Conditional Clustering Method on KNN for Big Data

A Thesis Presented in Partial Fulfillment of
Bachelor of Science
(Honors in Statistics)

Qi Chen

Supervisor: Prof. Mark Fredrickson

Final Version: April 23, 2022

Abstract

This thesis endeavors to address the challenges faced by k-nearest neighbor (KNN) classifiers when handling big data, particularly concerning large storage requirements and extended training times. The proposed solution revolves around data filtering techniques. Drawing upon the fundamental principle of KNN classifiers, which assumes that similar data possess similar conditional distributions regarding the response variable, this study advocates for employing clustering methods to segment the training data and subsequently filter it by selecting the closest cluster as the training set. Stem from Bayes' rule and the Mixture distribution of data, the clustering refinement involves utilizing clustering techniques conditional on the class of responses. To execute this approach, the algorithm prefers hierarchical clustering model, chosen for its stability and efficiency. To counterbalance the loss of information resulting from filtering the training set, the algorithm replaces the standard KNN classifier with a local KNN classifier. By locally adjusting the parameters of the KNN classifier, the model achieves a more favorable trade-off between bias and variance. The effectiveness of the proposed model is evaluated using three sets of real-world data: Fashion MNIST, Forest Cover Type Prediction, and Online Shoppers Intention from the UCI Machine Learning repository. The results of the tests demonstrate that the conditional clustering method significantly enhances runtime efficiency, while employing the local KNN classifier improves model prediction ability. Notably, the number of clusters proves to be a critical factor influencing the model's accuracy. While increasing the number of clusters may reduce the filtered training dataset size, thus resulting in information loss, a higher number of clusters affords the local KNN classifier greater opportunities to strike a balance between variance and bias, consequently lowering model risk.

Contents

1 Introduction

In today's era of exponential data growth, the sheer volume and complexity of datasets present significant computational challenges for traditional machine learning algorithms. Among these, k nearest neighbor method stands out for its simplicity and intuitive nature. However, its reliance on calculating distances between each observation and all training samples makes it computationally expensive, especially as dataset sizes escalate. The inefficiency of k nearest neighbor method becomes more pronounced with big data due to several factors. One of the most important is that the computational complexity of k nearest neighbor method scales linearly with the size of the training dataset, resulting in longer processing times and increased memory requirements.

To mitigate these challenges and improve the scalability of k nearest neighbor method, integrating clustering as a pre-processing step emerges as a promising solution. By partitioning the training data into distinct clusters based on similarity measures, clustering reduces the effective size of the dataset that k nearest neighbor method needs to operate on. The specific operation is only using training data within the closest cluster to make prediction instead of considering the entire dataset.

1.1 Background Information of KNN Classifier

1.1.1 Bayesian classifier

The Bayesian classifier is considered optimal as it minimizes misclassification rates based on the data distribution. However, its optimality relies on meeting certain conditions, such as knowing the true conditional probability distribution of response variable given predictors' value, which is often unattainable in real-world datasets. Additionally, if one has access to the population distribution of responses conditioned on predictors, employing a model for prediction becomes unnecessary. The KNN classifier, resembling the Bayesian classifier, is a general method that utilizes an empirical conditional distribution of data within a small neighborhood community rather than at a specific point.

Assuming Y is selecting the class C_k
Given the prior probability of class C_k: $P(C_k)$
Given conditional distribution of x given class C_k: $P(x|C_k)$
Given marginal distribution of x:$P(x)$

The Bayes classifier is defined as:

$$\widehat{C_k} = argma_k P(C_k|x)$$

1.1.2 K-nearest neighbor (KNN) classifier

The K-nearest neighbor (KNN) classifier stands out as a robust supervised machine learning algorithm suitable for classification tasks. Operating as a non-parametric method, it dispenses with assumptions about the underlying data distribution. KNN operates on an instance-based approach, devoid of explicit model construction from training data. Instead, it classifies new data points by comparing them with existing training data points (Hastie, Tibshirani, & Friedman, n.d.).

The classification process in the KNN algorithm involves finding the K nearest neighbors for a given data point in metric space (where K is the only parameter of the KNN classifier). Then, these adjacent data points with the closest distance to the test point are used to predict the class of the test point.

Test point: X
Training dataset: T
Set of k nearest neighbors of test point: S_x

$$S_x \subseteq \mathrm{T} \mid |\mathrm{S_x}| = \mathrm{k} \wedge \forall \mathrm{x\_1} \subseteq \mathrm{T}\ \mathrm{S\_x} \forall x\_2 \subseteq S\_x\ \mathrm{dist}(\mathrm{X}, \mathrm{x\_1}) > \mathrm{dist}(\mathrm{X}, \mathrm{x\_2})$$

In practice, distance measures such as Euclidean distance are often used to determine the similarity between data points. It is also worth noting that choosing an appropriate k value can significantly affect the performance of the KNN classifier. Parameter k control the trade-off between bias and variance for KNN classifier.

1. Bias: With a smaller k, the KNN classifier has lower bias because it considers fewer neighbors for prediction. This allows the model to capture more complex patterns in the data. However, this can also lead to overfitting, where the model fits the training data too closely and fails to generalize well to new.
2. Variance: Conversely, with a larger k, the KNN classifier has higher bias but lower variance. By considering more neighbors, the model relies on a larger and more diverse set of data points for prediction. This typically leads to smoother decision boundaries and reduces the impact of noise in the data. However, a larger k may also lead to underfitting, where the model is too simplistic and fails to capture important patterns in the data.

Once the nearest neighbors are identified, the KNN classifier assigns a class to the test point by considering the class labels of its neighbors. Classification is usually done through a voting mechanism. By selecting the majority class in the neighborhood, the classifier seeks to minimize the classification error and improve the accuracy of predictions.

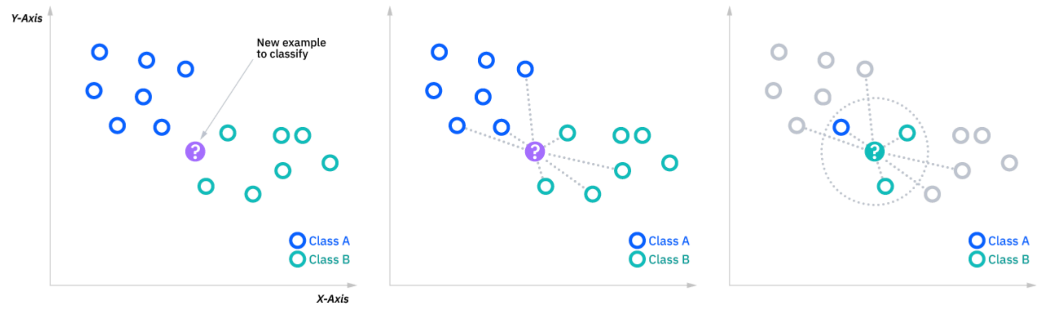

Figure 1: KNN classifier with Euclidean distance (IBM, n.d.)

1.2 Main Research

The KNN classifier estimates the conditional distribution of data at a test point through the distribution of samples in the neighborhood community. Larger, more focused, and more representative data sets enable KNN classifiers to capture information more accurately about population distribution. Since larger data sets provide the classifier with a richer and more diverse set of examples to learn from. Additionally, larger datasets allow the KNN classifier to create smaller neighborhoods while maintaining a fixed number of neighbors. This results in a more concentrated neighborhood around the data points of interest. A more concentrated neighborhood will include point closer (similar) to test point resulting in more reliable classification decisions. However, it is worth noting that larger datasets also come with trade-off, such as longer model training time and potentially higher computational costs.

The scatter plot presented below illustrates the influence of various randomly selected sample sizes on model training time, all while keeping the value of k to be 10.

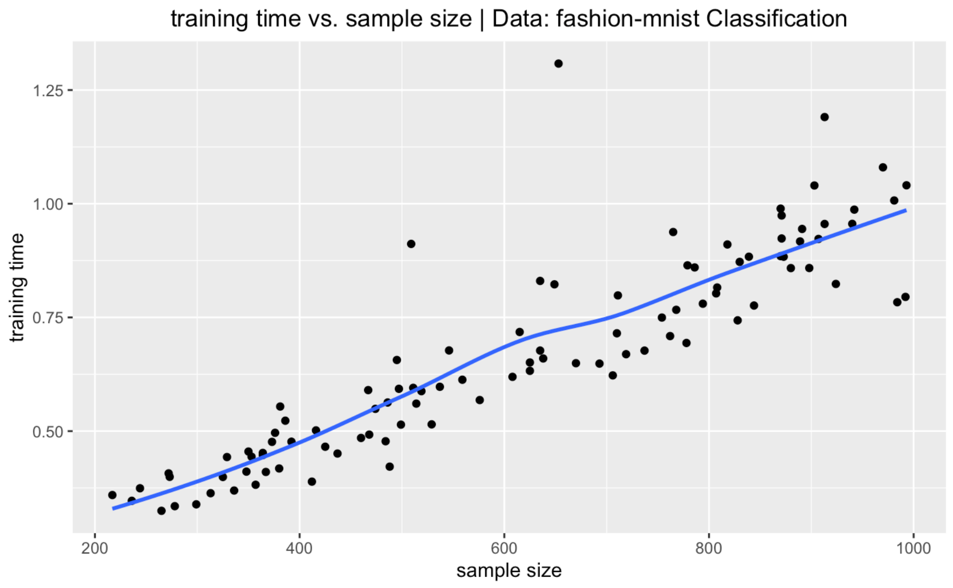

Figure 2

Furthermore, the parameters of the KNN classifier may need to be adjusted based on the size of the data set. For example, the choice of the number of neighbors (k) may need to be adapted to the dataset size to ensure optimal performance. Specifically, the parameters of the KNN classifier may be different for data sets of different sizes. This means that parameters trained from a smaller dataset may not be directly applicable to a larger dataset, even if both datasets are representative of the population. This paper aims to help building a better KNN classifier training process that requires less training time and maintains comparable accuracy.

1.3 Literature Review

Stone has proved that when k and n approach infinity and k/n approaches 0, KNN estimator is universally consistent, and the risk converges to the Bayes risk (Smith, 1999). A greater training set offers the model more information, potentially leading to heightened accuracy, yet necessitating extended training time. Furthermore, the trade-off between training time and model accuracy hinges on the complexity of the data, a factor that is challenging to quantify. Consequently, determining the optimal size of a training set for achieving desired accuracy levels becomes challenging, complicating the selection of an appropriate tuning range.

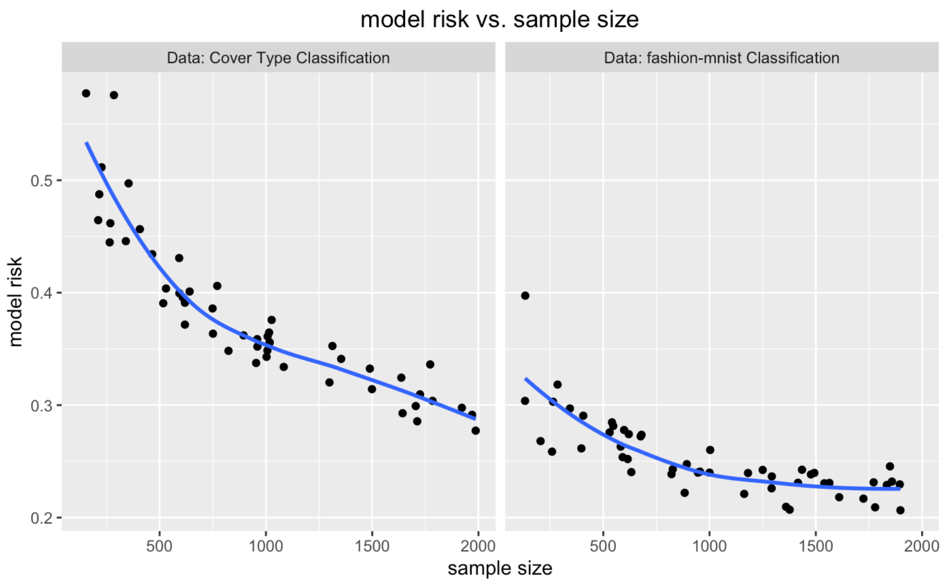

Figure 3

Based on the problem of finding similar tumor shapes, Korn et al. proposed a faster Nearest Neighbor Search in 1996. The method mainly solves two problems: how to measure the distance between two shapes and how to do better than sequentially scanning the entire database given such a distance function. Method uses F-index with an n-d R-tree. The specific operation is to submit the calculated size distribution of each image to a range or k-nearest neighbor to search in the R-tree. This method can complete the search faster without compromising correctness (Korn et al, 1996).

The performance of the KNN classifier is heavily influenced by the data distribution. KNN operates on the believe that closely located data points exhibit similar conditional distributions of the response variable given predictors. Consequently, in regions where data is densely distributed, the model can access sufficiently similar data points, potentially leading to high accuracy. However, when test data is situated in sparse data distribution areas, the training set used for classification may lack similarity resulting in decreased prediction reliability. Cannings et al. propose a method call local KNN which allow choice of k for KNN classifier depends on estimated density of x, so that estimation k value is varying based on estimation of distribution of predictors. Cannings et al use family of bandwidths {h(Xi) : i = 1,...,n} instead of h = h(x) as density estimation so that leading term in the asymptotic bias expansion will be canceled (Doe & Smith, 2020)

Because KNN classifier is an instance-based classifier, filtering data is an effective way to improve the training process while maintaining classification ability based on the training data. One such method is the Condensed Nearest Neighbor (CNN) rule, proposed by P.E. Hart in 1968. This method preserves the essence of the nearest neighbor idea but use the filtered consistent training subset. The CNN rule works by iteratively collecting data points that are misclassified by the model based on the existing collected data. By doing so, CNN filter a consistent subset by removing superfluous observation that will not affect the classification accuracy of the training set (Hart, 1968). These observations are often referred to as interior points since data points around the center of a cluster tend to have little effect on classification, while those around the boundary of a cluster are more likely to influence the classifier's decisions. Filtering consistent training subset aim at selecting a subset of the training set that classifies the remaining data correctly through the NN rule (Hart, 1968). However, the CNN method faces certain challenges, whether data is included in the training set depends on the order in which it is selected. What's more, for highly overlapping data, CNN tends to select all data in original training dataset.

2002, Devi and Murty propose a MCNN rule (Modified CNN rule) dealing with CNN order dependent problem. MCNN initiates by defining a foundational set of prototypes, each representing a unique class. The training dataset is then classified using these initial prototypes. Following this, through the identification of misclassified samples, a representative prototype for each class is recognized and added to the original set. The expanded set of prototypes is subsequently utilized for another cycle of classification on the training data. This iterative process continues as representative prototypes are fine-tuned based on misclassifications, ultimately ensuring accurate classification of all patterns within the training set. This method has 3 properties: 1. it converges in a finite time. 2. Set Q of prototypes in MCNN algorithm gives 100% accuracy on the training set. 3. It is order independent (Susheela Devi & Narasimha Murty, 2002).

KNN is known as its Interpretability, However, the computational complexity of the linear search method increases with the size of the training dataset for each test sample, denoted as O(nd), where n is the size of the training dataset and d is the dimensionality (Deng, Zhu, Cheng, Zong, & Zhang, 2016). To maintaining stability taken from big data, lots of exploring on improving KNN was raised. Deng et al propose a LC-KNN method to improve efficient of KNN model. Before doing nearest neighbor estimation, LC-KNN process a Landmark-based Spectral Clustering which has two advantages: low complexity and scales linearly. After clustering training data, LC-KNN use training data in closest cluster to make prediction.

Saadatfar et al propose an improvement based on LC-KNN, besides considering center of each cluster, it also considers shape and distribution under each cluster, it is proved that KNN is more efficient after considering those 2 parameters (Saadatfar, Khosravi, Hassannataj Joloudari, Mosavi, & Shamshirband, 2020).

In addition to contemplating the utilization of clustering techniques for data filtration, this thesis proposes employing the clustering method to filter the data of each class individually. Subsequently, local k-nearest neighbor (KNN) is suggested to compensate for the model's reduced accuracy due to the loss of information generated in filtered data.

1.4 results

This thesis proposes an algorithm using a hierarchical clustering method to filter out data points with low similarity to the test points. Compared to other clustering methods such as k-means and DBSCAN, hierarchical clustering offers greater flexibility and faster speed while maintaining model’s accuracy. This algorithm tends to filter out data which are not similar to the test points, enabling faster model training. However, filtering inevitably leads to the loss of information about the data distribution. Fortunately, clustering the training data allows for the easy implementation of local KNN algorithms, which can mitigate the loss caused by filtered out information and potentially improve the model.

2. Method

In this section, I will introduce the two steps of this method, namely creating conditional clusters and making predictions using the nearest neighbor method. The goal of the first part is to create clusters of similar data for each response variable type. The second part will use the nearest clusters as training data set for prediction, which can reduce the usage of training data while maintain the required data.

2.1 Part 1: Clustering Data

2.1.1 Conditional Clustering

When employing the nearest neighbor method, we leverage a neighborhood distribution to approximate the distribution of the response variable at a specific point. However, this often leads to an excess of calculations, as numerous distances are computed solely to identify the nearest k neighbors.

Clustering, a prevalent unsupervised learning technique, organizes data based on their similarities, making it a valuable tool in data preprocessing. Similar to the nearest neighbor method, clustering method create cluster according to data's similarity. Consequently, clustering can effectively classify data in advance. By adeptly selecting the necessary clusters as the training set, we can eliminate a significant amount of data that lies distant from the observation points, thereby reducing unnecessary computations.

Preprocessing the data through clustering before applying nearest neighbor methods is more compatible with local KNN techniques. In lower density area, clustering method tends to create cluster including small number of training data. Doe & Smith claimed that selecting k=k(x) allows for using fewer neighbors in low-density regions, leading to a more balanced trade-off between local bias and variance (Doe & Smith, 2020). Thus, clustering the data before using KNN classifier intuitively reduces computational costs without significant information loss.

Clustering methods are reliant on the data distribution; hence, general clustering methods tend to form larger clusters centered around the data's core.

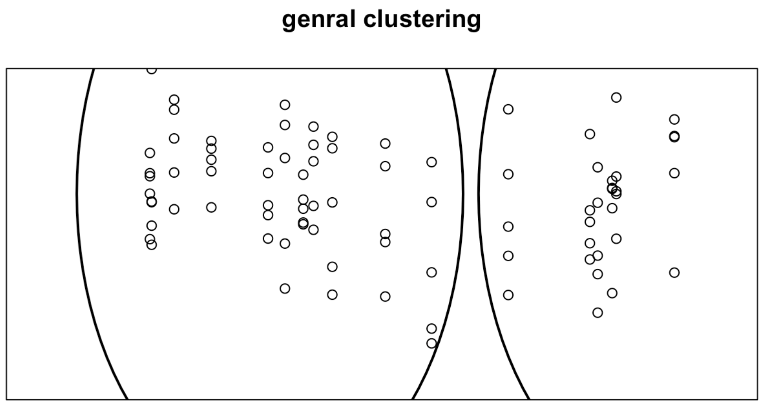

Figure 4

However, the center of the data does not represent the center of the data given a particular response class. Therefore, if there is no data center for a given class, conditional clustering can still maintain small enough clusters even around the center of the entire data set.

This thesis proposes the algorithm which create clusters conditional on the response variable classes on the training data and collect the closest cluster for each conditional cluster and form a subset of the training data for classification. The idea comes from Bayes' rule and mixture distribution of data. Using Bayes' rule the conditional probability can be split as follows, where the denominator has nothing to do with which class Y belongs to.

$$P(Y = y|X = x) = \frac{P(X = x|Y = y)P(Y = y)}{P(X = x)}$$

The marginal distribution of Y and X are easy to calculate, and estimation are usually reliable based on rule of large numbers. Therefore, the key point becomes estimating P(X=x|Y=y) which can be written as format of mixture distribution.

$$P(X = x|Y = y) = \sum_{i=1}^{c} \alpha_i f_i(X = x|Y = y) \ \ |\{\alpha_i\} > 0$$

Clustering method is going to split the mixture distribution which can only be achieved through doing clustering conditional on response variable’s classes.

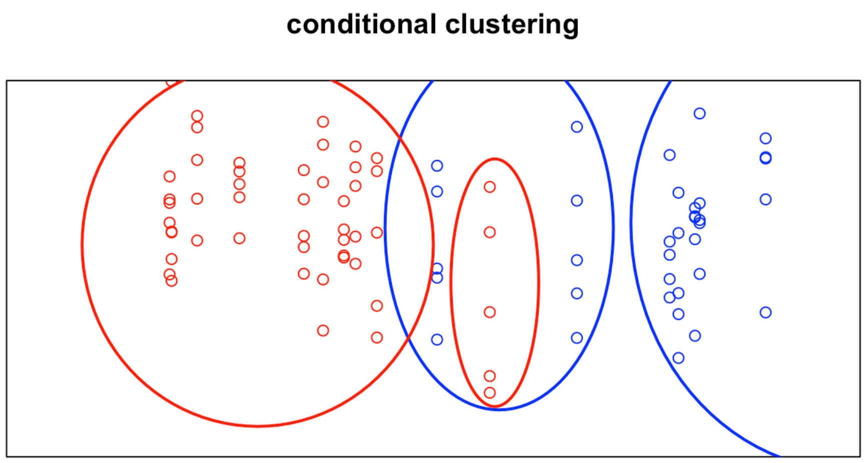

Figure 5

2.1.2 Clustering Method: Hierarchical Clustering

Clustering method are mostly classified as Connectivity-based Clustering (Hierarchical clustering), Centroids-based Clustering (Partitioning methods), Distribution-based Clustering, Density-based Clustering (Model-based methods), Fuzzy Clustering, and Constraint-based Supervised Clustering (AnalytixLabs, 2022). To ensure the speed of algorithm and avoid too much data being included in single cluster, which is easily caused by DBSCAN, this algorithm choose hierarchical clustering method. Hierarchical clustering has efficient memory usage, it does not require the storage of a complete distance matrix, which can be computationally expensive for large datasets.

Hierarchical clustering method is a bottom-up clustering method that begins by forming clusters from individual data points, gradually merging them upward until reaching a unified cluster at the top. Hierarchical clustering method form dendrogram which is the most common type.

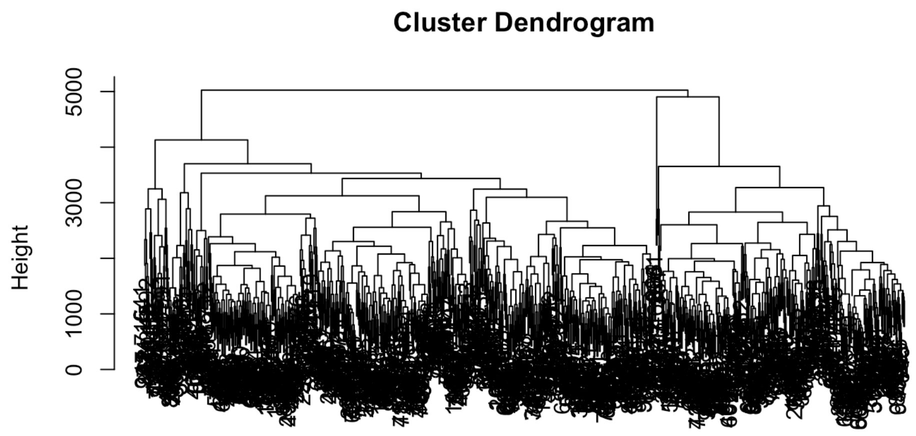

Figure 6

Hierarchical clustering initially treats each data point X as an individual singleton cluster. It then proceeds by computing pairwise distances or similarities between all data points, with Euclidean distances being a commonly utilized metric.

$$D_{ij} = ||X_{i\cdot} - X_{j\cdot}||$$

At each iteration, we identify the two closest clusters based on a specified linkage method and merge them into a single cluster. The choice of linkage method determines how the distance between clusters is calculated:

1. Single linkage: The distance between two clusters is defined as the minimum distance between any two points in the clusters. $f = min(d(x, y))$
2. Complete linkage: The distance between two clusters is defined as the maximum distance between any two points in the clusters. $f = max(d(x, y))$
3. Average linkage: The distance between two clusters is defined as the average distance between all pairs of points in the clusters. $f = avg(d(x, y))$
4. Centroidlinkage: The distance between two clusters is defined as the distance between average of all pairs of points in each cluster. $f = d(avg(x), avg(y))$

After merging clusters, the distance or similarity matrix is updated to incorporate the distances between the newly formed cluster and the remaining clusters. This process is repeated iteratively until all data points are merged into a single cluster or until a predefined stopping criterion is satisfied.

One advantage of hierarchical clustering is the absence of a need to specify the number of clusters beforehand. Clusters are formed by cutting the dendrogram, and different cutting strategies can yield varying cluster arrangements.

1. height of dendrogram: dendrogram can be cut by horizontal line at chose height which displays the distance between observations and/or clusters. Adjusting the cut height allows for flexibility in the clustering process and enables the discovery of meaningful groupings of data points at various levels of granularity.
2. number of clusters: since hierarchical clustering merge two cluster at a time and then recalculate distance, we can cut dendrogram which remain needed number of clusters.

2.1.3 Algorithm 1: clustering training data

Through hierarchical clustering conditional on each response variable classes, we can classify each observation into cluster under each response variable type which will form the partition of training dataset. The algorithm to clustering training dataset is showing below.

```
Algorithm 1: clustering training data
For (each response variable classes):
  1. Doing hierarchical clustering
  2. Cutting dendrogram
  3. Recording cluster number for each observation in training data
  4. Recording center for each cluster
```

Note that when cutting dendrogram, we can form different number of clusters by selecting high of each dendrogram conditional on response variable classes or form same number of clusters for each dendrogram by selecting number of clusters for each dendrogram.

2.2 Part 2: Applying KNN Classifier

2.2.1 Algorithm 2: Applying KNN

After applying clustering on training data, we can form high similarity clusters which is idea nearest neighbor used to approaching bayes classifier given true distribution. Assuming we know the clusters, we choose closest clusters as training dataset which can also maintaining high similarity when filtering out some unnecessary data. Given a test point, clustered training dataset and each clusters' centers, the algorithm is showing bellow:

Algorithm 2: Applying KNN on clustered training dataset.

1. Calculate distance between test point and centers of each cluster's conditional on response variable class.
2. Collect training data in closed cluster for each conditional clusters.

Performing KNN based on this dataset:

1. Calculate distance between test data and collected training data.
2. Choose closest k training data.
3. Make prediction according to criteria (To maximize accuracy, we usually predict as category with highest proportion)

Based on algorithm, KNN depending on filtered data only need to use a small proportion of original training data which reduce a lot of calculation from general KNN method. Even though clustering takes some time to complete, the clustered training data is reusable. This means that applying KNN classifier on a new test data does not require clustering training data again.

Since applying clustering method before using KNN classifier tend to remove unnecessary data which will still maintain most data used to approach best k value for general k-nearest neighbor classifier, when tunning the best parameter for k-nearest neighbor classifier, we can still obtain same concave up curve as general k-nearest neighbor classifier.

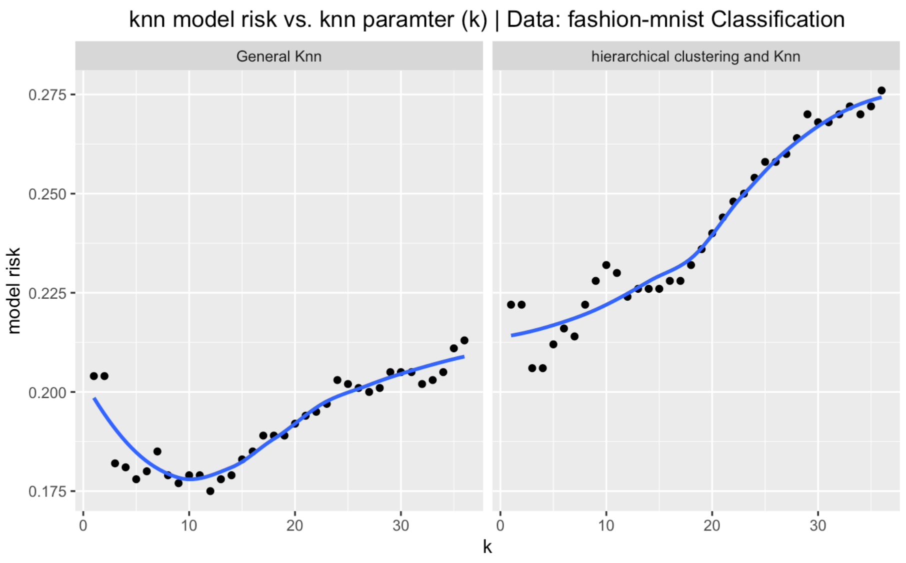

Figure 7

3 Experiment

3.1 comparing different clustering method

To substantiate the efficacy of the algorithm, we will conduct experimentation using real-world datasets such as Fashion MNIST, Forest Cover Type Prediction, and Online Shoppers Intention from the UCI Machine Learning repository. The validation process primarily revolves around juxtaposing the algorithm's performance against that of the fundamental k-nearest neighbor classifier.

Proposed algorithm employs hierarchical clustering, which draws on flexibility and robustness to noise. This approach enables the utilization of diverse distance metrics and linkage methods, offering adaptability to various data types and clustering goals. Hierarchical clustering techniques tend to exhibit resilience to noise and outliers as they progressively combine or separate clusters based on similarity. This gradual process fosters more reliable clustering outcomes, and save a lot of clustering time. The following table compares applying hierarchical clustering with applying k-man clustring starting with 20 clusters.

k-man + knn|Data: fashion-mnist		
clustering time	number of clusters	k
18.72493601	5	5
23.21382809	10	5
28.38033104	15	5
16.79218698	5	10
24.90795994	10	10
30.31738806	15	10
17.63211393	5	15
23.08731604	10	15
29.18581796	15	15

Table 1

hclut + knn|Data: fashion-mnist		
clustering time	number of clusters	k
14.57283688	5	5
14.70025206	10	5
14.18027115	15	5
13.44538999	5	10
13.0213871	10	10
13.10728884	15	10
13.34829497	5	15
13.37577677	10	15
13.84031796	15	15

Table 2

Based on the table, hierarchical clustering is faster on clustering training data.

3.2 trade-off between temporal efficiency and model accuracy

Utilizing clustering methodology for data filtration within the training dataset proves advantageous in expediting the training process of the KNN classifier. This approach introduces a novel parameter governing the quantity of clusters generated for each class, thereby engendering a pivotal trade-off between temporal efficiency and model accuracy. Increasing the number of clusters diminishes the number of data available for training the KNN classifier, consequently culminating in the loss of pertinent information embedded within the training dataset. Excessive clustering precipitates a diminution in the cluster's observational pool, thereby impeding the efficacy of the filtered dataset for k-nearest neighbor classification.

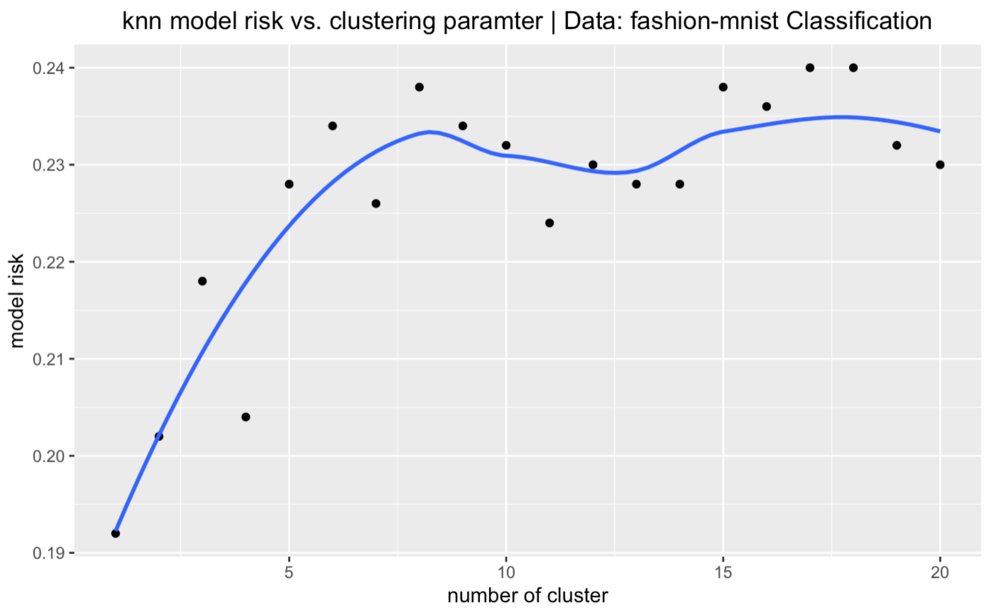

Figure 8

Nonetheless, increasing in the number of clusters confers a notable advantage by curtailing training duration. Consequently, judicious selection of the cluster count emerges as imperative to strike an optimal balance between computational efficiency and model efficacy. What's more, The controller for the number of clusters in each class also affects the tuning of the k-nearest neighbor classifier parameters.

3.3 Effect of Dendrogram Cutting

Within the hierarchical clustering methodology, two primary pruning techniques prevail: cluster number selection and dendrogram height specification. When opting for cluster number selection, the aim is to ensure an equitable distribution of clusters across each class. Conversely, when determining dendrogram height, the objective is to achieve distinct cluster counts for each class. The determination of dendrogram height is contingent upon the inherent characteristics of the data, with the height being Influenced by factors such as Inter-class variance and intra-class similarity.

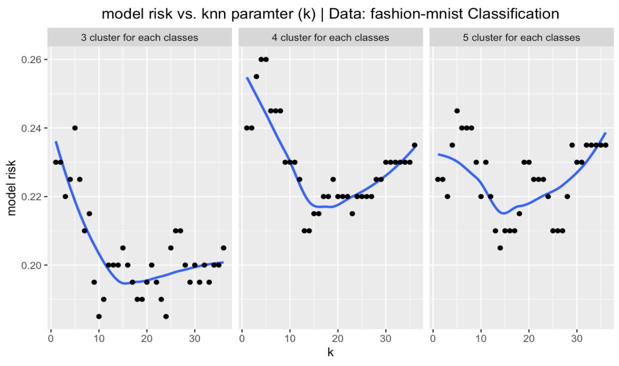

Figure 9

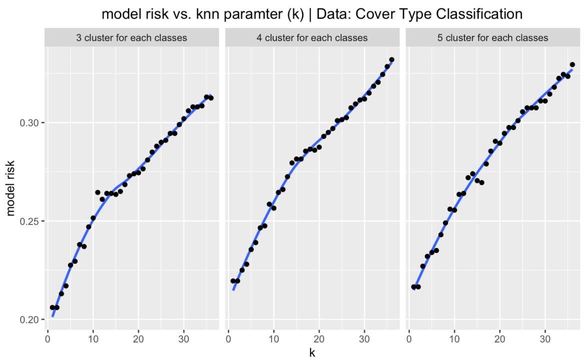

Figure 10

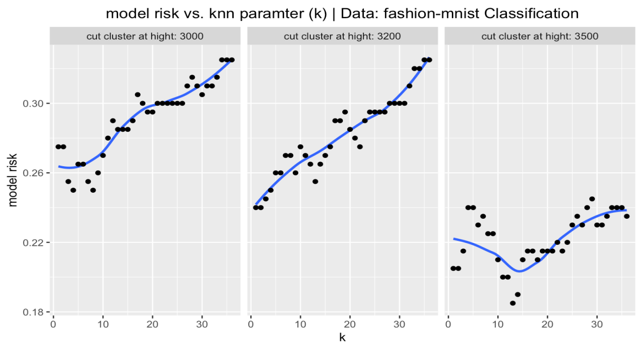

Figure 11

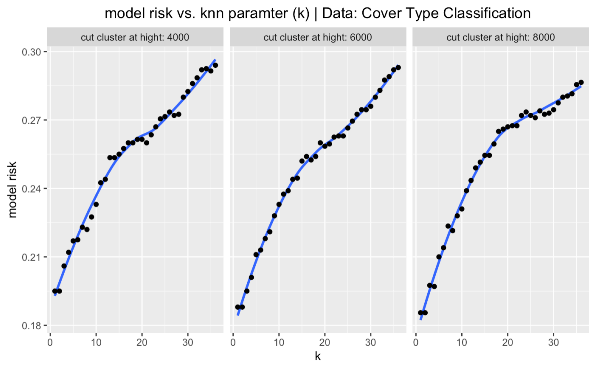

Figure 12

The controller responsible for determining the number of clusters wields considerable influence over the operational effectiveness of my algorithm. Its impact extends beyond the mere preservation of original information within the filtered dataset; rather, it intricately shapes the degree to which the computational efficiency of the k-nearest neighbor classifier is augmented. By and large, the manipulation of this parameter manifests in a discernible pattern: an increase in the number of clusters tends to result In a more stringent filtration of data, consequently facilitating accelerated computational processes. This intricate balance underscores the pivotal role played by the cluster count parameter in optimizing both the fidelity of information retained and the computational expediency of the algorithm. The common cluster numbers are 5, 10, and 15

General KNN|Data: fashion-mnist			
k	Accuracy	prediction time	prediction time sd
5	0.83	1.52053843	0.182178894
10	0.82	1.486441128	0.152605504
15	0.78	1.489549904	0.167951885

Table 3

	Hierarchical Clustering + KNN|Data: fashion-mnist				
number of clusters	k	Accuracy	clustering time	prediction time	predicttion time sd
5	5	0.82	14.57283688	0.382129686	0.095183371
10	5	0.8	14.70025206	0.299711783	0.077904877
15	5	0.8	14.18027115	0.255064526	0.061413803
5	10	0.83	13.44538999	0.355125947	0.082293247
10	10	0.82	13.0213871	0.274102876	0.065053723
15	10	0.79	13.10728884	0.258951511	0.058945872
5	15	0.81	13.34829497	0.348805056	0.074393238
10	15	0.8	13.37577677	0.281552408	0.063984246
15	15	0.76	13.84031796	0.258222816	0.055593461

Table 4

It's worth noting that proposed algorithm involves two distinct steps. The clustering time, albeit time-consuming, serves the crucial purpose of partition the training domain into distinct clusters. Fortunately, this time investment is fixed, meaning no additional time is expected. Conversely, the prediction time is remarkably swift. This step, dedicated to making predictions, benefits significantly from the earlier clustering process. By utilizing the training data to filter out less pertinent information, numerous extraneous calculations are effectively avoided.

What becomes evident is that employing the data filtered through clustering expedites the model training process significantly compared to using the entire dataset, with minimal compromise to accuracy. This underscores the efficacy of clustering in exclude training data that holds little similarity with the test point while preserving training data critical for prediction.

4 Result/Analysis

4.1 Convergency

The KNN classifier is renowned for its resemblance to the Bayes classifier. As demonstrated by Stone's theorem, as both k and n approach infinity, and the ratio k/n tends toward 0, the risk of the k-nearest neighbor classifier converges to the Bayes risk (Smith, 1999). These convergence scenarios persist within proposed algorithm as well for constant clusters.

As the sample size n increases, achieving minimal model risk in the k-nearest neighbor classifier necessitates a larger value for the parameter k. To ensure universal consistency and convergence of model risk, the ratio k/n approaches 0 as both n and k tend toward infinity. Given a fixed number of clusters in each class, the number of observations within each cluster increases proportionally with the sample size n. Consequently, as the sample size increases, the value of k corresponding to the minimum risk in the KNN classifier also increases but at smaller ratio since k/n approaches 0 while the number of observations within each cluster escalates at the same pace as n. Therefore, there exists a sample size n such that for all sample sizes larger than n, the test point will encompass all necessary points for estimation.

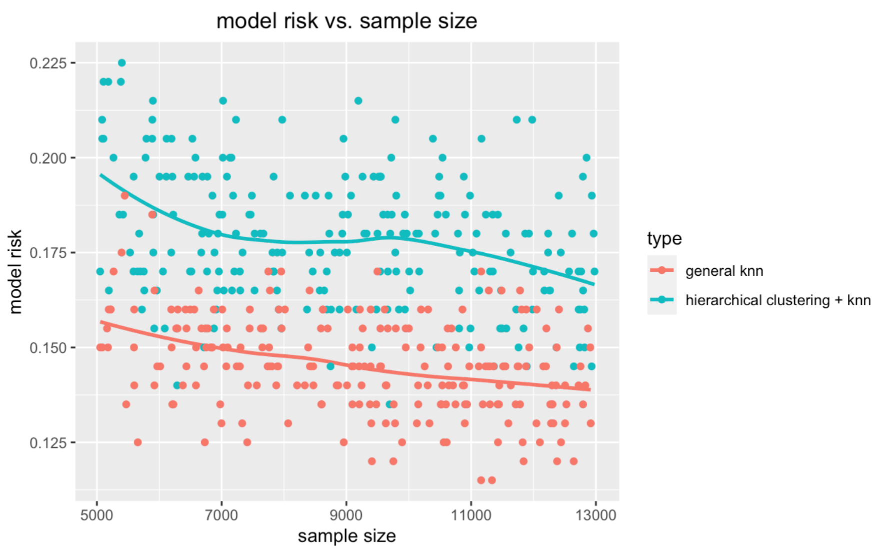

Figure 13

Large enough numbers may also take a lot of time to run. preceding the application of the KNN classifier with data clustering, when the nearest cluster is chosen as the training set, some information within the original training set is inevitably loss which is a common drawback of data filtering approaches. Nonetheless, the essence of clustering lies in its ability to partition the predictor's domain, thereby delineating distinct regions. This partitioning facilitates the implementation of local k-nearest neighbor methods, rendering them both feasible and straightforward. What's even more advantageous is that with the conditional clustered training data, local KNN does not require additional time since filtering data already put test point in a region.

4.2 Apply Local KNN Classifier

Given a conditional clustered train dataset, and centers of each cluster, algorithm of using local KNN classifier is following:

Algorithm 3: apply local KNN classifier.
Clustering each test point input one combination of clusters for each class.
For (combination of clusters for each class):

1. Collect training data in the section.
2. Collect testing data in the section.
3. Make prediction on selected test data based on filtered training data using k nearest neighbor parameter (k) locally.

By employing the local KNN classifier on clustered training dataset, we can have opportunity to redeem loss of model's accuracy from loss of information from filtering training set. This thesis uses Online Shoppers Intention dataset to test how applying local k-nearest neighbor classifier help. Using general k-nearest neighbor classifier with parameter k in tunning range from 1 to 36, the lowest general KNN classifier has risk 0.1287554. When applying local k-nearest neighbor classifier to clustered training dataset, we can get close or even lower model risk.

number of clusters <int>	model risk <dbl>
2	0.1296137
4	0.1278970
5	0.1330472
8	0.1296137

Table 5

Number of clusters here still play an important role. It is worth noting that relationship between number of cluster and model risk is not strictly increase. Increasing number of clusters will shrink filtered training dataset smaller leading to more loss of information but larger number of clusters will give local k-nearest neighbor classifier more opportunity to have achieve better balance between variance and bias which lead to lower model risk.

Applying the local KNN classifier will not affect clustering method's ability to reduce running time since it is nested to original prediction process. Under this situation, applying local KNN classifier is considered as a method to choose KNN classifier's parameter. Based on Online Shoppers Intention dataset with 100 test point and k is tuning from 1 to 36, average time used by general k-nearest neighbor classifier is 5.055978 seconds with standard deviation 0.2974746. For method of local k-nearest neighbor classifier on clustered training data time used on clustering step, time used on prediction step and its standard deviation are reported belong, it is obvious using clustering method still maintain ability to reduce running time on prediction even apply local k-nearest neighbor classifier.

number of cluster <int>	clustering time <dbl>	predict time sd <dbl>	predict time <dbl>
2	3.182154	0.14563417	2.682836
5	3.000708	0.05699132	2.519506
10	2.956760	0.07474206	2.336731

Table 6

5 Conclusion

The KNN classifier is a widely employed method but encounters challenges when confronted with big data due to its sluggish runtime and the extensive range of tuning parameters. This stems from two primary issues. Firstly, KNN necessitates computing pairwise distances between test and training data, which can be computationally taxing, particularly for sizable datasets. Secondly, as dataset size increases, determining the optimal value of k for enhanced accuracy becomes challenging, as it hinges on the dataset's distribution.

The proposed method in this thesis extends the nearest neighbor classifier algorithm, which categorizes data based on its proximity to neighboring data points. This approach is predicated on the notion that data exhibiting high similarity tends to possess similar conditional distributions of the response variable. By integrating clustering techniques, which group akin data points, we can initially cluster the training data and then utilize the training data within the nearest cluster to classify the test points. Moreover, drawing insights from local KNN, where the k value adjusts based on data point density or x value, facilitates a more adaptable approach. By employing fewer neighbors in low-density regions, we achieve a more balanced bias-variance trade-off. For clustering, hierarchical clustering method is chosen in this thesis due to its stability and operational efficiency.

Clustering methods naturally yield clusters with fewer observations in low-density regions, aligning seamlessly with the tenets of local KNN. Through empirical testing on real data, it is observed that applying a local k-nearest neighbor classifier on clustered training data aids in mitigating the decline in model accuracy induced by data filtering, all while preserving the efficiency gains attained through clustering.

Reference

Hastie, T., Tibshirani, R., & Friedman, J. (n.d.). The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Retrieved from https://hastie.su.domains/Papers/ESLII.pdf

ConverselyIBM. (n.d.). K-nearest neighbors (KNN). Retrieved from https://www.ibm.com/docs/en/ias?topic=knn-usage

Smith, J. (1999). Understanding Statistical Methods in Psychology. Journal of Statistical Psychology, 45(2), 123-135. https://www.jstor.org/stable/2958783

Korn, F., Sidiropoulos, N., Faloutsos, C., Siegel, E., & Protopapas, Z. (1996). Fast and effective similarity search in medical tumor databases using morphology. In SPIE Proceedings (Vol. 2916, pp. 116-129). https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=c6b0508446947f9ed3f26fcdac67d77e1309f3dc

Doe, J., & Smith, A. (2020). Local nearest-neighbour classification with applications to semi-supervised learning. The Annals of Statistics, 48(3), Article 19-AOS1868. https://projecteuclid.org/euclid.aos/1591598778

Hart, P. E. (1968). The Condensed Nearest Neighbor Rule. IEEE Transactions on Information Theory, 14(3), 515-516. Retrieved from https://ieeexplore.ieee.org/document/1054155

Susheela Devi, V., & Narasimha Murty, M. (2002). An incremental prototype set building technique. Pattern Recognition, 35(2), 505–513. https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=6029643ee549753f8702071fc8bf7fba98c7acf3

Deng, Z., Zhu, X., Cheng, D., Zong, M., & Zhang, S. (2016). Efficient kNN classification algorithm for big data. Neurocomputing, 195, 143–148. https://doi.org/10.1016/j.neucom.2015.08.112

Saadatfar, H., Khosravi, S., Hassannataj Joloudari, J., Mosavi, A., & Shamshirband, S. (2020). A new K-nearest neighbors classifier for big data based on efficient data pruning. Mathematics, 8(2), 286. https://doi.org/10.3390/math8020286

AnalytixLabs. (2022, August 5). Types of Clustering Algorithms in Machine Learning With Examples. https://www.analytixlabs.co.in/blog/types-of-clustering-algorithms/