 Running Time: 0.7534

Best Accuracy: 0.9894 (K = 3/4, Algorithm = brute; K = 7, Algorithm = ball_tree; K = 3, Algorithm = kd_tree)

Pair 3 A and B

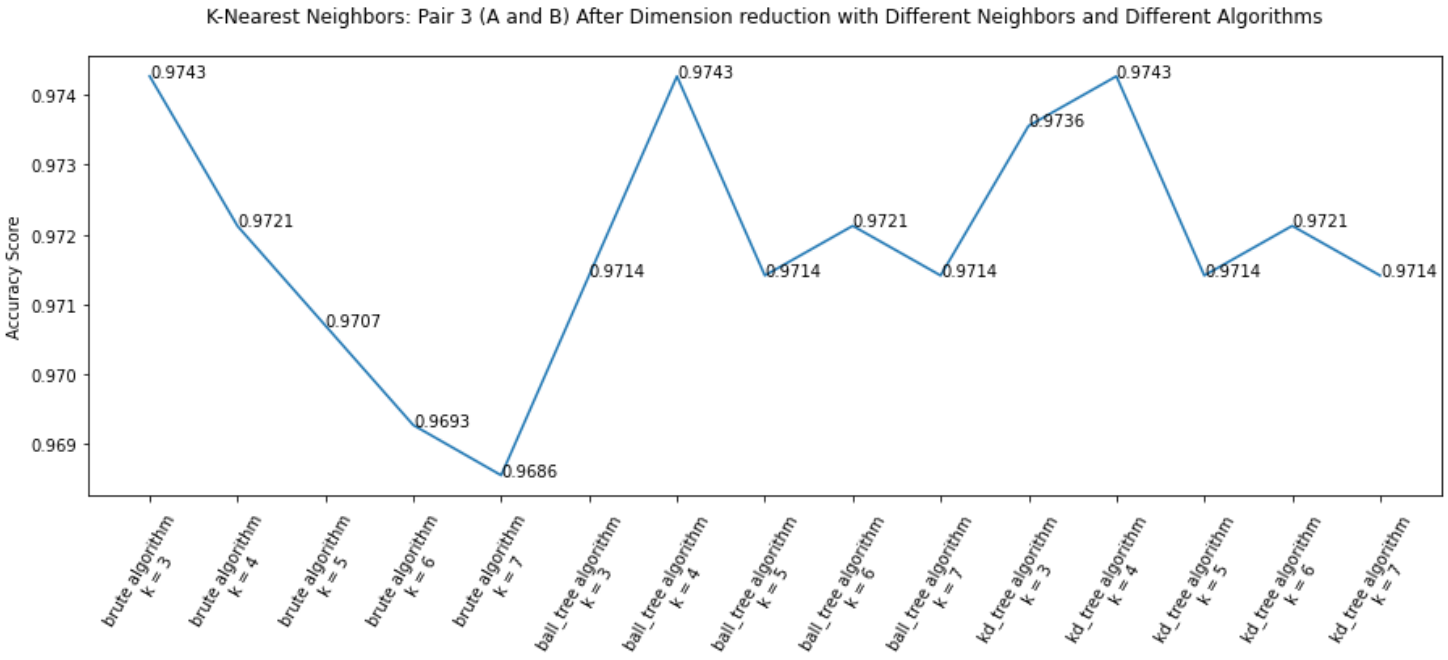
Without Dimension Reduction



Mean Accuracy: 0.9998 Running Time: 0.9035

Best Accuracy: 1.0 (K = 3/4/5/6, Algorithm = brute/ball_tree/kd_tree)

With Dimension Reduction

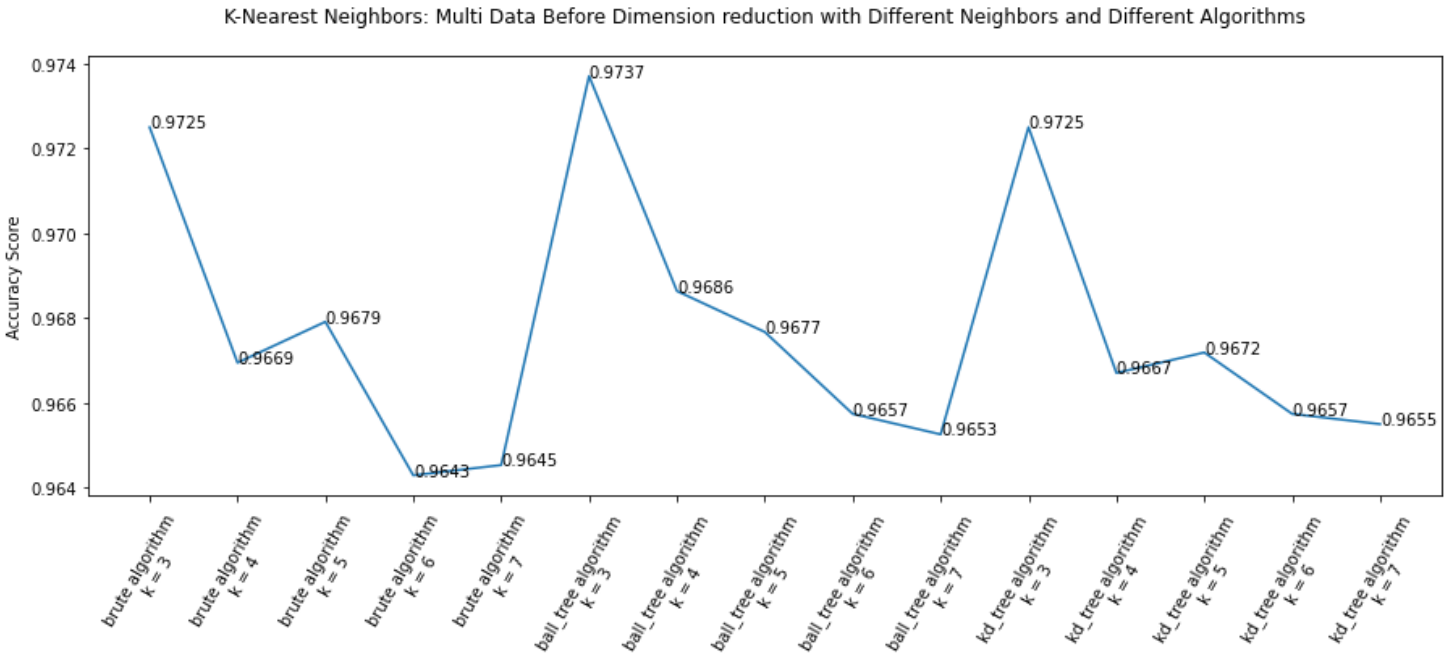


Mean Accuracy: 0.9719 Running Time: 0.719

Best Accuracy: 0.9743 (K = 3, Algorithm = brute/ball_tree; K = 4, Algorithm = kd_tree)

Multi Data

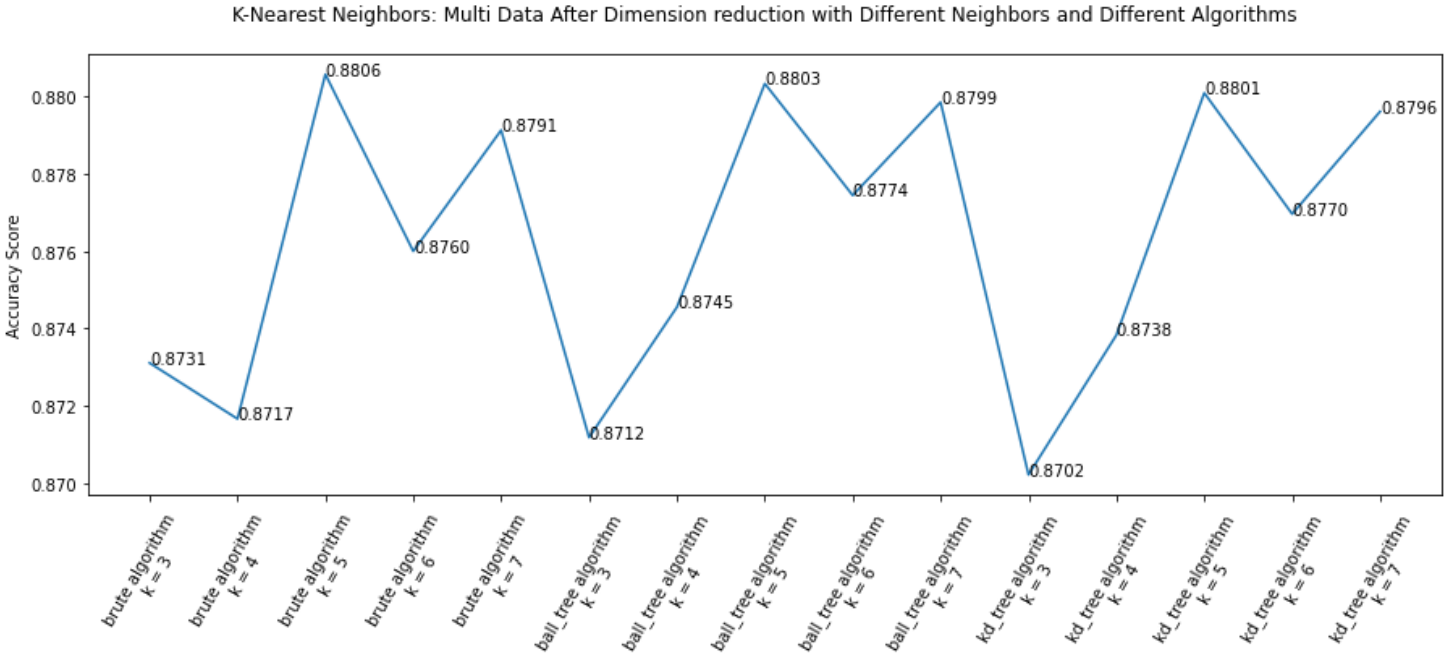
Without Dimension Reduction



Mean Accuracy: 0.9677 Running Time: 3.6447

Best Accuracy: 10.9737 (K = 3, Algorithm = ball_tree)

With Dimension Reduction



Mean Accuracy: 0.8763 Running Time: 2.7899

Best Accuracy: 0.8806 (K = 5, Algorithm = brute)

Decision Tree

Description

Decision Tree solves the problem of machine learning by transforming the data into a tree representation. Each internal node of the tree representation denotes an attribute and each leaf node denotes a class label.

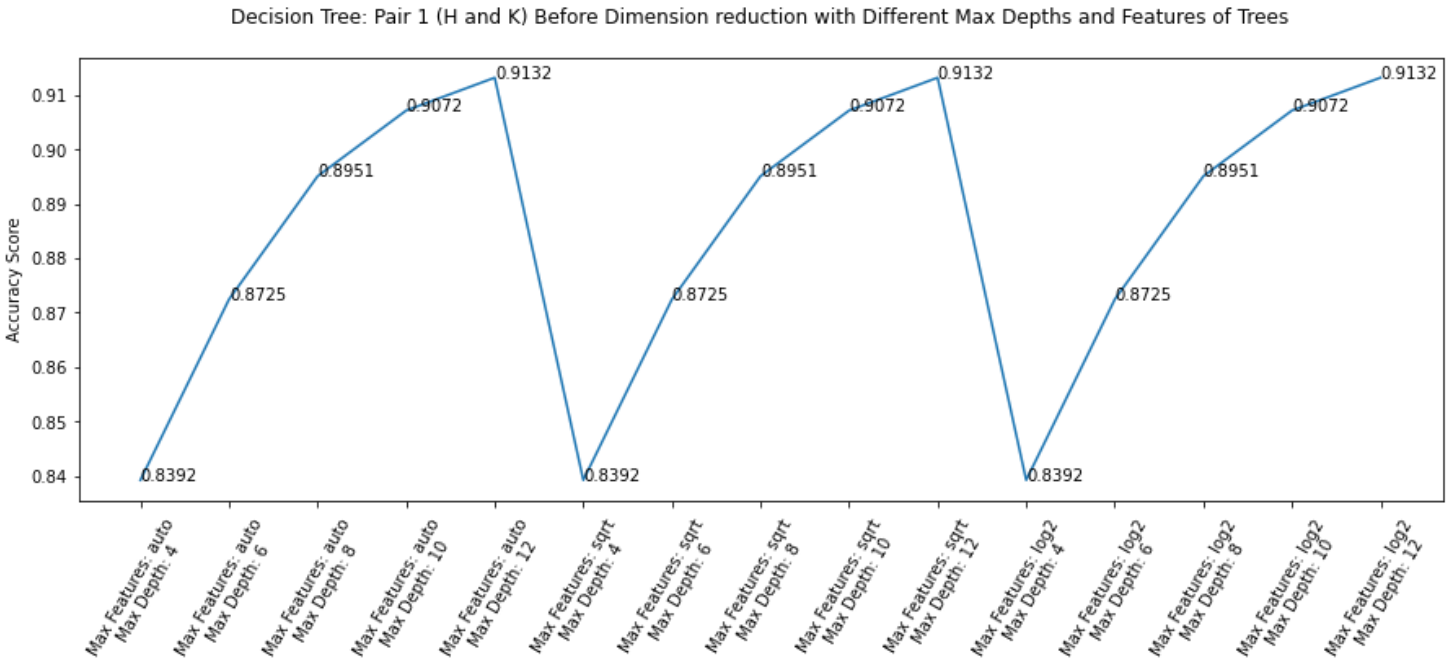
- Advantages：
  - **Less require for preprocessing**: Compared to other algorithms decision trees require less effort for data preparation during pre-processing such as normalization or scaling of data
  - **Interpretability**: A Decision tree model is very intuitive and easy to explain to technical teams as well as stakeholders.'
  - **No affected by missing values**: Missing values in the data also do NOT affect the process of building a decision tree to any considerable extent.
- Disadvantages:
  - **Easy affected by data change**: A small change in the data can cause a large change in the structure of the decision tree causing instability.
  - **High Training Cost**: Decision tree often involves higher time to train the model and is relatively expensive as the complexity and time have taken are more.

This project used algorithm from `sklearn.tree.DecisionTreeClassifier` , and tuned 2 hyperparameters, which are `max_depths = [4, 6, 8, 10, 12]` and `max_features = ['auto', 'sqrt', 'log2']` .

Cross Validation Results**

Pair 1 H and K

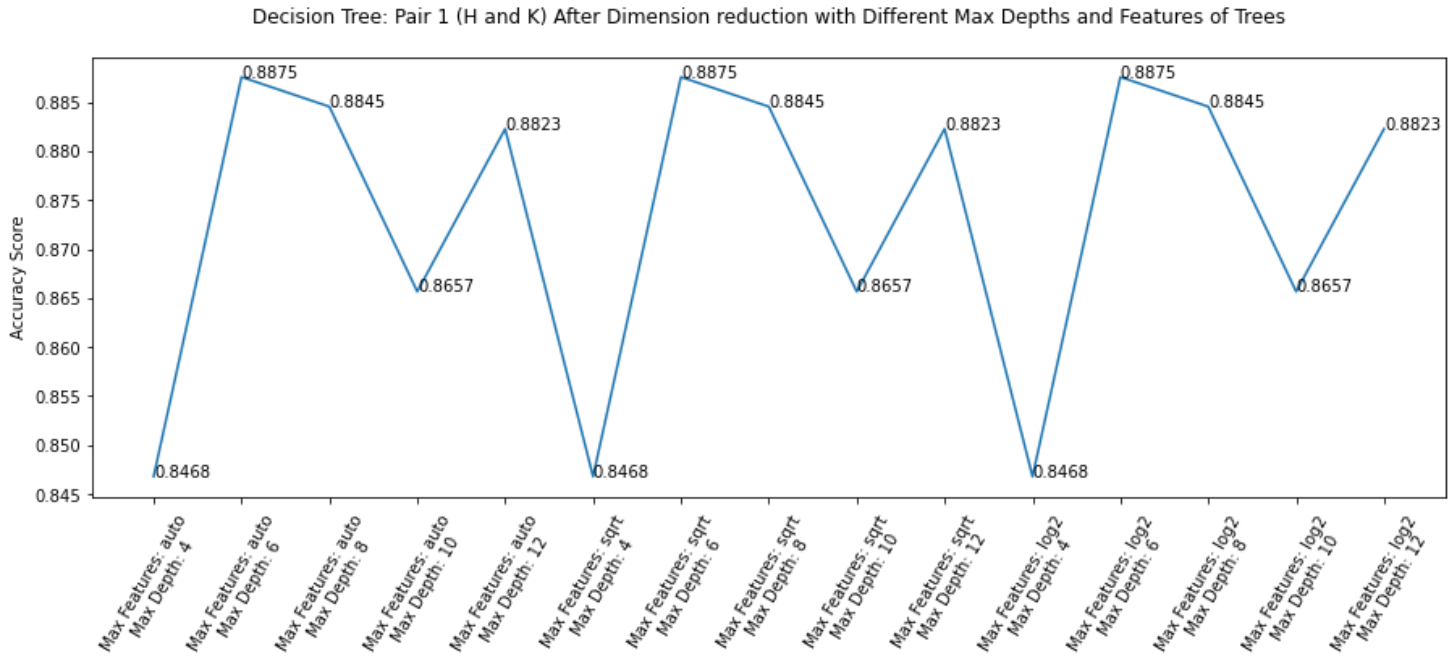
Without Dimension Reduction



Mean Accuracy: 0.8854 Running Time: 0.2719

Best Accuracy: 0.9132 (max_features = auto/sqrt/log2, max_depths = 12)

With Dimension Reduction

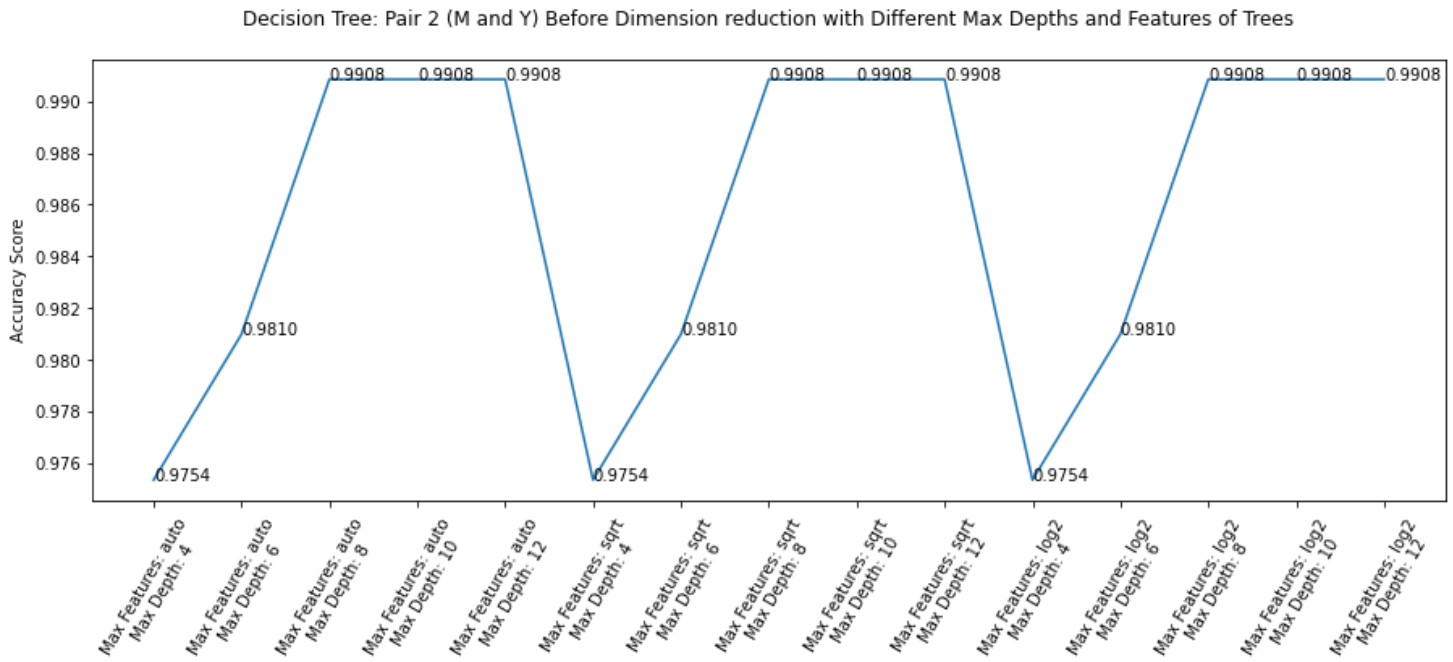


Mean Accuracy: 0.8734 Running Time: 0.2109

Best Accuracy: 0.8875 (max_features = auto/sqrt/log2, max_depths = 6)

Pair 2 M and Y

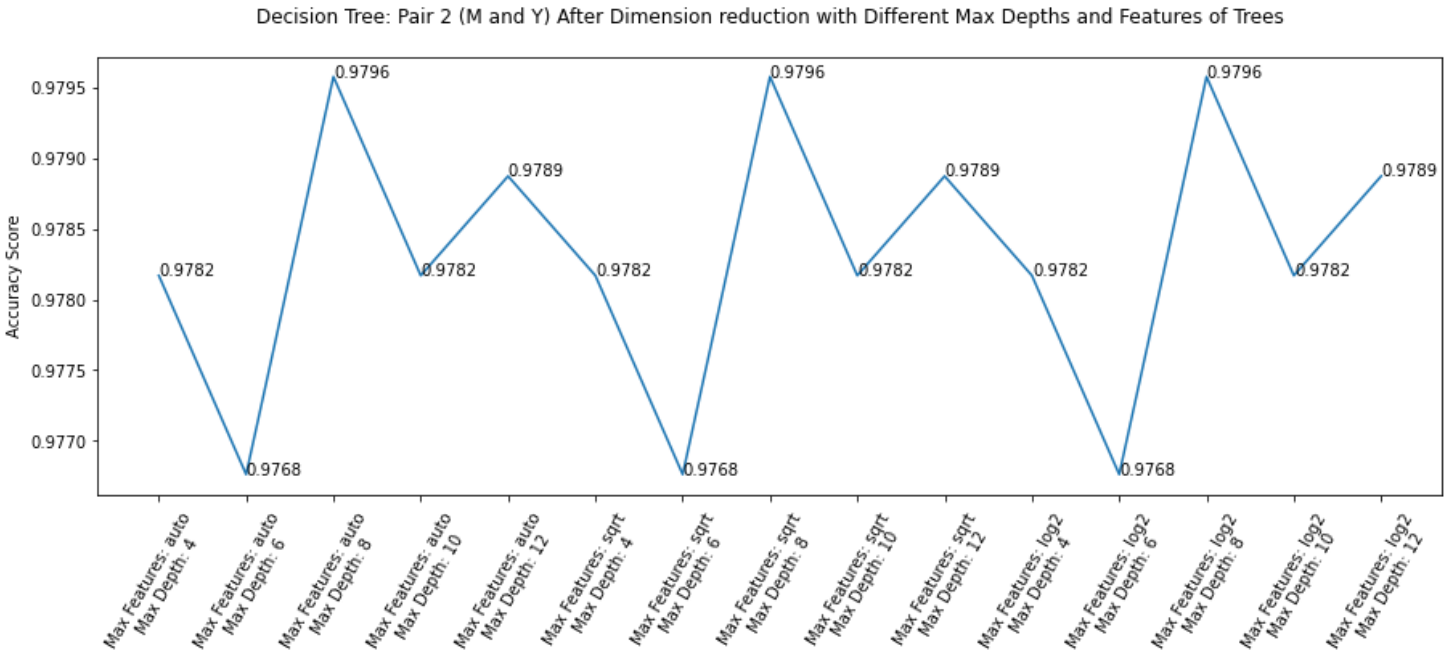
Without Dimension Reduction



Mean Accuracy: 0.9858 Running Time: 0.2615

Best Accuracy: 0.9908 (max_features = auto/sqrt/log2, max_depths = 8/10/12)

With Dimension Reduction

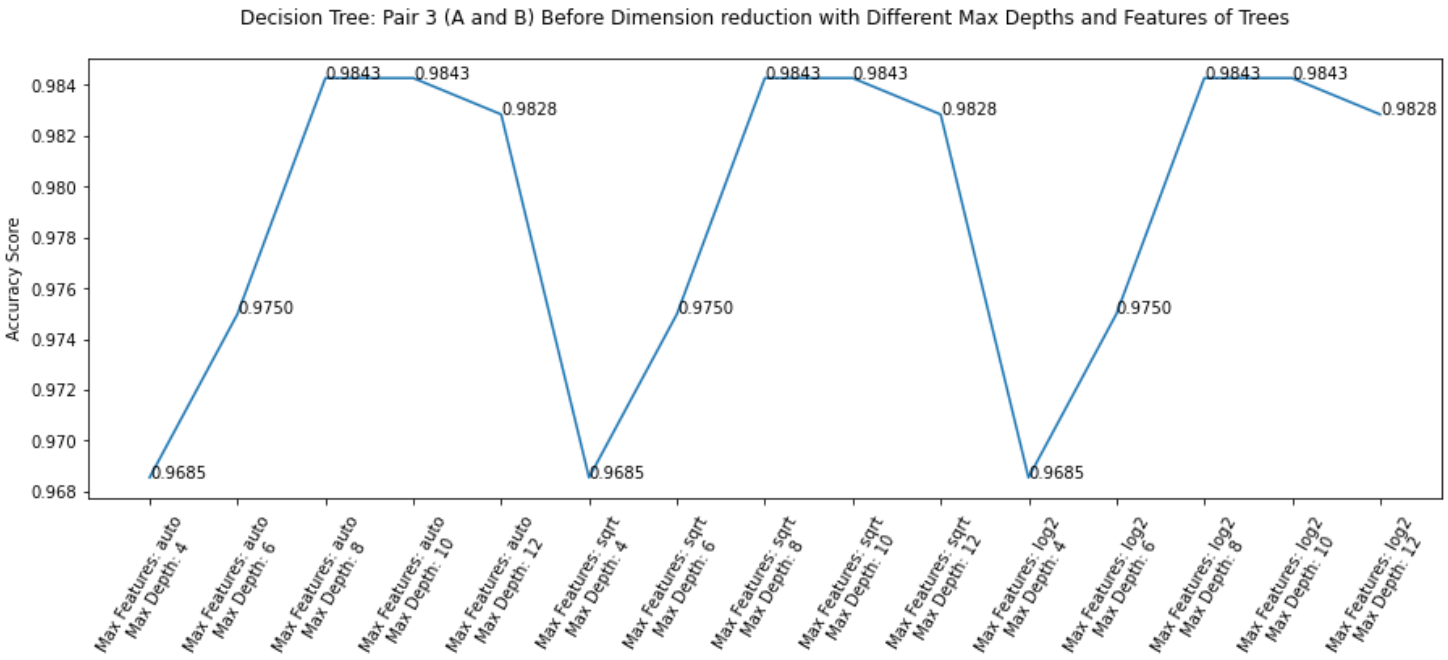


Mean Accuracy: 0.9783 Running Time: 0.2053

Best Accuracy: 0.9796 (max_features = auto/sqrt/log2, max_depths = 8)

Pair 3 A and B

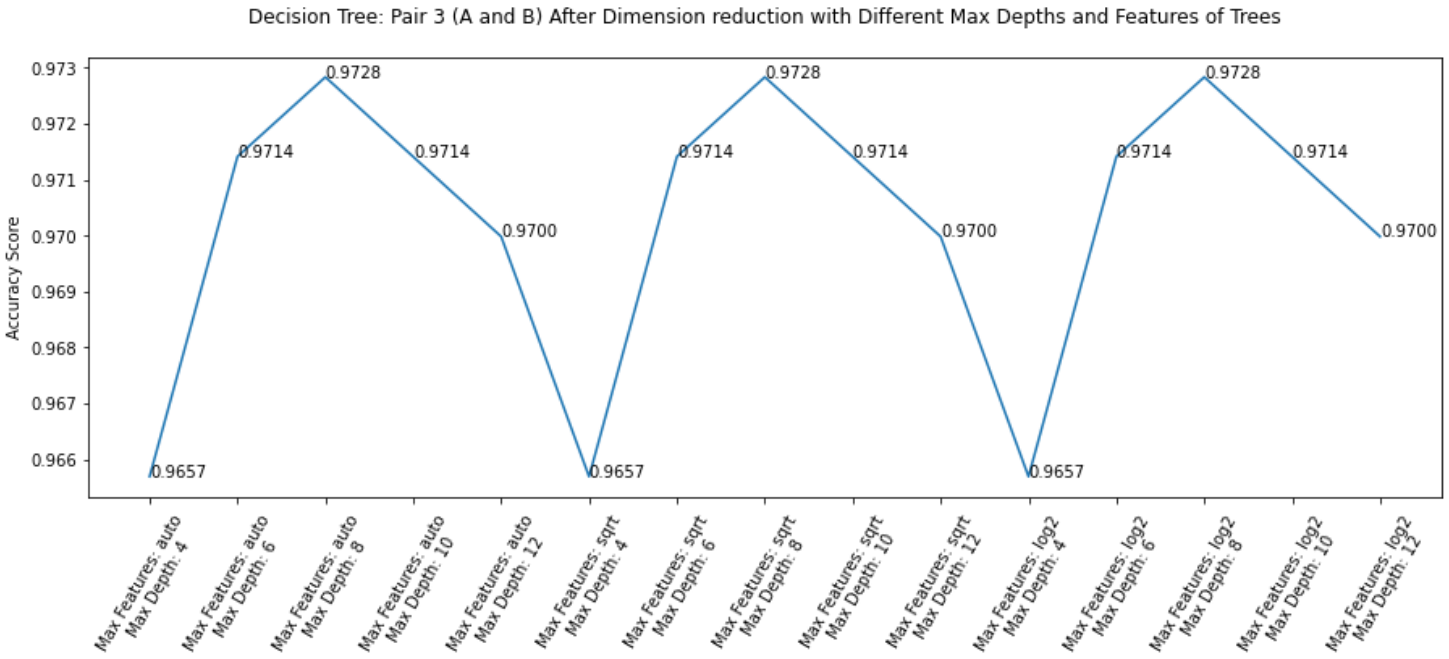
Without Dimension Reduction



Mean Accuracy: 0.979 Running Time: 0.2766

Best Accuracy: 0.9843 (max_features = auto/sqrt/log2, max_depths = 8/10)

With Dimension Reduction

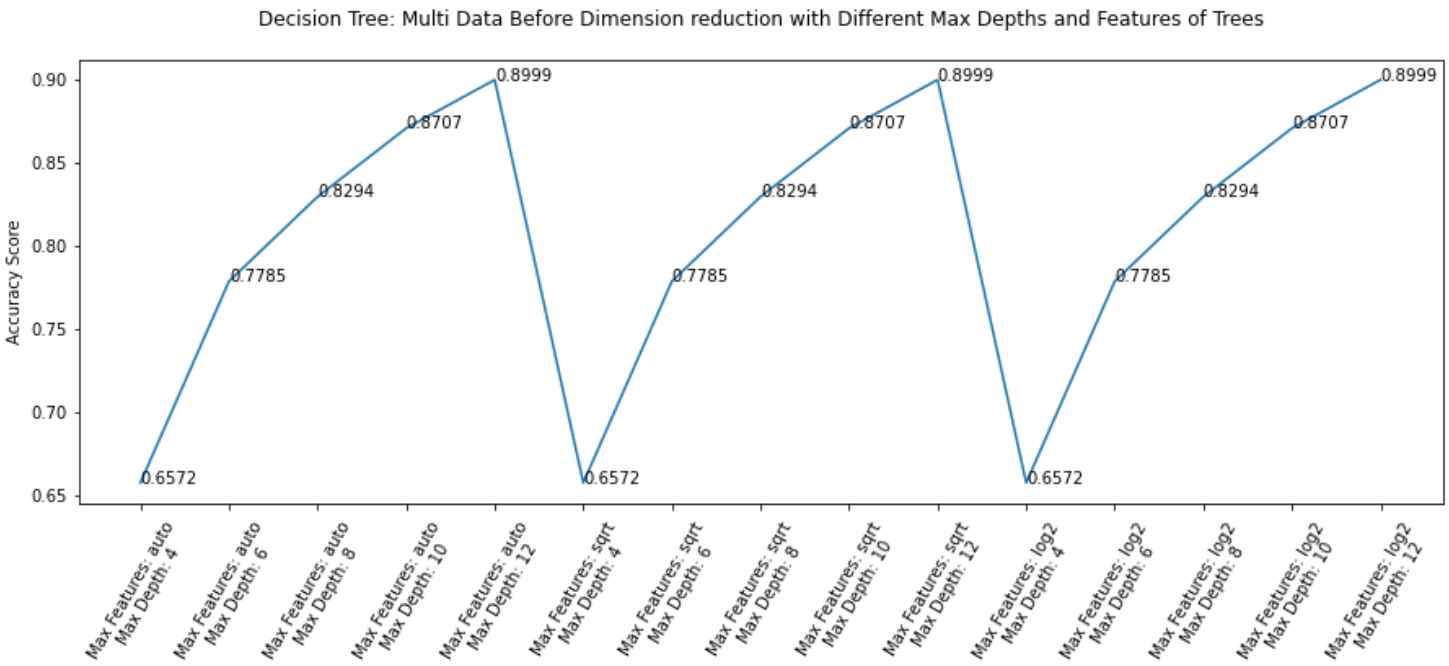


Mean Accuracy: 0.9703 Running Time: 0.1942

Best Accuracy: 0.9728 (max_features = auto/sqrt/log2, max_depths = 8)

Multi Data

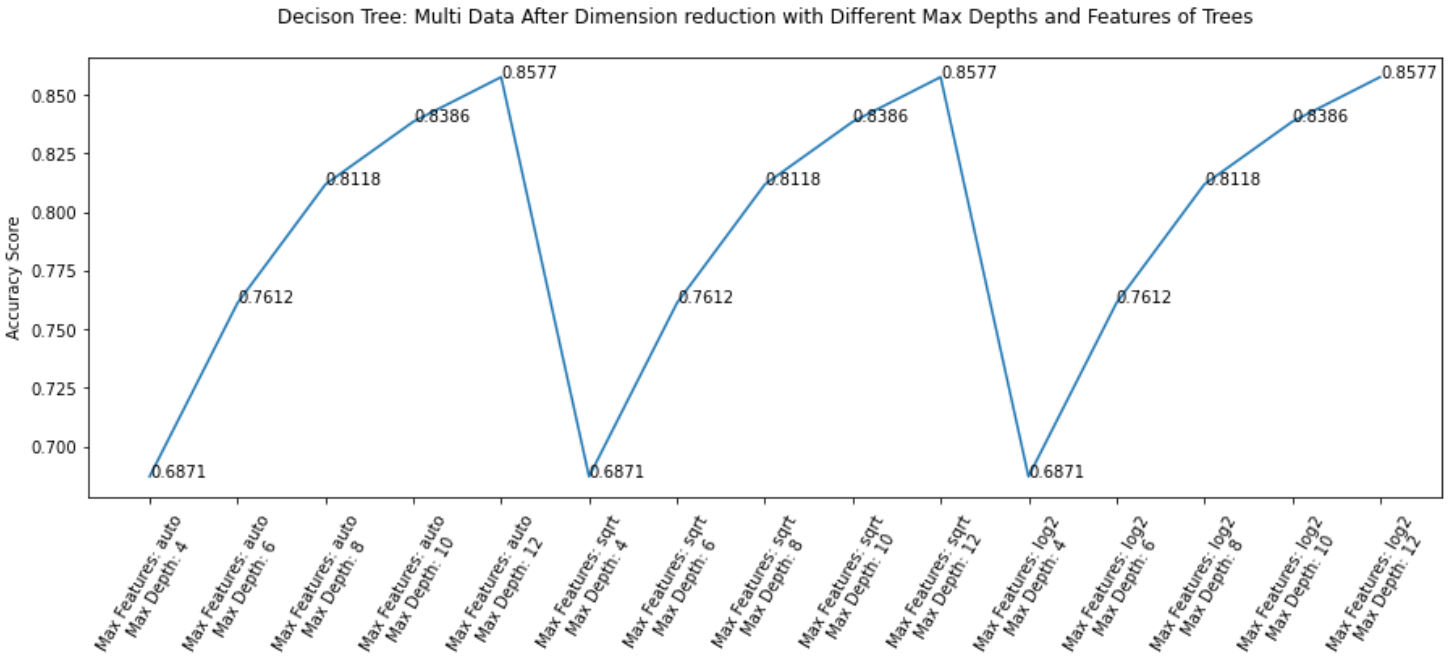
Without Dimension Reduction



Mean Accuracy: 0.8071 Running Time: 0.5648

Best Accuracy: 0.8999 (max_features = auto/sqrt/log2, max_depths = 12)

With Dimension Reduction



Mean Accuracy: 0.7913 Running Time: 0.5374

Best Accuracy: 0.8577 (max_features = auto/sqrt/log2, max_depths = 12)

Random Forest

Description

Random forest is a supervised learning algorithm. It builds a forest with an ensemble of decision trees. It is an easy-to-use machine learning algorithm that produces a great result, even without hyperparameter tuning.

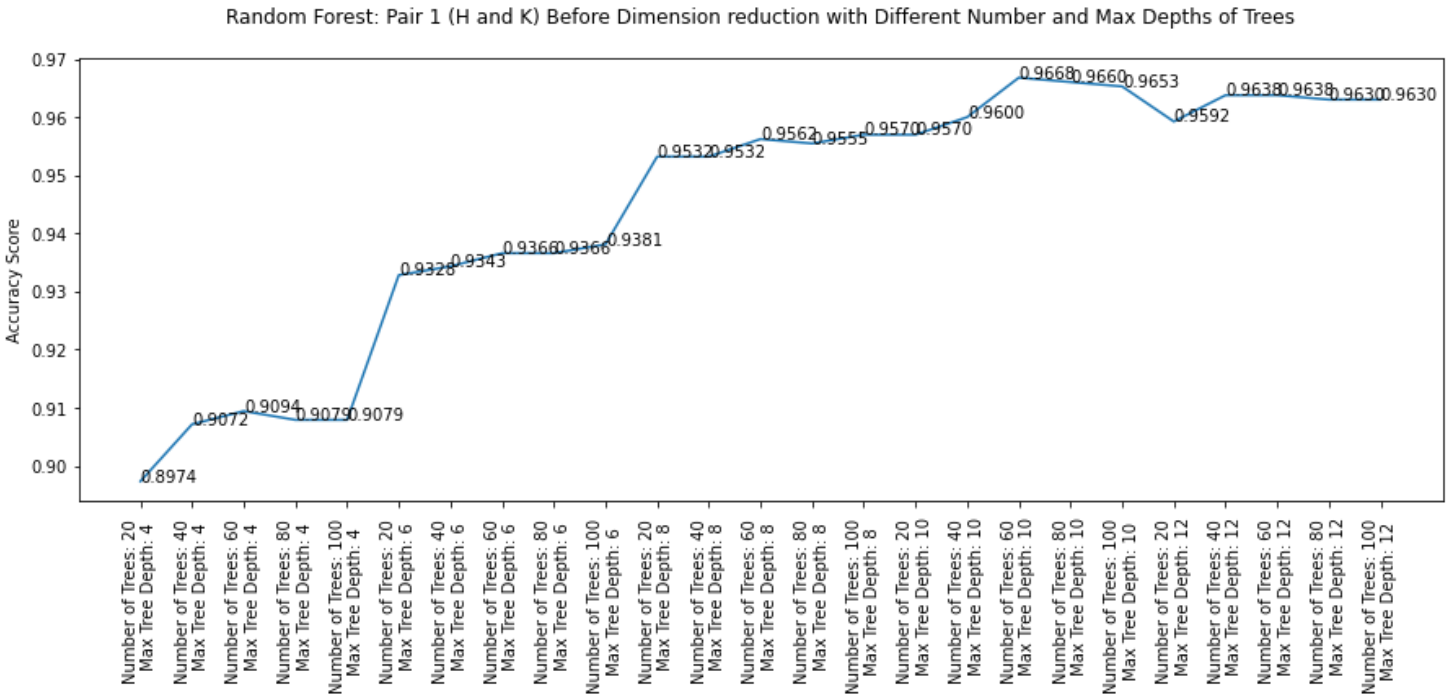
- Advantages：
  Since random forest consists of decision trees, it inherent advantages:
  - **Less require for preprocessing**: Compared to other algorithms random forest requires less effort for data preparation during pre-processing such as normalization or scaling of data
  - **No affected by missing values**: Missing values in the data also do NOT affect the process of building random forest to any considerable extent.
    Random forest unique advantages:
  - **Less Pruning**: Random Forest algorithm is less prone to overfitting than Decision Tree and other algorithms
- Disadvantages:
  - **Easy affected by data change**: A slight change in the data can cause a large difference in the structure of the random forest causing instability.
  - **High Training Cost**: Random forest often involves higher time to train the model and is relatively expensive as the complexity and time taken are more.

This project used algorithm from `sklearn.ensemble.RandomForestClassifier` , and tuned 2 hyperparameters, which are `max_depths = [4, 6, 8, 10, 12]` and `n_estimators = [20, 40, 60, 80, 100]` .

Cross Validation Results**

Pair 1 H and K

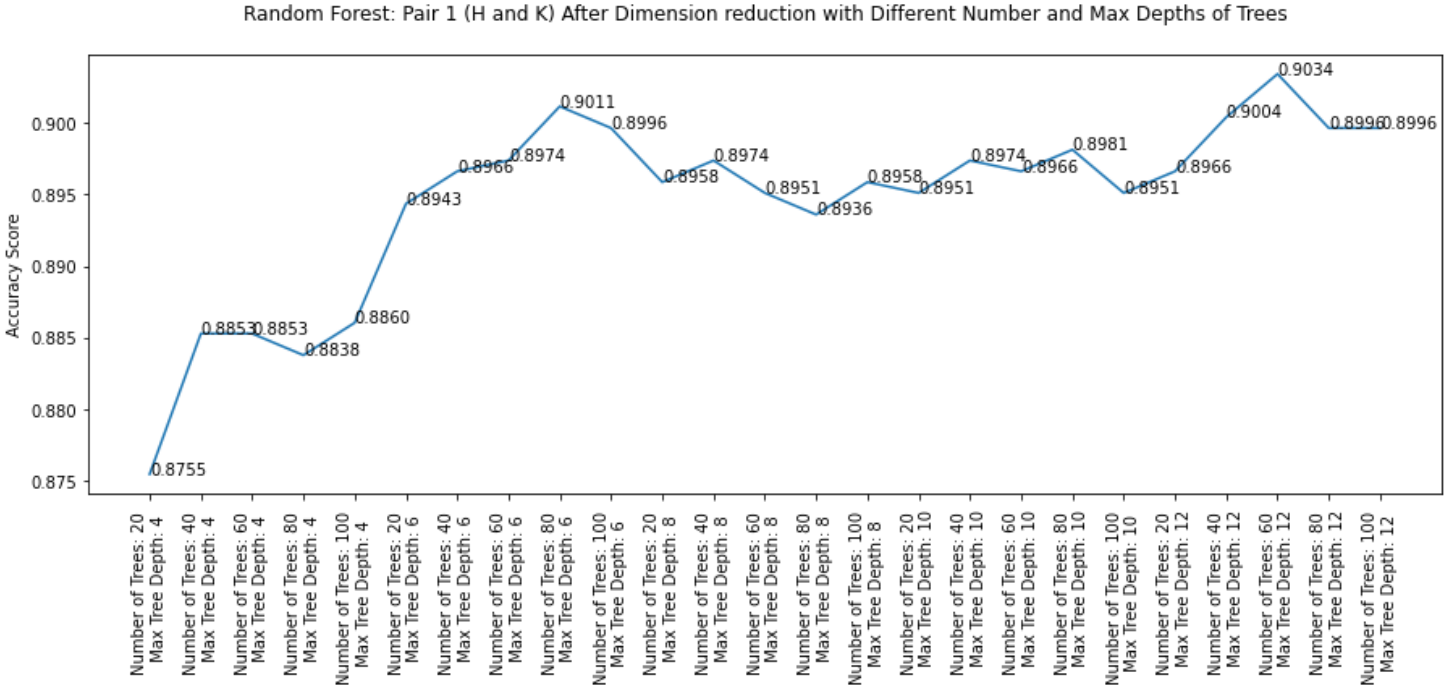
Without Dimension Reduction



Mean Accuracy: 0.9445 Running Time: 9.8011

Best Accuracy: 0.9668 (n_estimators = 60, max_depth = 10)

With Dimension Reduction

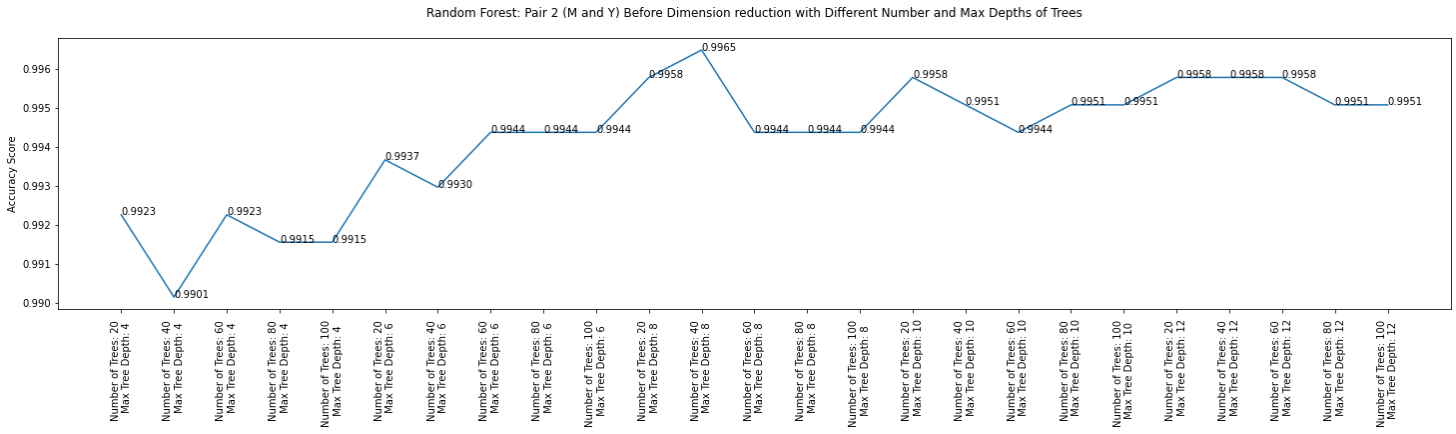


Mean Accuracy: 0.8946 Running Time: 10.6177

Best Accuracy: 0.9034 (n_estimators = 60, max_depth = 12)

Pair 2 M and Y

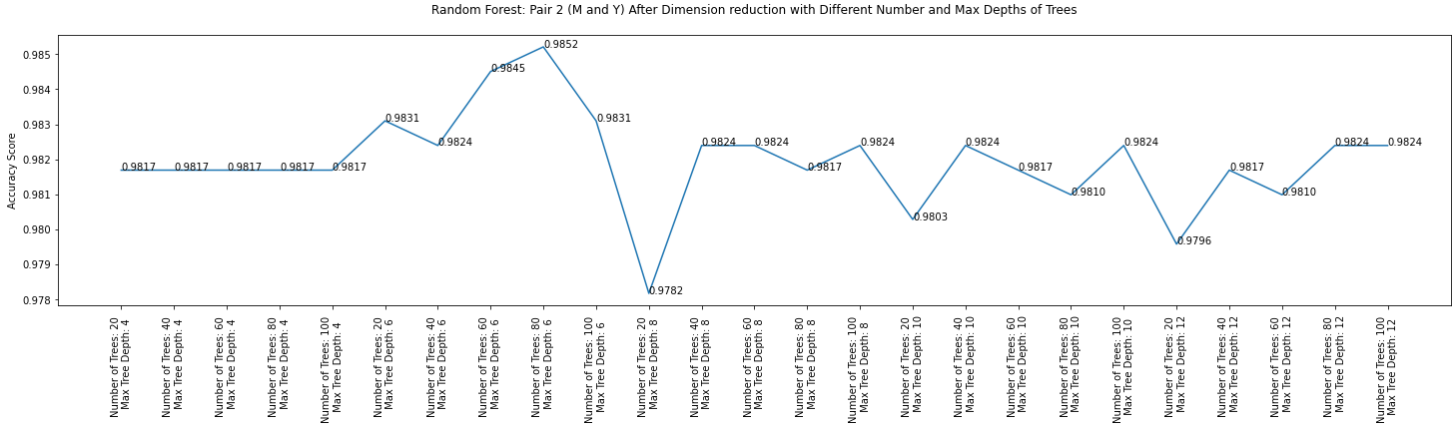
Without Dimension Reduction



Mean Accuracy: 0.9942 Running Time: 9.0003

Best Accuracy: 0.9968 (n_estimators = 40, max_depth = 8)

With Dimension Reduction

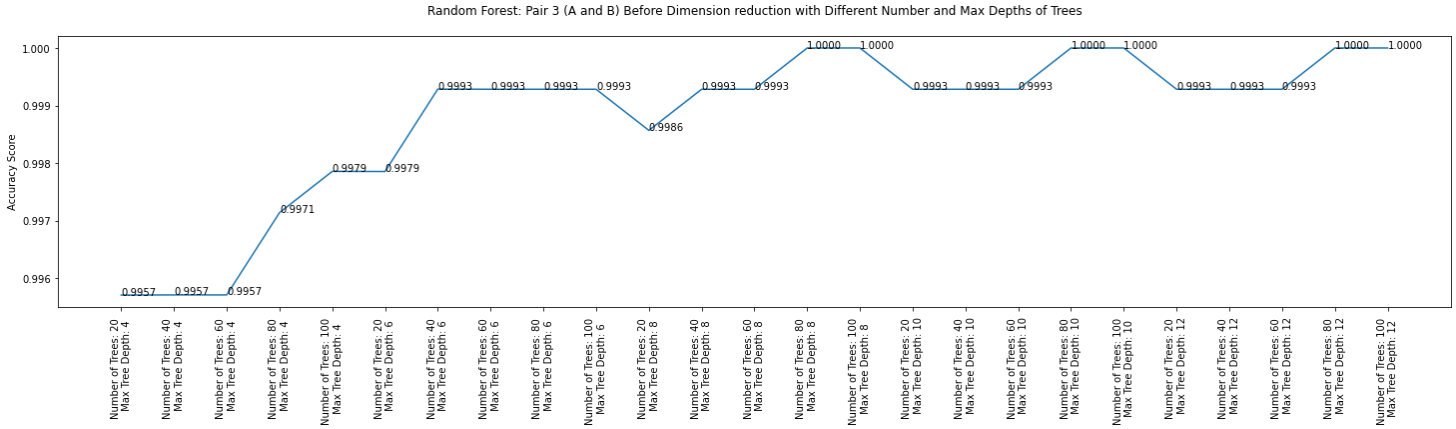


Mean Accuracy: 0.9818 Running Time: 9.9746

Best Accuracy: 0.9852 (n_estimators = 80, max_depth = 6)

Pair 3 A and B

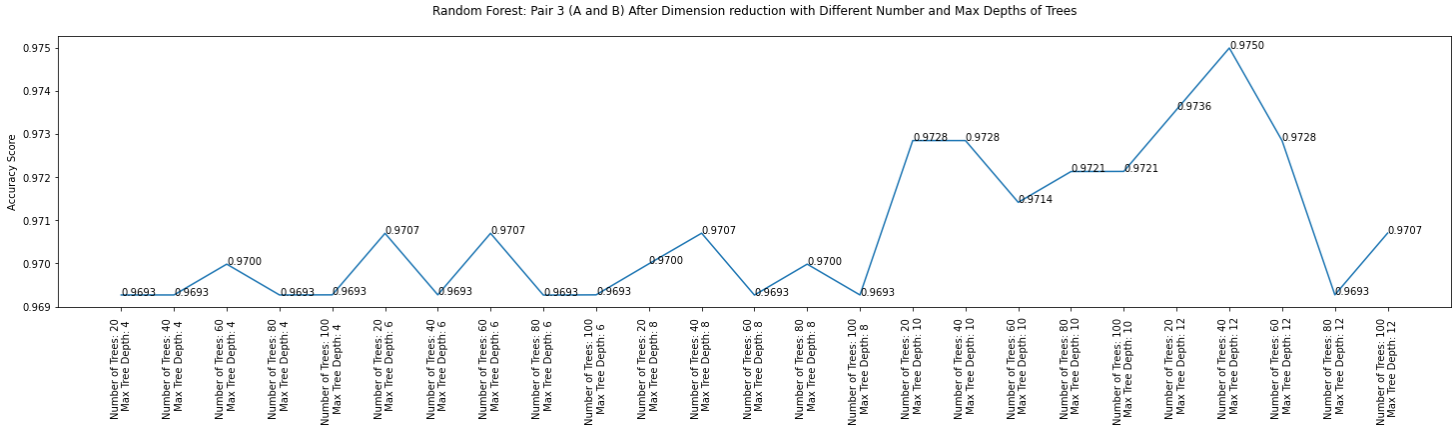
Without Dimension Reduction



Mean Accuracy: 0.9988 Running Time: 8.82

Best Accuracy: 1.0 (n_estimators = 80/100, max_depth = 8/10/12)

With Dimension Reduction

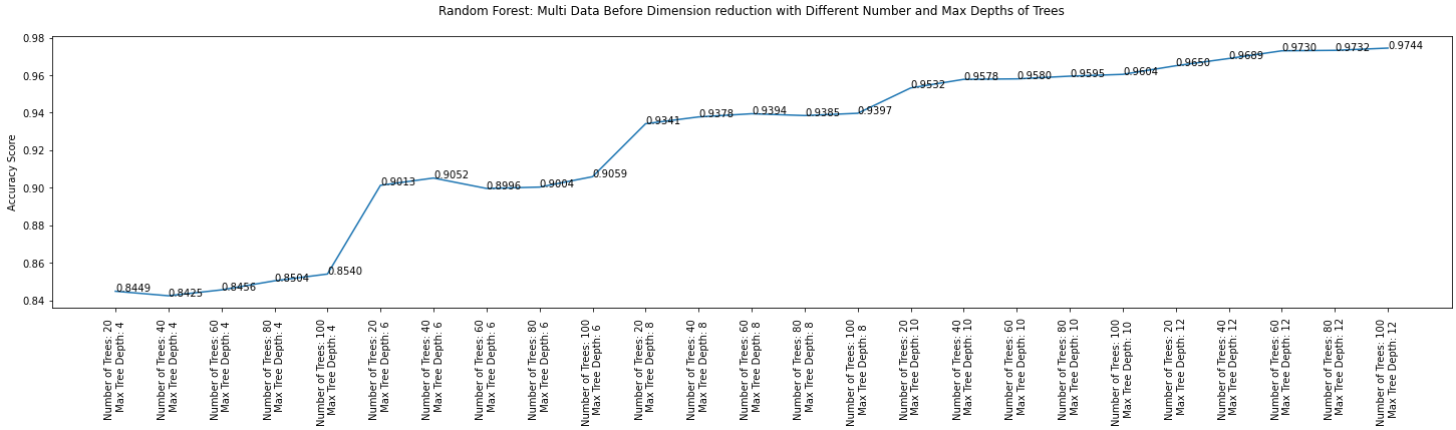


Mean Accuracy: 0.9707 Running Time: 9.6331

Best Accuracy: 0.975 (n_estimators = 40, max_depth = 12)

Multi Data

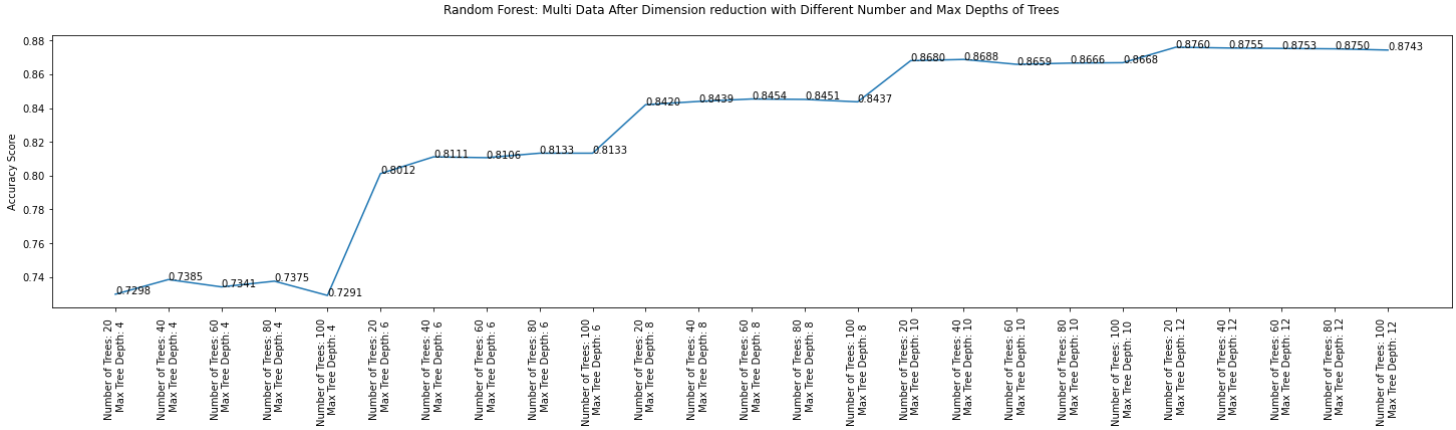
Without Dimension Reduction



Mean Accuracy: 0.9233 Running Time: 19.1403

Best Accuracy: 0.9744 (n_estimators = 100, max_depth = 12)

With Dimension Reduction



Mean Accuracy: 0.826 Running Time: 23.8312

Best Accuracy: 0.876 (n_estimators = 20, max_depth = 12)

SVM

Description

SVM (Support Vector Machine) classifies the data using a hyperplane that acts as a decision boundary between different classes. Extreme data points from each class are called Support Vectors. SVM tries to find the best and optimal hyperplane which has the maximum margin from each Support Vector.

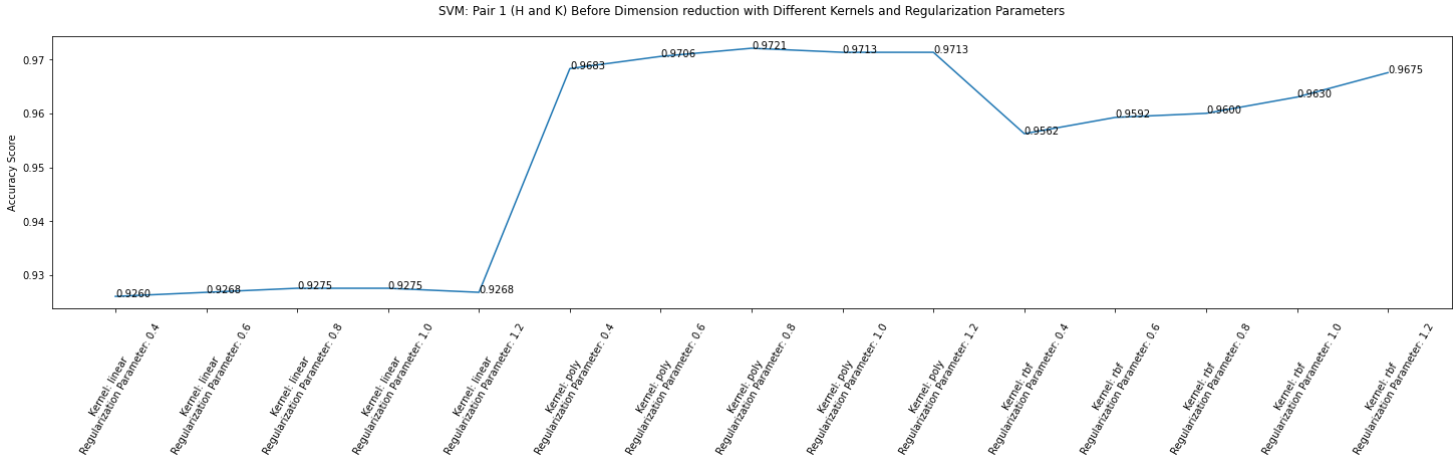
- Advantages：
  - **Regularization capabilities**: SVM has L2 Regularization feature. So, it has good generalization capabilities which prevent it from over-fitting.
  - **Handles non-linear data efficiently**: SVM can efficiently handle non-linear data using the Kernel trick.
  - **Stability**: A small change to the data does not greatly affect the hyperplane and hence the SVM. So the SVM model is stable.
- Disadvantages:
  - **Choosing an appropriate Kernel function is difficult**: Choosing an appropriate Kernel function (to handle the non-linear data) is tricky and complex. In the case of using a high dimension Kernel, it might generate too many support vectors which reduces the training speed drastically.
  - **Extensive memory requirement**: Algorithmic complexity and memory requirements of SVM are very high. It needs a lot of memory since it has to store all the support vectors in the memory and this number grows abruptly with the training dataset size.

This project used algorithm from `sklearn.svm.SVC` , and tuned 2 hyperparameters, which are `Cs = [0.4, 0.6, 0.8, 1.0, 1.2]` and `kernels = ['linear', 'poly', 'rbf']` .

Cross Validation Results**

Pair 1 H and K

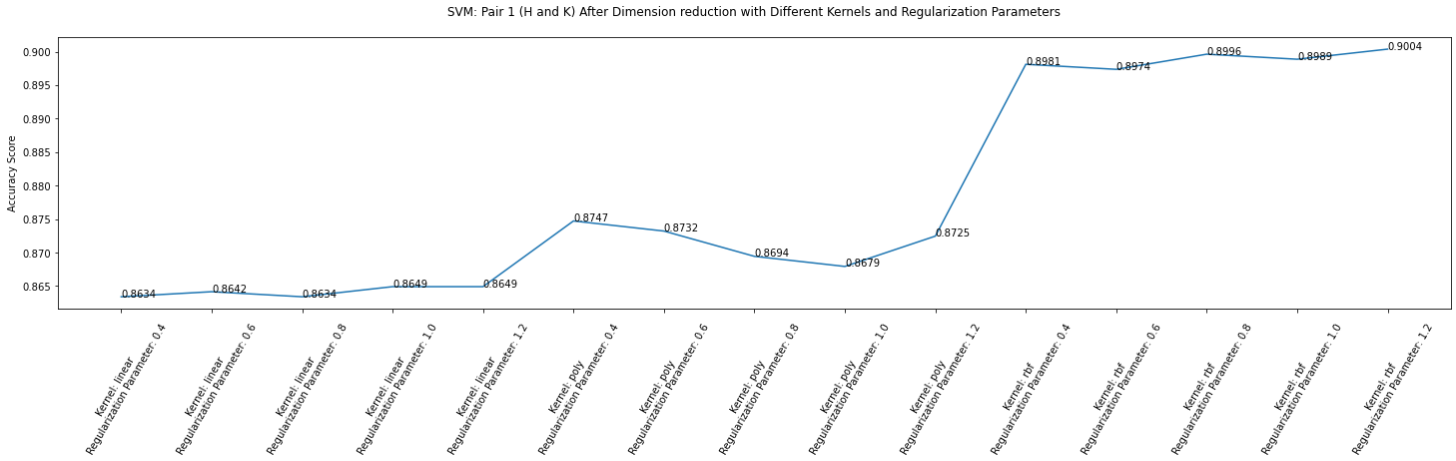
Without Dimension Reduction



Mean Accuracy: 0.953 Running Time: 2.5155

Best Accuracy: 0.9721 (kernel = poly, regularization parameter = 0.8)

With Dimension Reduction

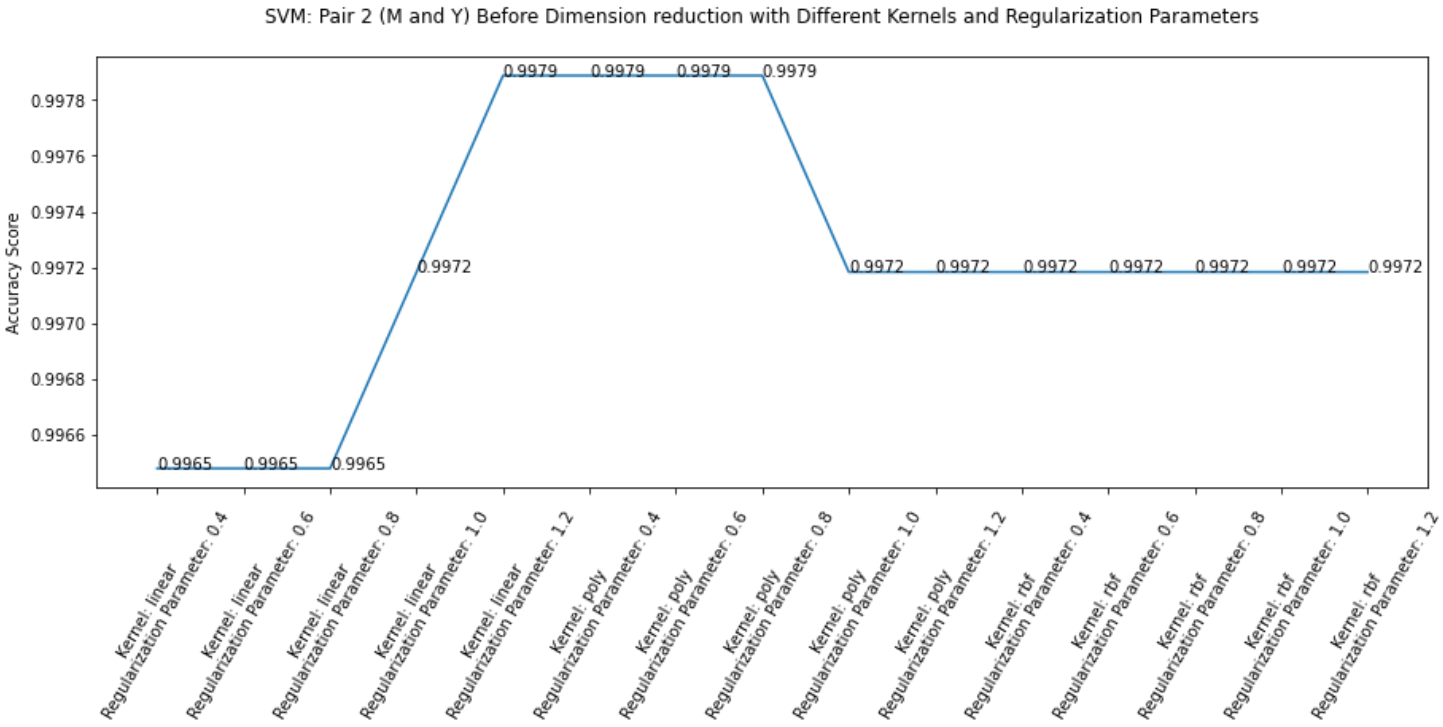


Mean Accuracy: 0.8782 Running Time: 2.4699

Best Accuracy: 0.9004 (kernel = rbf, regularization parameter = 1.2)

Pair 2 M and Y

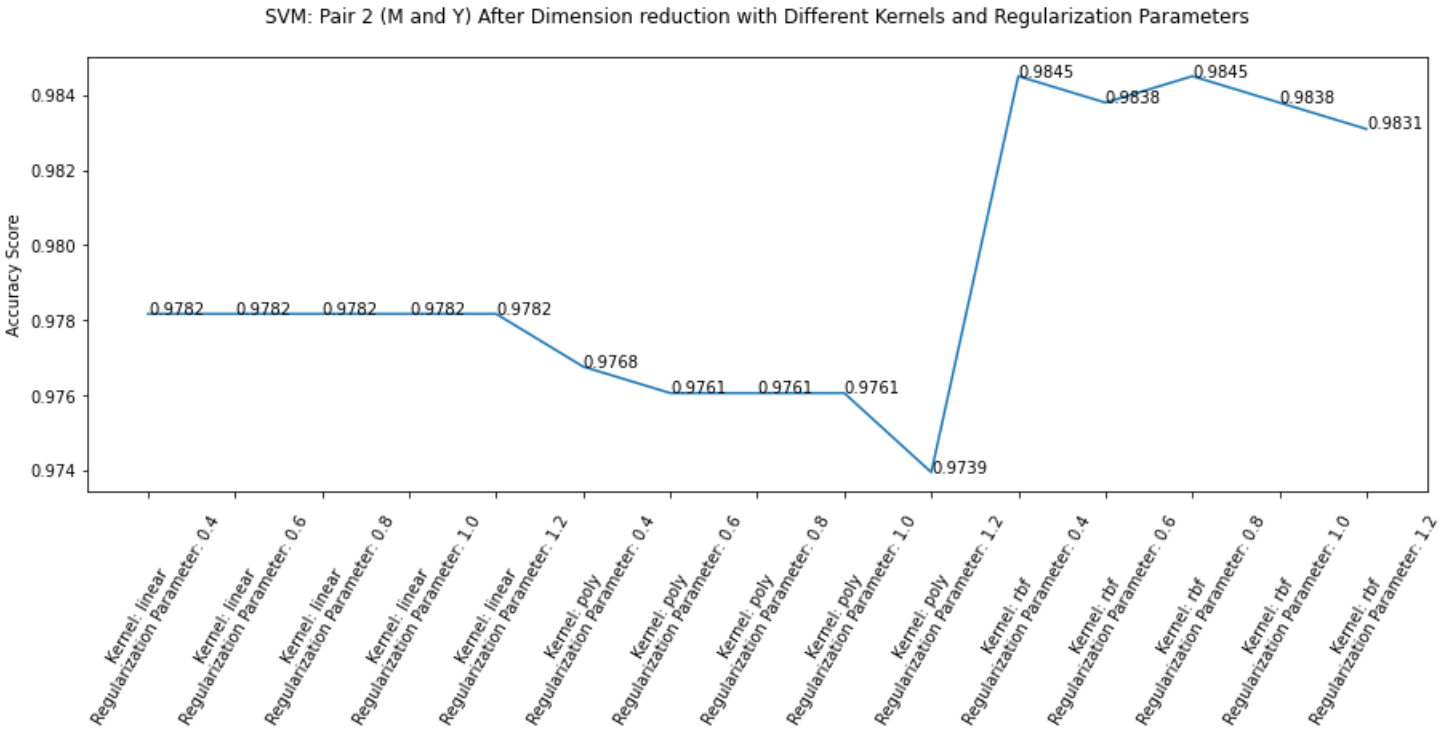
Without Dimension Reduction



Mean Accuracy: 0.9972 Running Time: 0.8041

Best Accuracy: 0.9979 (kernel = linear, regularization parameter = 1.2; kernel = poly, regularization parameter = 0.4/0.6/0.8)

With Dimension Reduction

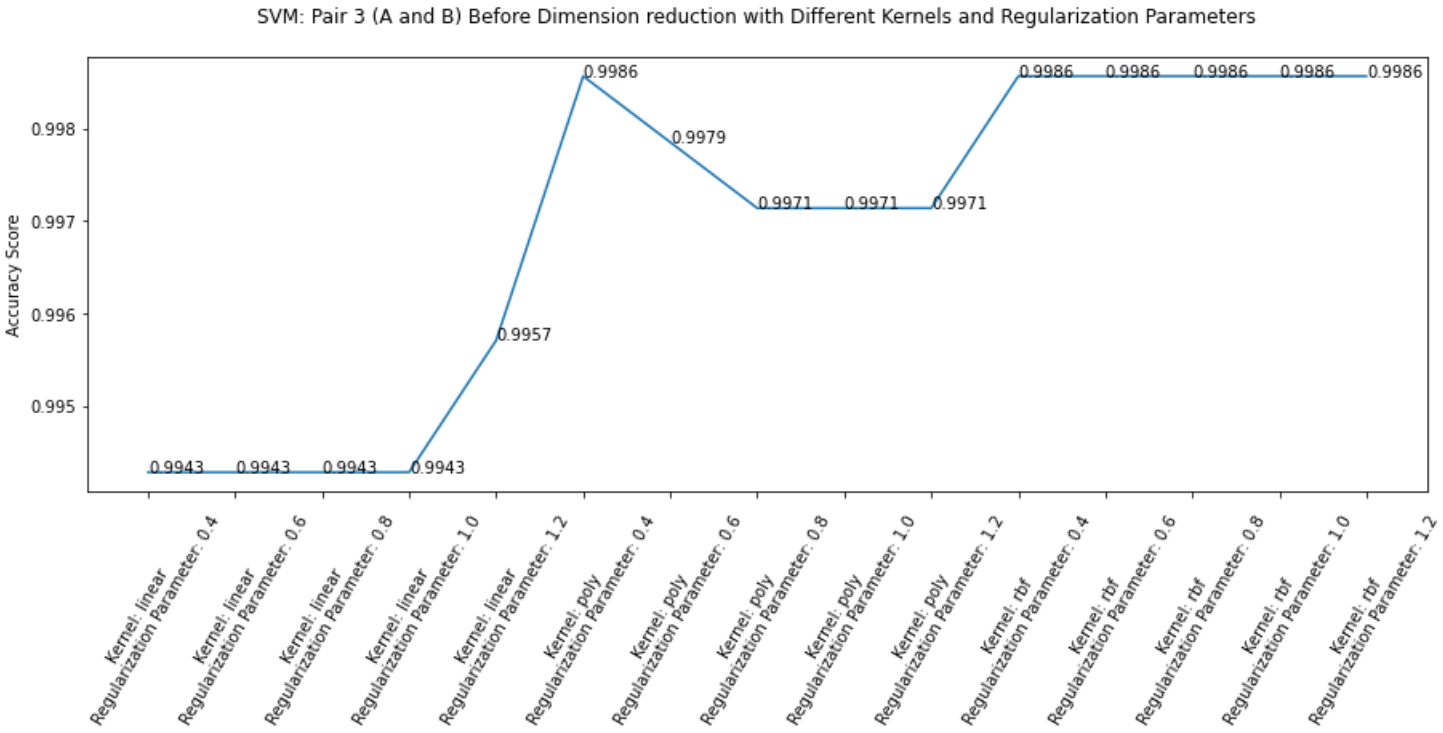


Mean Accuracy: 0.9793 Running Time: 0.7836

Best Accuracy: 0.9845 (kernel = rbf, regularization parameter = 0.4/0.8)

Pair 3 A and B

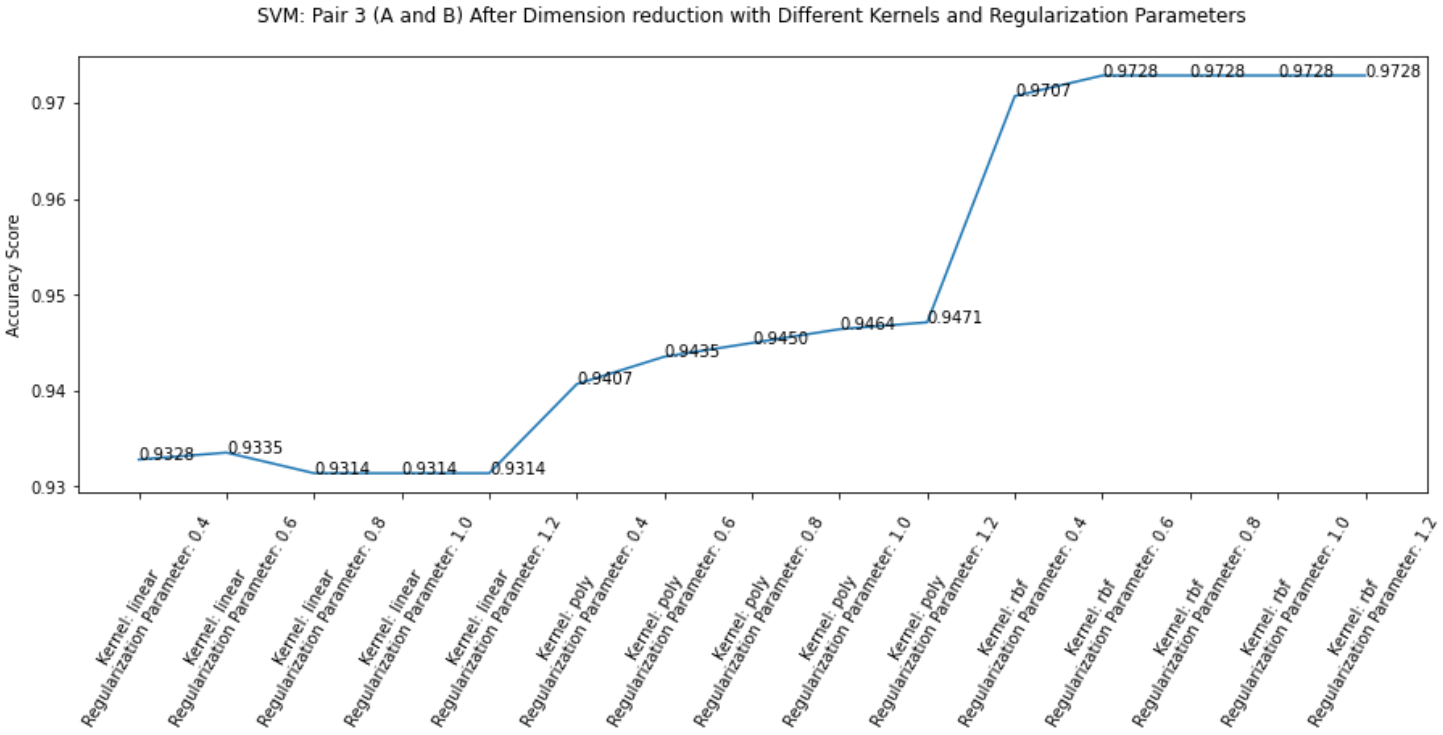
Without Dimension Reduction



Mean Accuracy: 0.9969 Running Time: 0.9303

Best Accuracy: 0.9986 (kernel = poly, regularization parameter = 0.4; kernel = rbf, regularization parameter = 0.4/0.6/0.8/1.0/1.2)

With Dimension Reduction

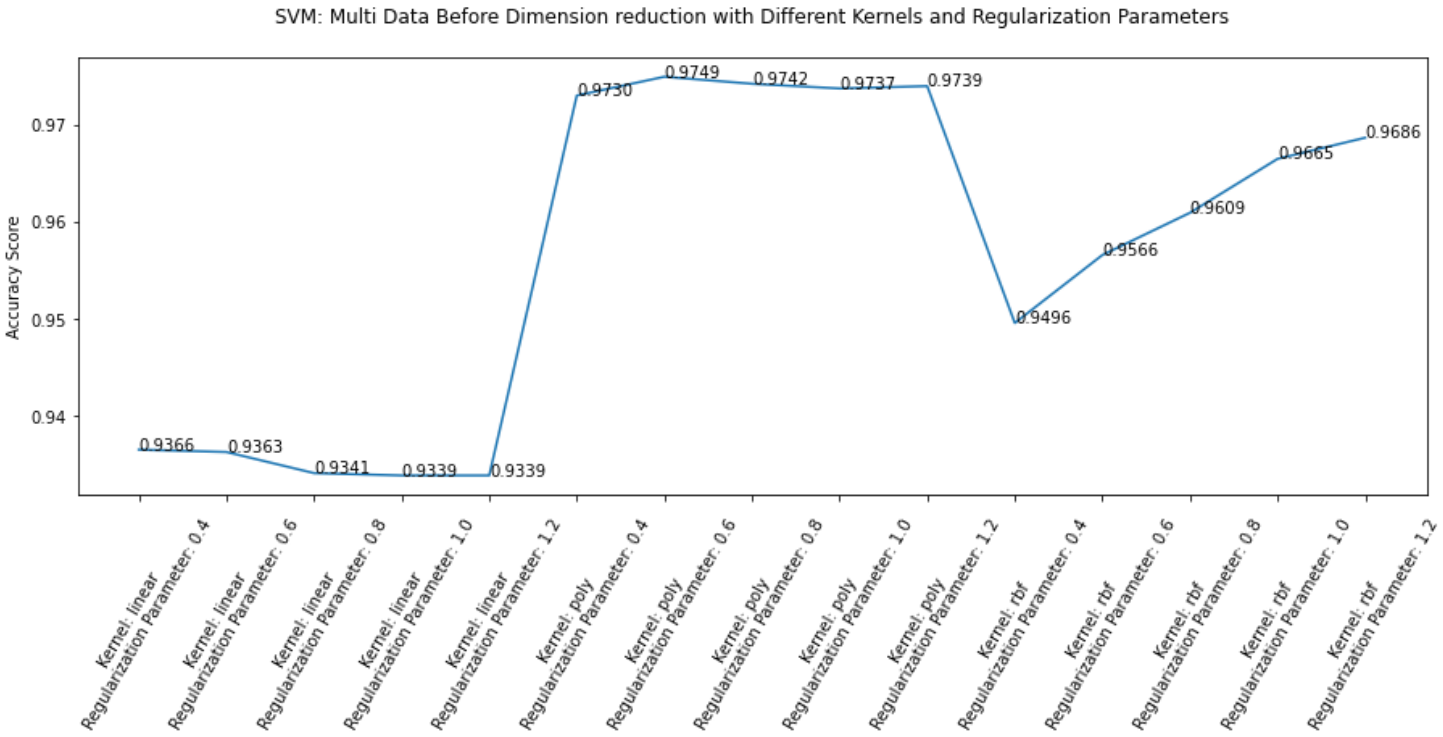


Mean Accuracy: 0.9497 Running Time: 1.2552

Best Accuracy: 0.9728 (kernel = rbf, regularization parameter = 0.6/0.8/1.0/1.2)

Multi Data

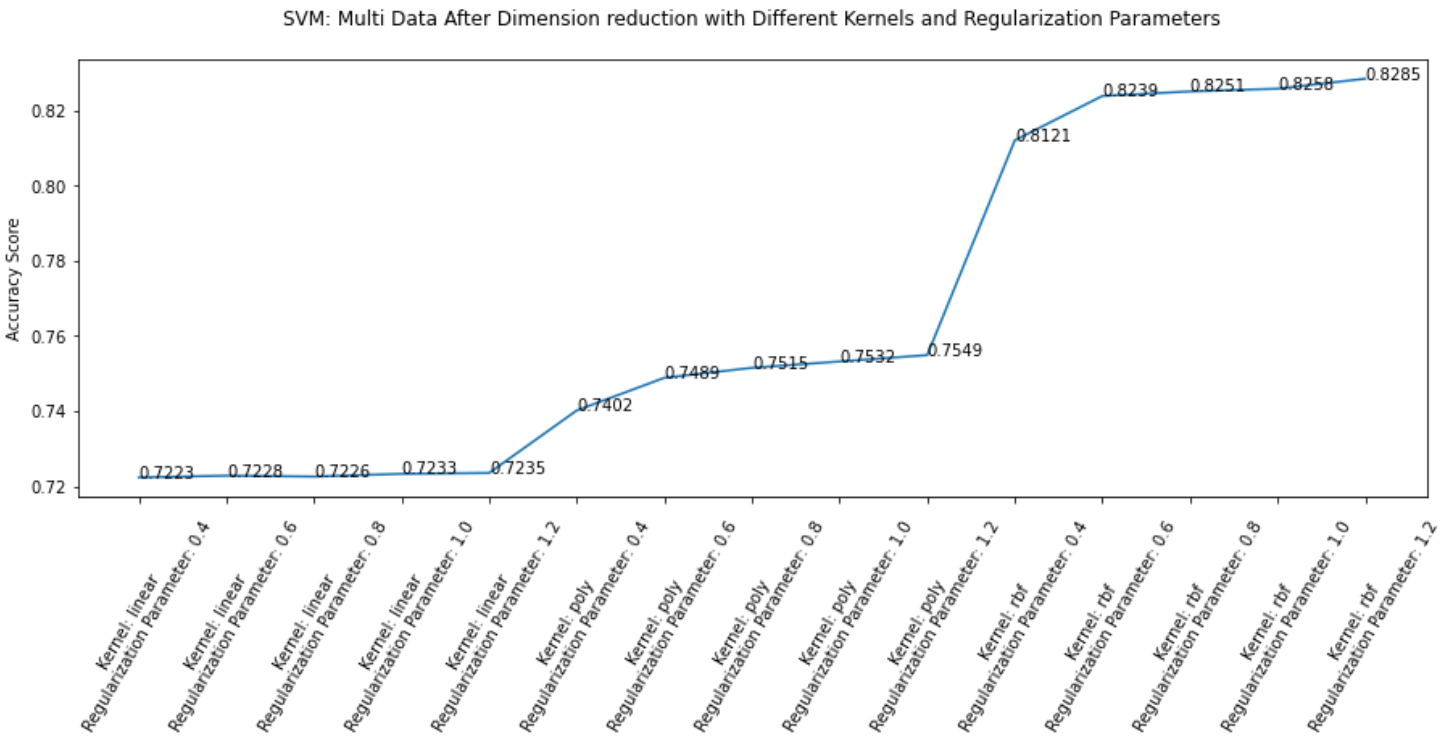
Without Dimension Reduction



Mean Accuracy: 0.9564 Running Time: 16.9737

Best Accuracy: 0.9749 (kernel = poly, regularization parameter = 0.6)

With Dimension Reduction



Mean Accuracy: 0.7652 Running Time: 26.1485

Best Accuracy: 0.8285 (kernel = rbf, regularization parameter = 1.2)

Artificial Neural Network

Description

An artificial neuron network (ANN) is a computational model that mimics the way nerve cells work in the human brain.

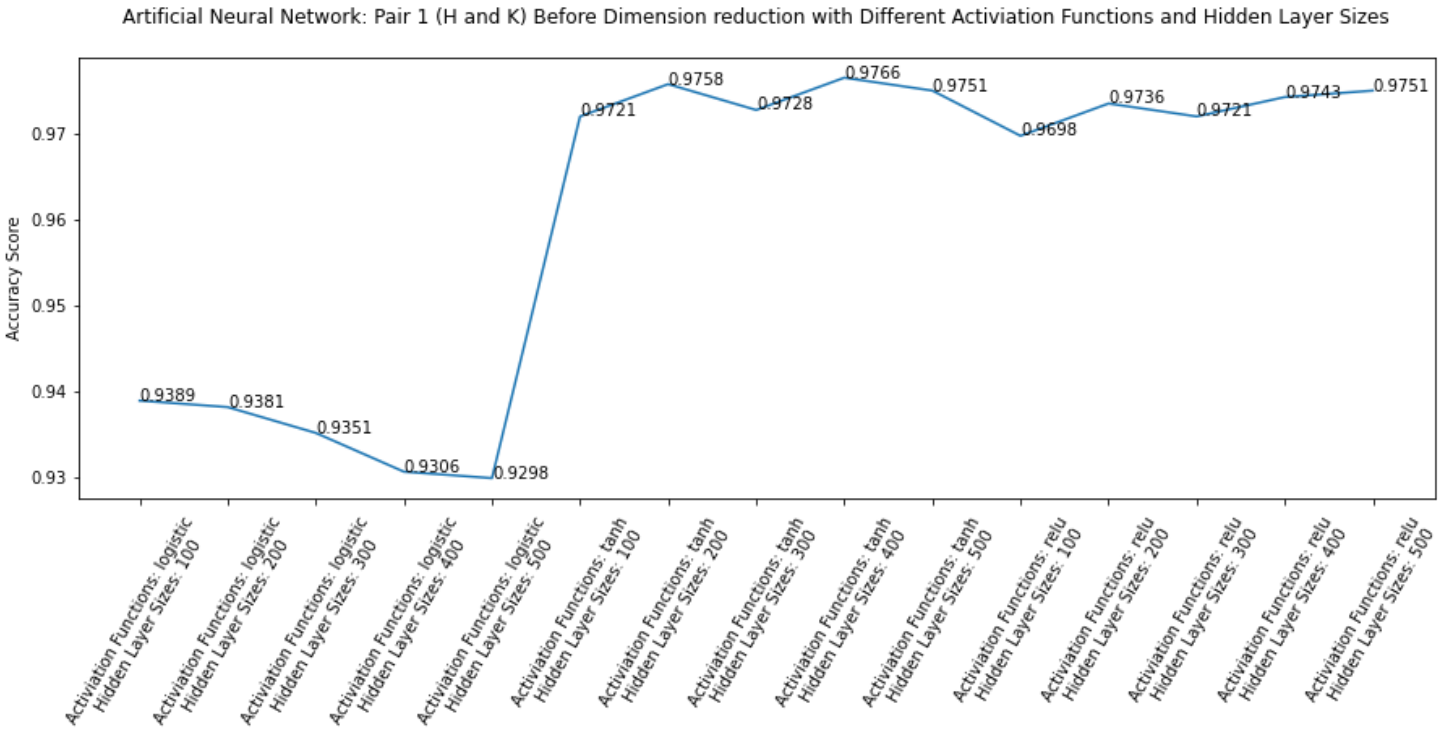
- Advantages：
  - **Fault tolerance**: Corruption of one or more cells of ANN does not prevent it from generating output.
  - **Distributed memory**: For ANN to be able to learn, it is necessary to determine the examples and to teach the network according to the desired output by showing these examples to the network. The network's progress is directly proportional to the selected instances, and if the event can not be shown to the network in all its aspects, the network can produce incorrect output
  - **Parallel processing ability**: Artificial neural networks have numerical strength that can perform more than one job at the same time.
- Disadvantages:
  - **Unexplained functioning of the network**: When ANN gives a probing solution, it does not give a clue as to why and how. This reduces trust in the network.
  - **Assurance of proper network structure**: There is no specific rule for determining the structure of artificial neural networks. The appropriate network structure is achieved through experience and trial and error.

This project used algorithm from `sklearn.neural_network.MLPClassifier` , and tuned 2 hyperparameters, which are `hidden_layer_sizes = [100, 200, 300, 400,500]` and `activations = ['logistic', 'tanh', 'relu']]` .

Cross Validation Results**

Pair 1 H and K

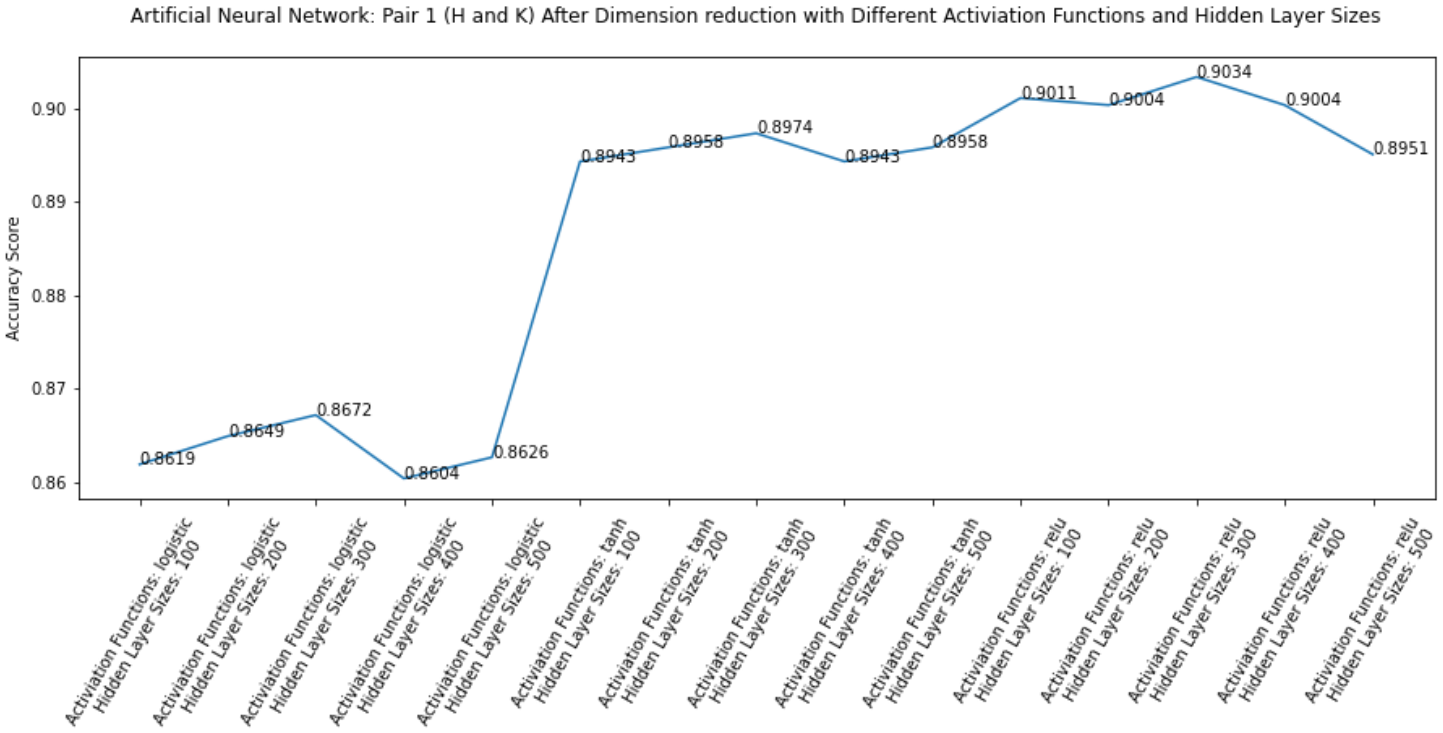
Without Dimension Reduction



Mean Accuracy: 0.9607 Running Time: 195.1661

Best Accuracy: 0.9766 (activation = tanh, hidden layer sizes = 400)

With Dimension Reduction

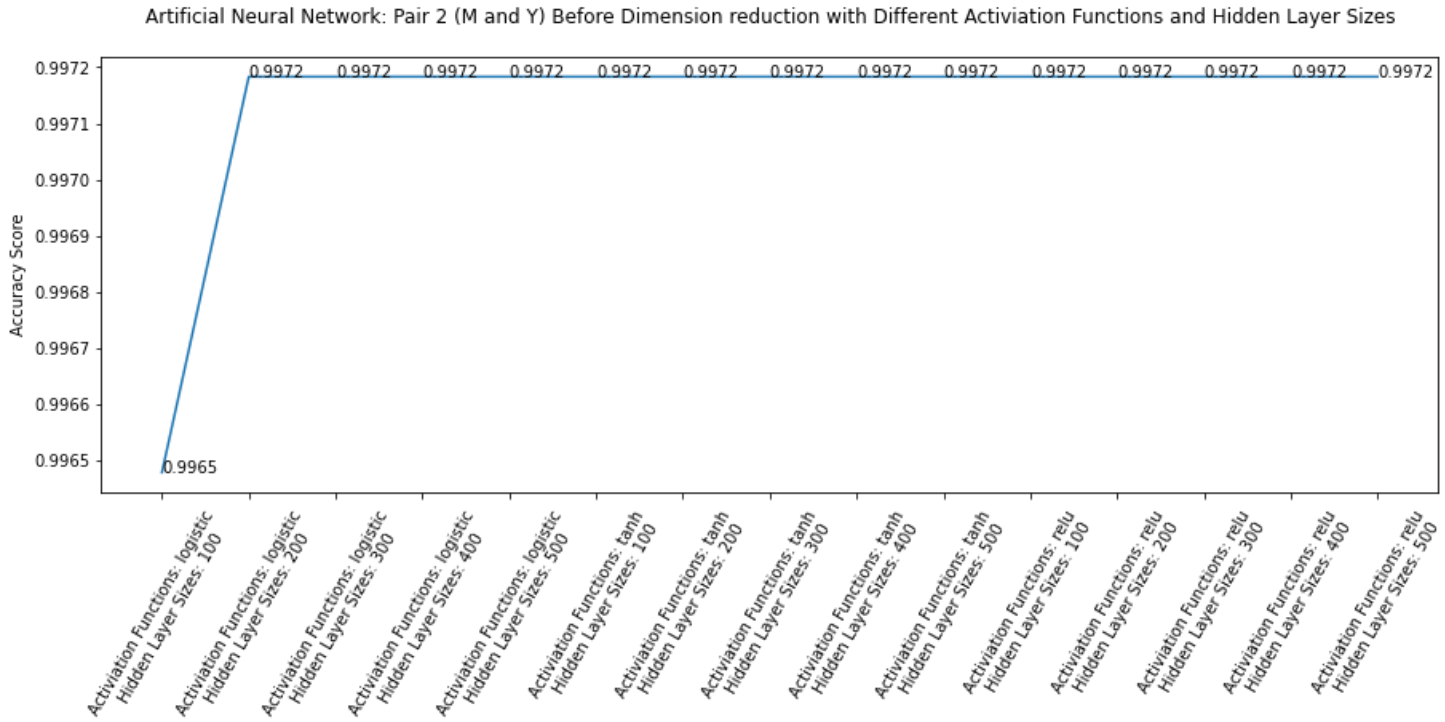


Mean Accuracy: 0.8863 Running Time: 174.353

Best Accuracy: 0.9034 (activation = relu, hidden layer sizes = 300)

Pair 2 M and Y

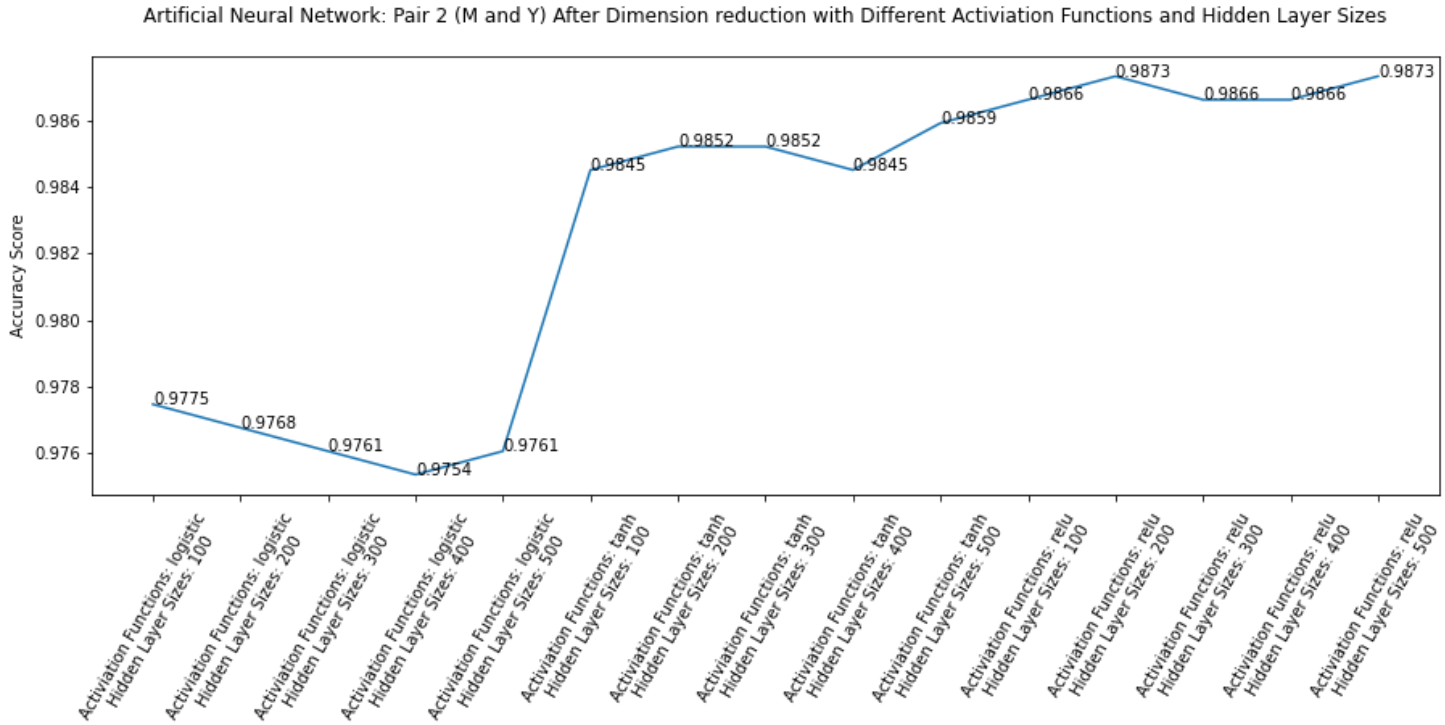
Without Dimension Reduction



Mean Accuracy: 0.9971 Running Time: 196.0437

Best Accuracy: 0.9972 (Except activation = logistic, hidden layer sizes = 100)

With Dimension Reduction

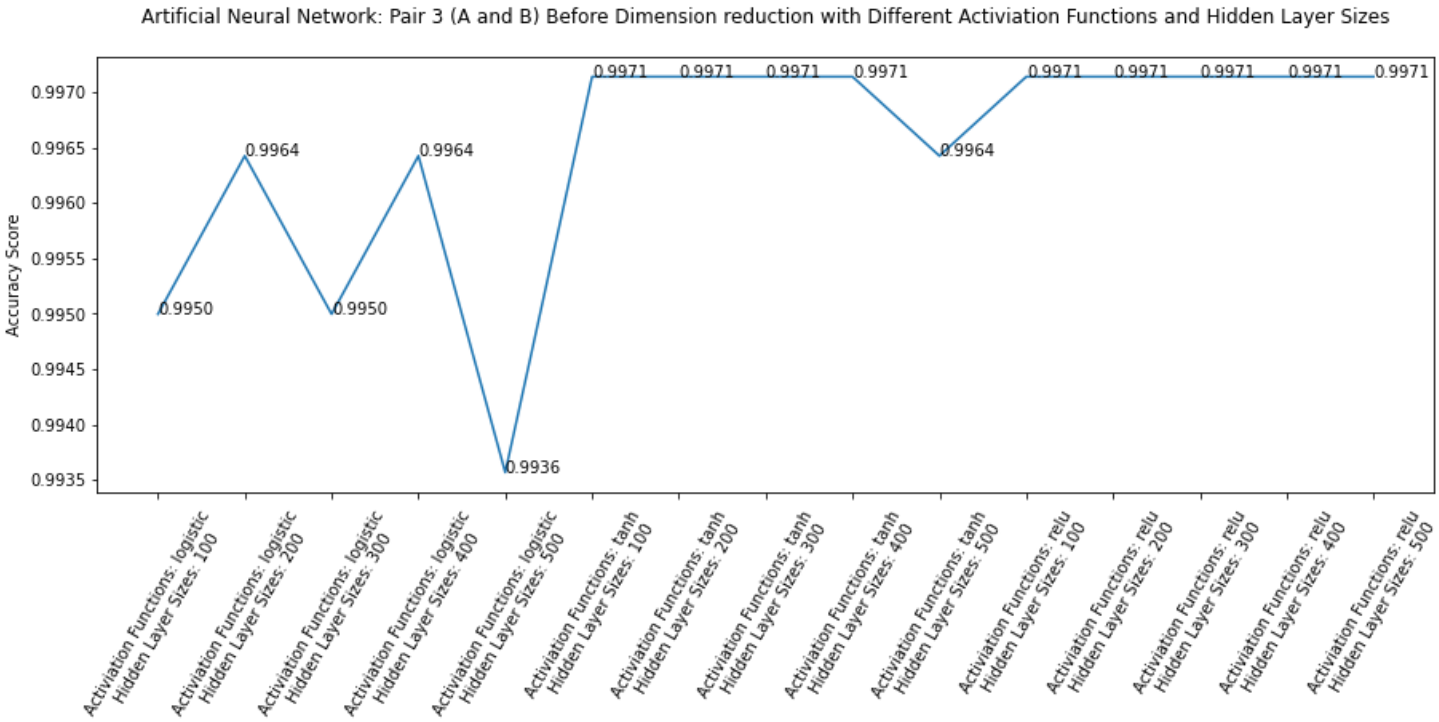


Mean Accuracy: 0.9828 Running Time: 195.1248

Best Accuracy: 0.9873 (activation = relu, hidden layer sizes = 200/500)

Pair 3 A and B

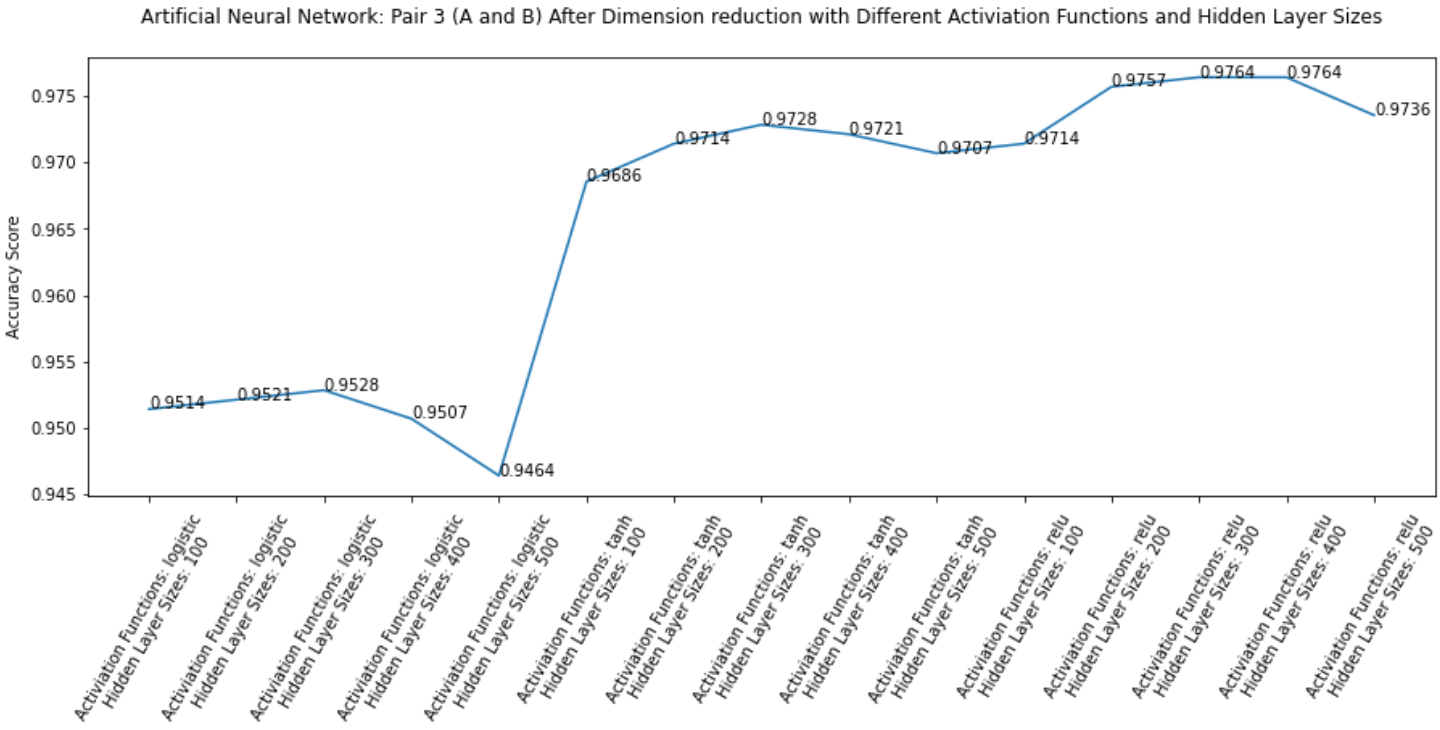
Without Dimension Reduction



Mean Accuracy: 0.9965 Running Time: 210.4061

Best Accuracy: 0.9971 (activation = tanh, hidden layer sizes = 100/200/300/400; activation = relu, hidden layer sizes = 100/200/300/400/500)

With Dimension Reduction



Mean Accuracy: 0.9655 Running Time: 180.8732

Best Accuracy: 0.9764 (activation = relu, hidden layer sizes = 300/400)

Extra Trees

Description

Extra Trees Classifier is a type of ensemble learning technique that aggregates the results of multiple de-correlated decision trees collected in a “forest” to output its classification result.

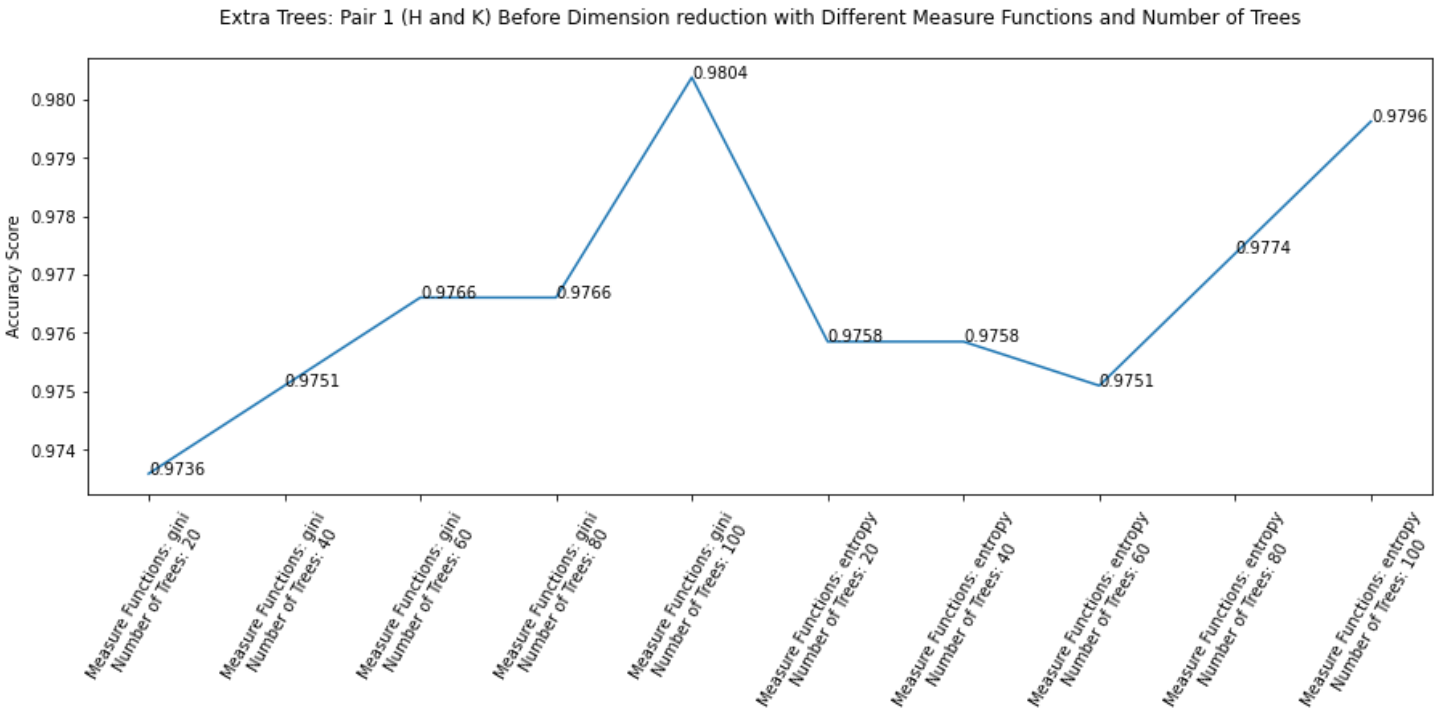
- Advantages：
  - **Reduction in bias**: This is in terms of sampling from the entire dataset during the construction of the trees. In general, different subsets of the data may introduce different biases in the results obtained, hence Extra Trees prevents this by sampling the entire dataset.
  - **Reduction variance**: This is a result of the randomized splitting of nodes within the decision trees, hence the algorithm is not heavily influenced by certain features or patterns in the dataset.
- Disadvantages
  - **Easy affected by data change**: Same as random forest, a slight change in the data can cause a large change in the structure of the extra trees causing instability.

This project used algorithm from `sklearn.ensemble.ExtraTreesClassifier` , and tuned 2 hyperparameters, which are `n_estimators = [20, 40, 60, 80, 100]` and `criterions = ['gini', 'entropy']` .

Cross Validation Results**

Pair 1 H and K

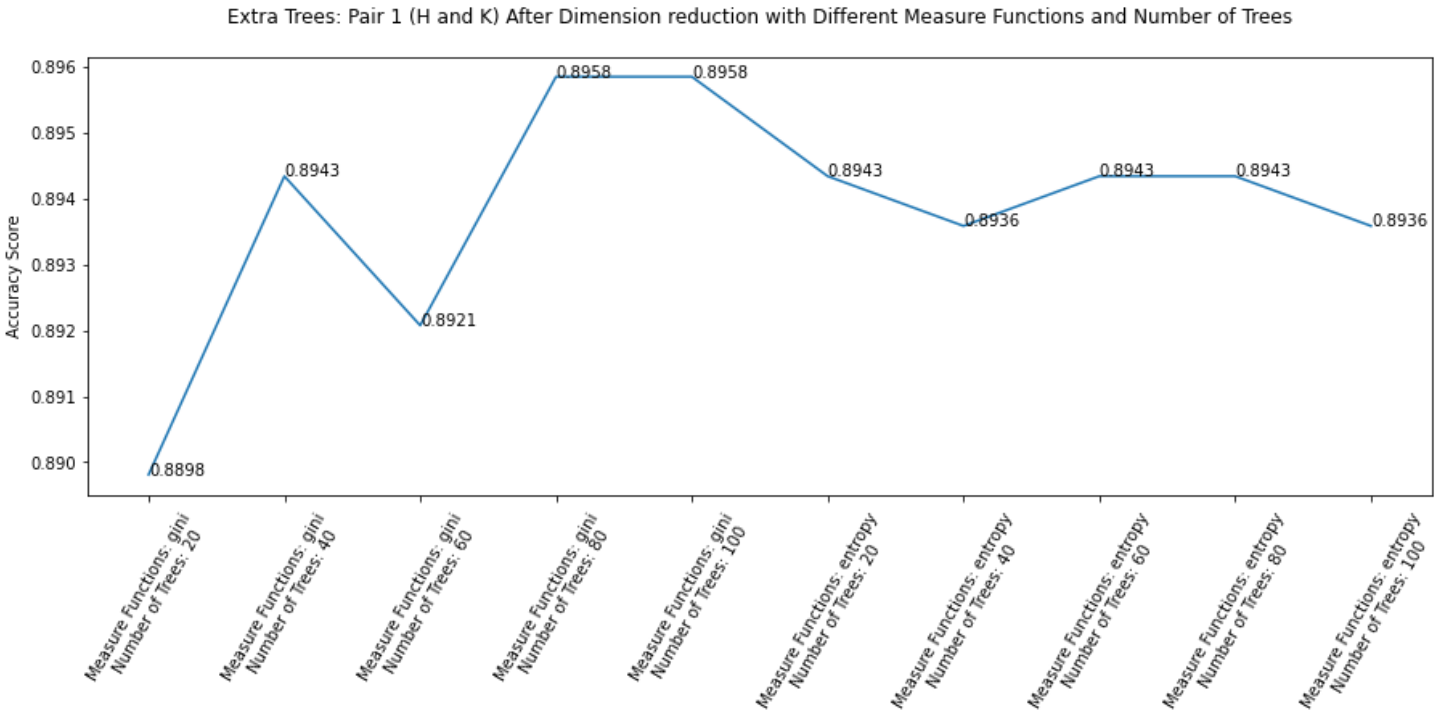
Without Dimension Reduction



Mean Accuracy: 0.9766 Running Time: 3.4654

Best Accuracy: 0.9804 (criterion = gini, n_estimator = 100)

With Dimension Reduction

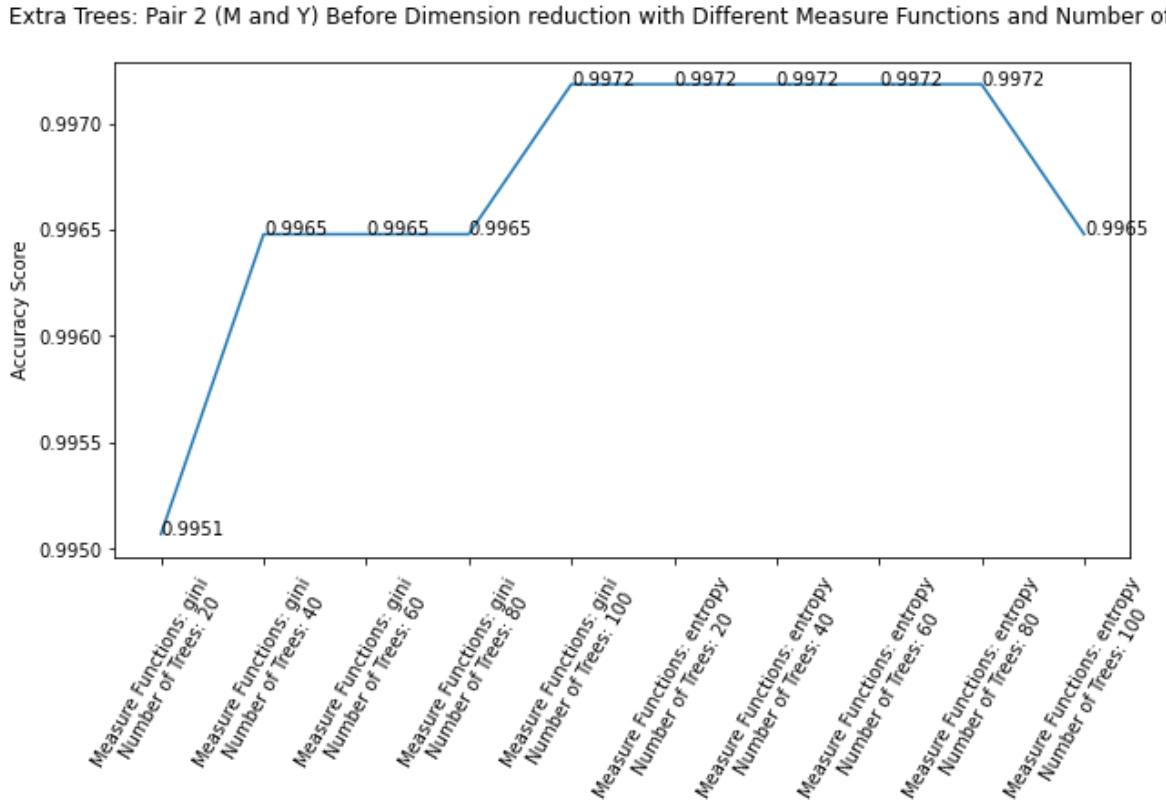


Mean Accuracy: 0.8938 Running Time: 2.9743

Best Accuracy: 0.8959 (criterion = gini, n_estimator = 80/100)

Pair 2 M and Y

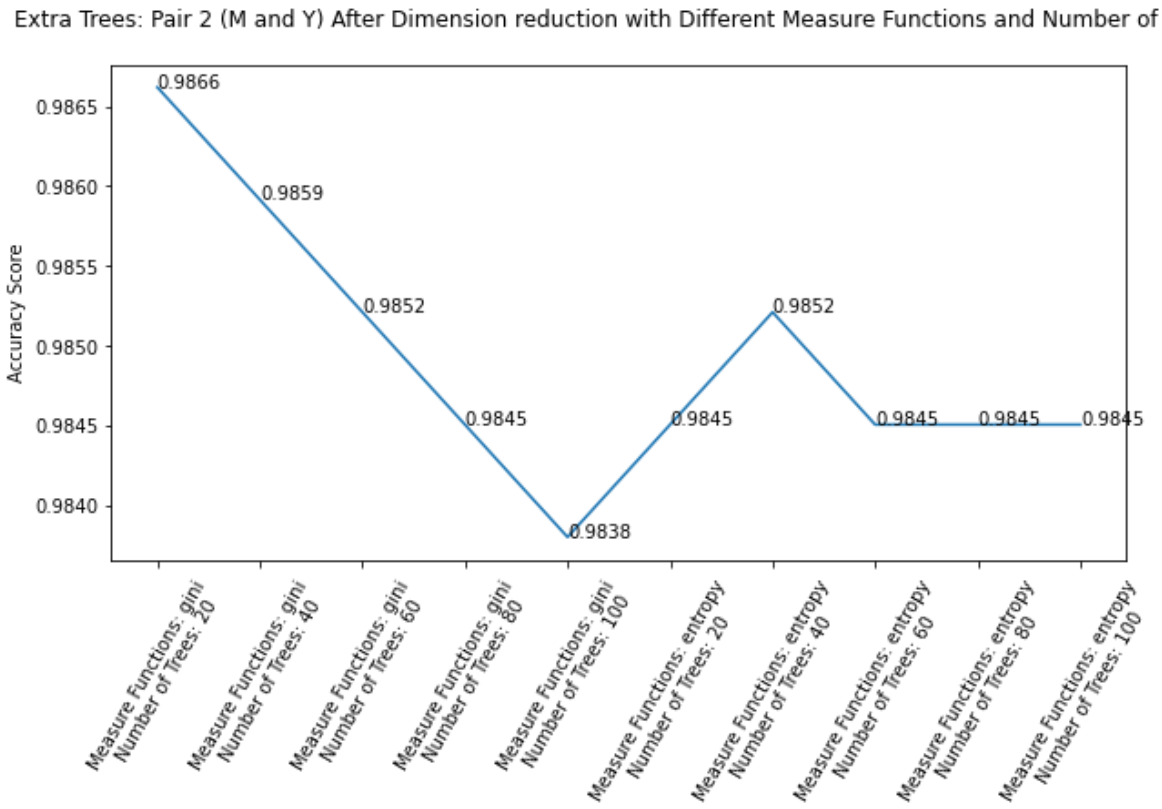
Without Dimension Reduction



Mean Accuracy: 0.9967 Running Time: 2.8379

Best Accuracy: 0.9972 (criterion = gini, n_estimator = 100; criterion = entropy, n_estimator = 20/40/60/80)

With Dimension Reduction

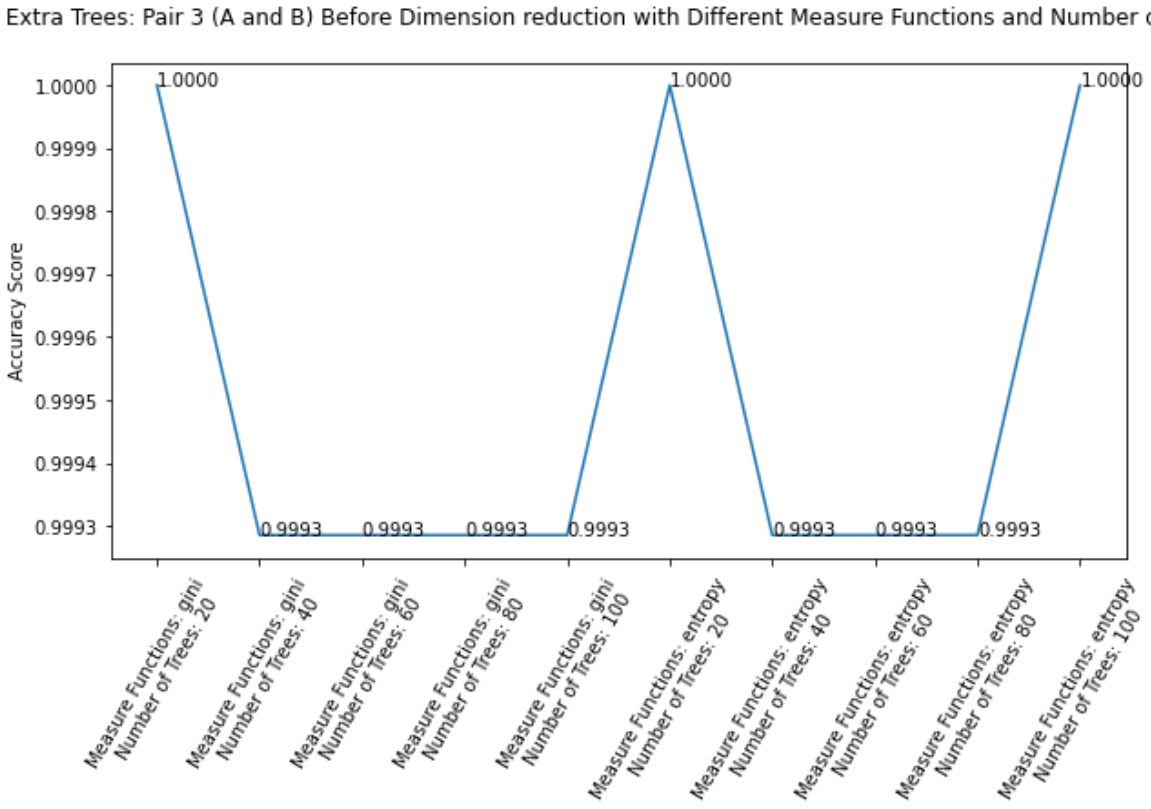


Mean Accuracy: 0.9849 Running Time: 2.5978

Best Accuracy: 0.9866 (criterion = gini, n_estimator = 20)

Pair 3 A and B

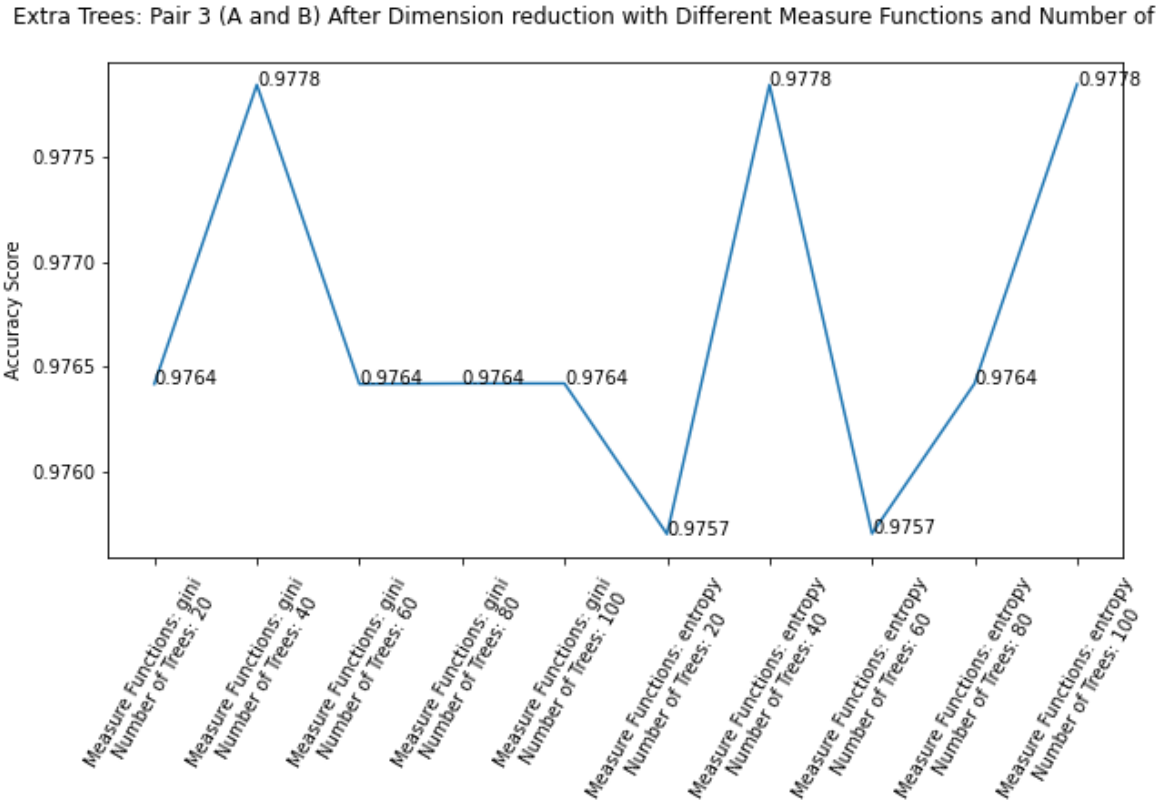
Without Dimension Reduction



Mean Accuracy: 0.9995 Running Time: 2.9602

Best Accuracy: 1.0 (criterion = gini, n_estimator = 20; criterion = entropy, n_estimator = 20/100)

With Dimension Reduction

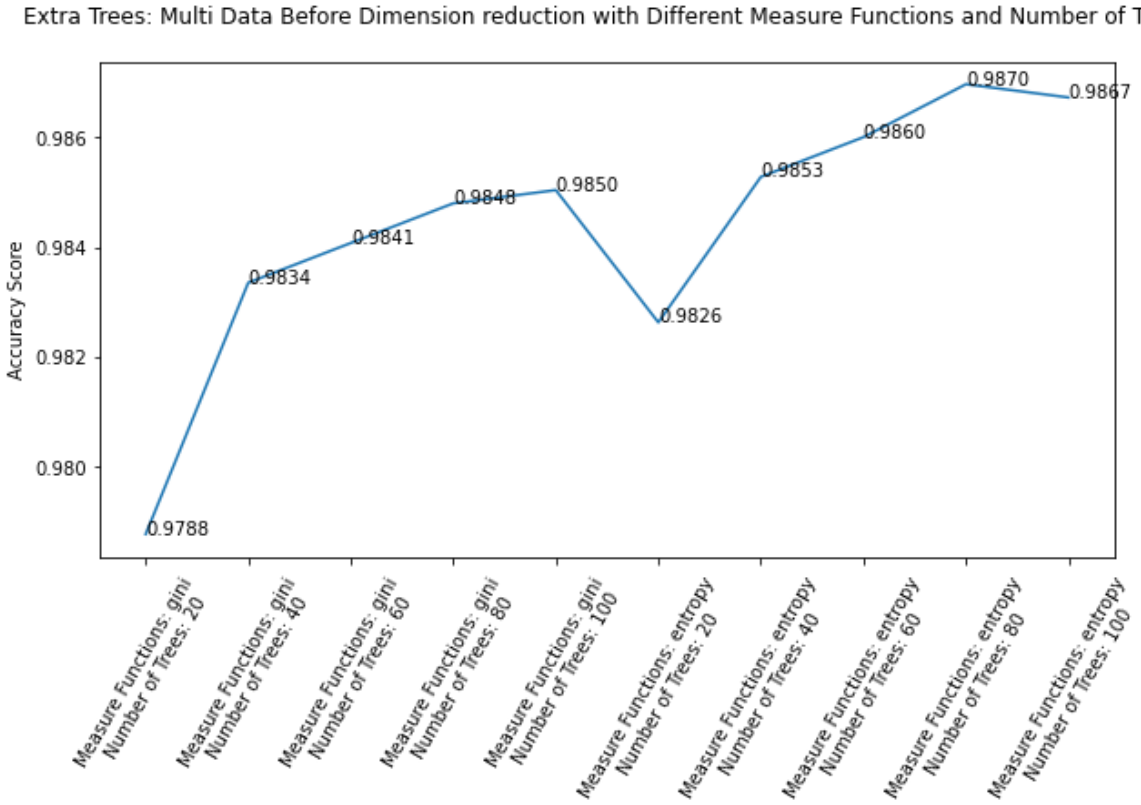


Mean Accuracy: 0.9767 Running Time: 2.7239

Best Accuracy: 0.9778 (criterion = gini, n_estimator = 40; criterion = entropy, n_estimator = 40/100)

Multi Data

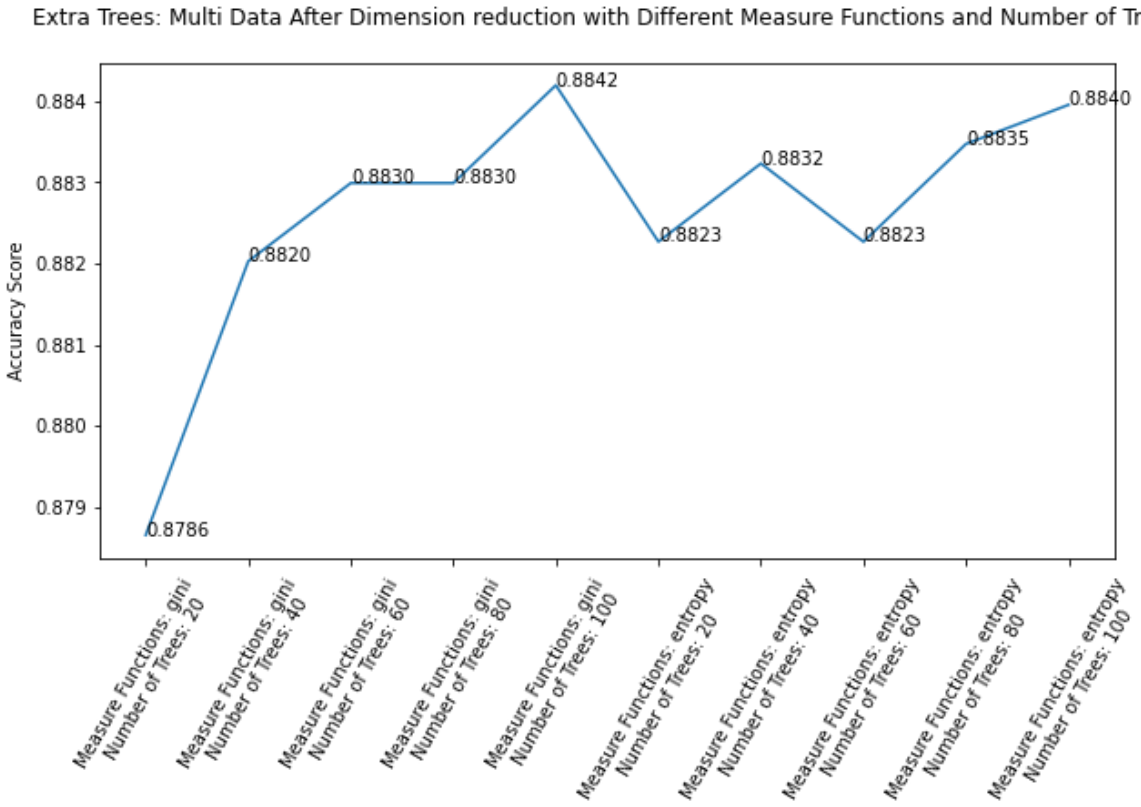
Without Dimension Reduction



Mean Accuracy: 0.9844 Running Time: 8.2623

Best Accuracy: 0.987 (criterion = entropy, n_estimator = 80)

With Dimension Reduction



Mean Accuracy: 0.8826 Running Time: 6.746

Best Accuracy: 0.8842 (criterion = gini, n_estimator = 100)

Ada Boost

Description

Ada Boost (Adaptive Boosting) classifier is a meta-estimator that begins by fitting a classifier on the original dataset and then fits additional copies of the classifier on the same dataset but where the weights of incorrectly classified instances are adjusted such that subsequent classifiers focus more on difficult cases.

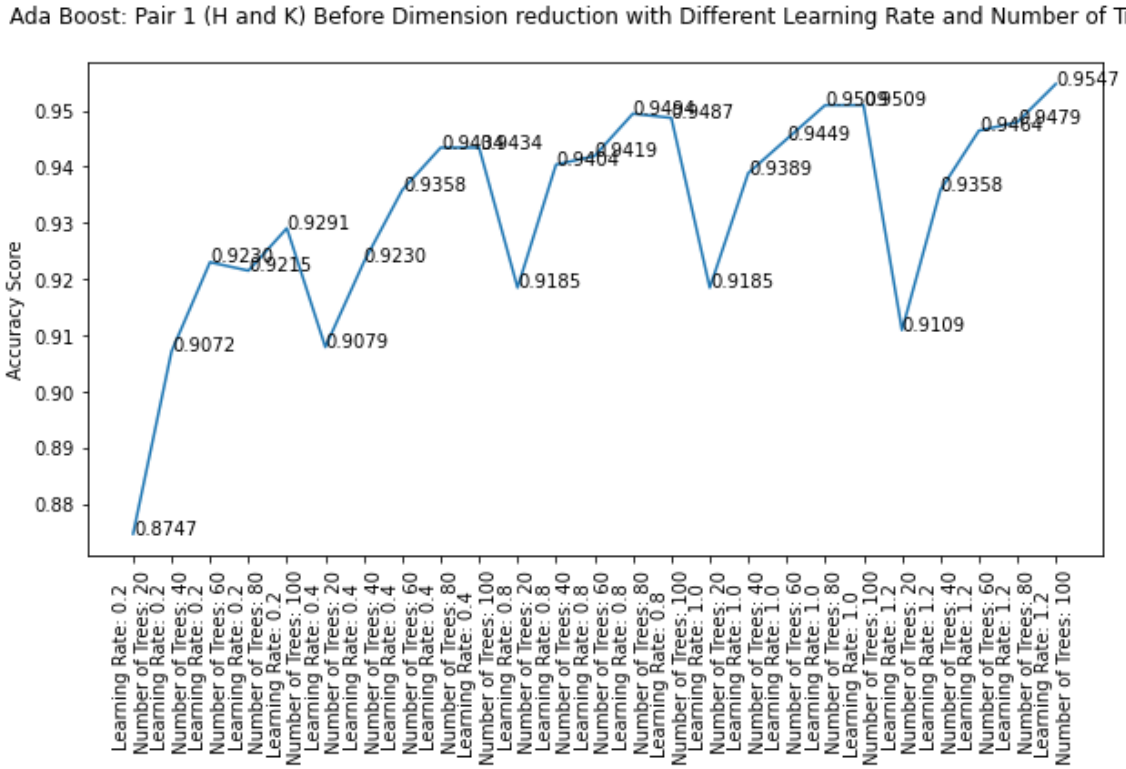
- Advantages：
  - **Fast speed and easy to implement**.
  - **Collaborative**: Ada Boost can be combined with any other machine learning algorithm without the requirement of fine-tuning parameters.
  - **Ada Boost can handle text as well as numeric data**.
- Disadvantages:
  - **Vulnerable to noise**: Due to Ada boost's empirical evidence, it's potentially vulnerable to noise
  - **Highly susceptible to outlier**.

This project used algorithm from `sklearn.ensemble.AdaBoostClassifier` , and tuned 2 hyperparameters, which are `learning_rates = [0.2, 0.4, 0.8, 1.0, 1.2]` and `n_estimators = [20, 40, 60, 80, 100]` .

Cross Validation Results**

Pair 1 H and K

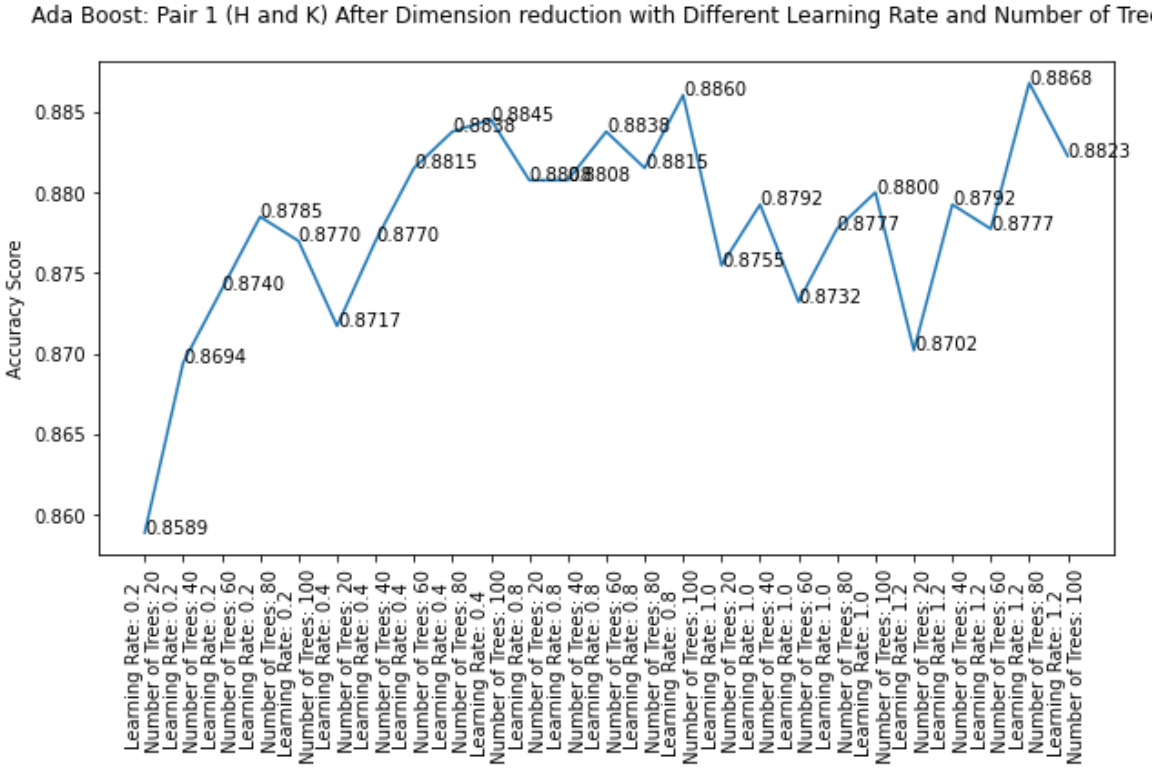
Without Dimension Reduction



Mean Accuracy: 0.9323 Running Time: 15.5958

Best Accuracy: 0.9547 (learning rate = 1.2, n_estimators = 100)

With Dimension Reduction

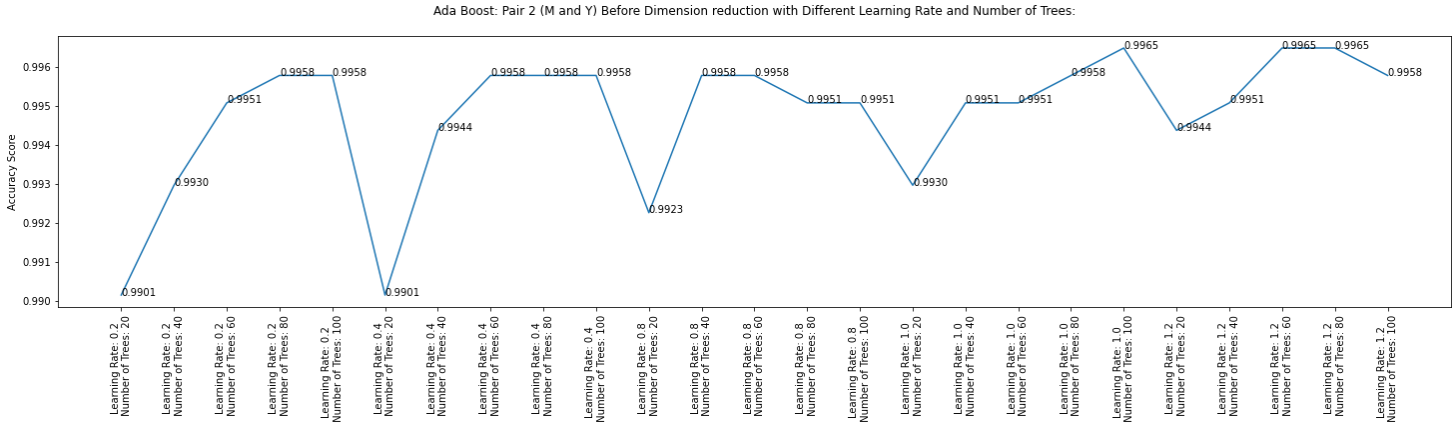


Mean Accuracy: 0.878 Running Time: 14.8702

Best Accuracy: 0.8868 (learning rate = 1.2, n_estimators = 80)

Pair 2 M and Y

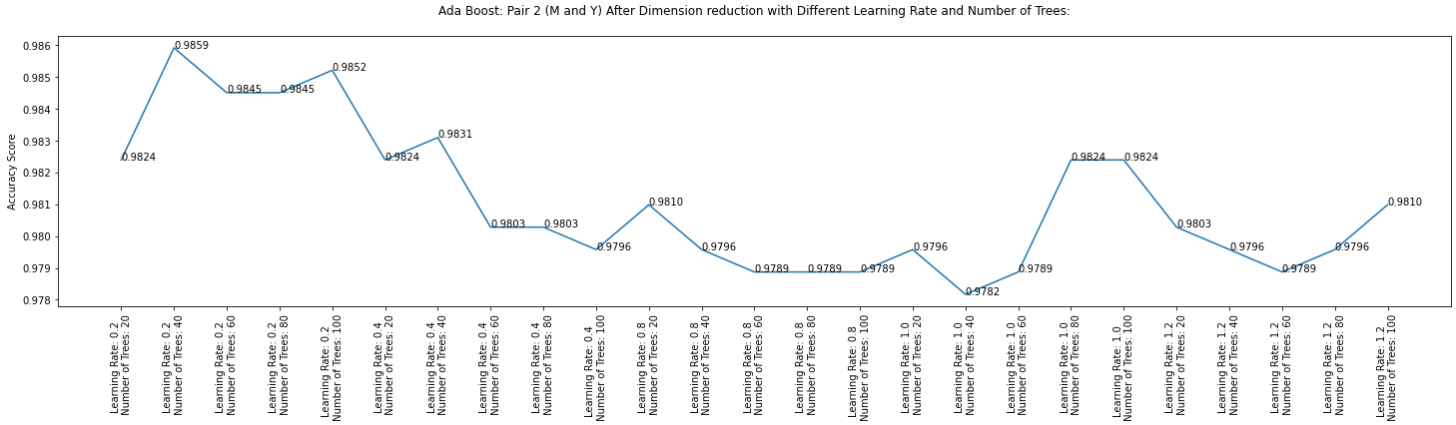
Without Dimension Reduction



Mean Accuracy: 0.9948 Running Time: 16.5292

Best Accuracy: 0.9965 (learning rate = 1.0, n_estimators = 100; learning rate = 1.2, n_estimators = 60/80)

With Dimension Reduction

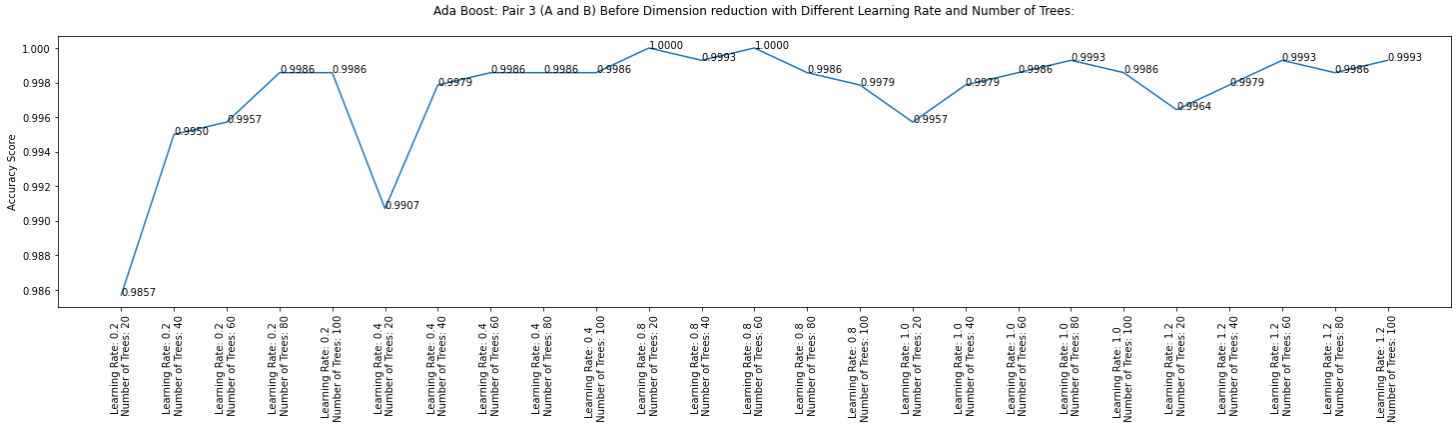


Mean Accuracy: 0.981 Running Time: 14.6039

Best Accuracy: 0.9859 (learning rate = 0.2, n_estimators = 40)

Pair 3 A and B

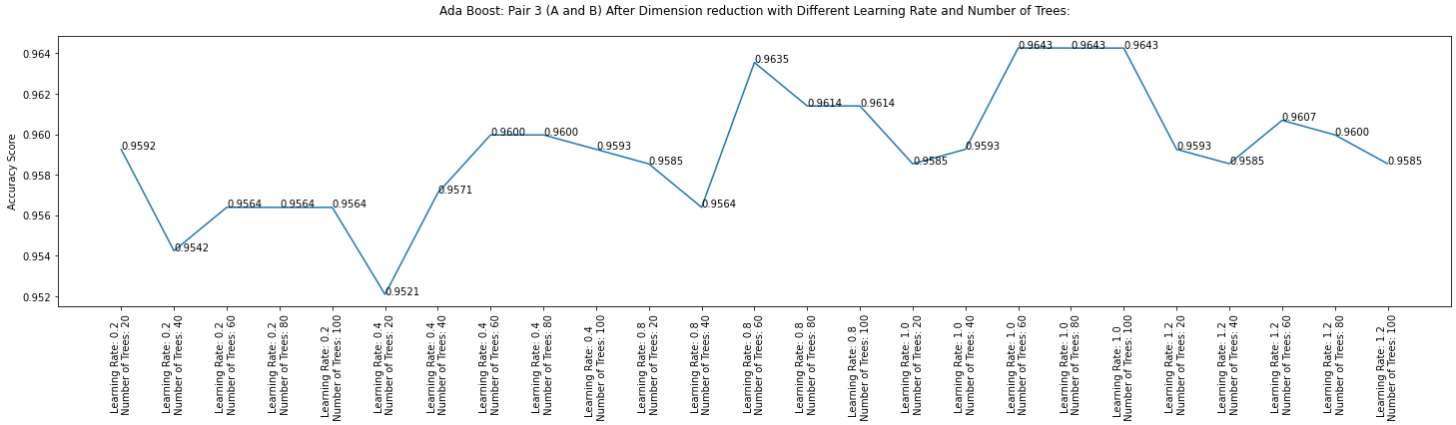
Without Dimension Reduction



Mean Accuracy: 0.9974 Running Time: 16.3481

Best Accuracy: 1.0 (learning rate = 0.8, n_estimators = 20/60)

With Dimension Reduction

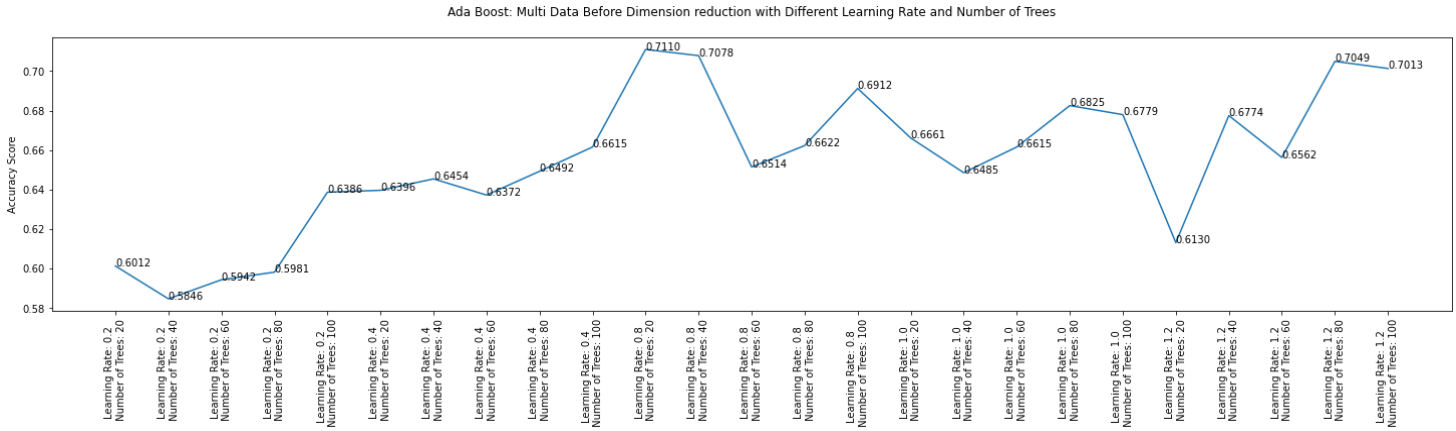


Mean Accuracy: 0.9592 Running Time: 14.0689

Best Accuracy: 0.9643 (learning rate = 1.0, n_estimators = 60/80/100)

Multi Data

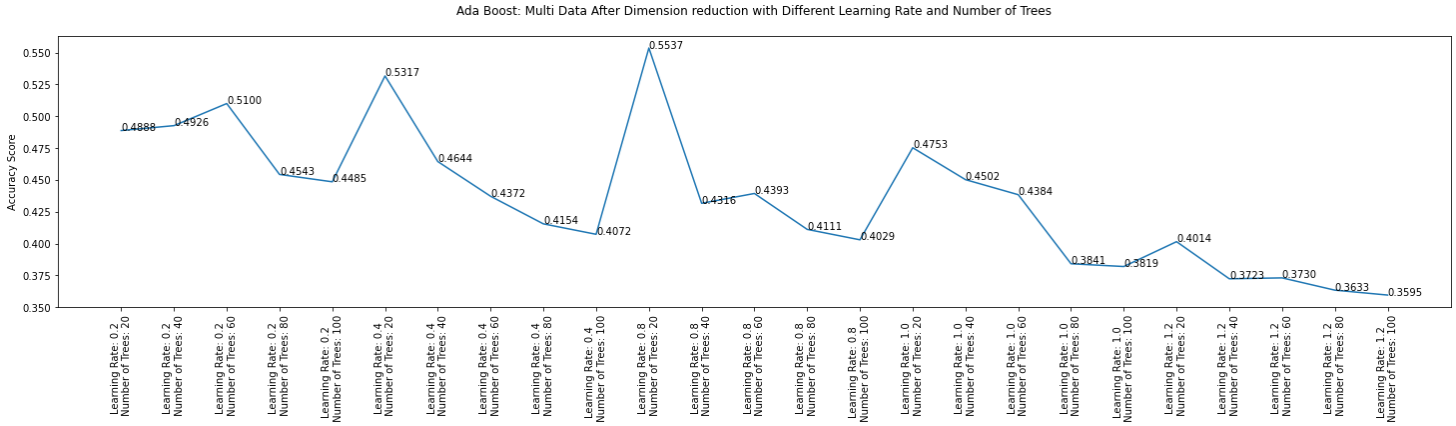
Without Dimension Reduction



Mean Accuracy: 0.6545 Running Time: 42.804

Best Accuracy: 0.711 (learning rate = 0.8, n_estimators = 20)

With Dimension Reduction



Mean Accuracy: 0.4355 Running Time: 41.2981

Best Accuracy: 0.5537 (learning rate = 0.8, n_estimators = 20)

Discussion

Performance and Running Time Comparison Tables

Mean Accuracy	K-Nearest Neighbors	Decision Tree	Random Forest	SVM	Artificial Neural Network	Extra Trees	Ada Boost	Mean
Pair 1 HK (Before)	0.943	0.8854	0.9445	0.953	0.9607	0.9766	0.9323	0.9422
Pair1 HK (After)	0.8921	0.8734	0.8946	0.8782	0.8863	0.8938	0.878	0.8852
Pair 2 MY (Before)	0.998	0.9858	0.9942	0.9972	0.9971	0.9967	0.9948	0.9948
Pair 2 MY (After)	0.9884	0.9783	0.9819	0.9793	0.9828	0.9849	0.981	0.9824
Pair 3 AB (Before)	0.9998	0.979	0.9988	0.9969	0.9965	0.9995	0.9974	0.9954
Pair 3 AB (After)	0.9719	0.9703	0.9707	0.9497	0.9655	0.9767	0.9592	0.9663
Multi Data (Before)	0.9677	0.8071	0.9233	0.9564		0.9844	0.6545	0.8822
Multi Data (After)	0.8763	0.7913	0.826	0.7652		0.8826	0.4355	0.7628

Mean Accuracy	K-Nearest Neighbors	Decision Tree	Random Forest	SVM	Artificial Neural Network	Extra Trees	Ada Boost	Mean
Mean Result (Before)	0.9803	0.9501	0.9792	0.9824	0.9848	0.9909	0.9748	
Mean Result (After)	0.9508	0.9407	0.9491	0.9357	0.9449	0.9518	0.9394	
Mean Result (Total)	0.9655	0.9454	0.9641	0.9591	0.9648	0.9714	0.9571	

Best Accuracy	K-Nearest Neighbors	Decision Tree	Random Forest	SVM	Artificial Neural Network	Extra Trees	Ada Boost	Mean
Pair 1 HK (Before)	0.9509	0.9132	0.9668	0.9721	0.9766	0.9804	0.9547	0.9592
Pair1 HK (After)	0.9004	0.8875	0.9034	0.9004	0.9034	0.8958	0.8868	0.8968
Pair 2 MY (Before)	0.9986	0.9908	0.9965	0.9979	0.9972	0.9972	0.9965	0.9964
Pair 2 MY (After)	0.9894	0.9796	0.9852	0.9845	0.9873	0.9866	0.9859	0.9855
Pair 3 AB (Before)	1	0.9843	1	0.9986	0.9971	1	1	0.9971
Pair 3 AB (After)	0.9734	0.9728	0.975	0.9728	0.9764	0.9778	0.9643	0.9732
Multi Data (Before)	0.9737	0.8999	0.9744	0.9749		0.987	0.711	0.9202
Multi Data (After)	0.8806	0.8577	0.876	0.8285		0.8842	0.5537	0.8135
Mean Result (Before)	0.9832	0.9628	0.9878	0.9895	0.9903	0.9925	0.9837	

Best Accuracy	K-Nearest Neighbors	Decision Tree	Random Forest	SVM	Artificial Neural Network	Extra Trees	Ada Boost	Mean
Mean Result (After)	0.9544	0.9466	0.9545	0.9526	0.9557	0.9534	0.9457	
Mean Result (Total)	0.9688	0.9547	0.9712	0.9711	0.973	0.973	0.9647	

Running Time (s)	K-Nearest Neighbors	Decision Tree	Random Forest	SVM	Artificial Neural Network	Extra Trees	Ada Boost	Mean
Pair 1 HK (Before)	0.8464	0.2719	9.8011	2.5155	195.1661	3.4654	15.5958	32.5232
Pair1 HK (After)	0.7735	0.2109	10.6117	2.4699	174.353	2.9734	14.8702	29.4661
Pair 2 MY (Before)	0.9804	0.2615	9.0003	0.8041	196.0437	2.8379	16.5295	32.3511
Pair 2 MY (After)	0.7534	0.2053	9.9746	0.7863	195.1248	2.5978	14.6039	32.0066
Pair 3 AB (Before)	0.9035	0.2766	8.82	0.9303	210.4061	2.9602	16.3481	34.3778
Pair 3 AB (After)	0.719	0.1942	9.6331	1.2552	180.8732	2.7239	14.0689	29.9239
Multi Data (Before)	3.6447	0.5648	19.1403	16.9737		8.2623	42.804	15.2316
Multi Data (After)	2.7899	0.5374	23.8312	26.1485		6.746	41.2981	16.8919

Running Time (s)	K-Nearest Neighbors	Decision Tree	Random Forest	SVM	Artificial Neural Network	Extra Trees	Ada Boost	Mean
Mean Result (Before)	0.9101	0.27	9.2071	1.4166	200.5386	3.0878	16.1578	
Mean Result (After)	0.7486	0.2035	10.0731	1.5038	183.4503	2.765	14.5143	
Mean Result (Total)	0.8294	0.2367	9.6401	1.4602	191.9945	2.9264	15.3361	

Results Analysis and Discussion

The first table shows that before dimension reduction, all classifiers' mean accuracies pass 0.95, and the Extra Trees classifier performs best with 0.9909 accuracy. But after the dimension reduction, all mean accuracies decrease and some even decrease dramatically such as the Ada Boost classifier decrease from 0.9748 to 0.9394. Therefore, dimension reduction has a negative effect on the classifier. It's may be caused by reducing some informative dimensions which lead to classifier models can't fit the dataset well. And when talking about multi-class classification, the first table states that due to the increase of classes to classify, the difficulty of the problem leads to a decrease in mean accuracy. In addition, comparing 3 binary classifications, pair 1 is the hardest to predict for all 7 classifiers and pair 3 is the easiest one. But they are easier than multi-class classification. Consider all models, Extra Tree performance best before dimension reduction and Artificial Neural Network performance best after dimension reduction. And these two models' performance on total results (the average mean accuracy of 3 results before dimension reduction and 3 results after dimension reduction) are the same (both are 0.973). So, it needs to consider another factor to determine choose which model.

The third table shows that except for random forest and SVM, all other models' running time decreased after dimension reduction, especially for Artificial Neural Network decreased from 200s to 183s. And multi-class classification undoubtedly spends more time than binary classification. And for three binary classifications, pair 3 spend the most time and pair 2 spends the least time. No matter before or after dimension reduction, the Decision Tree classifier has the fastest running time (Before: 0.27s, After: 0.2035s). But considering its mean accuracy (0.9454), choosing a model between Extra Tree and Artificial Neural Network is still better than choosing it when considering accuracy as a high priority. Obviously, the Extra Trees classifier has less running time Artificial Neural Network (2.9264s < 191.9945s), therefore, choosing the Extra Trees classifier is better than all the other 6 classifier models. If considering running time as the highest priority, choosing Decision Trees is the best choice since its mean accuracy passes the 0.95 threshold which is acceptable although worse than the Extra Trees classifier.

When given the same task of new datasets, except for the above analysis, I also will do differently considering the second table (best accuracy table). Because in some situations, people just want to get the best result without caring about cost and stability. In this way, choosing classifiers based on the second table have higher priority than other considerations. According to this thinking, Random Forest (Best Accuracy:0.9712) and SVM (Best Accuracy: 0.9711) would be chosen, and considering their mean accuracy and running time, SVM would be the better choice since it saves large time than Random Forest (1.4602s < 9.6401s).