Bankruptcy Clustering

Zeedan Shaikh

2024-09-06

Introduction

Clustering banks based on bankruptcy risk is a crucial analytical approach for understanding and managing financial stability within the banking sector. By grouping banks with similar risk profiles, stakeholders can better assess potential vulnerabilities, allocate resources efficiently, and implement targeted strategies. This method provides a structured way to address the complexities of financial risk and improve overall oversight and intervention efforts.

motivation

- **Risk Assessment and Management:**
  - Identifies banks with similar bankruptcy risks for targeted risk management.
  - Facilitates the development of specific strategies for managing financial distress.
- **Resource Allocation:**
  - Enables efficient allocation of monitoring and support resources to high-risk banks.
  - Prioritizes intervention efforts based on cluster risk levels.
- **Regulatory Oversight:**
  - Helps regulators design tailored policies and regulations for different risk clusters.
  - Supports focused oversight and preventive measures.
- **Predictive Analysis:**
  - Enhances predictive models by analyzing groups of banks with comparable financial profiles.
  - Improves the forecasting of potential financial crises and systemic risks.
- **Strategic Planning:**
  - Assists banks in understanding their position relative to peers within the same risk cluster.
  - Aids in formulating strategies for risk reduction and financial stability.
- **Investment Decisions:**
  - Guides investors in evaluating the risk profiles of banks and making informed investment choices.
  - Highlights potential investment risks and opportunities based on cluster analysis.
- **Crisis Management:**
  - Improves crisis preparedness by identifying high-risk clusters requiring immediate attention.
  - Supports the development of tailored contingency plans for different risk levels.
- **Performance Benchmarking:**
  - Facilitates comparison and benchmarking of banks within similar risk clusters.
  - Helps assess performance metrics and identify best practices among peers.
- **Data-Driven Insights:**
  - Provides a structured approach to analyzing bankruptcy risks, leading to more informed decision-making.
  - Enhances understanding of systemic vulnerabilities and strengths.
- **Enhanced Communication:**
  - Clarifies risk discussions between banks, regulators, and stakeholders through a structured framework.
  - Improves transparency and rationale behind regulatory decisions and interventions.

Objective

Here our objective is to cluster the data points.

Methodology

Data collection

The data is obtained from kaggle Dataset name : **Company Bankruptcy Prediction**

▶ **Variable Details**

Data preparation

Getting the required packages:

```
pacman::p_load(MASS,clValid,cluster,dbscan,ggplot2)
```

Importing and cleaning the dataset:

```
mydata=read.csv("C:\\Users\\zeeda\\Downloads\\archive (3)\\data.csv")
```

- First check if the data is balanced or not i.e. whether the no of bankrupt and non-bankrupt banks are equal.

```
table(mydata$Bankrupt.)
```

```
   0    1 
6599  220 
```

Clearly the data is highly unbalanced as we have a lot of observation on banks that are not bankrupt and less on bankrupt. So we will use undersampling technique and reduce the cardinality of majority class (non-bankrupt class)

```
no_bankrupt=subset(mydata,mydata$Bankrupt.==0)
mydata_index=sample(1:nrow(no_bankrupt),201,F)
no_bankrupt_data=no_bankrupt[mydata_index,]
bankrupt_data=subset(mydata,mydata$Bankrupt.==1)
mydata=rbind(no_bankrupt_data,bankrupt_data)
table(mydata$Bankrupt.)
```

```
  0   1 
201 220 
```

- Now we have a balanced data.

▶ **Summary of the data**

```
dim(mydata)
```

```
[1] 421  96
```

- Clearly we have a really high number of predictor variable so we need to use derived inputs techniques (ex. Principle Component Analysis)

Data Reduction

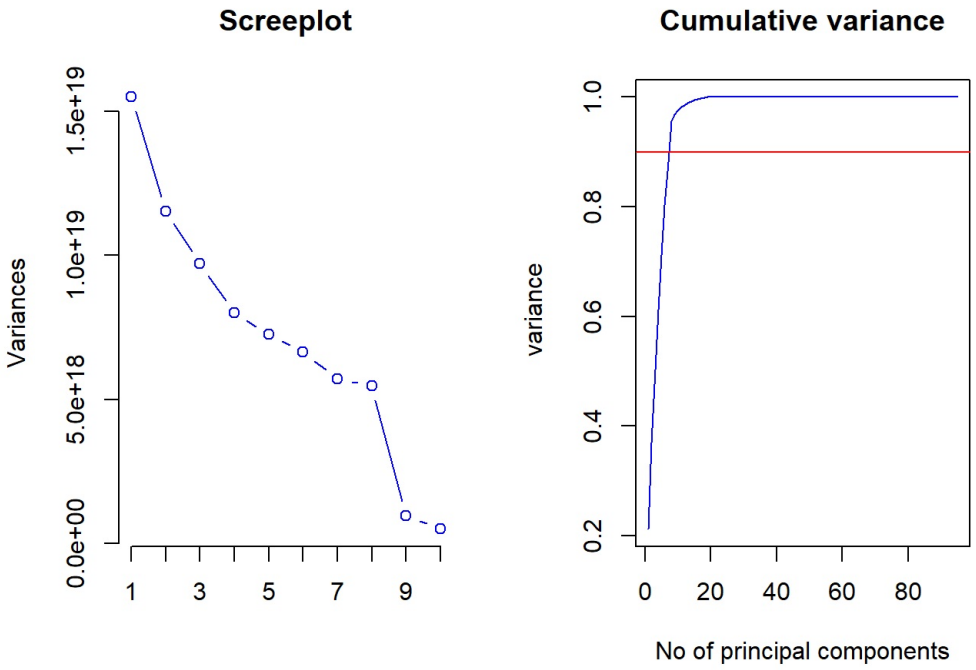
Principle Component Analysis

```
data=mydata[,-1]
pca_model=prcomp(data,center = TRUE)
```

▶ **Summary PCA model**

We will obtain Screeplot and Cumulative variance plot

```
par(mfrow=c(1,2))
plot(pca_model,typ="l",main= "Screeplot",col="blue")
variance_plot=cumsum((pca_model$sdev^2)/sum(pca_model$sdev^2))
plot(variance_plot,type="l",main="Cumulative variance",xlab="No of principal components",ylab=" variance",col="bl
ue")
abline(h=0.9,col="red")
```



- *7 Principle components capture 90 percent variance of the data*

Now we will reduce the dimension of the data using these 7 PC's as predictors

```
rotation_matrix=pca_model$rotation
reduced_rotation_matrix=rotation_matrix[,-c(8:95)]
final_data=as.matrix(data)%*%reduced_rotation_matrix
dim(final_data)
```

```
[1] 421   7
```

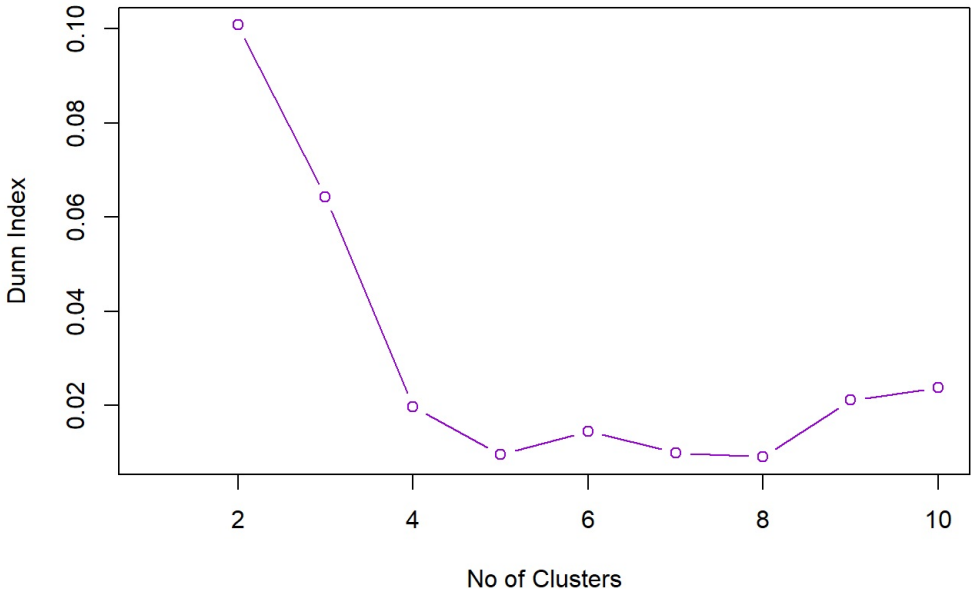
Now we will apply different Clustering Techniques

Clustering

K-means

- We will use kmeans to obtain different no of clusters.
- compute **Dunn Index** for each of the model and plot it.
- Choose the model with largest Dunn Index value and select the corresponding no of cluster.

```
set.seed(12)
dun_index=c()
for(i in 2:10){
  obj=kmeans(x=final_data,i)
  dun_index[i]=dunn(dist(final_data),as.vector(obj$cluster))
}
plot(dun_index,typ="b",xlab="No of Clusters",ylab="Dunn Index",col="darkviolet")
```



- *The Dunn Index is max when the cluster number is*

```
which.max(dun_index)
```

```
[1] 2
```

- Now we will check the accuracy how well the method is been able to classify the two data

```
#--optimal clustering
set.seed(12)
obj=kmeans(final_data,2)
cluster_vector=obj$cluster
#--class labels
class_vector=mydata$Bankrupt.
```

```
#--Accuracy
True_Positive=0
False_Negetive=0
True_Negetive=0
False_Positive=0
for(i in 1:nrow(final_data)){
  if(class_vector[i]==0){
    if(cluster_vector[i]==1)
      True_Positive=True_Positive+1
    else
      False_Negetive=False_Negetive+1
    }
  else{
    if(cluster_vector[i]==2)
      True_Negetive=True_Negetive+1
    else
      False_Positive=False_Positive+1
  }
}
Confusion_mat=matrix(c(True_Positive,False_Negetive,True_Negetive,False_Positive),nc=2)
Accuracy=(sum(diag(Confusion_mat))/sum(Confusion_mat))*100
Accuracy
```

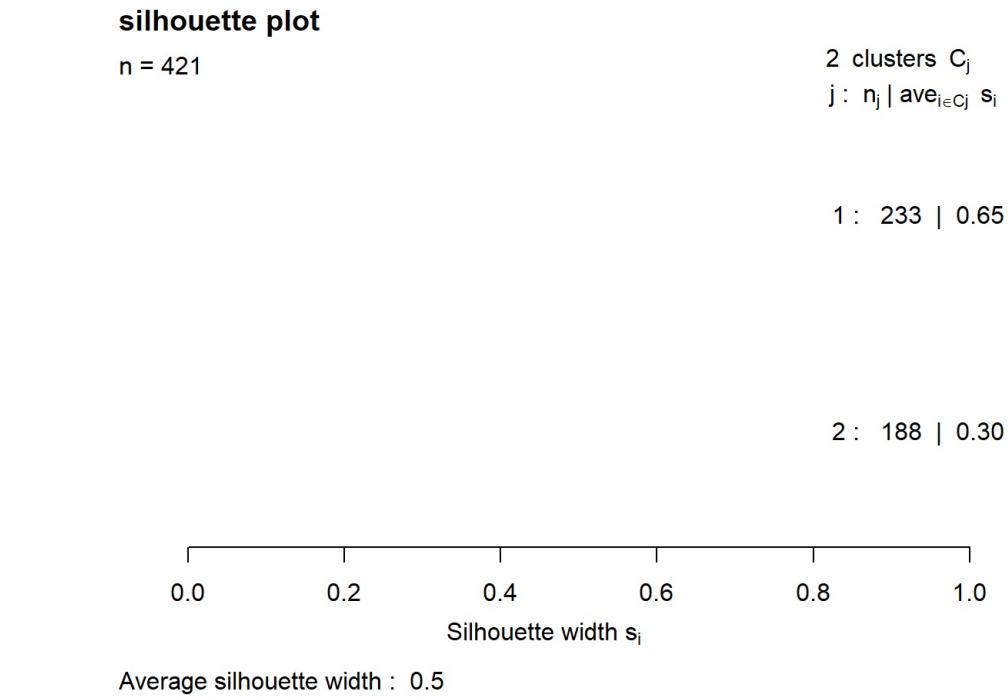
```
[1] 47.981
```

- Which is Quiet less.

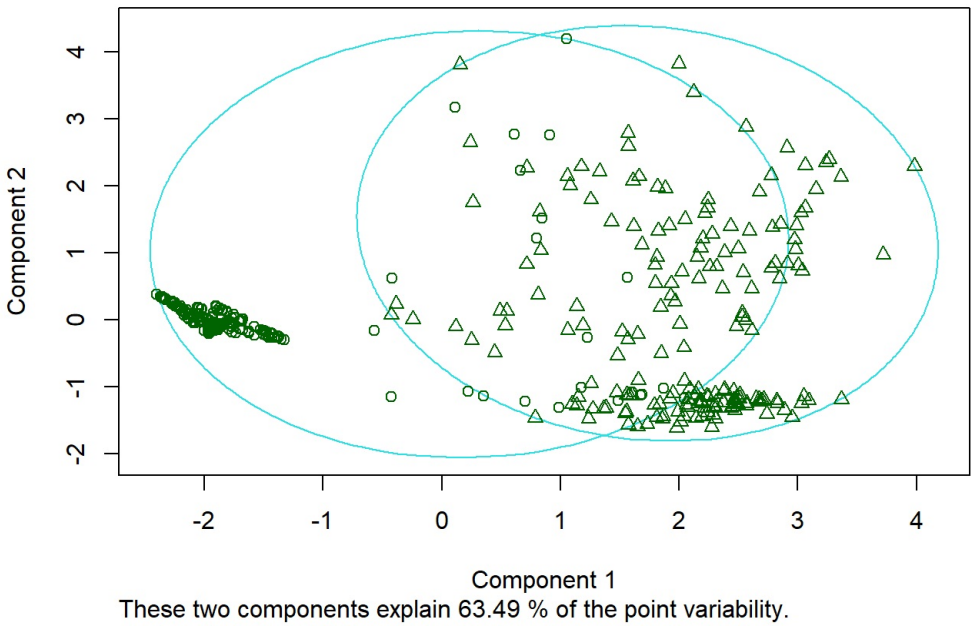
PAM (Partioning Around Medoids)

- We will use PAM to obtain different no of clusters.
- compute **Dunn Index** for each of the model and plot it.
- Choose the model with largest Dunn Index value and select the corresponding no of cluster.
- We will also use silhouette plot to judge the number of clusters.

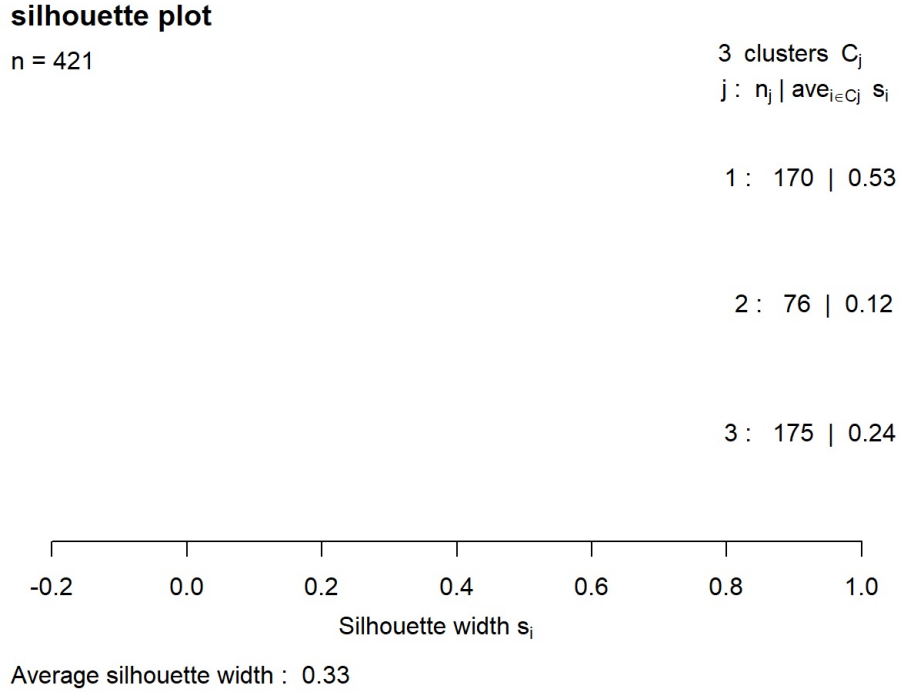
```
set.seed(12)
for(i in 2:8){
  obj2=pam(final_data,i)
  dun_index[i]=dunn(dist(final_data),obj2$clustering) }
obj2=pam(final_data,2)
plot(obj2,which=2,main="silhouette plot")
```



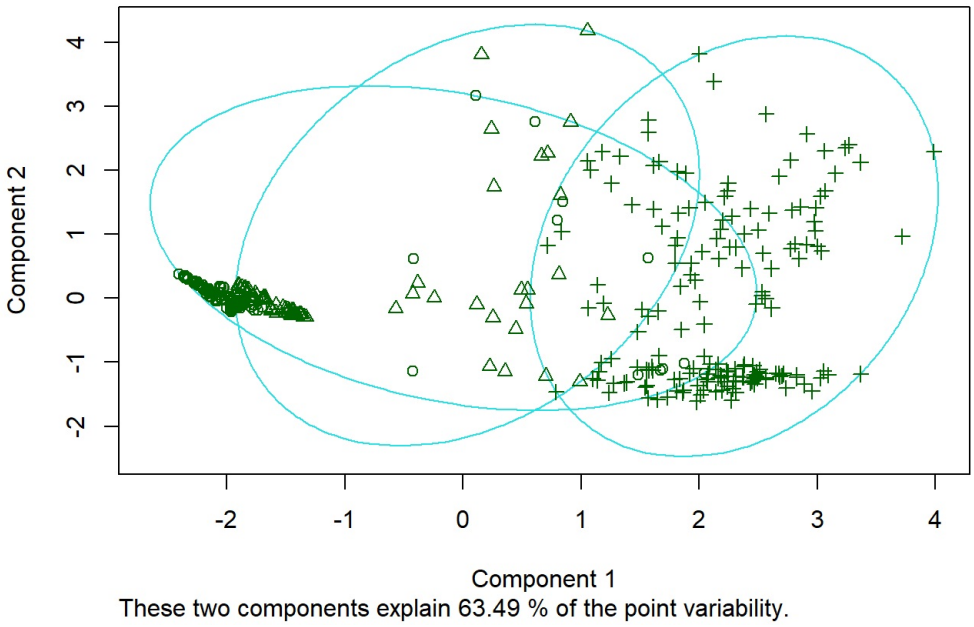

```
plot(obj2,which=1,main="")
```



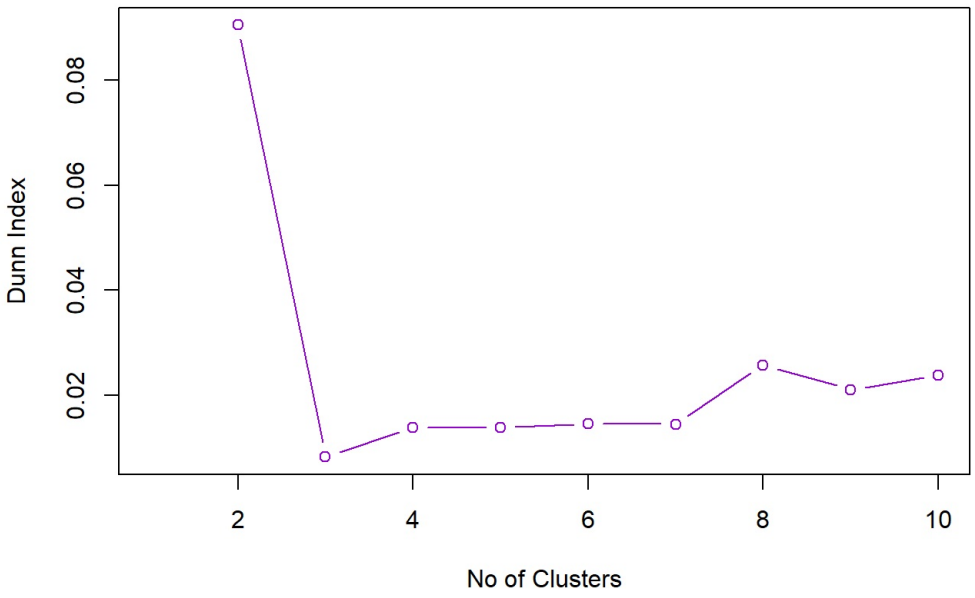

```
obj2=pam(final_data,3)
plot(obj2,which=2,main="silhouette plot")
```




```
plot(obj2,which=1,main="")
```




```
plot(dun_index,typ="b",xlab="No of Clusters",ylab="Dunn Index",col="darkviolet")
```



- From the silhouette plot we get the maximum value of the silhouette coefficient 0.5 when the number of clusters is 2.
- 0.5 means the clusters are moderately well separated
- *The Dunn Index is max when the cluster number is*

```
which.max(dun_index)
```

```
[1] 2
```

- Now we will check the accuracy how well the method is been able to classify the two data

```
#--optimal clustering
set.seed(12)
obj2=pam(final_data,2)
cluster_vector=obj2$clustering
```

```
#--Accuracy
True_Positive=0
False_Negetive=0
True_Negetive=0
False_Positive=0
for(i in 1:nrow(final_data)){
  if(class_vector[i]==0){
    if(cluster_vector[i]==1)
      True_Positive=True_Positive+1
    else
      False_Negetive=False_Negetive+1
    }
  else{
    if(cluster_vector[i]==2)
      True_Negetive=True_Negetive+1
    else
      False_Positive=False_Positive+1
  }
}
Confusion_mat=matrix(c(True_Positive,False_Negetive,True_Negetive,False_Positive),nc=2)
Accuracy=(sum(diag(Confusion_mat))/sum(Confusion_mat))*100
Accuracy
```

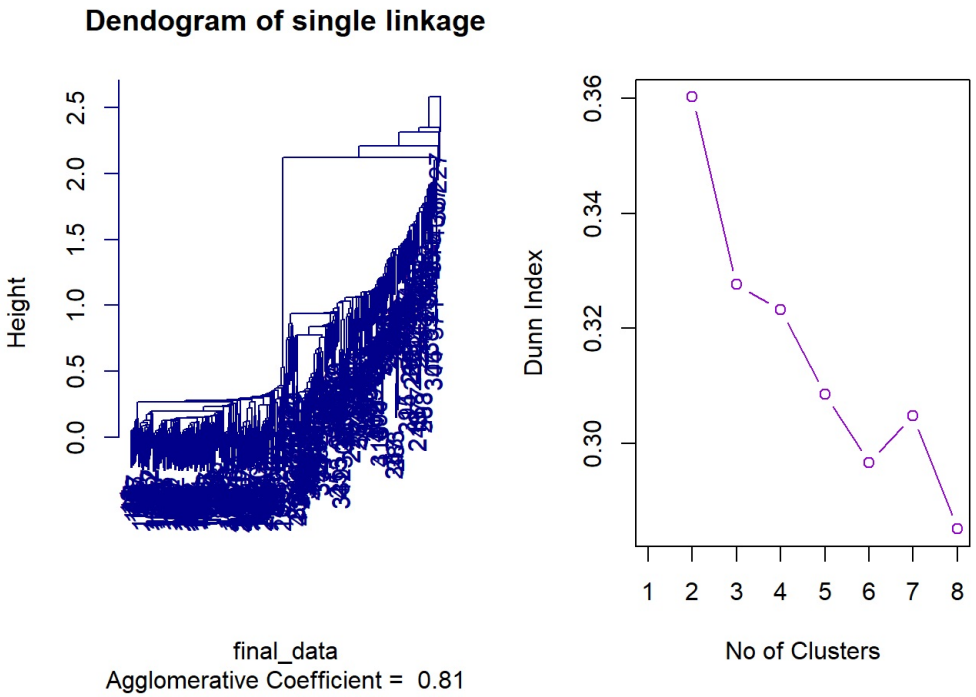
```
[1] 55.34442
```

- Which is still Quiet less.

AGNES (Single Linkage Algorithm)

- We will plot the dendogram using the above method.
- Cut the dendogram at different lengths to get different no of clusters.
- Then calculate Dunn Index based on it and get the optimal no of clusters.

```
par(mfrow=c(1,2))
obj3=agnes(final_data,method="single")
plot(obj3,which=2,main="Dendogram of single linkage",col="darkblue")
dun_index=c()
for(i in 2:8){
  clust=cutree(obj3,i)
  dun_index[i]=dunn(dist(final_data),clust) }
plot(dun_index,typ="b",xlab="No of Clusters",ylab="Dunn Index",col="darkviolet")
```



- The Agglomerative Coefficient 0.81 which means clusters are well formed.
- *The Dunn Index is max when the cluster number is*

```
which.max(dun_index)
```

```
[1] 2
```

- Now we will check the accuracy how well the method is been able to classify the two data

```
#--optimal clustering
obj3=agnes(final_data,method = "single")
clust=cutree(obj3,2)
cluster_vector=clust
```

```
#--Accuracy
True_Positive=0
False_Negetive=0
True_Negetive=0
False_Positive=0
for(i in 1:nrow(final_data)){
  if(class_vector[i]==0){
    if(cluster_vector[i]==1)
      True_Positive=True_Positive+1
    else
      False_Negetive=False_Negetive+1
    }
  else{
    if(cluster_vector[i]==2)
      True_Negetive=True_Negetive+1
    else
      False_Positive=False_Positive+1
  }
}
Confusion_mat=matrix(c(True_Positive,False_Negetive,True_Negetive,False_Positive),nc=2)
Accuracy=(sum(diag(Confusion_mat))/sum(Confusion_mat))*100
Accuracy
```

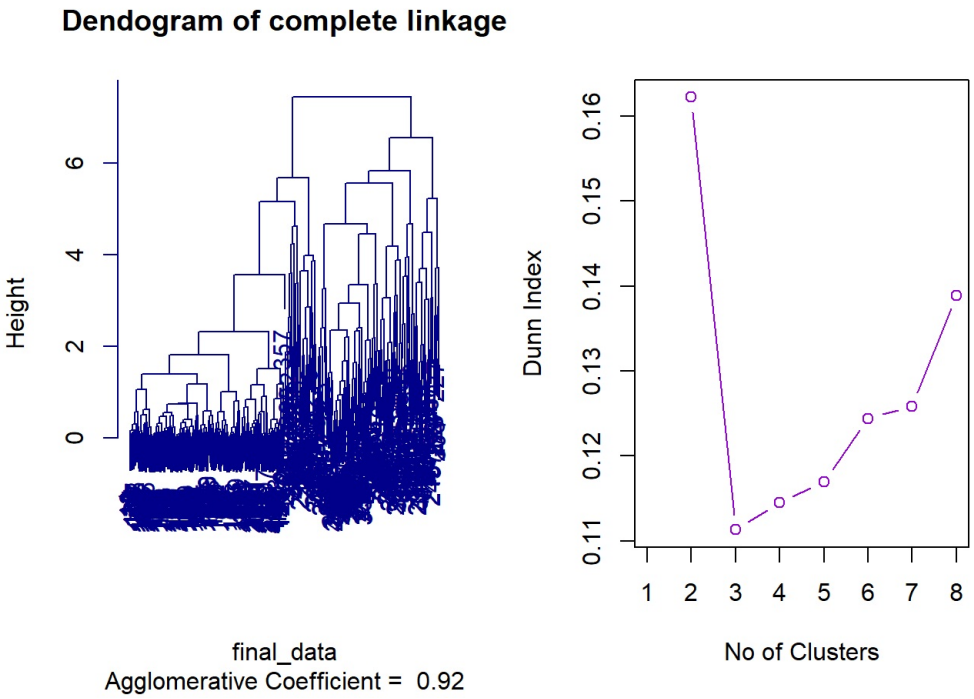
```
[1] 99.76247
```

- Which is very accurate.

AGNES (complete Linkage Algorithm)

- We will plot the dendogram using the above method.
- Cut the dendogram at different lengths to get different no of clusters.
- Then calculate Dunn Index based on it and get the optimal no of clusters.

```
par(mfrow=c(1,2))
obj3=agnes(final_data,method="complete")
plot(obj3,which=2,main="Dendogram of complete linkage",col="darkblue")
dun_index=c()
for(i in 2:8){
  clust=cutree(obj3,i)
  dun_index[i]=dunn(dist(final_data),clust) }
plot(dun_index,typ="b",xlab="No of Clusters",ylab="Dunn Index",col="darkviolet")
```



- The Agglomerative Coefficient 0.92 which means clusters are very well formed.
- *The Dunn Index is max when the cluster number is*

```
which.max(dun_index)
```

```
[1] 2
```

- Now we will check the accuracy how well the method is been able to classify the two data

```
#--optimal clustering
obj3=agnes(final_data,method = "complete")
clust=cutree(obj3,2)
cluster_vector=clust
```

```
#--Accuracy
True_Positive=0
False_Negetive=0
True_Negetive=0
False_Positive=0
for(i in 1:nrow(final_data)){
  if(class_vector[i]==0){
    if(cluster_vector[i]==1)
      True_Positive=True_Positive+1
    else
      False_Negetive=False_Negetive+1
    }
  else{
    if(cluster_vector[i]==2)
      True_Negetive=True_Negetive+1
    else
      False_Positive=False_Positive+1
  }
}
Confusion_mat=matrix(c(True_Positive,False_Negetive,True_Negetive,False_Positive),nc=2)
Accuracy=(sum(diag(Confusion_mat))/sum(Confusion_mat))*100
Accuracy
```

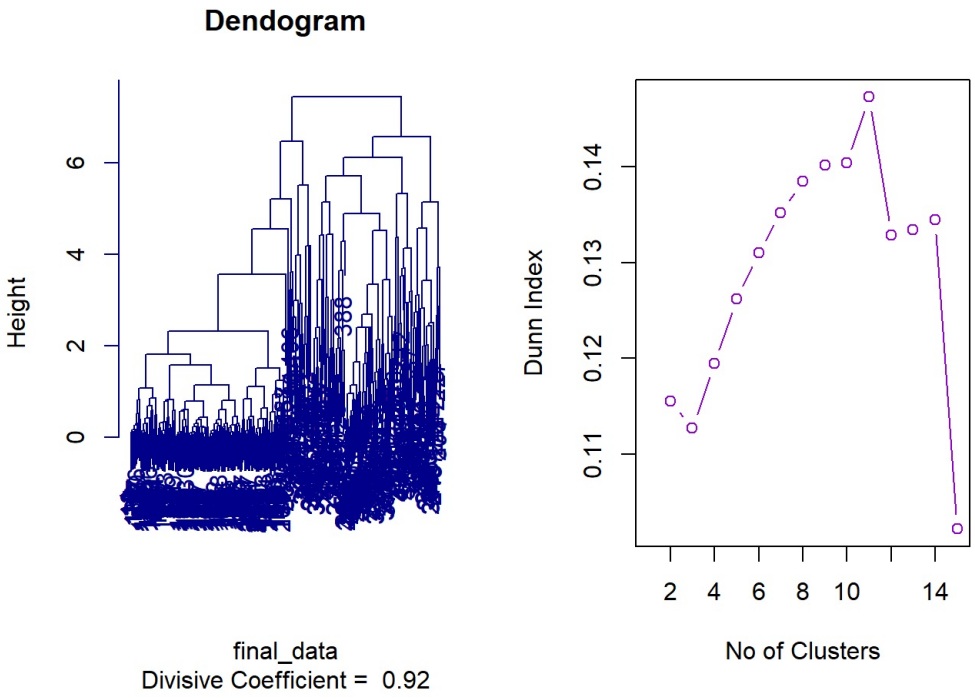
```
[1] 60.09501
```

- Which is moderate.

DIANA (Divisive Analysis)

- We will plot the dendogram using the above method.
- Cut the dendogram at different lengths to get different no of clusters.
- Then calculate Dunn Index based on it and get the optimal no of clusters.

```
par(mfrow=c(1,2))
obj5=diana(final_data)
plot(obj5,which=2,main="Dendogram",col="darkblue")
dun_index=c()
for(i in 2:15){
  clust=cutree(obj5,i)
  dun_index[i]=dunn(dist(final_data),clust) }
plot(dun_index,typ="b",xlab="No of Clusters",ylab="Dunn Index",col="darkviolet")
```



- The divisive Coefficient 0.92 which means clusters are very well formed.
- *The Dunn Index is max when the cluster number is*

```
which.max(dun_index)
```

```
[1] 11
```

- Which is quiet different from the rest of the results obtained from different methods.
- Now we will check the accuracy how well the method is been able to classify the two data

```
#--optimal clustering
obj5=diana(final_data)
clust=cutree(obj3,2)
cluster_vector=clust
```

```
#--Accuracy
True_Positive=0
False_Negetive=0
True_Negetive=0
False_Positive=0
for(i in 1:nrow(final_data)){
  if(class_vector[i]==0){
    if(cluster_vector[i]==1)
      True_Positive=True_Positive+1
    else
      False_Negetive=False_Negetive+1
    }
  else{
    if(cluster_vector[i]==2)
      True_Negetive=True_Negetive+1
    else
      False_Positive=False_Positive+1
  }
}
Confusion_mat=matrix(c(True_Positive,False_Negetive,True_Negetive,False_Positive),nc=2)
Accuracy=(sum(diag(Confusion_mat))/sum(Confusion_mat))*100
Accuracy
```

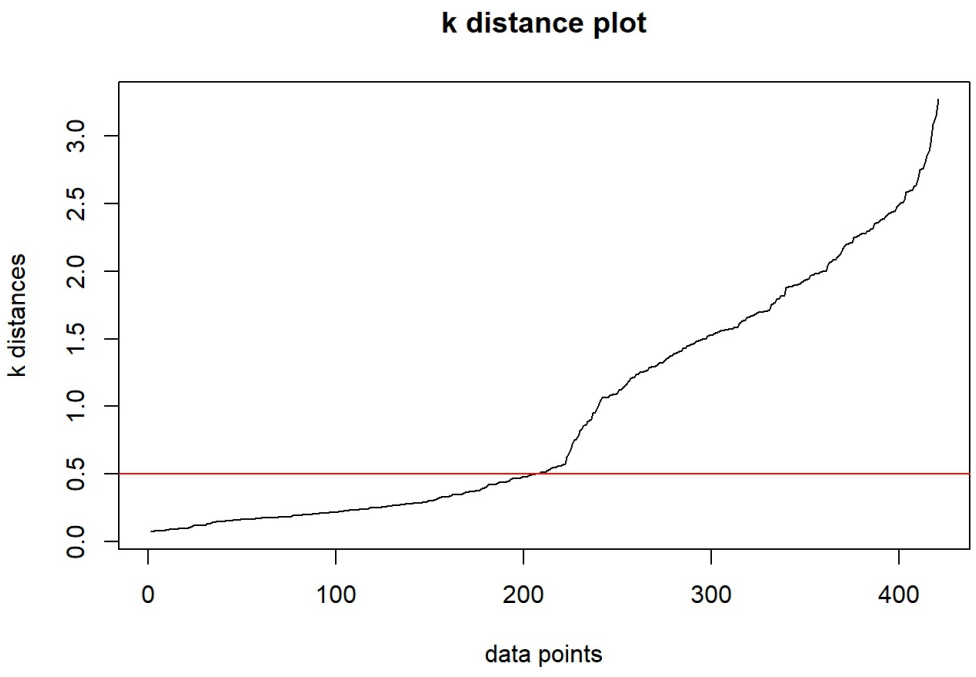
```
[1] 60.09501
```

- Which is Very moderate.

DBSCAN (Density Based Spatial Clustering of Application with Noise)

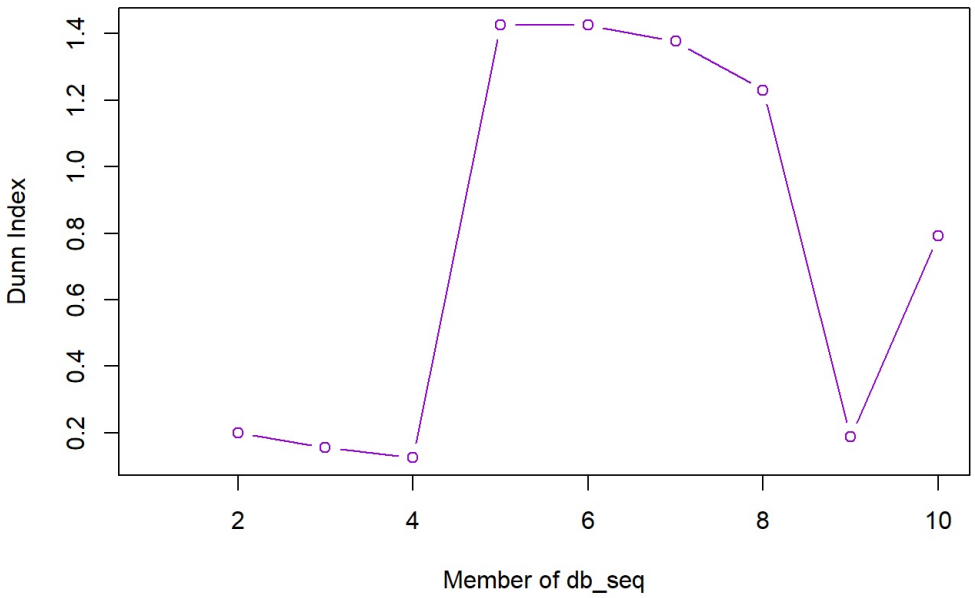
- Here we will need initial value of two parameter *eps* and *minPts*
- Selecting *minPts* = (no of predictors in data +1)=8
- For *eps* we will obtain k-distance plot

```
mat=kNN(final_data, k = 8)
k_dist=sort(mat$dist[,8])
plot(k_dist,typ="l",xlab="data points",ylab="k distances",main="k distance plot")
abline(h=0.5,col="red")
```



- We will use a sequence of values around 0.5 .
- Create a Dunn Index plot and Choose the member of the sequence for which Dunn index will be maximum.

```
db_seq=seq(0.1,2.0,0.1)
dun_index=c()
for(i in 2:10){
  obj6=dbscan(final_data,eps=db_seq[i],minPts=8)
  dun_index[i]=dunn(dist(final_data),obj6$cluster) }
plot(dun_index,typ="b",xlab="Member of db_seq",ylab="Dunn Index",col="darkviolet")
```




```
which.max(dun_index)
```

```
[1] 5
```

The 5'th member is selected so we will choose **eps** =0.5

- Since we also have Noise points here we cannot directly calculate accuracy.
- We remove the noise points and calculate Accuracy with rest of the points.

```
db=dbscan(final_data,eps=db_seq[5],minPts=8)
db
```

```
DBSCAN clustering for 421 objects.
Parameters: eps = 0.5, minPts = 8
Using euclidean distances and borderpoints = TRUE
The clustering contains 2 cluster(s) and 194 noise points.

  0   1   2
194 208  19

Available fields: cluster, eps, minPts, metric, borderPoints
```

```
clus_vec=db$cluster
index_remove=which(clus_vec==0)
cluster_vector=clus_vec[-index_remove]
class_vector=mydata[-index_remove,]$Bankrupt.
```

```
#--Accuracy
True_Positive=0
False_Negetive=0
True_Negetive=0
False_Positive=0
for(i in 1:nrow(final_data[-index_remove,])){
  if(class_vector[i]==0){
    if(cluster_vector[i]==1)
      True_Positive=True_Positive+1
    else
      False_Negetive=False_Negetive+1
    }
  else{
    if(cluster_vector[i]==2)
      True_Negetive=True_Negetive+1
    else
      False_Positive=False_Positive+1
  }
}
Confusion_mat=matrix(c(True_Positive,False_Negetive,True_Negetive,False_Positive),nc=2)
Accuracy=(sum(diag(Confusion_mat))/sum(Confusion_mat))*100
Accuracy
```

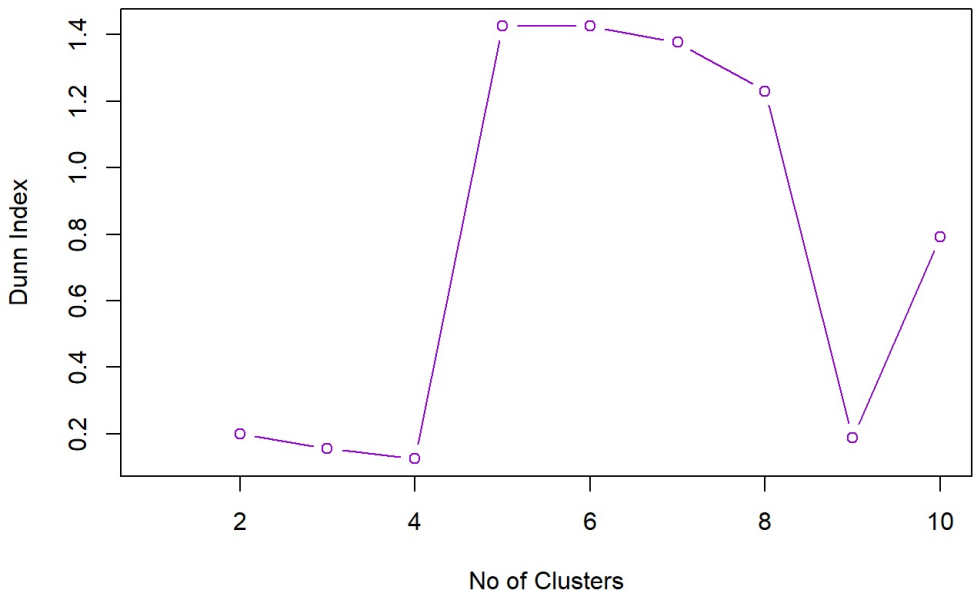
```
[1] 91.62996
```

- Which is Very Good.

OPTICS (Ordering Points To Identify Clustering Structures)

- Here we used the value of **minPts** same as before.
- Select **eps** using the same analogy as before.
- We will use a sequence of values around 0.5 .
- Create a Dunn Index plot and Choose the member of the sequence for which Dunn index will be maximum.

```
dun_index=c()
for(i in 2:10){
  obj6=optics(final_data,eps=db_seq[i],minPts=8)
  dun_index[i]=dunn(dist(final_data),extractDBSCAN(obj6,db_seq[i])$cluster) }
plot(dun_index,typ="b",xlab="No of Clusters",ylab="Dunn Index",col="darkviolet")
```




```
which.max(dun_index)
```

```
[1] 5
```

The 5'th member is selected so we will choose **eps** =0.5

- Since we also have Noise points here we cannot directly calculate accuracy.
- We remove the noise points and calculate Accuracy with rest of the points.
- We will visualize the the Reachability Plot

```
op=optics(final_data,eps=db_seq[5],minPts=8)
table(extractDBSCAN(op,db_seq[5])$cluster)
```

```
  0   1   2
194 208  19
```

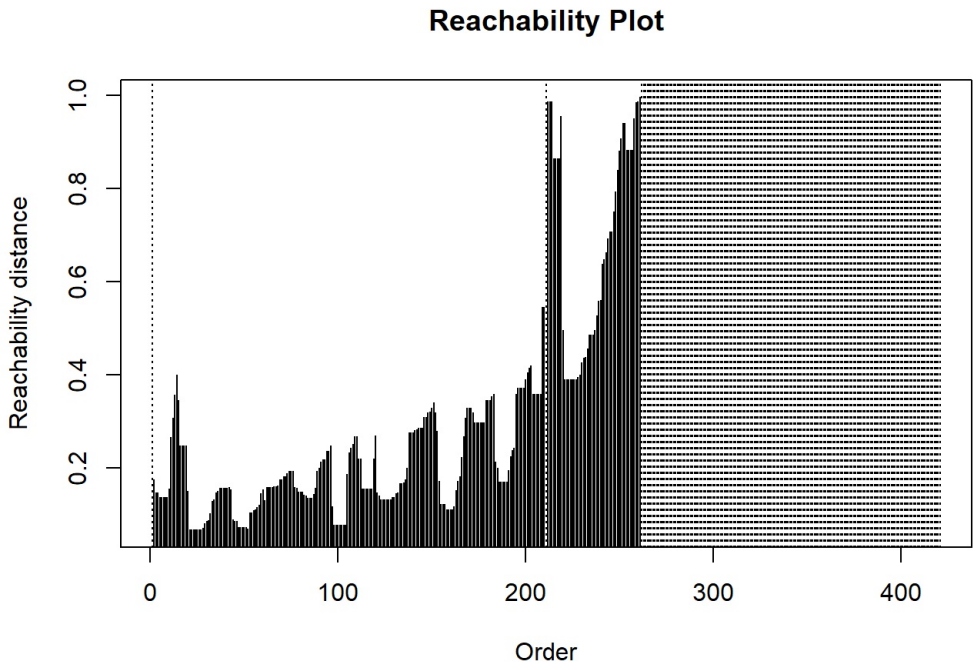
```
clus_vec=extractDBSCAN(optics(final_data,eps=db_seq[5],minPts=8),db_seq[5])$cluster
index_remove=which(clus_vec==0)
cluster_vector=clus_vec[-index_remove]
```

```
#--Accuracy
True_Positive=0
False_Negetive=0
True_Negetive=0
False_Positive=0
for(i in 1:nrow(final_data[-index_remove,])){
  if(class_vector[i]==0){
    if(cluster_vector[i]==1)
      True_Positive=True_Positive+1
    else
      False_Negetive=False_Negetive+1
    }
  else{
    if(cluster_vector[i]==2)
      True_Negetive=True_Negetive+1
    else
      False_Positive=False_Positive+1
  }
}
Confusion_mat=matrix(c(True_Positive,False_Negetive,True_Negetive,False_Positive),nc=2)
Accuracy=(sum(diag(Confusion_mat))/sum(Confusion_mat))*100
Accuracy
```

```
[1] 91.62996
```

- Which is Very Good.

```
plot(obj6,ylab="Reachability distance")
```



** Clearly We have 2 Clusters here with varying density.

Project Report

Summary findings

- The findings from different methods are summarized in the table below

Method Applied	Optimal no of Clusters	Accuracy
K-Means	2	47.98
PAM	2	55.35
AGNES(Single Linkage)	2	99.76247
AGNES(complete Linkage)	2	60.09
DIANA	11	60.09
DBSCAN	2	91.63
OPTICS	2	91.63

Conclusion

In our analysis of bankruptcy data using clustering methods, we observed that several algorithms performed exceptionally well, achieving high classification accuracy. Specifically, the clustering methods produced two distinct clusters, which effectively separated the data into “bankrupt” and “non-bankrupt” categories.

The high accuracy achieved by 3 methods indicates that the clustering algorithms were successful in identifying and grouping data points according to their bankruptcy status. This suggests that the underlying structure of the data is well-represented by the two-cluster model, and that the chosen clustering techniques are well-suited for this classification task.

Overall, the results underscore the effectiveness of clustering as a tool for distinguishing between bankrupt and non-bankrupt entities, validating the robustness of the algorithms used. This not only demonstrates the algorithms’ capability in handling this specific classification problem but also highlights their potential for broader applications in similar data analysis scenarios.