1. **Introduction**
    a. **Your description of the problem and the practical impacts of solving it.**
        i. We have a dataset of 20,000 samples of 26 alphabets with each alphabet having 16 features that describe how the alphabet consists of. We separated the dataset into 4 kinds. i.e., pair 1 with only H and K letters, pair 2 with only M and Y letters, pair 3 with only A and B letters, and lastly all letters. In the problem, we are trying to cluster each pair of datasets with different models. We are able to compare different models in terms of their performance. By looking at the appearance of letters, we can also have a pre-sense of how certain letters are harder to classify. For instance, D and Q would make models harder to classify apart. We also have tried clustering after dimension reduction, to see whether reducing features dimension impacts the results.
    b. **Motivation for multiple classifiers. What factors should be considered in determining a classifier as the "best,".**
        i. When evaluating a model, I prioritize several key factors, including computational complexity, validation accuracy, and model interpretability. However, I also take into account the model's ability to generalize to new data and its robustness to noise. It's important that the computation cost of a model does not outweigh its value, and that it performs better than the average. A good model should be intuitive and not too far removed from common sense. Additionally, it's crucial that the model doesn't overfit the training data and can handle variations in the data without being overly sensitive. Meeting these criteria is hard, but it determines the best model.
    c. **Motivation for dimension reduction. Which methods are "better," and what factors should be considered in determining a dimension reduction method as "good" or "bad."**
        i. The motivation behind dimension reduction is to simplify data by reducing the number of variables or features, while retaining as much important information as possible. A good dimension reduction method should be computationally efficient and scalable, while preserving the relevant information and discarding the irrelevant or redundant information. On the other hand, a bad method would be one that does not achieve these traits, such as being inefficient or discarding too much important information.
    d. **Brief description of the dimension reduction method(s) you chose.**
        i. I chose Principal Component Analysis (PCA) to do feature extraction. PCA is able to transcribe data into lower dimensions by projecting onto the principal components, which are linear combinations of the original variables that capture the most variation in the data. I suppose it can capture and retain most information of the data while do dimension reduction.

e. **Speculate on the binary classification problems. Which pair of letters did you choose for the third problem? Which pair do you predict will be the easiest or hardest to classify?**
    i. I chose A and B, because they look pretty unsimilar to each other. I have also considered T and Y, or K and O, which are all letters pairs that are easy to distinguish. Harder pairs would be D and Q, O and Q, or E and F, which are pairs that are sometimes also hard for humans to distinguish if with limited information.
f. **Discuss the advantages/disadvantages of using a multi-class classifier instead of a set of binary classifiers.**
    i. Advantages of using a multi-class classifier include a simplified training process and potentially better results. This is because multi-class classifiers can input multiple data samples at the same time, which is simpler than using a single binary classifier. Possibly better results is because of its ability to look at multiple correlations between features and leverage them. However, a disadvantage of using a multi-class classifier is that it takes longer to compute compared to a single binary classifier. Additionally, the results of a multi-class classifier may be harder to interpret, as the multiplication of layers is more complicated than with binary classifiers.

2. **Results**
    a. **For each classifier:**
        i. **Brief description of the classifier and its general advantages and disadvantages**.

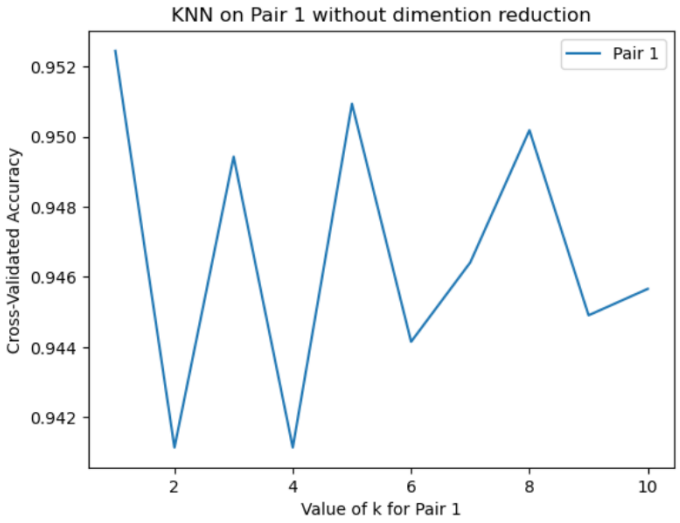
For KNN, it can handle classification and regression problems and also simple to implement and fairly easy to interpret. However, it is sensitive to how I choose the distance metrics, which will be seen in more hyperparameter tuning.
For ANN, it is able to compute more complex problems with different relationships between features. It can also handle unstructured data fairly well. However, it is prone to overfit when the model gets too complicated, and therefore also making it hard to interpret. Also, the most obvious downside is it takes the longest to train.
For SVM, it can handle both linear and non-linear data because of its nature of projecting data into higher dimension to find solutions. However, choosing the right kernel may be tricky and it is sensitive to the choice of hyperparameters.
For Random Forest, it is easy to interpret when having small dataset, and it is relatively less prone to overfitting. However, when the dataset gets larger, it takes pretty long time to train and hard to interpret, which I have tried to print out the first few forests, it was indeed hard to comprehend.

        ii. **Figure: Graph the cross validation results (from fitting the classification model *without* dimension reduction) over the range of hyperparameter values you tested. There should be three sets of values, one for each binary classification problem.**

There are a total of 4 binary classifiers: 1. KNN, 2. Random Forest. Additional classifiers: 3. ANN, and 4. SVM.

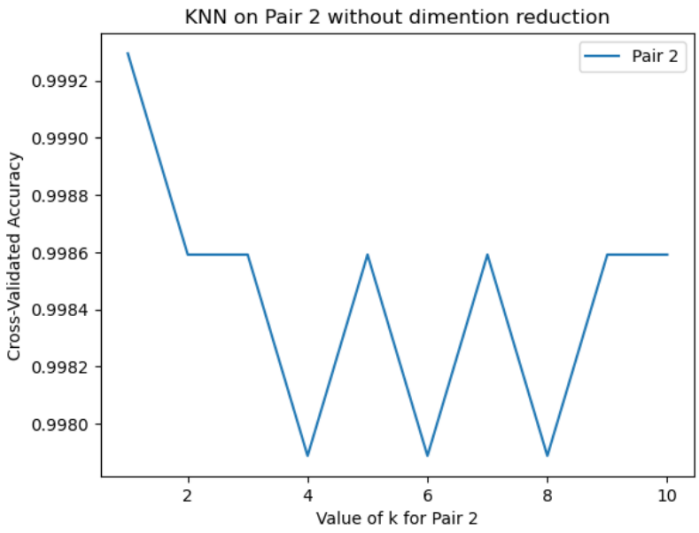
1. KNN on Pair 1 (tuning k)

```
Best k: 1
Best k accuracy:0.9524528301886793
confusion matrix
[[73  2]
 [ 3 70]]
Accuracy on testing 0.9662162162162162
testing time:  0.009001970291137695
```



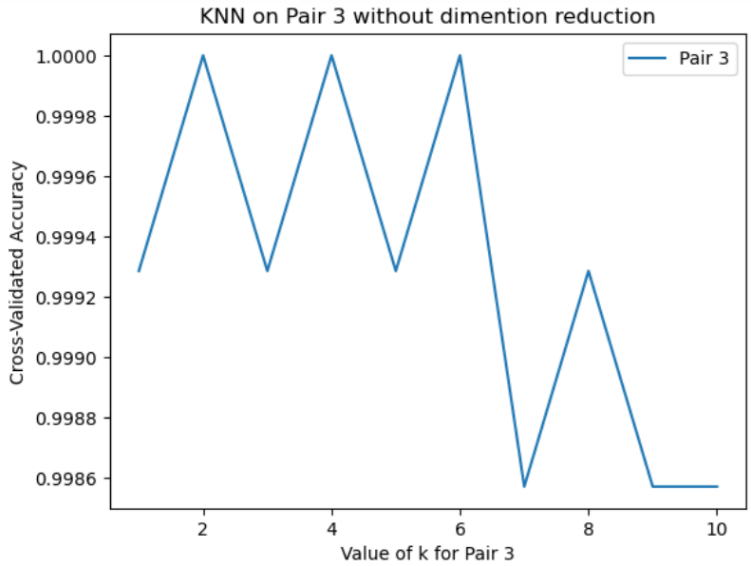
2. KNN on Pair 2 (tuning k)

```
Best k: 1
Best k accuracy:0.9992957746478872
Accuracy on testing 1.0
confusion matrix:
[[95  0]
 [ 0 63]]
testing time:  0.010001182556152344
```



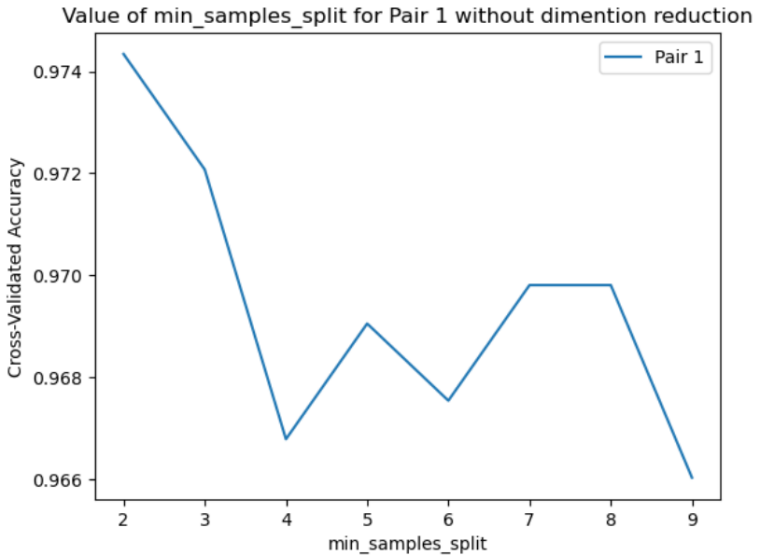
3. KNN on Pair 3 (tuning k)

```
Best k: 2
Best k accuracy:1.0
confusion matrix
[[87  0]
 [ 0 69]]
Accuracy on testing 1.0
testing time:  0.010002374649047852
```



4. Random Forest on Pair 1 (tuning min_samples_split)

```
Best split: 2
Best split accuracy:0.9743396226415095
Accuracy on testing 0.9932432432432432
confusion matrix:
[[74  1]
 [ 0 73]]
testing time:  0.14110517501831055
```

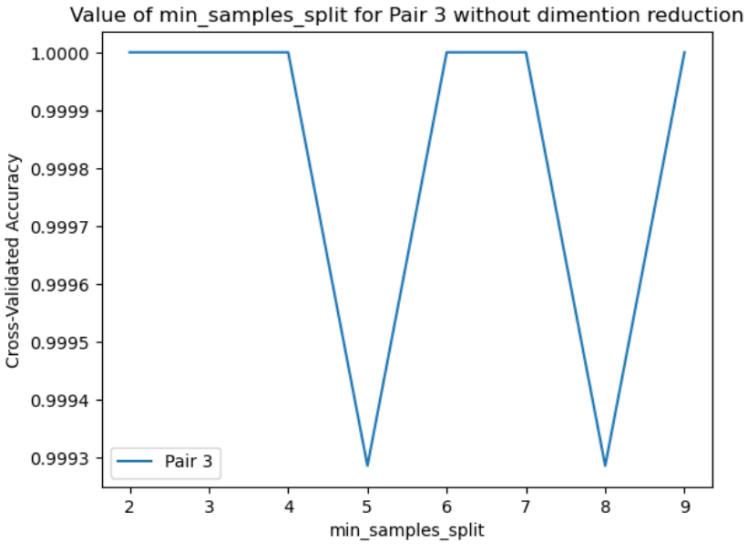


5. Random Forest on Pair 2 (tuning min_samples_split)

```
Best split: 2
Best split accuracy:0.9971830985915492
Accuracy on testing 0.9936708860759493
testing time:  0.12840890884399414
confusion matrix:
[[95  0]
 [ 1 62]]
```

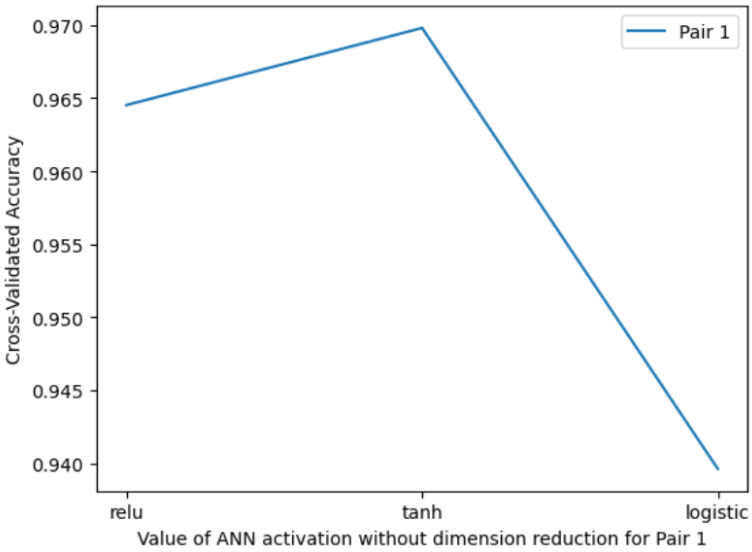
6. Random Forest on Pair 3 (tuning min_samples_split)

```
Best split: 2
Best split accuracy:1.0
Accuracy on testing 1.0
confusion matrix:
[[87  0]
 [ 0 69]]
testing time:  0.12302684783935547
```



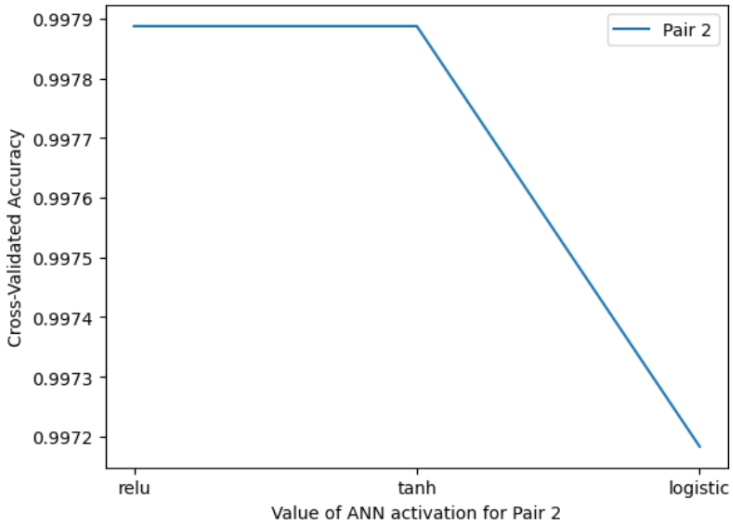
7. ANN on Pair 1 (tuning activation function)

```
Best acti: tanh
Best acti accuracy:0.969811320754717
Accuracy on testing: 0.9797297297297297
confusion matrix:
[[73  2]
 [ 1 72]]
testing time 1.7293446063995361
```



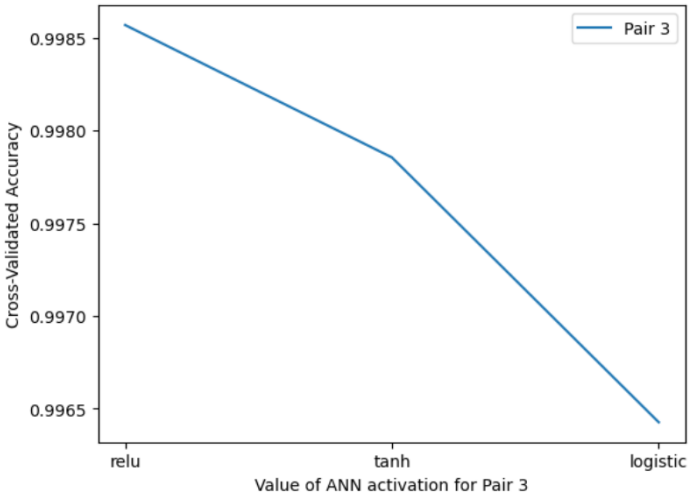
8. ANN on Pair 2 (tuning activation function)

```
Best acti: relu
Best acti accuracy:0.997887323943662
Accuracy on testing: 0.9936708860759493
confusion matrix:
[[95  0]
 [ 1 62]]
testing time 1.0560534000396729
```



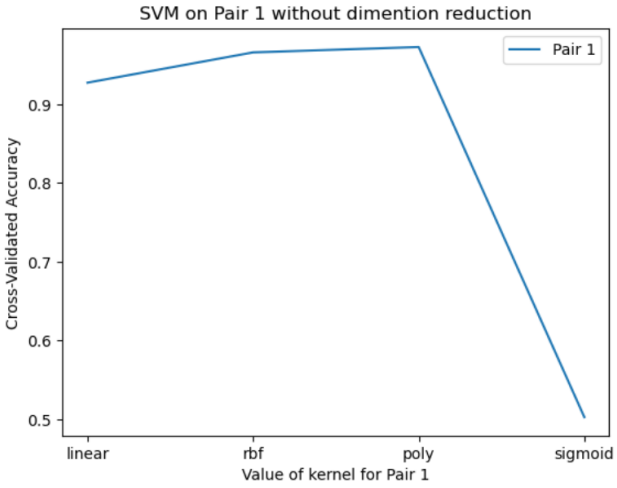
9. ANN on Pair 3 (tuning activation function)

```
Best acti: relu
Best acti accuracy:0.9985688684075781
Accuracy on testing: 0.9935897435897436
confusion matrix:
[[87  0]
 [ 1 68]]
testing time 1.5750536918640137
```



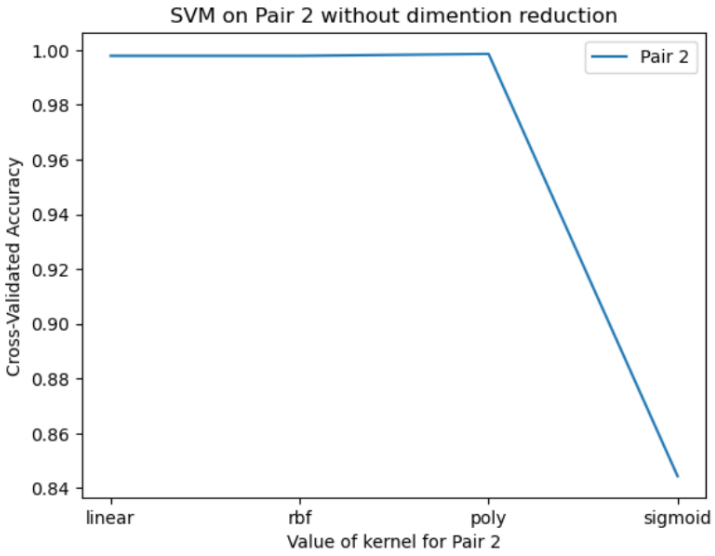
10. SVM on Pair 1 (tuning kernel)

```
Best kernel: poly
Best kernel accuracy:0.9728301886792453
Accuracy on testing: 0.9797297297297297
confusion matrix:
[[73  2]
 [ 1 72]]
testing time 0.028017520904541016
```



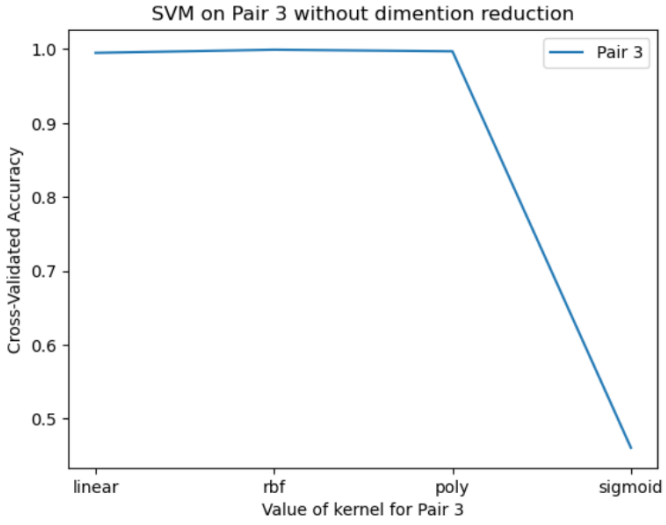
11. SVM on Pair 2 (tuning kernel)

```
Best kernel: poly
Best kernel accuracy:0.9985915492957746
Accuracy on testing: 1.0
confusion matrix:
[[95  0]
 [ 0 63]]
testing time 0.00899648666381836
```



12. SVM on Pair 3 (tuning kernel)

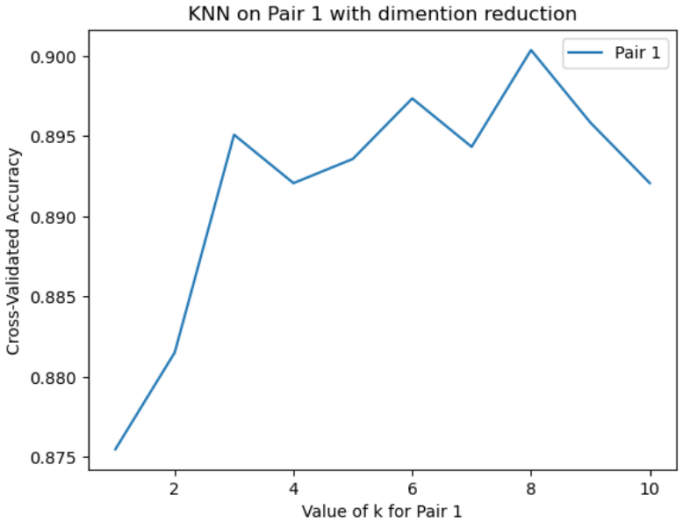
```
Best kernel: rbf
Best kernel accuracy:0.9992857142857143
confusion matrix:
[[87  0]
 [ 1 68]]
Accuracy on testing: 0.9935897435897436
testing time 0.013003110885620117
```



iii. Figure: Graph the cross validation results (from fitting the classification model ***with*** dimension reduction) over the range of hyperparameter values you tested. There should be three sets of values, one for each binary classification problem.

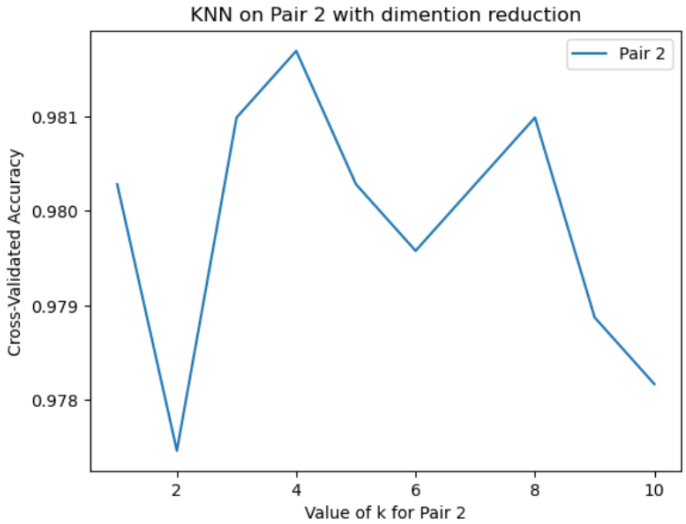
1. KNN on Pair 1 (tuning k)

```
Best k: 8
Best k accuracy: 0.9003773584905661
Accuracy on testing:  0.5337837837837838
confusion matrix:
[[45 30]
 [39 34]]
testing time:  0.005995273590087891
```



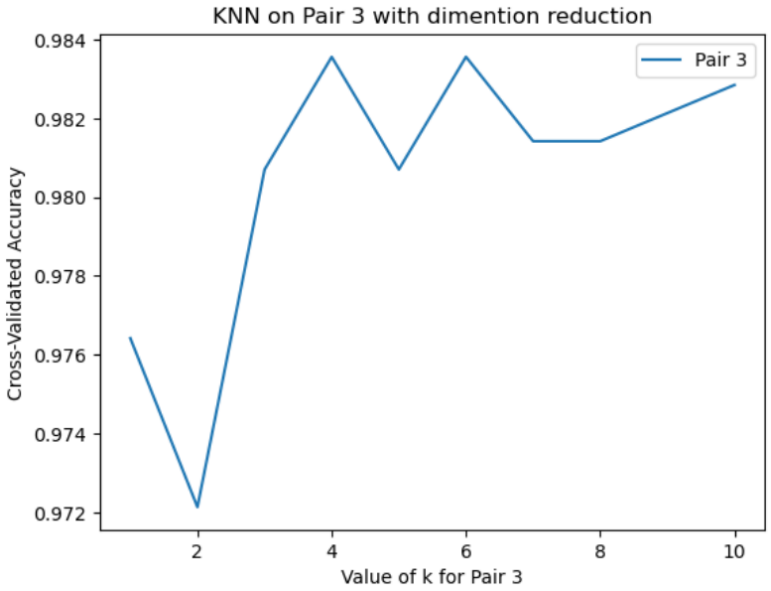
2. KNN on Pair 2 (tuning k)

```
Best k: 4
Best k accuracy: 0.9816901408450704
Accuracy on testing:  0.4430379746835443
confusion matrix:
[[44 51]
 [37 26]]
testing time:  0.007984399795532227
```



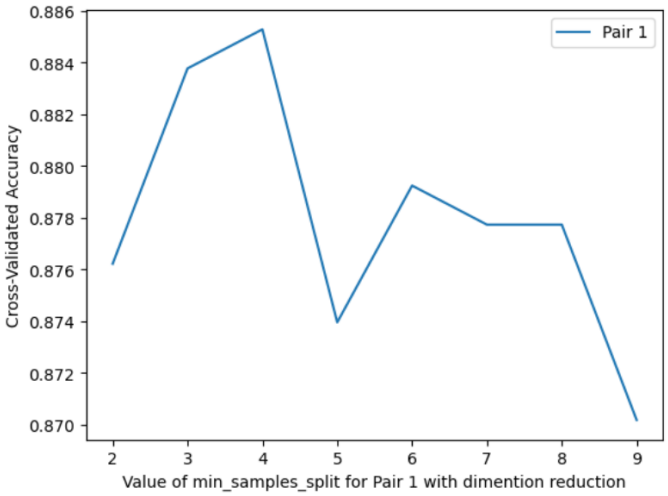
3. KNN on Pair 3 (tuning k)

```
Best k: 6
Best k accuracy: 0.9835663082437277
Accuracy on testing:  0.5897435897435898
confusion matrix:
[[54 33]
 [31 38]]
testing time:  0.006990909576416016
```



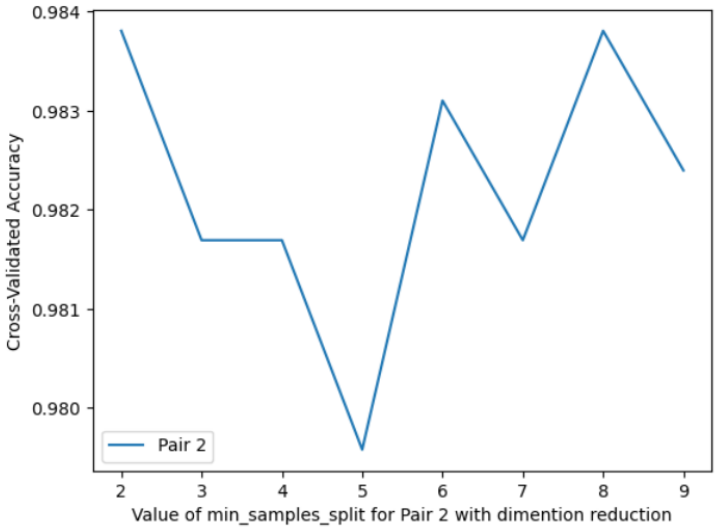
4. Random Forest on Pair 1 (tuning min_samples_split)

```
Best split: 4
Best split accuracy:0.8852830188679246
Accuracy on testing 0.49324324324324326
confusion matrix:
[[38 37]
 [38 35]]
testing time:  0.17003226280212402
```



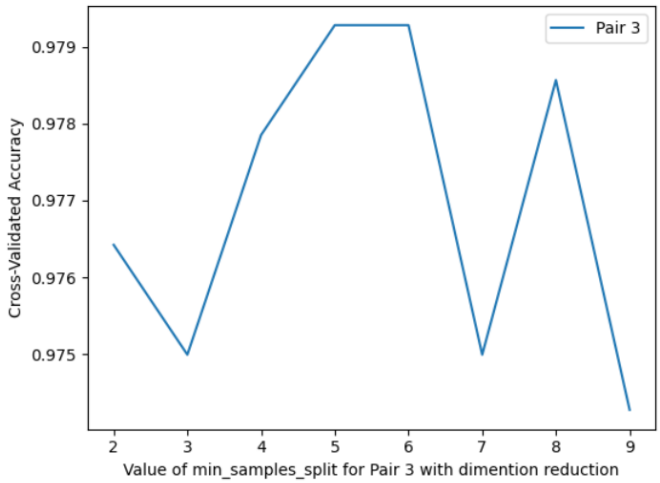
5. Random Forest on Pair 2 (tuning min_samples_split)

```
Best split: 2
Best split accuracy:0.9838028169014084
confusion matrix:
[[47 48]
 [35 28]]
Accuracy on testing 0.47468354430379744
testing time:  0.15364480018615723
```



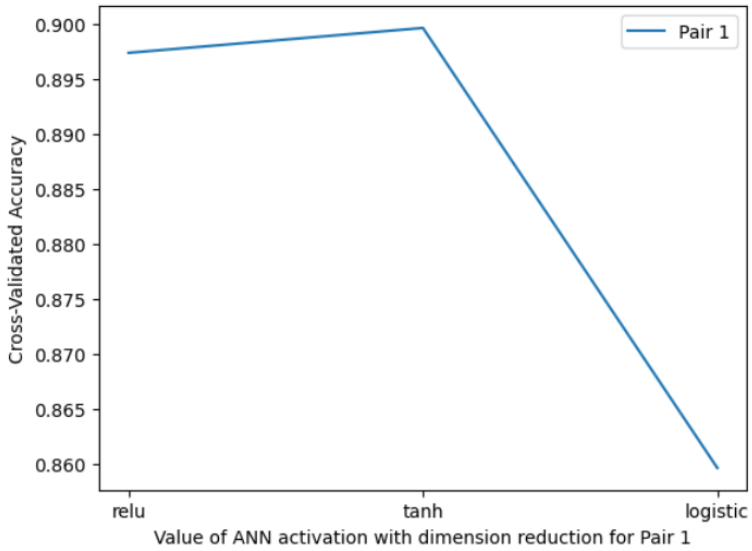
6. Random Forest on Pair 3 (tuning min_samples_split)

```
Best split: 5
Best split accuracy:0.9792805939580133
confusion matrix:
[[54 33]
 [30 39]]
Accuracy on testing 0.5961538461538461
testing time:  0.15039873123168945
```



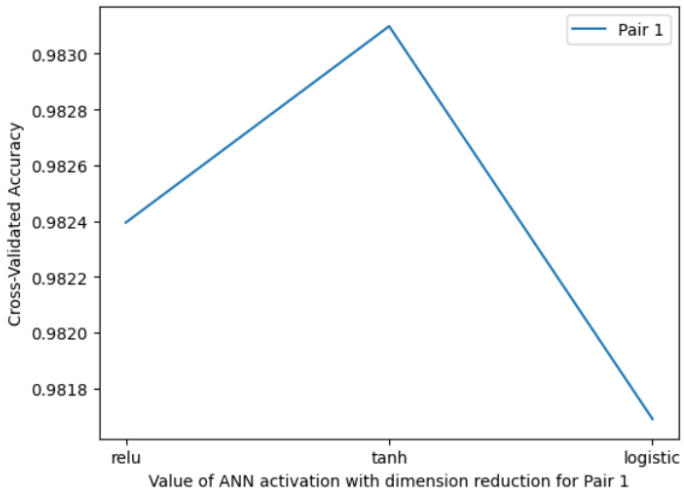
7. ANN on Pair 1 (tuning activation function)

```
Best acti: tanh
Best acti accuracy:0.8996226415094342
confusion matrix:
[[37 38]
 [38 35]]
Accuracy on testing: 0.4864864864864865
testing time 1.5612521171569824
```



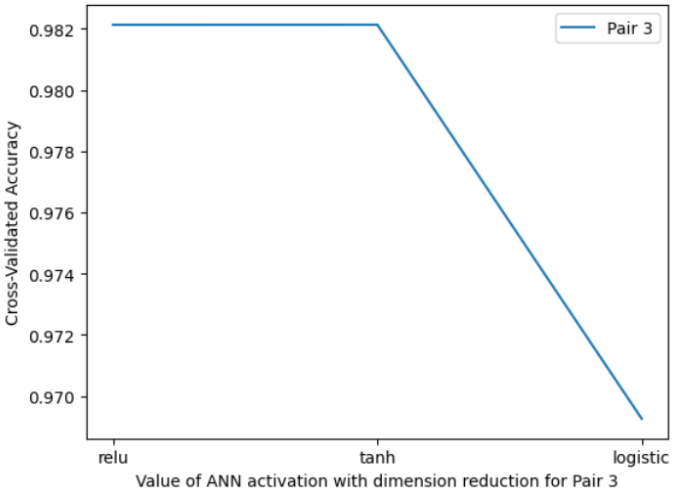
8. ANN on Pair 2 (tuning activation function)

```
Best acti: tanh
Best acti accuracy:0.9830985915492958
confusion matrix:
[[41 54]
 [37 26]]
Accuracy on testing: 0.4240506329113924
testing time 0.783151388168335
```



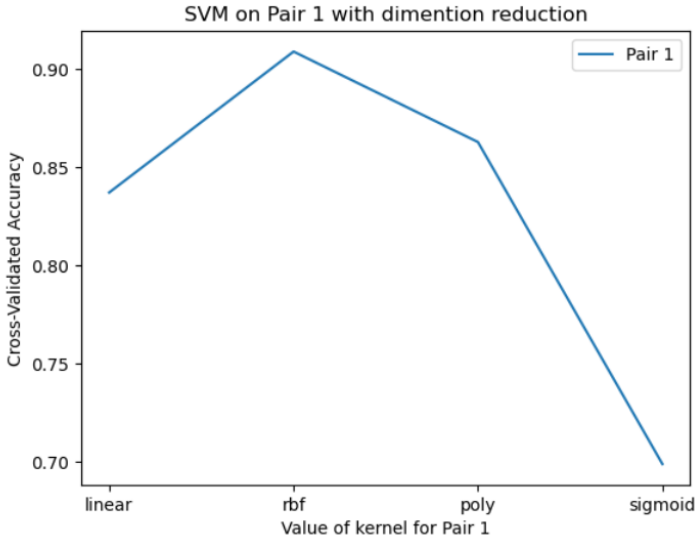
9. ANN on Pair 3 (tuning activation function)

```
Best acti: tanh
Best acti accuracy:0.9821326164874552
confusion matrix:
[[55 32]
 [40 29]]
Accuracy on testing: 0.5384615384615384
testing time 0.94681715965271
```



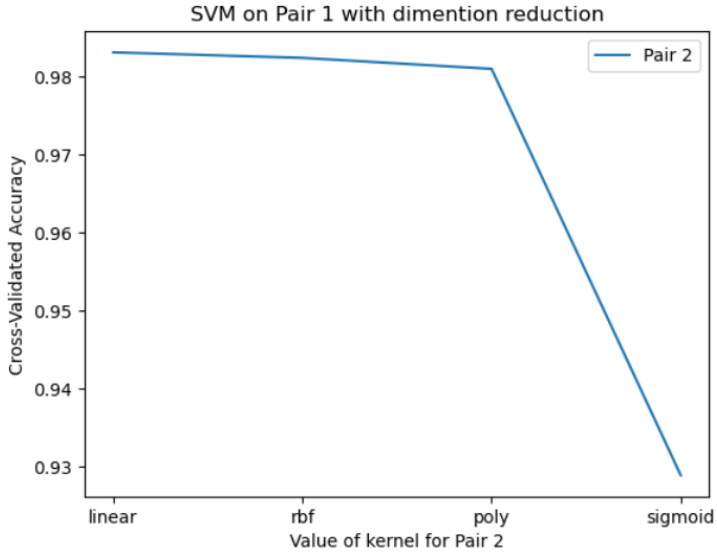
10. SVM on Pair 1 (tuning kernel)

```
Best kernel: rbf
Best kernel accuracy:0.9086792452830188
confusion matrix:
[[33 42]
 [36 37]]
Accuracy on testing: 0.47297297297297297
testing time 0.03701496124267578
```



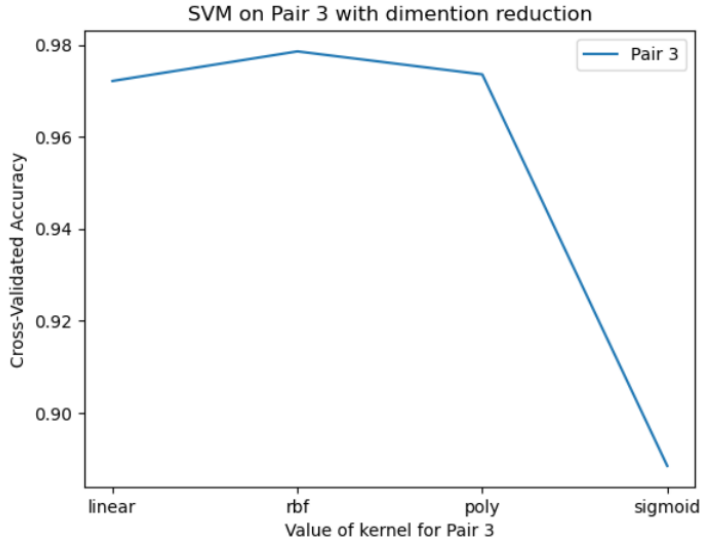
11. SVM on Pair 2 (tuning kernel)

```
Best kernel: linear
Best kernel accuracy:0.9830985915492958
confusion matrix:
[[42 53]
 [36 27]]
Accuracy on testing: 0.43670886075949367
testing time 0.011017322540283203
```



12. SVM on Pair 3 (tuning kernel)

```
Best kernel: rbf
Best kernel accuracy:0.9785560675883257
confusion matrix:
[[53 34]
 [33 36]]
Accuracy on testing: 0.5705128205128205
testing time 0.013021707534790039
```



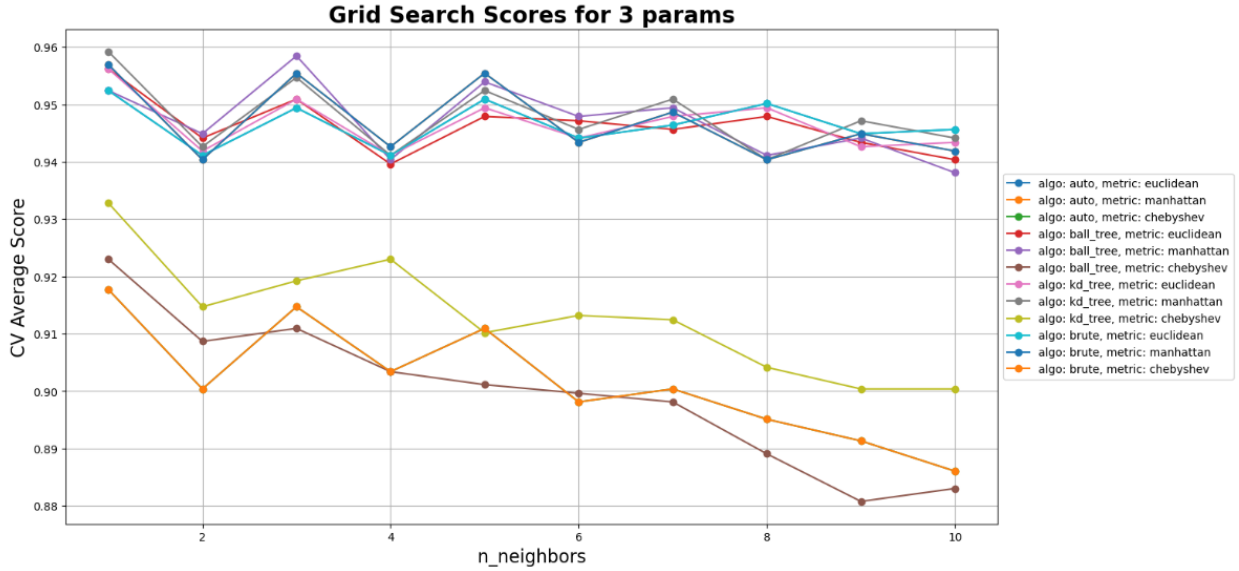
iv. for additional hyperparameters tuned

There are a total of 5 additional hyperparameters tuned. KNN additionally tuning distance metrics, and algorithms. ANN additionally tuning solver, and max iterations. SVM additionally tuning constant C.

The followings are without dimension reductions:

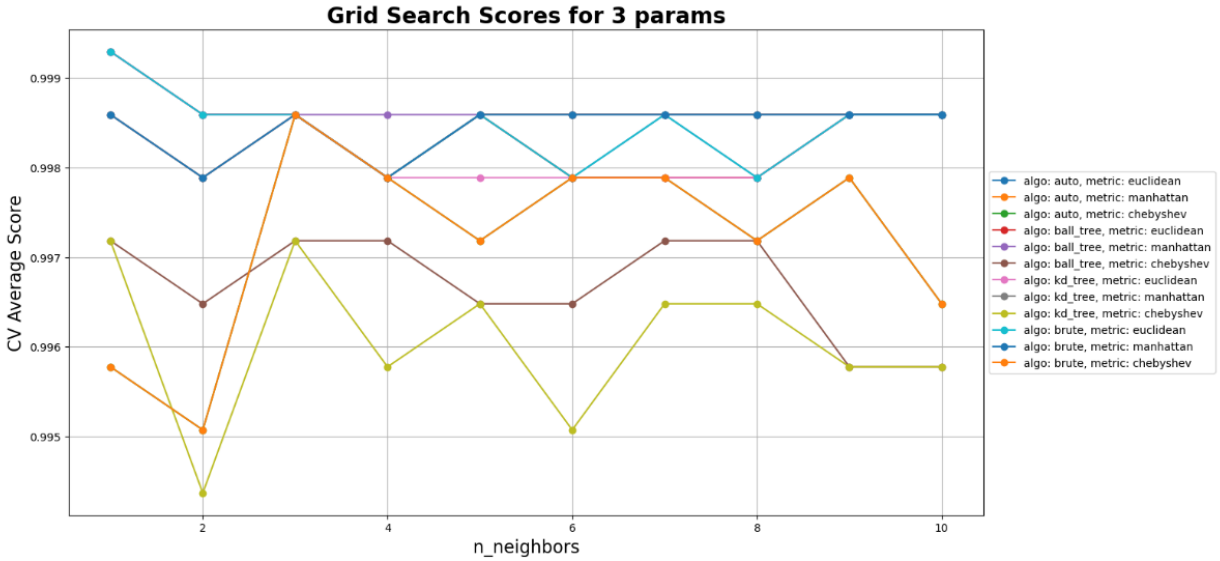
1. KNN on Pair 1

```
Best parameters:  {'algorithm': 'kd_tree', 'metric': 'manhattan', 'n_neighbors': 1}
Best cross-validated score:  0.959245283018868
confusion matrix:
[[72  3]
 [ 3 70]]
Accuracy on testing: 0.9594594594594594
testing time:  0.011651039123535156
```



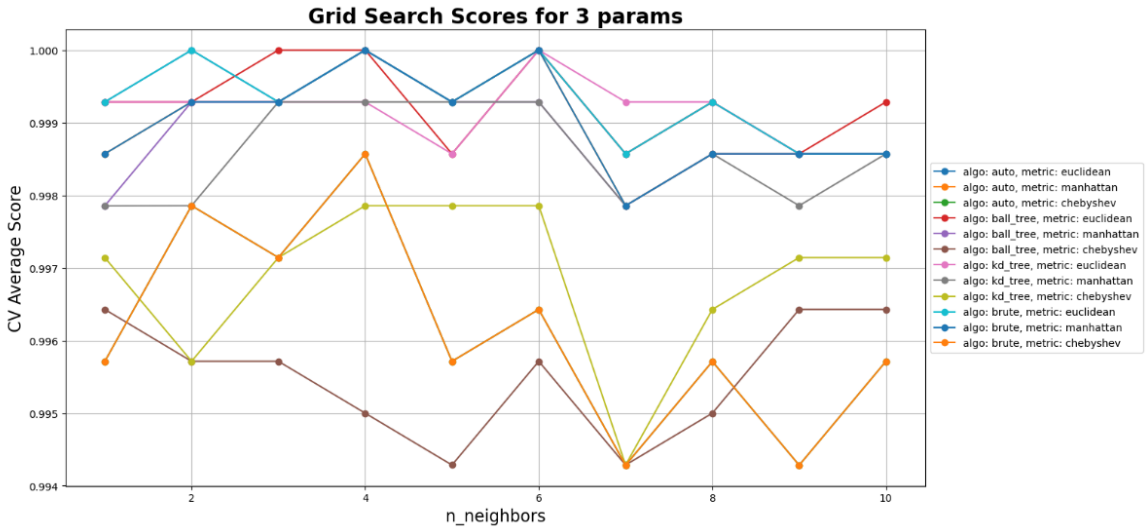
2. KNN on Pair 2

```
Best parameters:  {'algorithm': 'auto', 'metric': 'euclidean', 'n_neighbors': 1}
Best cross-validated score:  0.9992957746478872
confusion matrix
[[95  0]
 [ 0 63]]
Accuracy on testing: 1.0
testing time:  0.008564233779907227
```



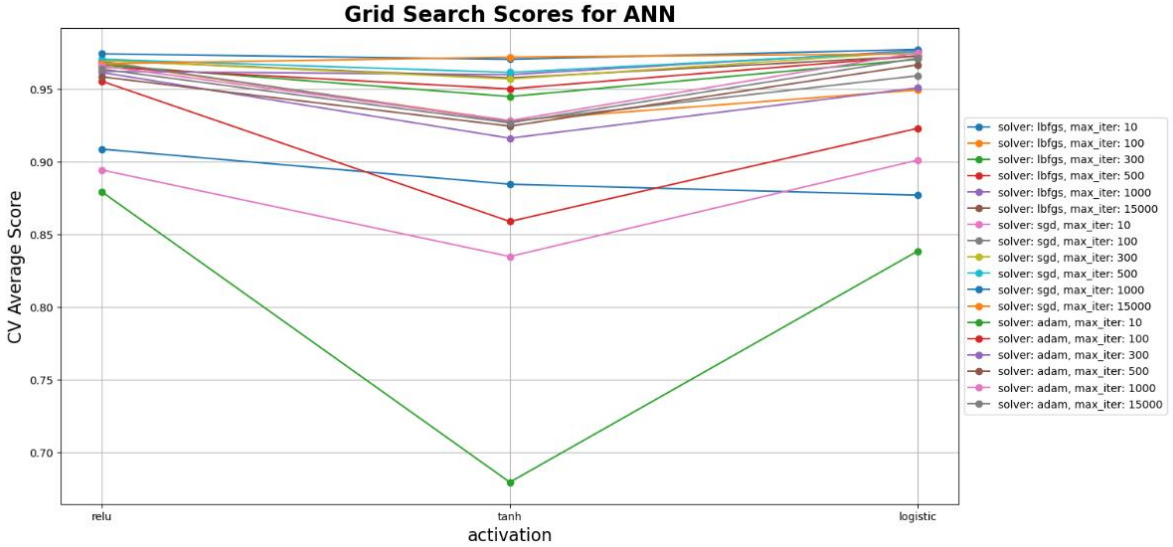
3. KNN on Pair 3

```
Best parameters:  {'algorithm': 'auto', 'metric': 'euclidean', 'n_neighbors': 2}
Best cross-validated score:  1.0
Accuracy on testing: 1.0
confusion matrix:
[[87  0]
 [ 0 69]]
testing time:  0.009020566940307617
```



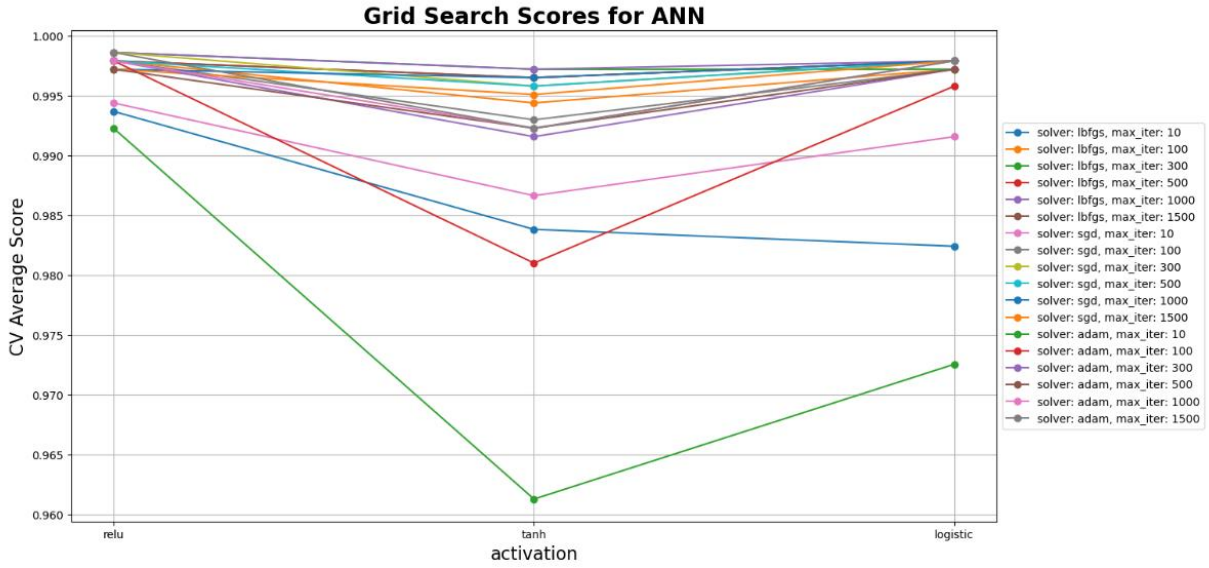
4. ANN on Pair 1

```
Best parameters:  {'activation': 'tanh', 'max_iter': 1000, 'solver': 'adam'}
Best cross-validated score:  0.9773584905660379
confusion matrix:
[[74  1]
 [ 0 73]]
Accuracy on testing: 0.9932432432432432
testing time 0.0030007362365722656
```



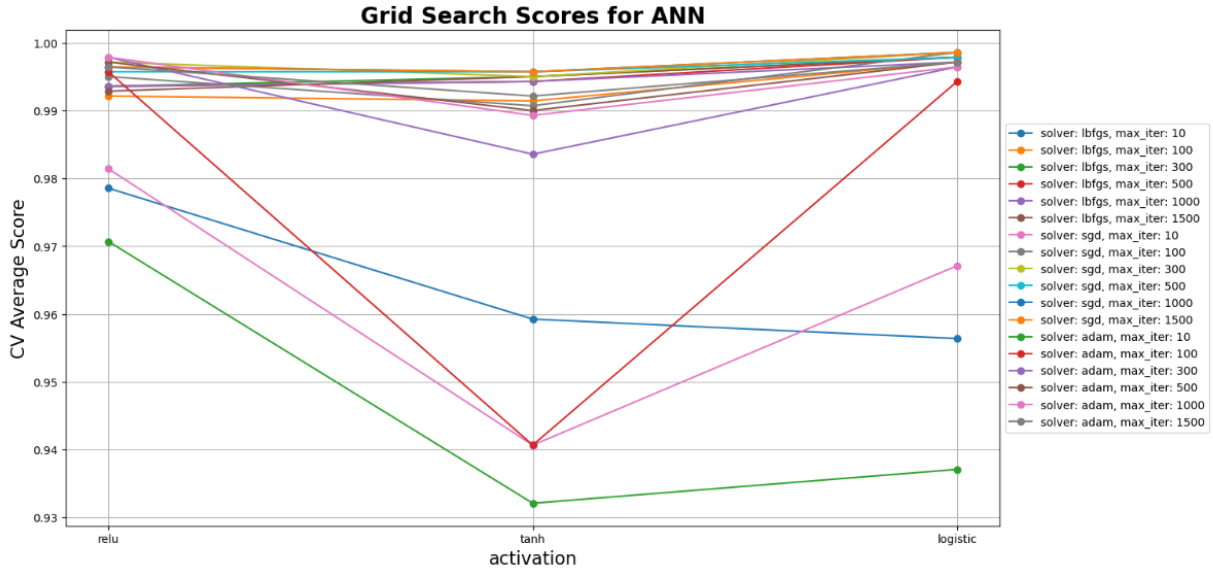
5. ANN on Pair 2

```
Best parameters:  {'activation': 'relu', 'max_iter': 300, 'solver': 'lbfgs'}
Best cross-validated score:  0.9985915492957746
Accuracy on testing: 1.0
confusion matrix:
[[95  0]
 [ 0 63]]
testing time 0.0019981861114501953
```



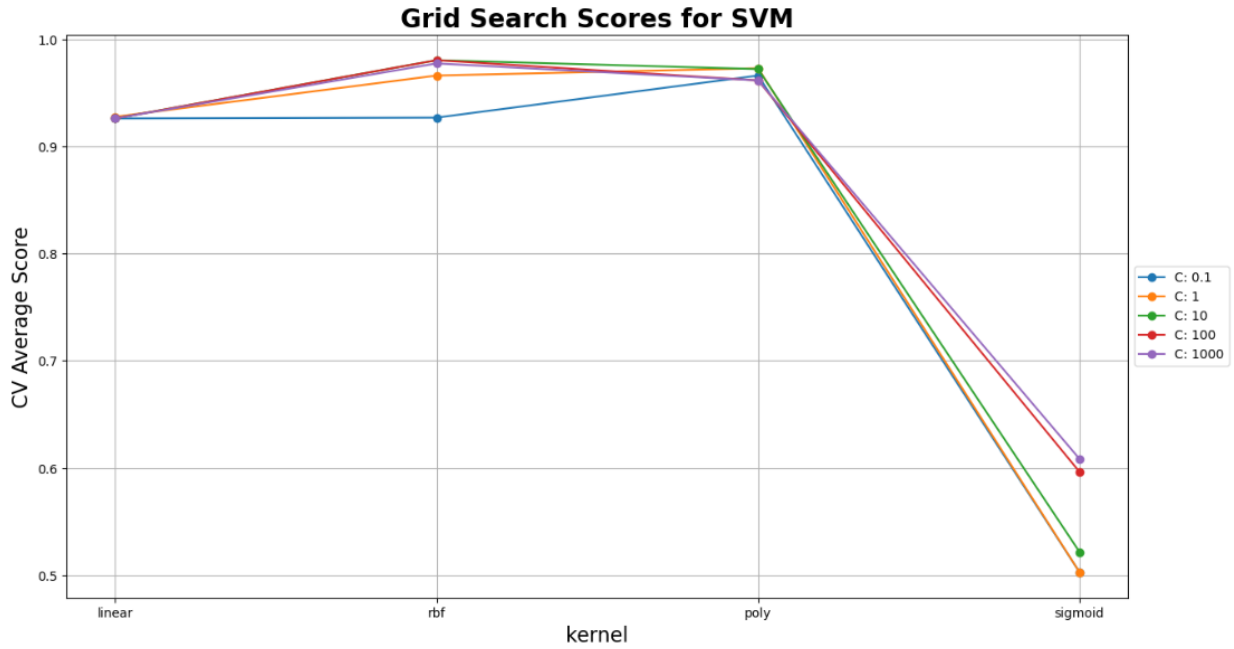
6. ANN on Pair 3

```
Best parameters:  {'activation': 'tanh', 'max_iter': 100, 'solver': 'adam'}
Best cross-validated score:  0.9985688684075781
Accuracy: 1.0
confusion matrix:
[[87  0]
 [ 0 69]]
testing time 0.0020012855529785156
```



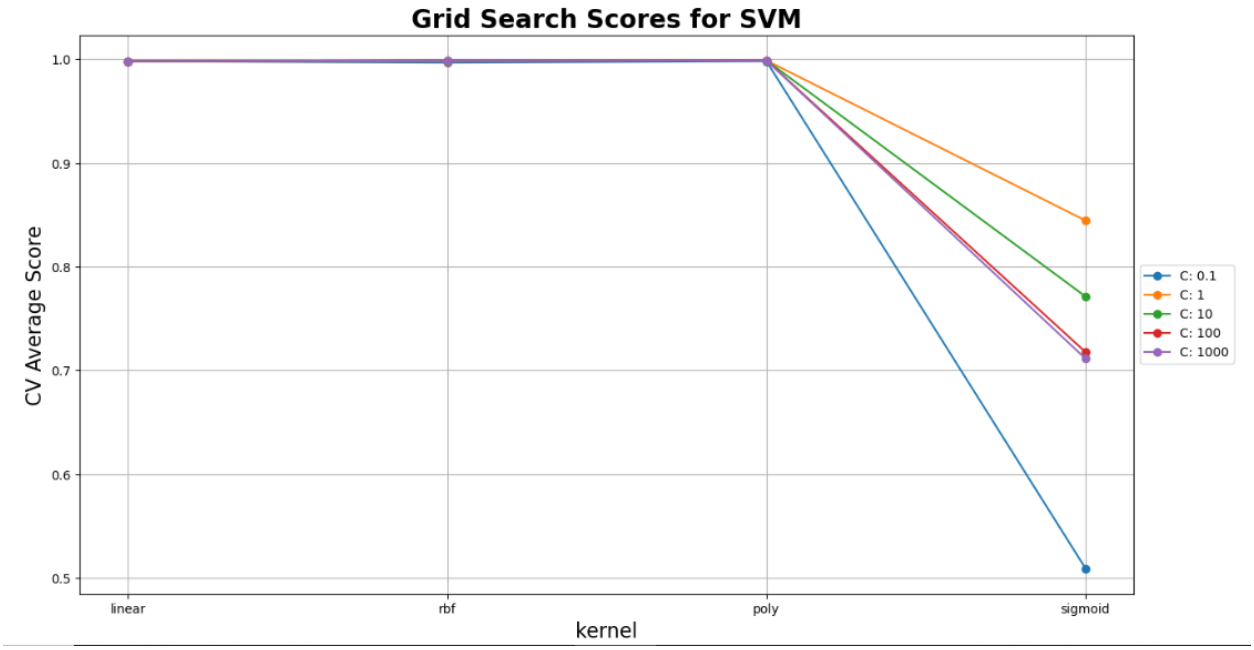
7. SVM on Pair 1

```
Best parameters:  {'C': 100, 'kernel': 'rbf'}
Best cross-validated score:  0.9803773584905662
Accuracy on testing: 0.9932432432432432
confusion matrix:
[[74  1]
 [ 0 73]]
testing time 0.0040013790130615234
```



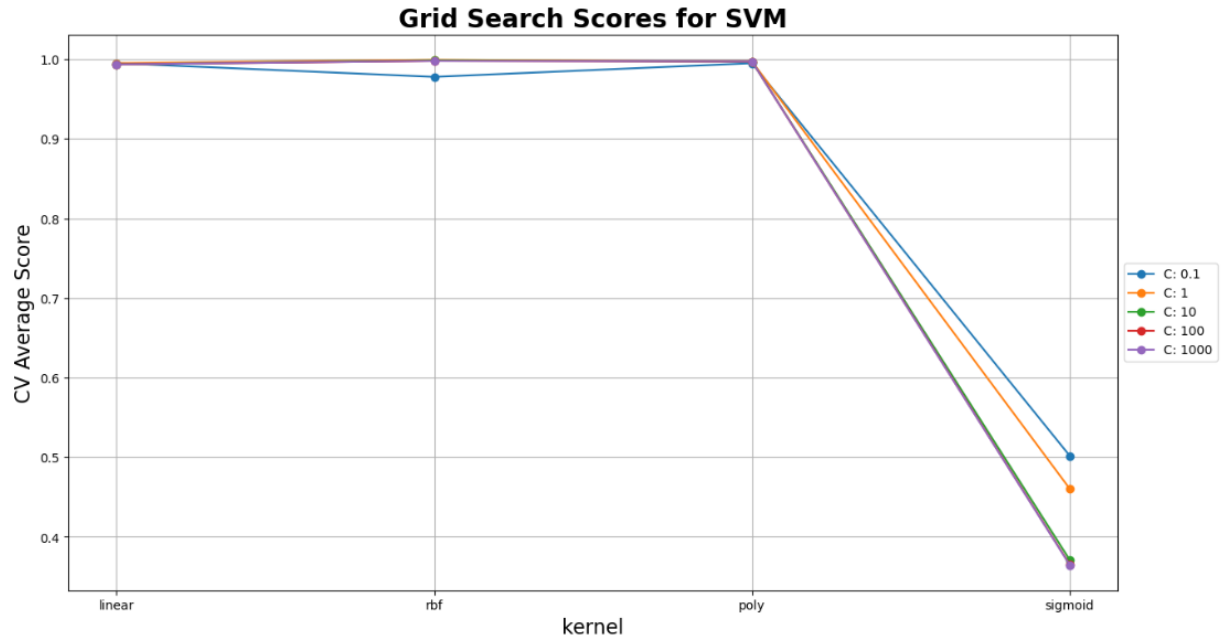
8. SVM on Pair 2

```
Best parameters:  {'C': 1, 'kernel': 'poly'}
Best cross-validated score:  0.9985915492957746
confusion matrix:
[[95  0]
 [ 0 63]]
Accuracy on testing: 1.0
testing time 0.002020120620727539
```



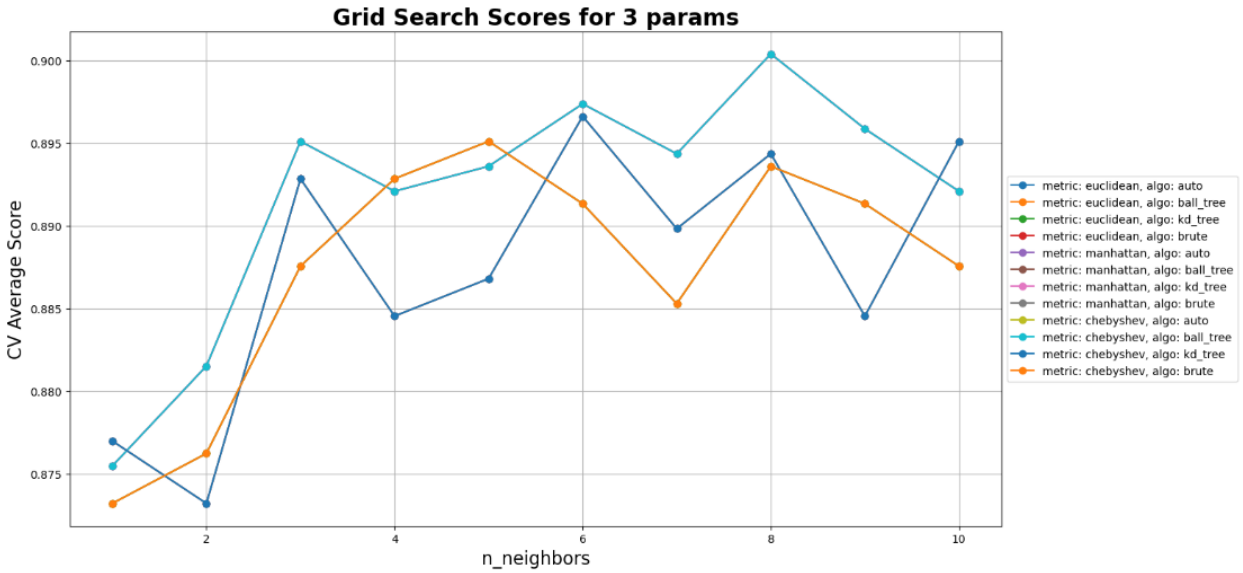
9. SVM on Pair 3

```
Best parameters:  {'C': 1, 'kernel': 'rbf'}
Best cross-validated score:  0.9992857142857143
confusion matrix:
[[87  0]
 [ 1 68]]
Accuracy on testing: 0.9935897435897436
testing time 0.003000497817993164
```

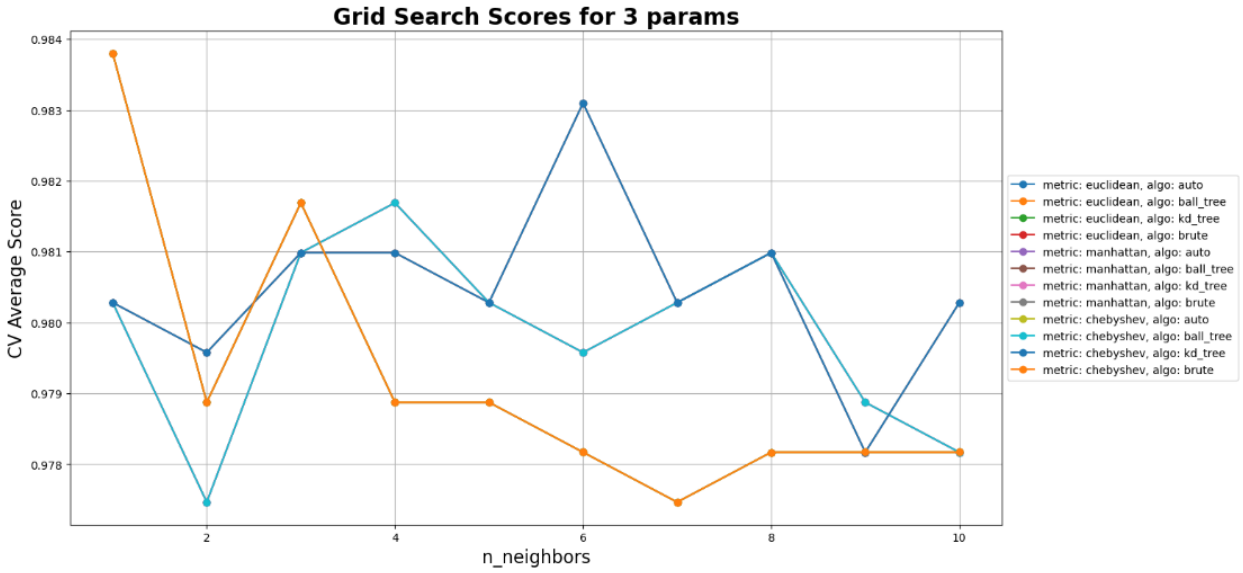


The followings are with dimension reduction (some lines overlapped with each other):

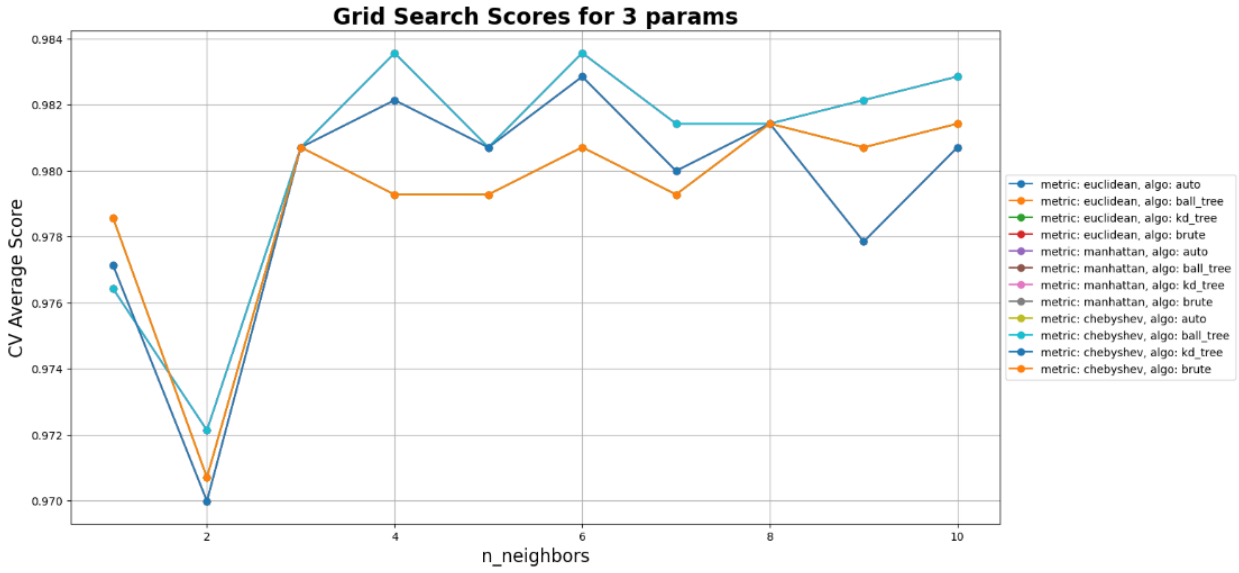
1. KNN on Pair 1



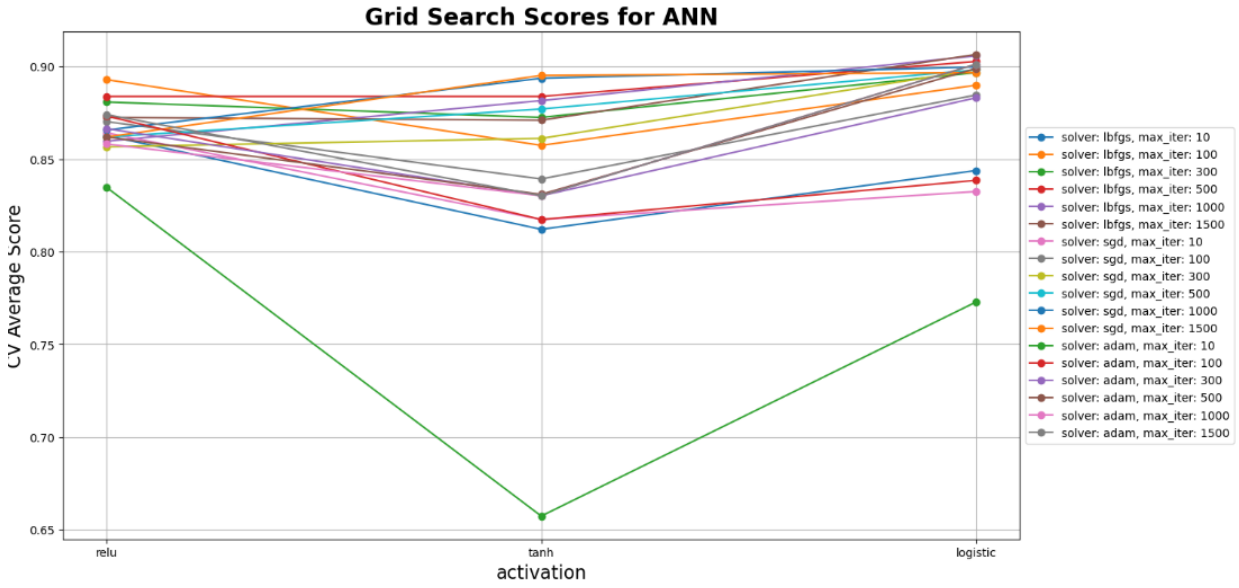
2. KNN on Pair 2



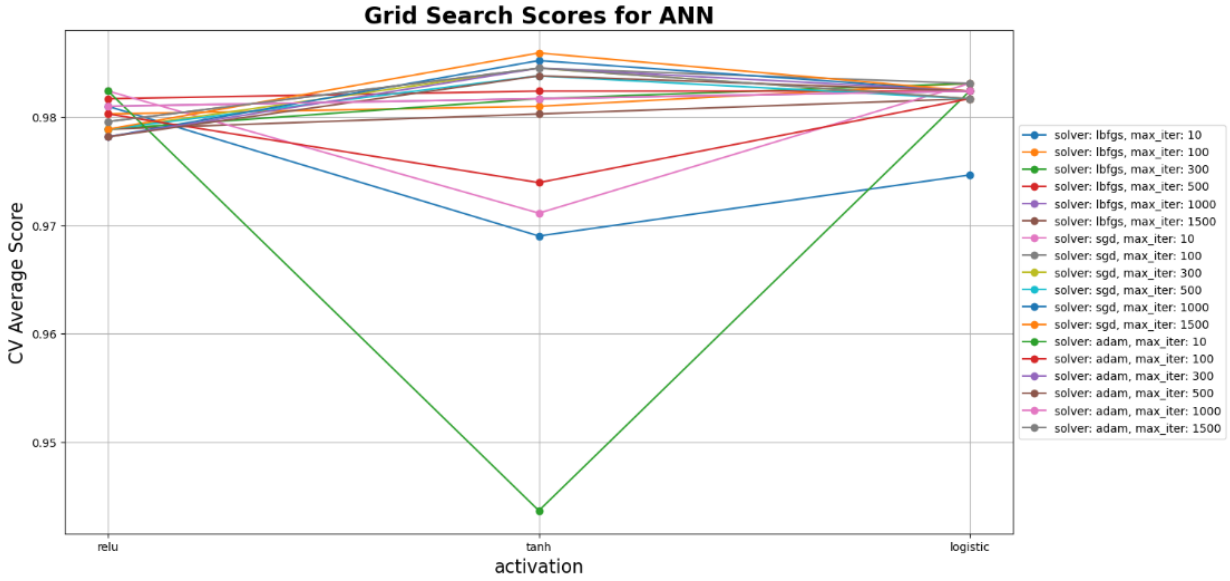
3. KNN on Pair 3



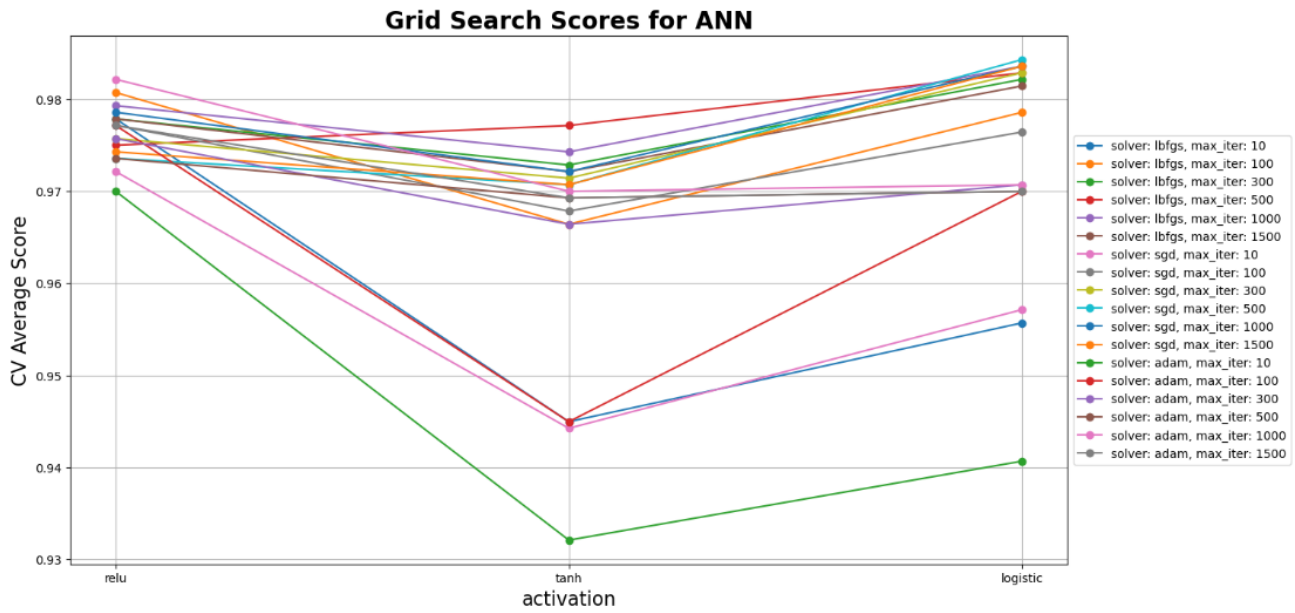
4. ANN on Pair 1



5. ANN on Pair 2

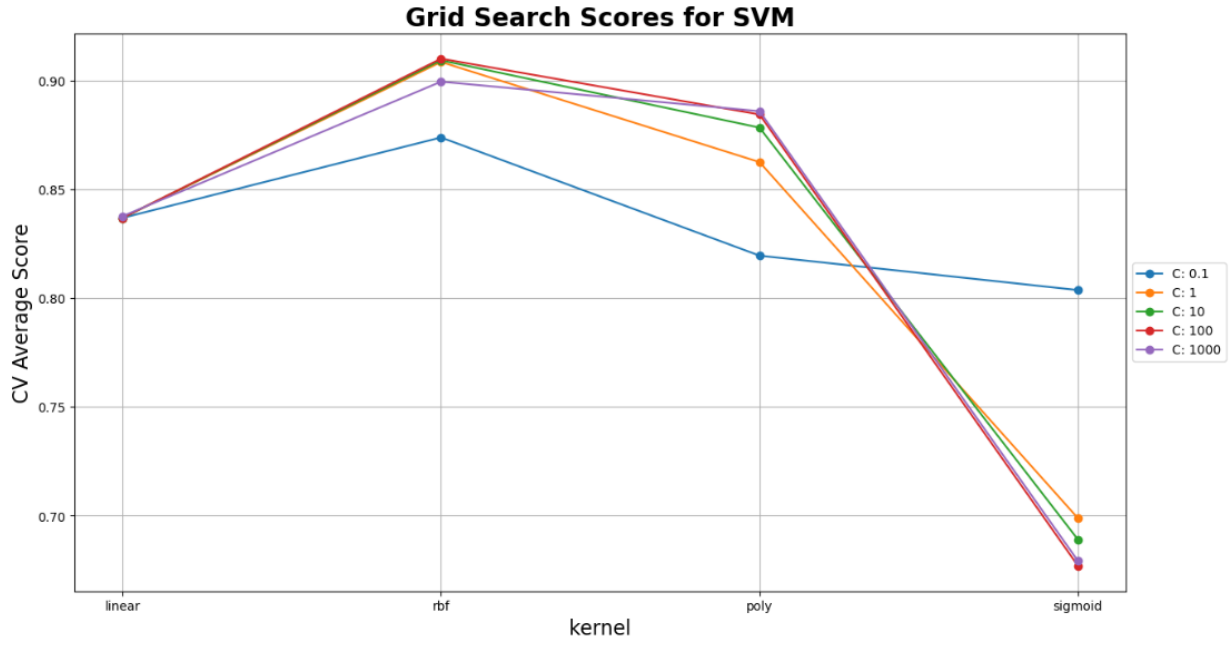


6. ANN on Pair 3



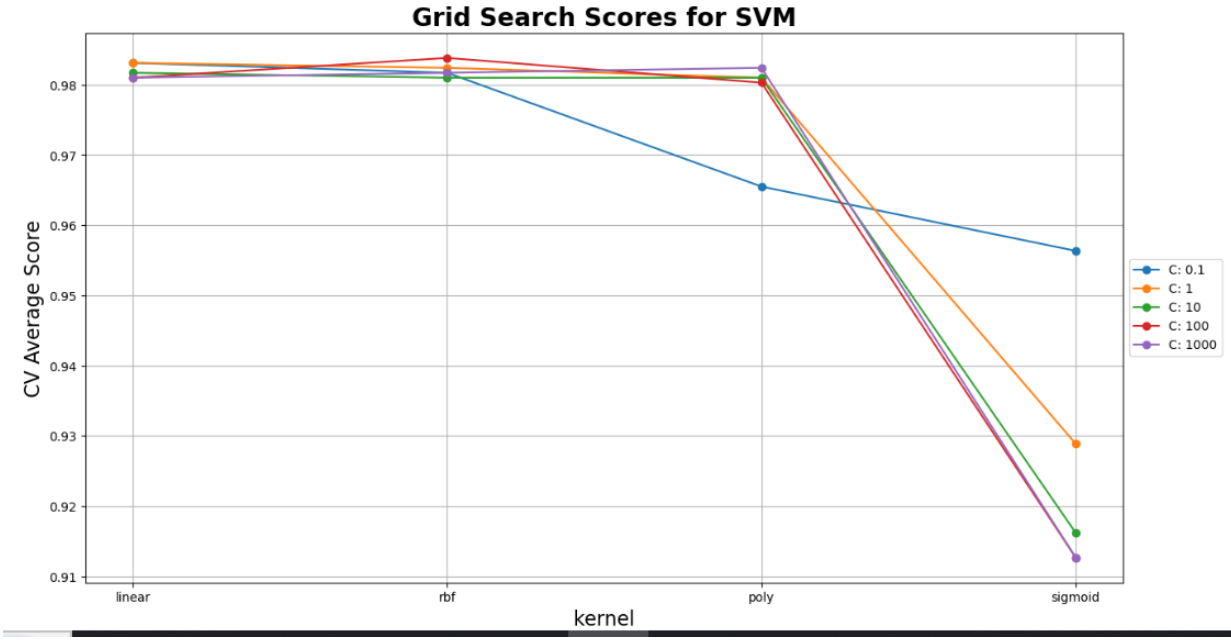
7. SVM on Pair 1

```
Best parameters:  {'C': 100, 'kernel': 'rbf'}
Best cross-validated score:  0.9101886792452831
confusion matrix:
[[29 46]
 [40 33]]
Accuracy on testing: 0.4189189189189189
testing time 0.006007671356201172
```



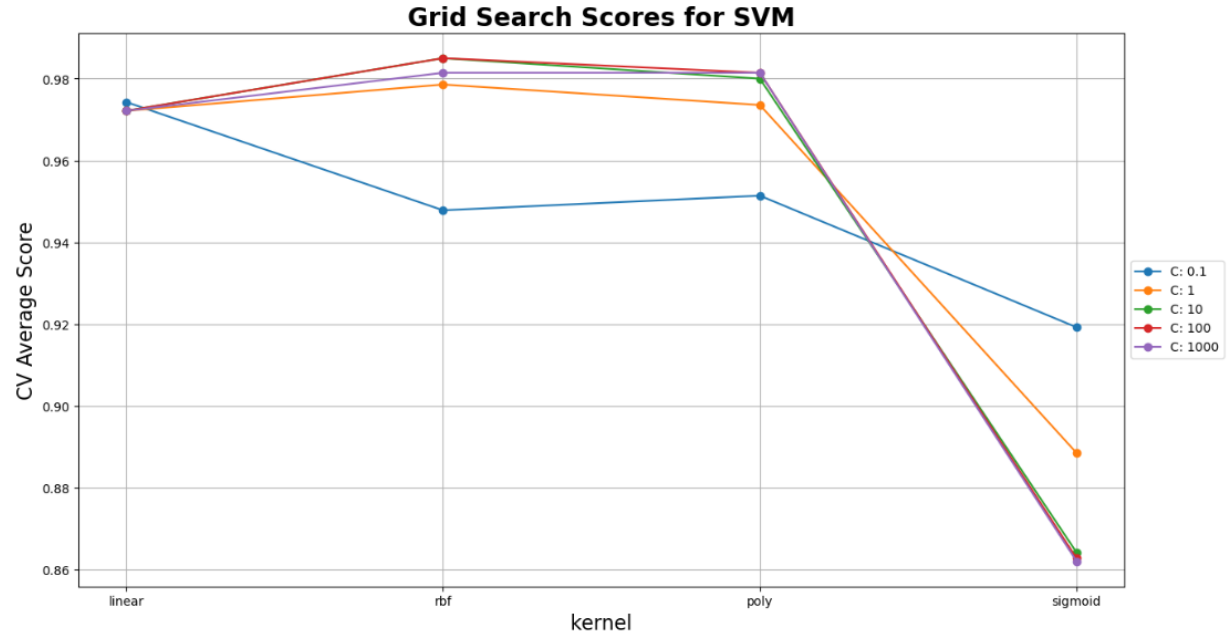
8. SVM on Pair 2

```
Best parameters:  {'C': 100, 'kernel': 'rbf'}
Best cross-validated score:  0.9838028169014086
confusion matrix:
[[45 50]
 [35 28]]
Accuracy on testing: 0.4620253164556962
testing time 0.0020003318786621094
```



9. SVM on Pair 3

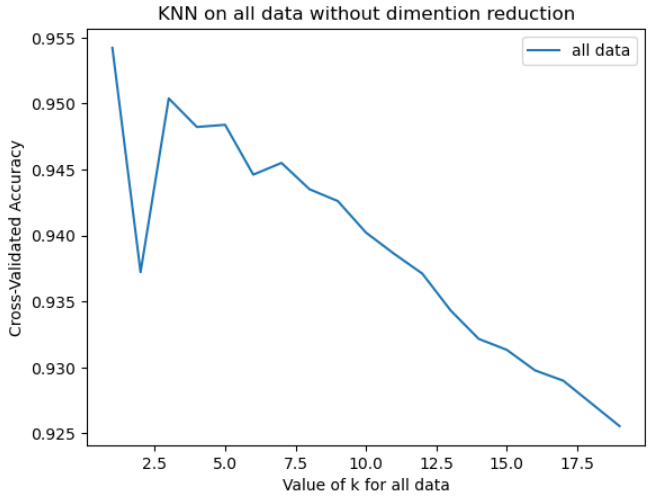
```
Best parameters:  {'C': 100, 'kernel': 'rbf'}
Best cross-validated score:  0.9849948796722992
confusion matrix:
[[42 45]
 [33 36]]
Accuracy on testing: 0.5
testing time 0.002000570297241211
```



v. multi-class classification (using KNN)

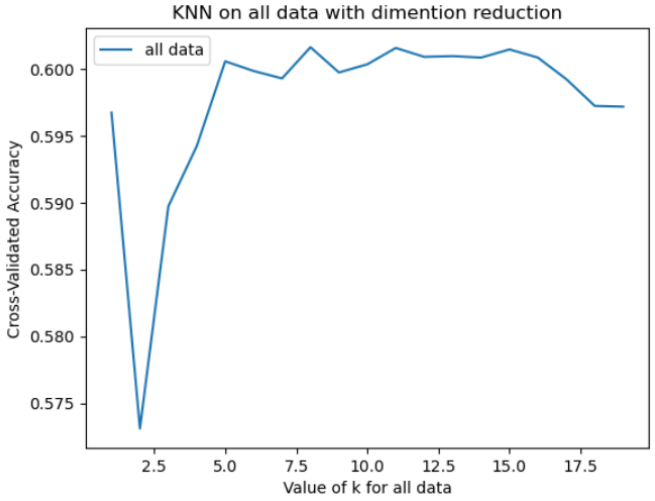
1. without dimension reduction

```
Best k: 1
Best k accuracy:0.9542222222222222
Accuracy on testing 0.9575
testing time:  0.44828176498413086
```



2. with dimension reduction

```
Best k: 8
Best k accuracy:0.601611111111111
Accuracy on testing 0.5595
testing time:  0.07370352745056152
```



3. Discussion

a. Compare the performance and run time of the different classifiers on the final validation sets with either a table or a figure. And including the additional classifiers and multi-class classifier in your discussion.

runtime	KNN	RF	ANN	SVM	Multi class

Pair 1	0.009002	0.14110517	1.72934460	0.02801752	0.0737035
Pair 2	0.0100012	0.12840890	1.05605340	0.00899646	
Pair 3	0.0100023	0.12302684	1.57505369	0.01300311	

performance	KNN	RF	ANN	SVM	Multi class
Pair 1	Best k accuracy:0.95245 Accuracy on testing 0.966216	Best split accuracy:0.974339 Accuracy on testing 0.9932	Best acti accuracy:0.96981132 Accuracy on testing: 0.9797	Best score: 0.980377358490566 Accuracy on testing: 0.99324	Best k accuracy:0.954222 Accuracy on testing 0.9575
Pair 2	Best k accuracy:0.99929577464 Accuracy on testing 1.0	Best split accuracy:0.9971985 Accuracy on testing 0.993670	Best acti accuracy:0.997887323 Accuracy on testing: 0.99367	Best kernel accuracy:0.9985915492 Accuracy on testing: 1.0	
Pair 3	Best k accuracy: 1.0 Accuracy on testing 1.0	Best split accuracy:0.9971830 Accuracy on testing 0.993670	Best acti accuracy:0.99788732 Accuracy on testing: 0.99367	Best score: 0.980377735849056 Accuracy on testing: 0.99324	

b. Compare the performance and run time of the different classifiers after dimension reduction on the final validation sets with either a table or a figure. And including the additional classifiers and multi-class classifier in your discussion.

Runtime	KNN	RF	ANN	SVM	Multi class
Pair 1	0.0059953	0.1700322	1.56125211	0.03701496	0.44828176
Pair 2	0.0079843	0.1536448	0.78315138	0.01101732	
Pair 3	0.0069909	0.1503987	0.94681715	0.01302170	

Performance	KNN	RF	ANN	SVM	Multi class
Pair 1	Best k accuracy:0.90245 Accuracy on testing 0.5337837	Best split accuracy:0.8852830188679246 Accuracy on testing 0.4932	Best acti accuracy:0.9698113 Accuracy on testing: 0.48972	Best score: 0.9803773584905 Accuracy on testing: 0.47324	Best k accuracy:0.601611111111111 Accuracy on testing 0.5595
Pair 2	Best k accuracy: 0.9816901408450704	Best split accuracy:0.9871985	Best acti accuracy:0.9878873	Best kernel accuracy:0.988591549	

	Accuracy on testing: 0.4430379746835	Accuracy on testi ng 0.473670	Accuracy on te sting: 0.42367	Accuracy on testi ng: 0.432848	
Pair 3	Best k accuracy: 0.9 835663082437277 Accuracy on testing: 0.5897435	Best split accurac y:0.9771830 Accuracy on testi ng 0.593670	Best acti accur acy:0.9888732 Accuracy on te sting: 0.53367	Best score: 0.970 3773 Accuracy on testi ng: 0.5729324	

c. Lessons learned: What model would you choose for this problem and why? How did dimension reduction effect the accuracy and/or run times of the different classifiers? What would you do differently if you were given this same task for a new dataset? Include at least one additional topic of discussion.

For the first topic, the focus is on dimension reduction. In general, models that use dimension reduction tend to achieve testing accuracy of around 50-60%, which is considered relatively poor. This could be due to overfitting during the training phase, as many models perform very well with training accuracies over 95%, but generalize poorly during testing. In my experience, using dimension reduction can reduce the overall computation time, but the reduction may not be significant. This could be because the dataset used was not sufficiently large or complex, and fewer hyperparameters were required to achieve optimal results.

The second topic would be on the run time, ANN took the longest to train and are really prone to overfitting. If I could choose based the results on the assignments, I would choose KNN, then probably SVM. Given that KNN is fairly easy to interpret and takes less time to train.