CSE514 AS2
Ashley Wang , Clinton Jules , Tony Ngok

1 Introduction

1.1 problem description and practical impacts

This study aims to handle several binary classification challenges using a dataset focused on letter recognition, encompassing 20000 instances with diverse features corresponding to different letters. The successful resolution of these classification issues holds practical implications for applications like optical character recognition, automated handwriting interpretation, and font style classification. Effectively addressing this problem has the potential to greatly improve the efficiency and accuracy of automated text processing in fields such as digital document management and assistive technologies.

1.2 Motivation for training and testing multiple classifiers

We need to try different models on the same dataset and problem because various models come with their unique strengths and weaknesses, making them suitable for different types of data and classification tasks. By experimenting with multiple models, we can comprehensively assess their performance on a specific problem and identify the most effective one. This multi-model comparison helps us understand factors such as computational complexity, validation accuracy, and model interpretability, allowing us to better tailor our choice of model to meet practical requirements.

Using multiple classifiers helps in understanding how different models generalize to unseen data. A model that performs well on the training data might not necessarily do well on new, unseen data. Testing multiple classifiers can help identify models that are more robust and generalize better. Relying on a single classifier increases the risk of overfitting, especially if the model is complex. Testing multiple simpler models can provide a benchmark to understand if a more complex model is genuinely learning patterns or just memorizing the data. The ease with which a model can be tuned (hyperparameter optimization) and the sensitivity of the model to these parameters can be crucial, especially when fine-tuning performance.

1.3 Motivation for Dimension Reduction

The primary reason for dimensionality reduction lies in addressing challenges associated with high-dimensional data. High-dimensional datasets not only escalate computational complexity but also risk encountering the ″curse of

dimensionality", where data sparsity increases, leading to lower sample density in high-dimensional space. This can result in overfitting, diminishing the model's generalization capabilities. Dimensionality reduction allows us to decrease the number of features, improving computational efficiency while retaining essential information in the data. It aids in enhancing model performance, mitigating the risk of overfitting, facilitating better understanding and visualization of data. Additionally, dimensionality reduction helps eliminate redundancy among features, enhancing model robustness, and, in some cases, accelerating the training process.

1.4 Chosen Dimension Reduction Method

We're choosing to use PCA for dimensionality reduction in this situation. PCA, or Principal Component Analysis, is like a smart way to simplify complex data. It finds the most important stuff in the data, kind of like discovering the main patterns. Then, it represents the data in a simpler way, making it easier for us to handle and understand

1.5 Speculation on Binary Classification Problems

- Pair 1: H and K
- Pair 2: M and Y
- Pair 3: X and O

We choose X and O as the third set of classification tasks. The reason for this choice is that these two letters have significant differences in the spatial structure of the pixel points and should be less difficult to recognize. We can test the basic performance of the classifier with this task set.

The pairs of letters included in the first two sets of categorization tasks have some structural similarity. In terms of pixel space characteristics, the similarities outweigh the differences, which leads to greater difficulty in discrimination. We hypothesize that the classifier should perform worse on the first two classification tasks than on the last one.

2 Results

2.1 kNN Classifier

2.1.1 Description of the kNN Classifier

The basic principle of kNN is that it works by finding the k nearest data points to a given sample and predicting the label based on the majority vote of

these neighbours. It's a type of instance-based or lazy learning, where generalization of the model to new data is delayed until a query is made. It is used for both classification and regression tasks but is more commonly associated with classification.

General Advantages

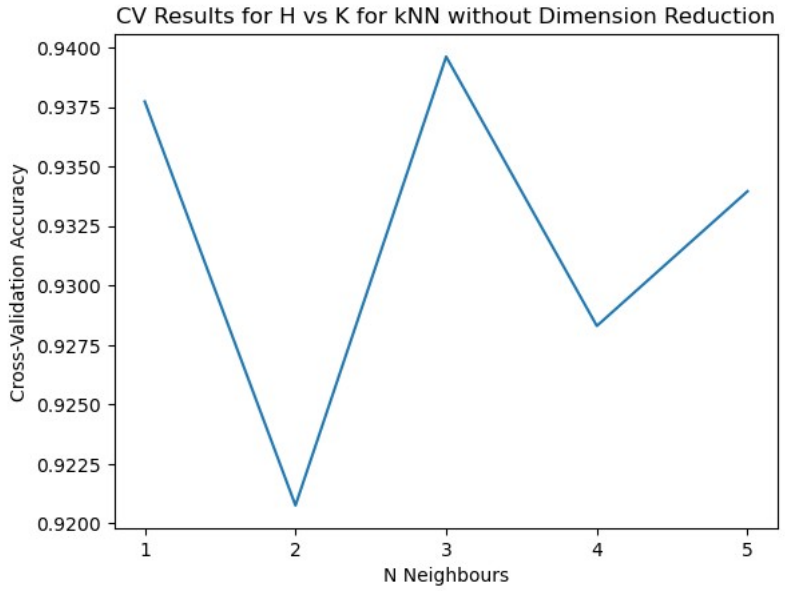
kNN is straightforward to understand and implement, making it a good starting point for classification problems. Unlike many algorithms, kNN doesn't make any assumptions about the underlying data distribution, which is beneficial in real-world scenarios where such assumptions often don't hold. It is flexible in that it can be used for both binary and multi-class classification tasks. Furthermore, kNN can easily handle scenarios with more than two classes. When there's a significant amount of representative data, kNN can be very effective, as it relies on the proximity to the nearest observed data points.

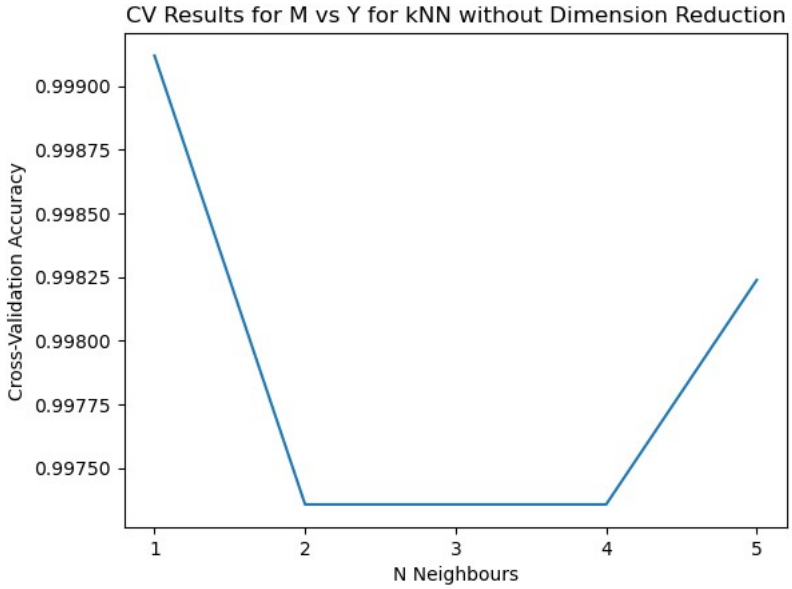
General Disadvantages

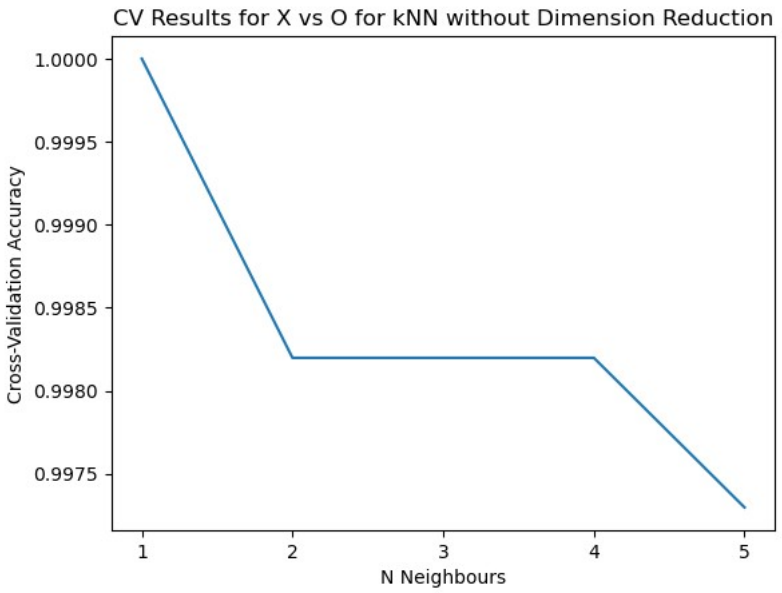
As the dataset grows, kNN can become impractically slow because it searches for the nearest neighbours in the entire dataset for each prediction. It also requires storing the entire dataset, which can be a limitation for large datasets. kNN is sensitive to irrelevant or redundant features because they can lead to inaccurate distances between samples. In an imbalanced dataset, kNN can be biased towards the majority class. The choice of the distance metric (Euclidean, Manhattan, etc.) can significantly impact the performance, and there is no universal rule for which one to choose. The choice of k (number of neighbours) greatly affects the results. A small value of k can lead to noise having a higher influence on the result, while a large value makes the computation slower. kNN suffers in high-dimensional spaces (many features) due to the "curse of dimensionality", as distinguishing between near and far points becomes less meaningful.

2.1.2 kNN Classifier Results without Dim Reduction

Method: kNN Classifier
Hyperparameter: N (nearest) neighbours
Hyperparameter values: 1, 2, 3, 4, 5

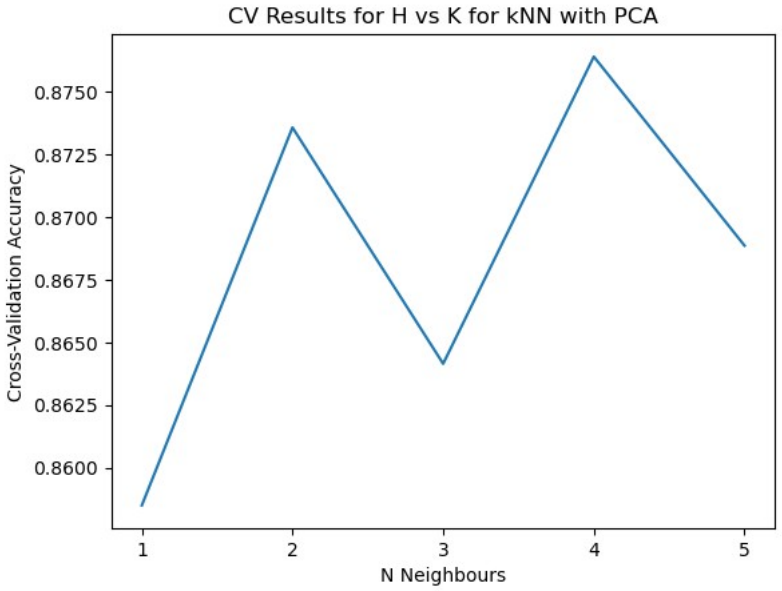

CV Results for H vs K for kNN without Dimension Reduction
Cross-Validation Accuracy
0.9400
0.9375
0.9350
0.9325
0.9300
0.9275
0.9250
0.9225
0.9200
1
2
3
4
5
N Neighbours

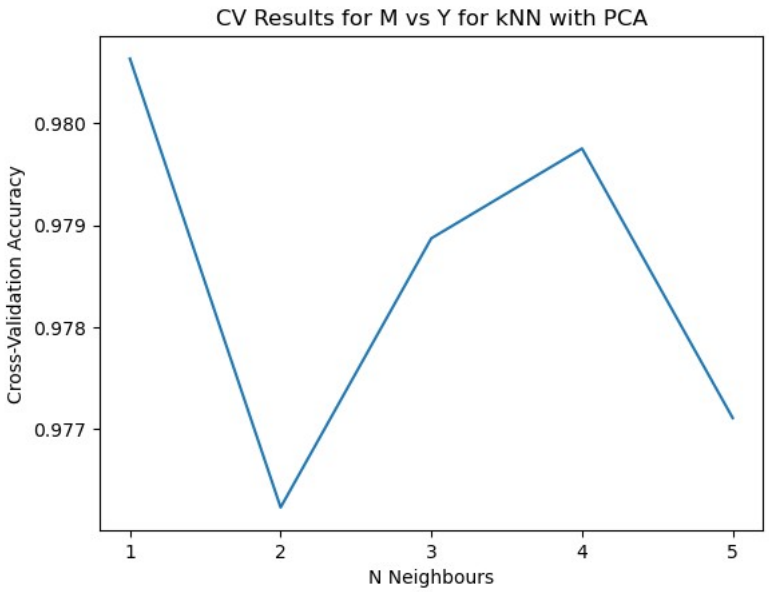

CV Results for M vs Y for kNN without Dimension Reduction
Cross-Validation Accuracy
0.99900
0.99875
0.99850
0.99825
0.99800
0.99775
0.99750
1
2
3
4
5
N Neighbours

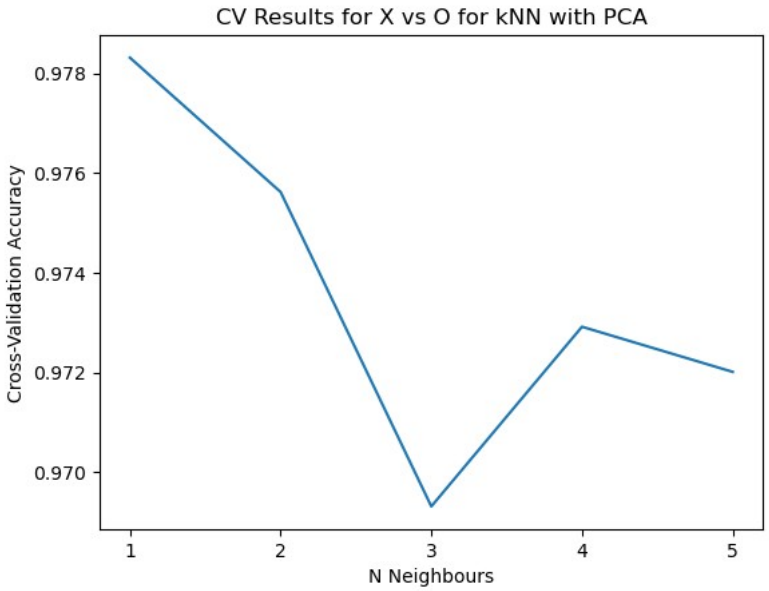


2.1.3 kNN Classifier with Dim Reduction using PCA

Method: kNN Classifier; PCA(4PC)
Hyperparameter: N (nearest) neighbours
Hyperparameter values: 1, 2, 3, 4, 5







2.2 ANN Classifier

2.2.1 Description of the ANN Classifier

ANNs consist of layers of interconnected nodes (neurons), including an input layer, one or more hidden layers, and an output layer. Each connection has a weight that is adjusted during the learning process. They learn by adjusting these weights based on the error of the output compared to the expected result. This process typically involves a method known as backpropagation combined with an optimization technique like gradient descent.

General Advantages

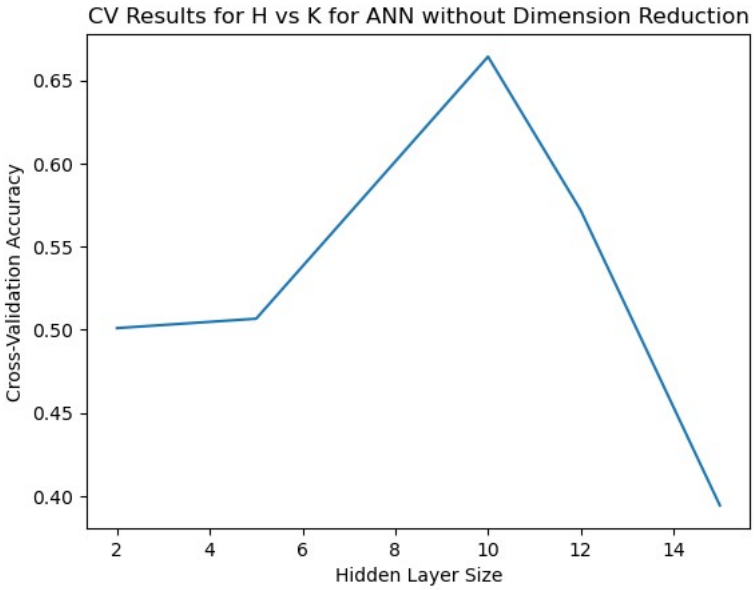
ANNs can model complex non-linear relationships and interactions between features, making them suitable for a wide range of applications, from image and speech recognition to time series prediction. They are capable of handling and processing large-scale and high-dimensional data effectively. Unlike many traditional algorithms, ANNs can automatically discover and learn relevant features from the input data. Theoretically, ANNs can approximate any continuous function, given enough neurons and data.

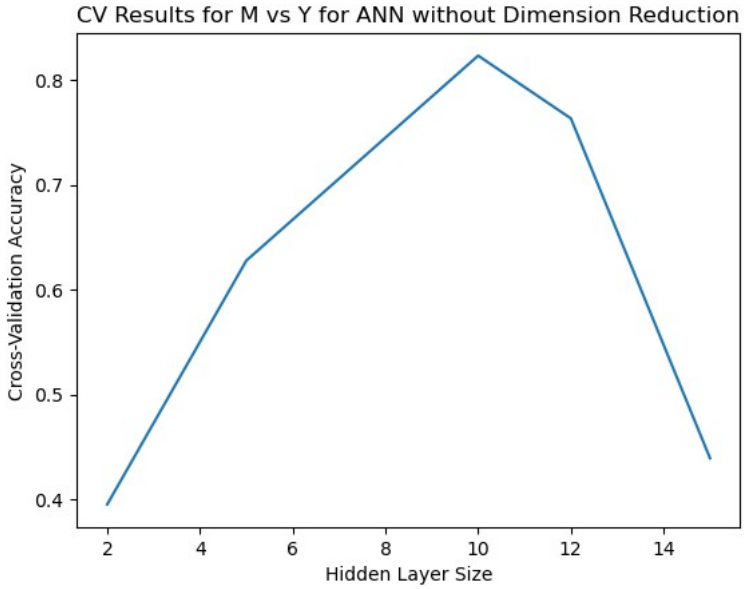
General Disadvantages

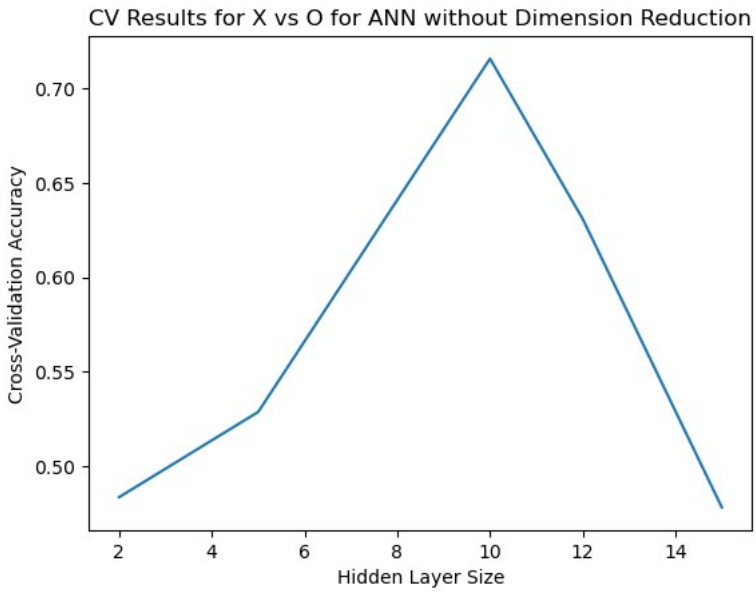
They require large amounts of data and computational resources for training, which might not be feasible for all projects. Without proper regularization and tuning, ANNs can easily overfit to the training data, especially if the network is too complex relative to the amount and diversity of the data. ANNs, especially deep networks, are often considered "black boxes" because it's challenging to interpret how they make decisions, which can be a significant drawback in fields requiring explainability. The performance of ANNs is highly sensitive to the choice of architecture and hyperparameters (like the number of layers, neurons, learning rate, etc.), and finding the right combination can be time-consuming — for example, for the classification problems in this assignment, it has taken us a long time to find the appropriate learning rate that does not cause overshooting via trial and error.

2.2.2 ANN Classifier Results without Dim Reduction

Method: ANN Classifier
Hyperparameter: hidden layer size
Hyperparameter values: 2, 5, 10, 12, 15

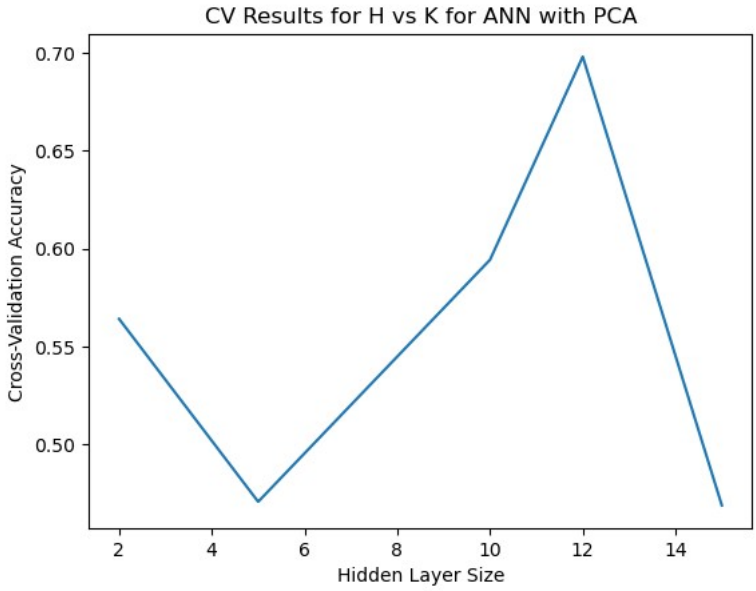

CV Results for H vs K for ANN without Dimension Reduction
Cross-Validation Accuracy
0.65
0.60
0.55
0.50
0.45
0.40
2
4
6
8
10
12
14
Hidden Layer Size

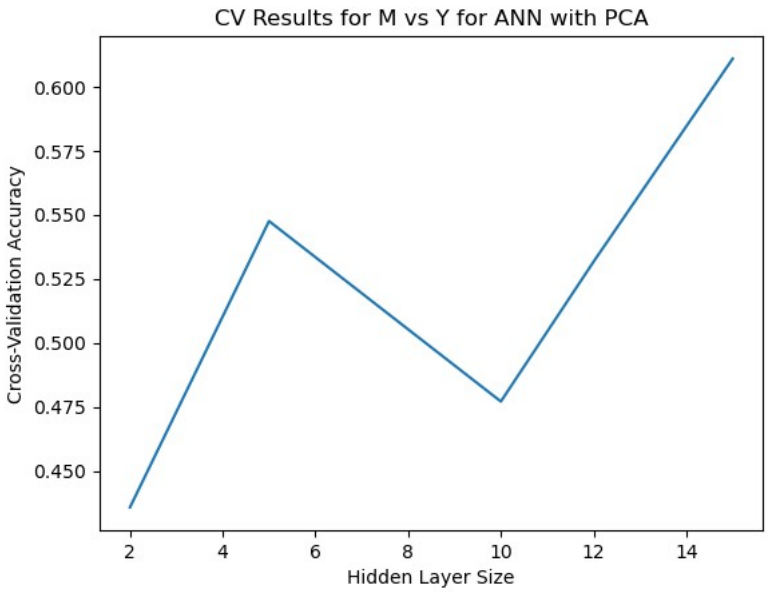

CV Results for M vs Y for ANN without Dimension Reduction
Cross-Validation Accuracy
0.8
0.7
0.6
0.5
0.4
2
4
6
8
10
12
14
Hidden Layer Size

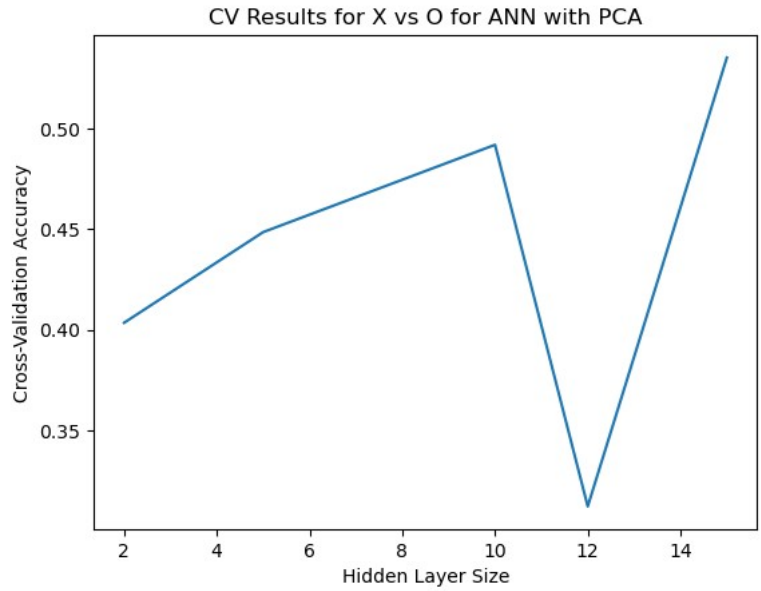


2.2.3 ANN Classifier with Dim Reduction using PCA

Method: ANN Classifier; PCA(4PC)
Hyperparameter: hidden layer size
Hyperparameter values: 2, 5, 10, 12, 15







2.3 Decision Tree Classifier

2.3.1 Description of the Decision Tree Classifier

A Decision Tree has a flowchart-like structure, where each internal node represents a "test" on an attribute, each branch represents the outcome of the test, and each leaf node represents a class label (decision taken after computing all attributes). The paths from root to leaf represent classification rules. In the context of classification, the learning process involves the algorithm deciding which features or combinations of features best split the data based on certain criteria, like Gini impurity or information gain, in

order to categorize the data into different classes as accurately as possible.

General Advantages

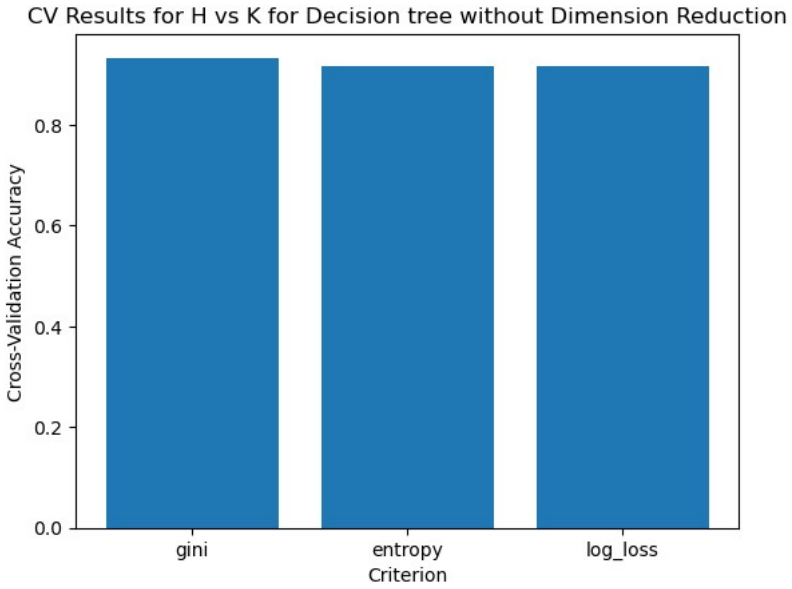
One of the key strengths of Decision Trees is their simplicity and interpretability. They are easy to understand and visualize, even for people without a background in data science. Decision Trees do not require feature scaling (normalization or standardization) before training, unlike many other classifiers. They are capable of handling non-linear relationships between features and labels effectively. They work well with both types of data, which makes them versatile. Decision Trees implicitly perform feature selection, which can be an advantage in situations with many features.

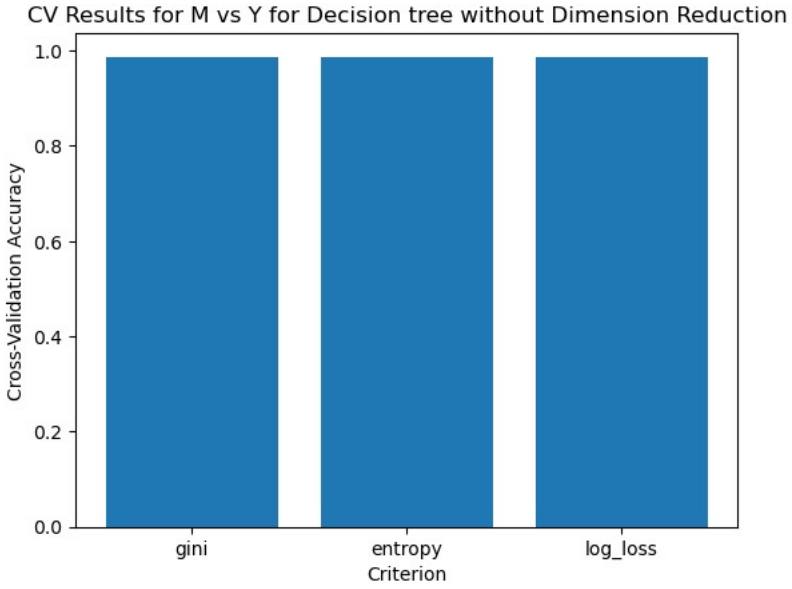
General Disadvantages

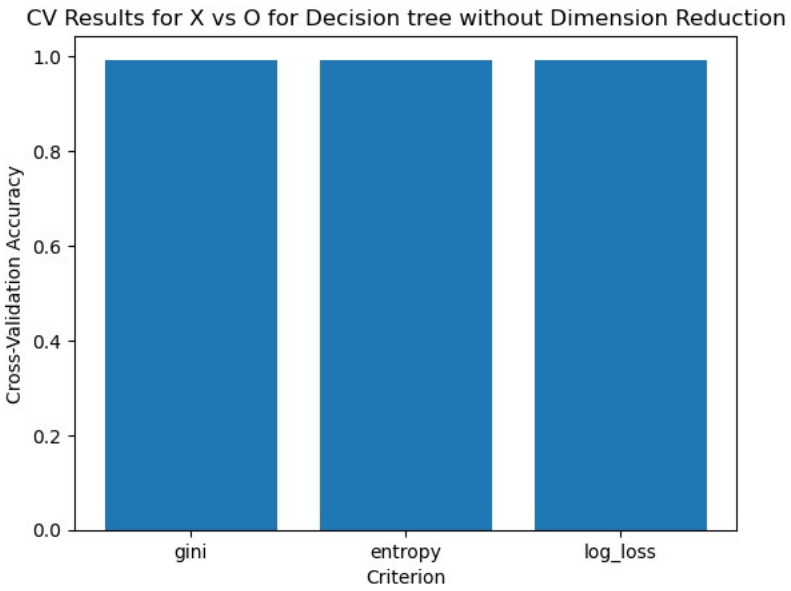
A common problem with Decision Trees, especially if they are deep, is overfitting to the training data. They tend to memorize the training data rather than learning to generalize from patterns. Techniques like pruning (removing parts of the tree that provide little power to classify instances) are necessary to avoid this. Alternatively, mall variations in the data can result in a completely different tree being generated. This is because the algorithm is highly sensitive to the data on which it is trained. If some classes dominate, decision trees can create biased trees. Balancing the dataset before training the tree can help mitigate this. For datasets with a lot of features and data points, the tree can become very complex and unwieldy, making it hard to interpret and slow to make predictions.

2.3.2 Decision Tree Classifier Results without Dim Reduction

Method: Decision Tree Classifier
Hyperparameter: criterion (for splitting)
Hyperparameter values: gini, entropy, log_loss

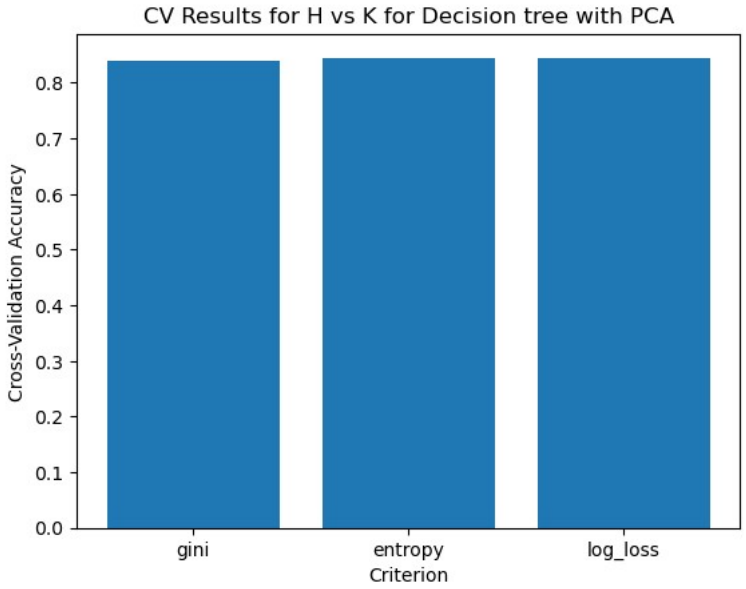

CV Results for H vs K for Decision tree without Dimension Reduction
Cross-Validation Accuracy
0.0
0.2
0.4
0.6
0.8
gini
entropy
log_loss
Criterion

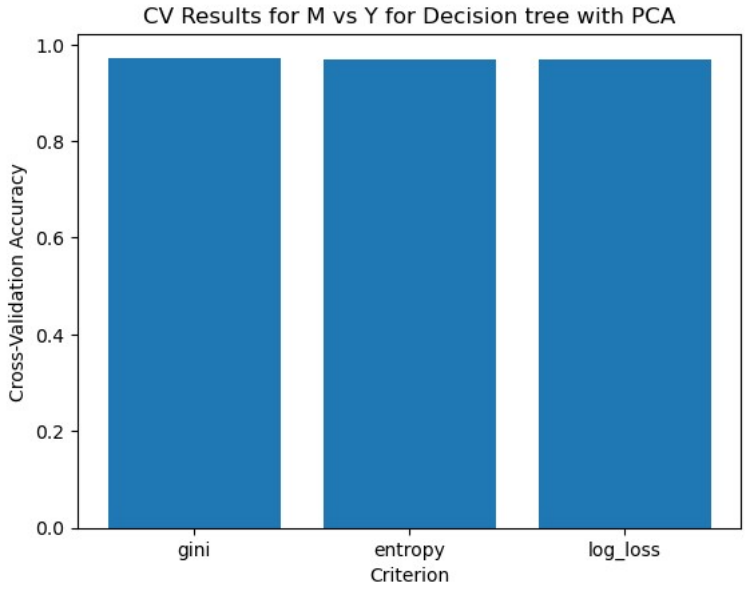

CV Results for M vs Y for Decision tree without Dimension Reduction
Cross-Validation Accuracy
0.0
0.2
0.4
0.6
0.8
1.0
gini
entropy
log_loss
Criterion

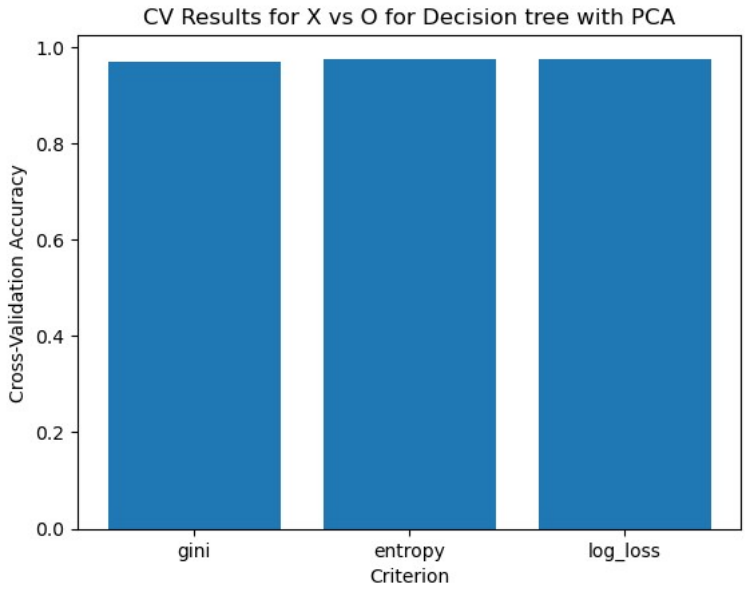


2.3.3 Decision Tree Classifier with Dim Reduction using PCA

Method: Decision Tree Classifier; PCA(4PC)
Hyperparameter: criterion (for splitting)
Hyperparameter values: gini, entropy, log_loss







2.4 Random Forest Classifier (RFC)

2.4.1 Description of the Random Forest Classifier

Random Forest is an ensemble learning method in machine learning that constructs a multitude of decision trees during training. The final prediction is determined by averaging or voting over the predictions of individual trees for classification tasks or by averaging for regression tasks.

General Advantages

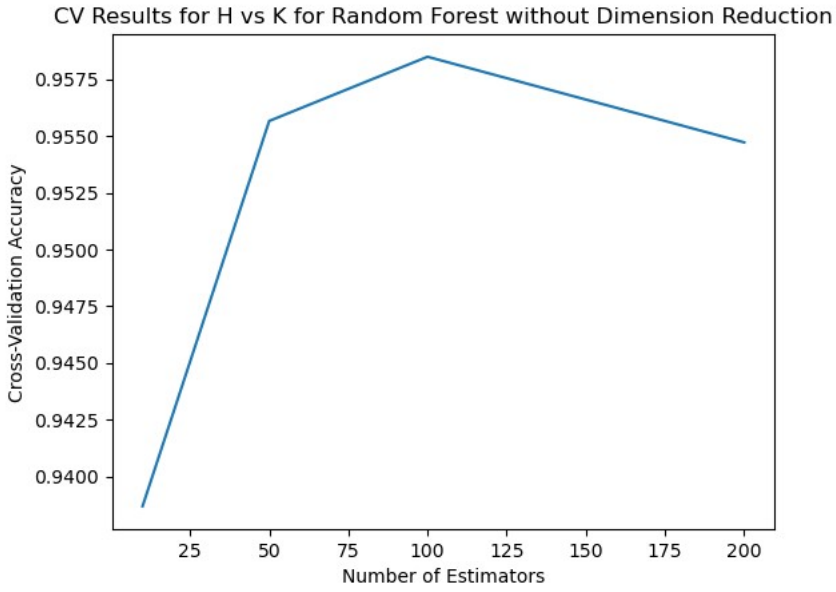
High Accuracy: Random Forest generally yields high accuracy by combining the

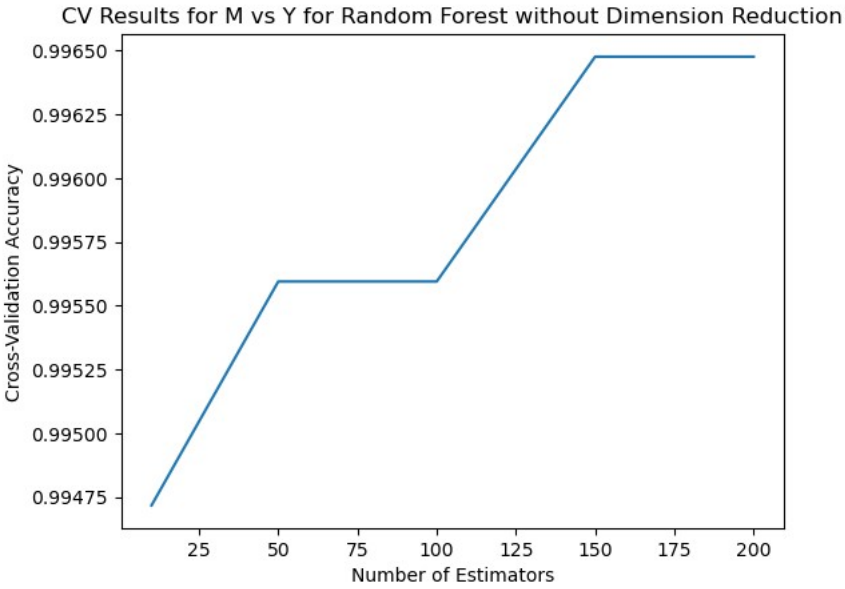
predictions of multiple decision trees, reducing the risk of overfitting.
Robust to Overfitting: The ensemble approach and the use of random subsets of features for each tree make Random Forest robust to overfitting and improve generalization performance.

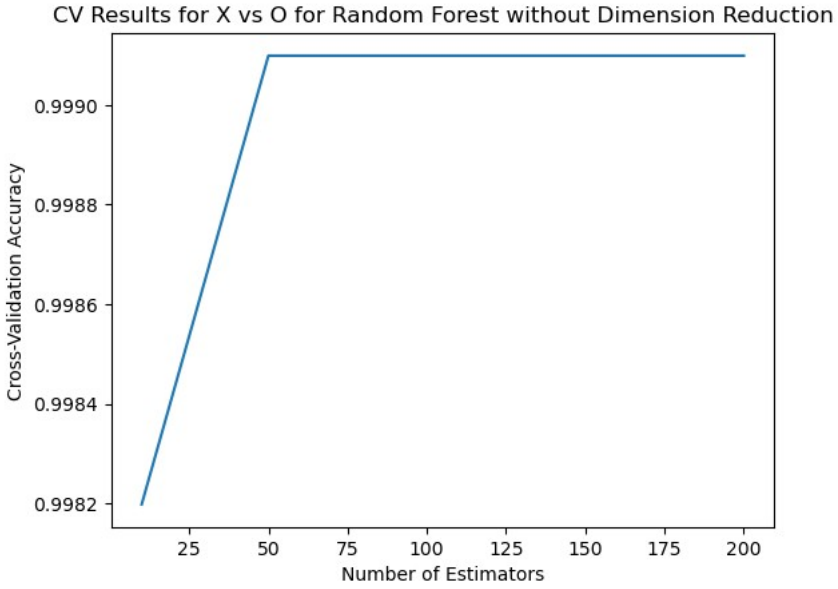
General Disadvantages
Complexity: The model can become complex and harder to interpret as it consists of multiple decision trees.
Computationally Intensive: Training a large number of trees can be computationally intensive, especially for large datasets and complex models.

2.4.2 RFC Results without Dim Reduction

Method: RFC; PCA(4PC)
Hyperparameter: number of estimators
Hyperparameter values: 10, 50, 100, 150, 200

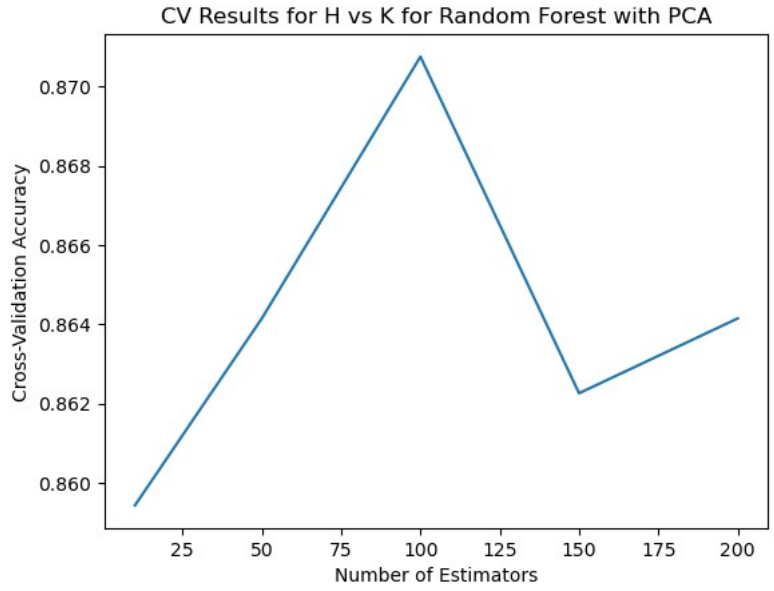


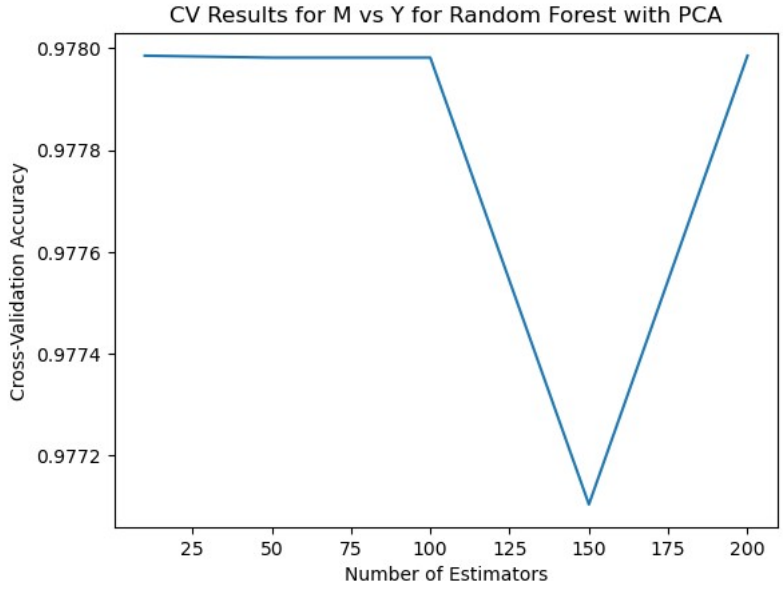


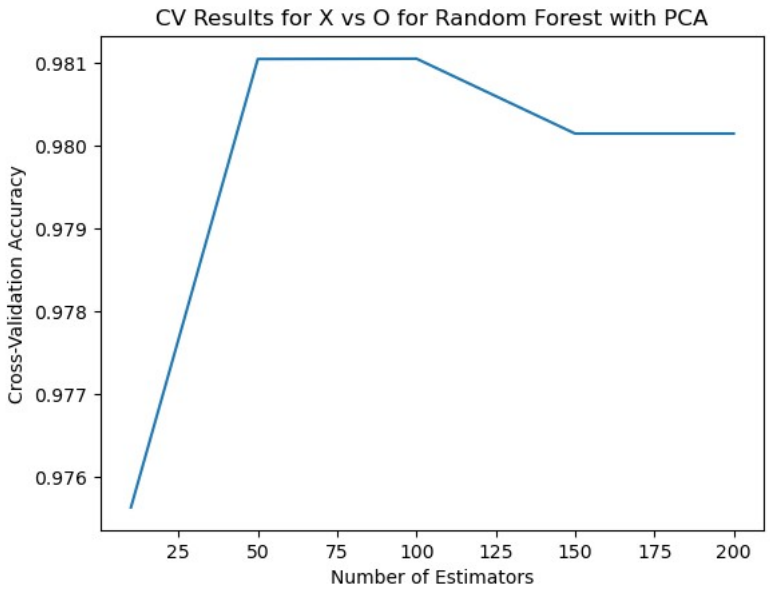


2.4.3 RFC with Dim Reduction using PCA

Method: RFC; PCA(4PC)
Hyperparameter: number of estimators
Hyperparameter values: 10, 50, 100, 150, 200


CV Results for H vs K for Random Forest with PCA
Cross-Validation Accuracy
0.870
0.868
0.866
0.864
0.862
0.860
25
50
75
100
125
150
175
200
Number of Estimators


CV Results for M vs Y for Random Forest with PCA
Cross-Validation Accuracy
0.9780
0.9778
0.9776
0.9774
0.9772
25
50
75
100
125
150
175
200
Number of Estimators



2.5 Naïve Bayes Classifier (NBC)

2.5.1 Description of the Naïve Bayes Classifier

Naïve Bayes Classifier is a probabilistic machine learning model based on Bayes' Theorem, assuming independence among features. Known for its simplicity and efficiency, it is commonly used for tasks like text classification, such as spam filtering and sentiment analysis.

General Advantages

Simplicity and Efficiency: Naïve Bayes is a simple and computationally efficient classification algorithm. It is particularly effective for large datasets and performs well in situations where the independence assumption holds, making it easy to implement and scale.

Good Performance with Categorical Data: Naïve Bayes performs well when dealing with categorical data and is robust in situations where the features are discrete. It is especially suitable for text classification tasks, such as spam filtering and sentiment analysis, where the presence or absence of specific words determines the classification.

General Disadvantages

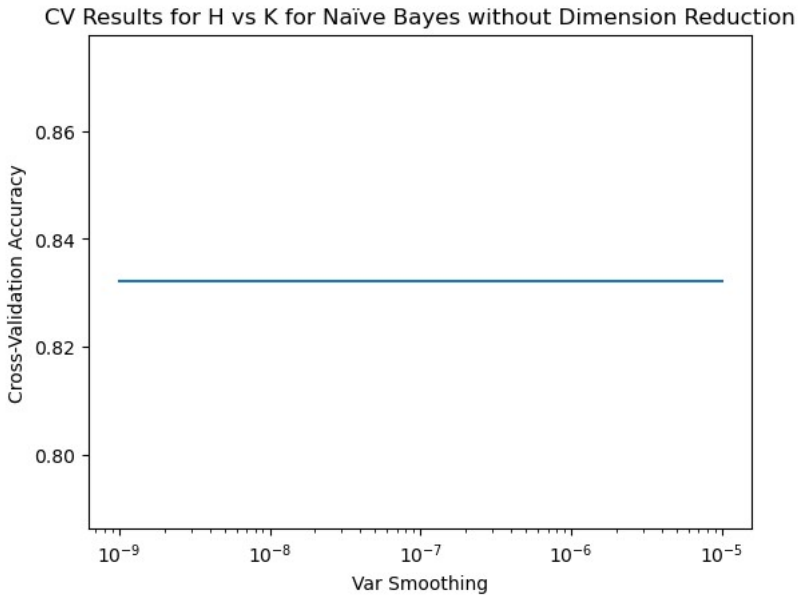
Assumption of Feature Independence: One major limitation of Naïve Bayes is its assumption of feature independence. In real-world scenarios, it's common for features to be correlated, and this assumption may not always hold true. This can lead to suboptimal performance when dealing with highly dependent features.

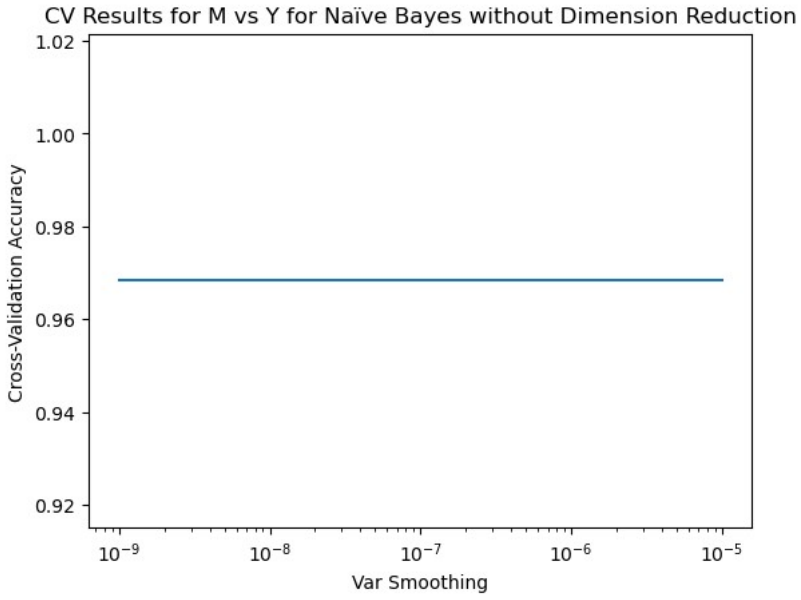
Sensitive to Input Data Quality: Naïve Bayes can be sensitive to the quality of the input data. If the training dataset is small or contains irrelevant or misleading features, the classifier's performance may be adversely affected. It

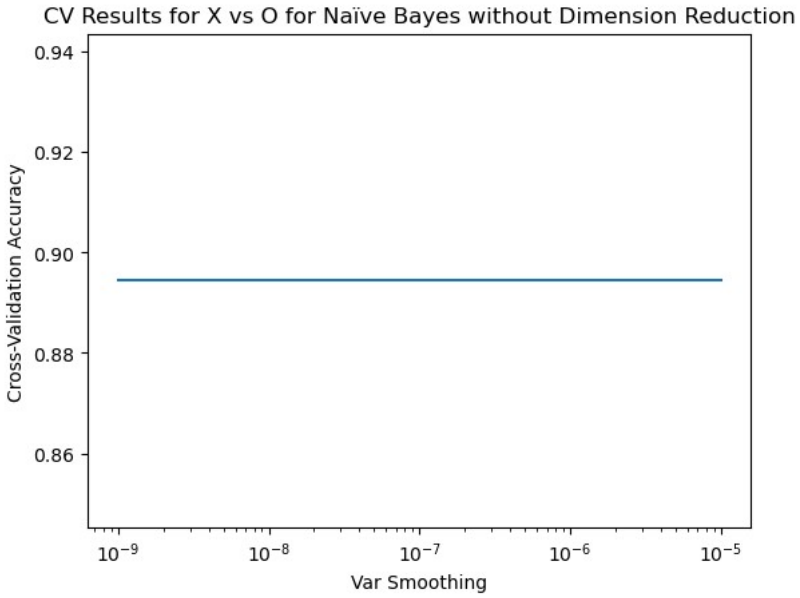
may struggle with complex relationships and nuances present in the data, leading to less accurate predictions compared to more sophisticated algorithms.

2.5.2 NBC Results without Dim Reduction

Method: Gaussian Naïve Bayes Classifier
Hyperparameter: var smoothing
Hyperparameter values: 1e-9, 1e-8, 1e-7, 1e-6, 1e-5

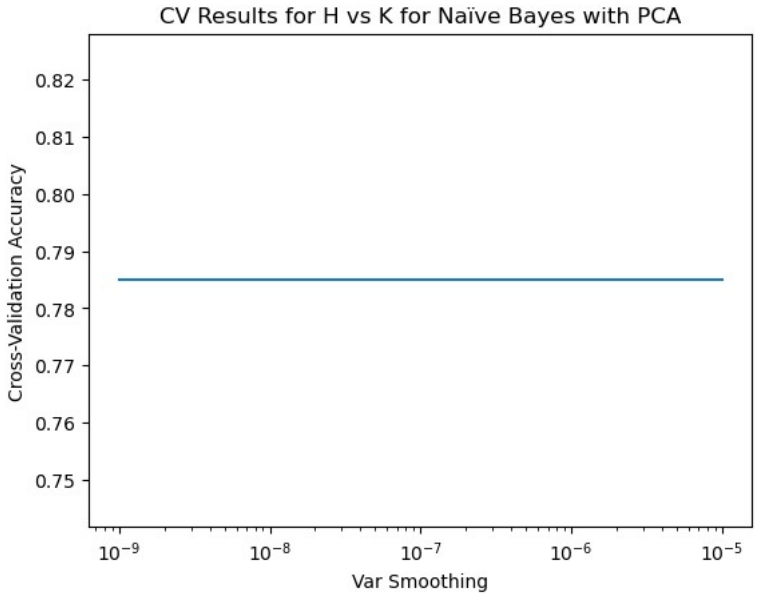


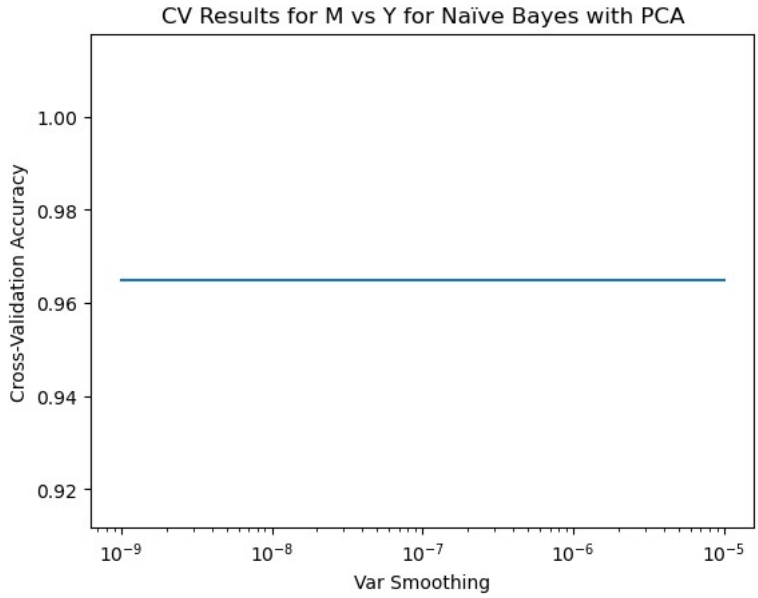


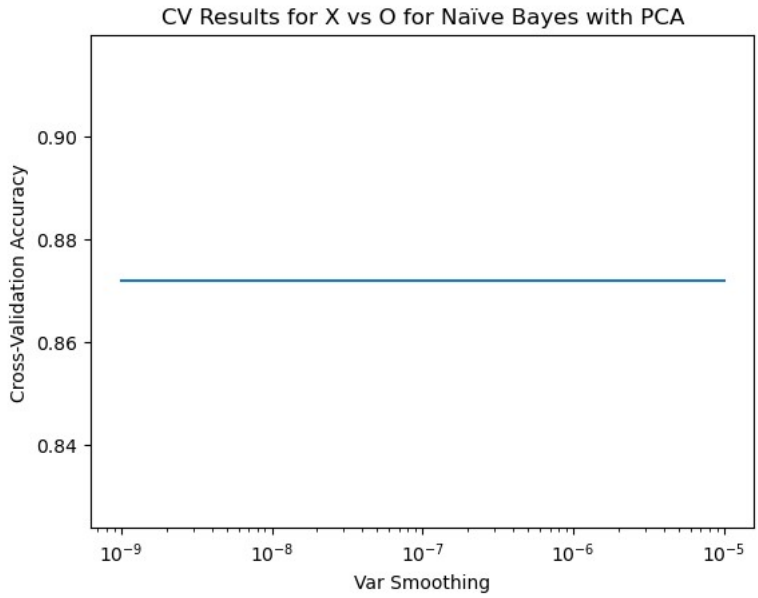


2.5.3 NBC Results with Dimen Reduction using PCA

Method: Gaussian Naïve Bayes Classifier; PCA(4PC)
Hyperparameter: var smoothing
Hyperparameter values: 1e-9, 1e-8, 1e-7, 1e-6, 1e-5







2.6 SVM Classifier

2.6.1 Description of the SVM Classifier

The core idea behind SVM is to find a hyperplane in an N-dimensional space that distinctly classifies the data points into different categories. In simpler terms, SVM finds the best boundary that separates classes of data. SVM aims to maximize the margin (distance) between the data points of different classes, thereby creating a more robust classifier. One of the key features of SVM is its use of the kernel trick, which allows it to handle non-linearly separable data by transforming it into a higher-dimensional space where it can be

linearly separated.

General Advantages

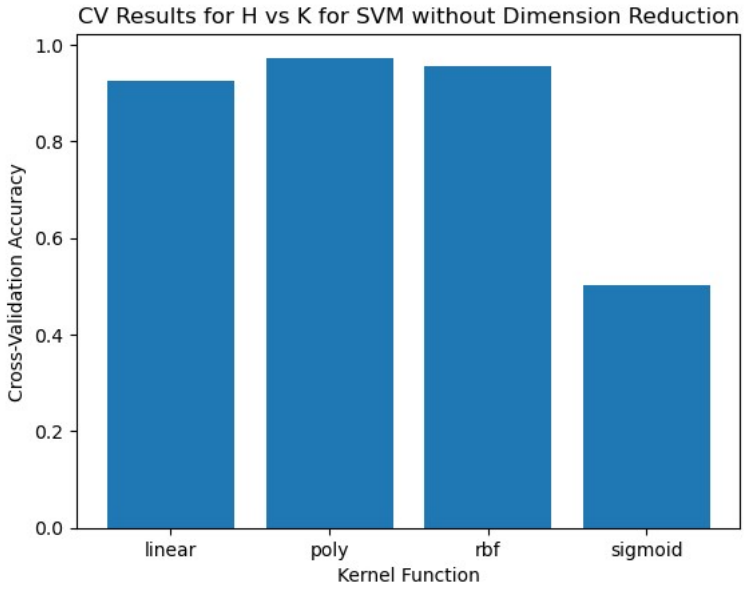
SVMs are particularly effective in situations where the number of dimensions is greater than the number of samples, which is a common scenario in text and image classification. Using different kernel functions (linear, polynomial, radial basis function, sigmoid), SVMs can be adapted for various decision boundaries, from linear to complex non-linear relationships. Due to the regularization parameter, SVMs have a high generalization capability, which prevents overfitting effectively. SVMs are well-suited for sparse data, which is common in text and other NLP problems.

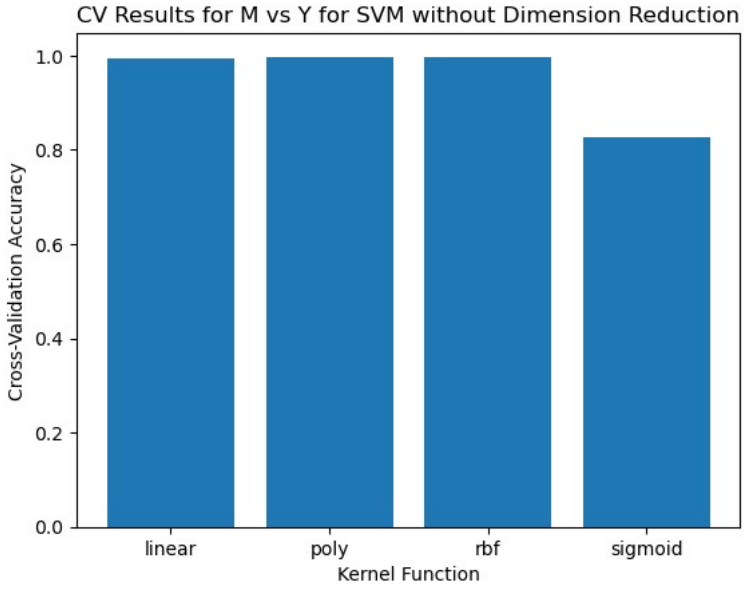
General Disadvantages

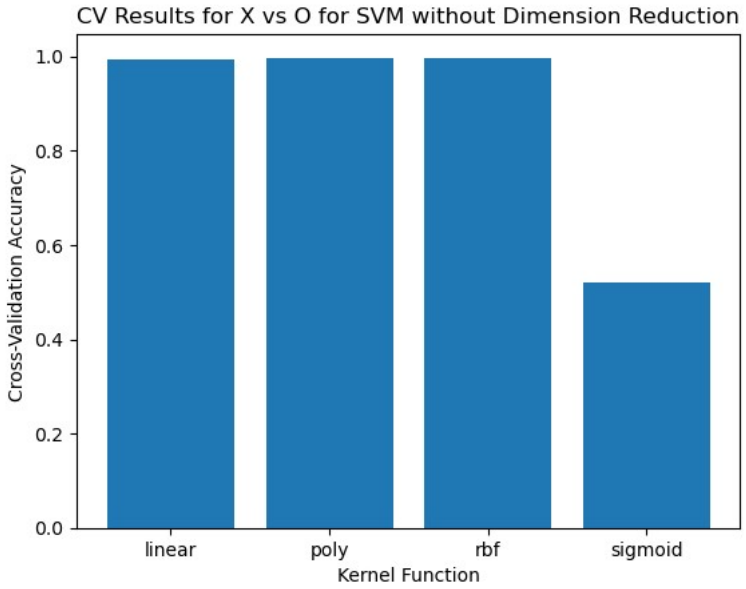
The effectiveness of SVM largely depends on the selection of the kernel. The choice is not always straightforward and can require careful tuning and domain knowledge. For large datasets, the training time with SVM can be quite long, which makes it less practical in situations where computational efficiency is a priority. SVMs can be memory-intensive during training, which can be a limitation for large datasets. Understanding and interpreting the final model can be challenging, especially with non-linear kernels. SVMs are sensitive to the choice of the regularization and kernel parameters, and inappropriate choices can lead to overfitting or underfitting. In scenarios where classes are significantly overlapping, the performance of SVM can decrease.

2.6.2 SVM Classifier Results without Dim Reduction

Method: SVM Classifier
Hyperparameter: kernel function
Hyperparameter values: linear, poly, rbf, sigmoid

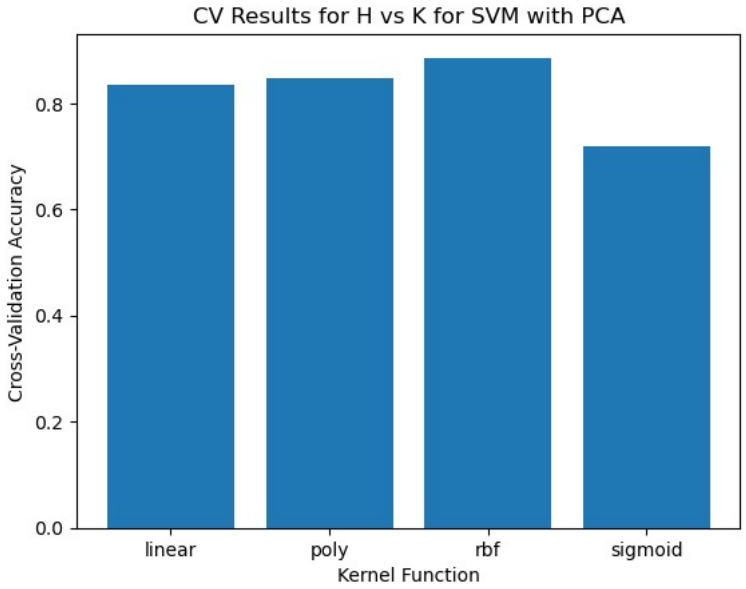

CV Results for H vs K for SVM without Dimension Reduction
Cross-Validation Accuracy
1.0
0.8
0.6
0.4
0.2
0.0
linear
poly
rbf
sigmoid
Kernel Function

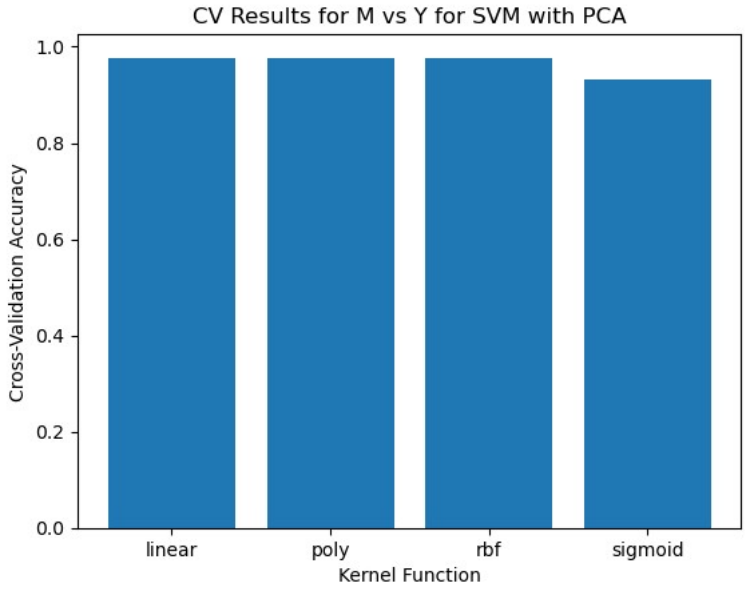

CV Results for M vs Y for SVM without Dimension Reduction
Cross-Validation Accuracy
1.0
0.8
0.6
0.4
0.2
0.0
linear
poly
rbf
sigmoid
Kernel Function

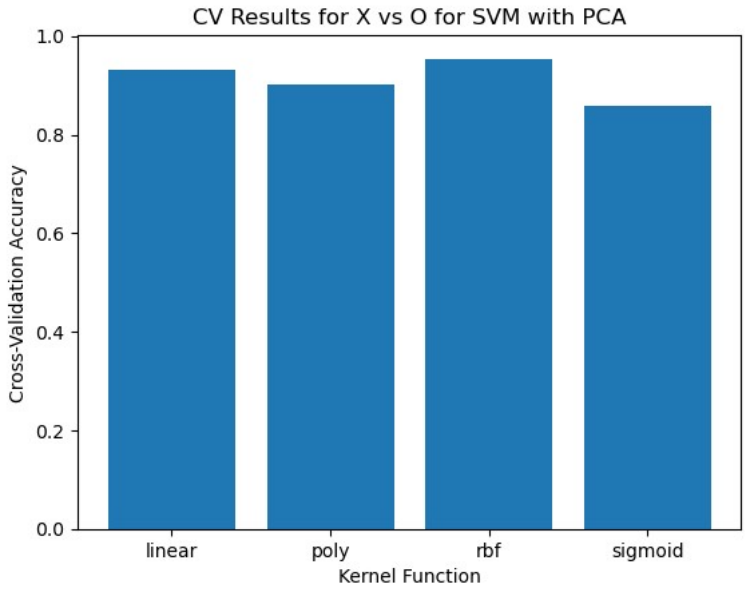


2.6.3 SVM Classifier Results with Dimen Reduction using PCA

Method: SVM Classifier; PCA(4PC)
Hyperparameter: kernel function
Hyperparameter values: linear, poly, rbf, sigmoid







3 Discussion

3.1 Comparison of Classifier Performance and Run Time

AUC		H vs K	M vs Y	X vs O
kNN Classifier	without DR	cv: 0,940 val: 0,980 time: 0,79	cv: 0,999 val: 1,000 time: 0,84	cv: 1,000 val: 1,000 time: 0,93

	with DR	cv: 0,876 val: 0,932 time: 0,53	cv: 0,981 val: 0,956 time: 0,56	cv: 0,978 val: 0,981 time: 0,51
ANN Classifier	without DR	cv: 0,664 val: 0,676 time: 0,95	cv: 0,823 val: 0,848 time: 0,90	cv: 0,716 val: 0,649 time: 0,91
	with DR	cv: 0,698 val: 0,720 time: 1,17	cv: 0,611 val: 0,560 time: 1,23	cv: 0,535 val: 0,470 time: 1,17
Decision Tree Classifier	without DR	cv: 0,933 val: 1,000 time: 0,18	cv: 0,988 val: 0,987 time: 0,16	cv: 0,994 val: 1,000 time: 0,14
	with DR	cv: 0,845 val: 0,910 time: 0,33	cv: 0,973 val: 0,970 time: 0,29	cv: 0,976 val: 1,000 time: 0,28
RFC	without DR	cv: 0,958 val: 1,000 time: 6,65	cv: 0,996 val: 0,994 time: 6,08	cv: 0,999 val: 1,000 time: 5,86
	with DR	cv: 0,871 val: 0,950 time: 10,42	cv: 0,978 val: 0,990 time: 8,95	cv: 0,981 val: 1,000 time: 9,01
NBC	without DR	cv: 0,832 val: 0,885 time: 0,22	cv: 0,968 val: 0,956 time: 0,25	cv: 0,894 val: 0,922 time: 0,21
	with DR	cv: 0,785 val: 0,845 time: 0,13	cv: 0,965 val: 0,987 time: 0,12	cv: 0,872 val: 0,903 time: 0,13
SVM Classifier	without DR	cv: 0,973 val: 0,987 time: 0,92	cv: 0,997 val: 0,994 time: 0,66	cv: 0,997 val: 1,000 time: 0,67
	with DR	cv: 0,886 val: 0,926 time: 0,70	cv: 0,977 val: 0,987 time: 0,30	cv: 0,954 val: 0,968 time: 0,42

3.1.1 Performance and Run Time without Dim Reductions

We compared the accuracy and runtime of the six classifiers. Below are our findings.

1. kNN (k-nearest neighbours) has a small time overhead and a very high

accuracy.

2. Artificial Neural Network has a small time overhead, but it gives much lower accuracy (which is even worsened with PCA).
3. Decision tree has a very small time overhead and a very high accuracy.
4. Random Forest would have a longer time overhead while obtaining significantly higher accuracy. The average training time for Random Forest is around 6,2 seconds, and it extends to around 9,5 seconds with PCA.
5. Naïve Bayes has a very small time overhead and an acceptable sacrifice in accuracy. While performing specific tasks, we need to take such trade-offs into account and make choices that fit the needs.
6. SVM has a small time overhead and a high accuracy similar to that of Random Forest.

3.1.2 Performance and Run Time with Dim Reductions

PCA improved the performances of the models almost in no case.

1. In the training of each classification model, we found that the use of of we found that the use of PCA in different tasks led to a decrease in accuracy in general.
2. Although the introduction of PCA has resulted in less training time and higher accuracy in some problems, this effect is not stable and significant enough.
3. In kNN, Naïve Bayes and SVM models, PCA led to a slight reduction in time overhead; while in ANN, Decision tree and Random Forest models, PCA led to an increase (even doubling) of the training duration.

Taken together, we judge that PCA is not suitable for dimensionality reduction in this problem, and the reason may lie in the characteristics of the dataset itself.

3.2 Lessons Learned and Model Selection

Model Choice for the Problem

We will choose the Random Forest model because this model has shown excellent performance in the training of each task and demonstrated robust performance after applying PCA. The fact that the random forest model performs better on the validation set than on the training set indicates that this model has good generalization performance on this problem. At the same time, it has an acceptable time overhead.

Impact of Dim Reduction

PCA shows different impacts on different classifiers, but not positively overall.

1. For k-nearest neighbours, PCA slightly reduced the time overhead while

making the accuracy slightly lower as well.

2. For Artificial Neural Network, PCA not only significantly lowered the accuracy, but also increased the time overhead as well.
3. For Decision tree, PCA slightly decreased the accuracy and almost doubled the time overhead (although the original time overhead is very small).
4. For Random Forests, PCA did not effectively improve accuracy while significantly increasing the time overhead.
5. For Naïve Bayes, PCA slightly reduced the time overhead while making the accuracy significantly lower.
6. For SVM, PCA slightly reduced the time overhead while making the accuracy slightly lower as well.

Therefore, PCA is not recommended for this problem.

Future Approaches

For different datasets, the applicable methods can be significantly different, and we need to explore how datasets of different nature show different characteristics under different dimension and different distance measures respectively to choose a more optimal method. Specifically, we should try more choices in dimensionality reduction, and we should also try to adjust more parameters, which may be helpful.

Addition Considerations

I would recommend more feature engineering before training to understand the characteristics of the dataset, based on which the appropriate model can be selected. This involves processes such as identifying and creating relevant features that could improve model performance, handling missing values, encoding categorical variables, and normalizing or standardizing features. By thoroughly understanding and engineering the features, it becomes easier to choose a model that aligns well with the data's structure and the problem's requirements.