CSCI-B 565 DATA MINING
Homework 3
Evening Class
Computer Science Core
Fall 2013
Indiana University

Tousif Ahmed
touahmed@indiana.edu

15 October, 2013

All the work herein is solely mine.

Problem One: Missing Data

For the data series $\Delta = a, b, a, b, b, a, ?, a, b, b$ where ? is missing data, we can replace the missing data using uniform distribution, random distribution and binomial distribution $\beta(p, n)$. If we use uniform distribution the missing value will be replaced by a. Because there are 5 b's and 4 a's besides the missing value. So, for the case of uniform distribution the number of the probability of getting$a$ and b must be equal. So, as we have one a short, to get equal probability we can replace the missing value by a.
In case of random distribution, we can replace the missing value by either a or b. In the random distribution, it is uncertain what will be the outcome is. So, we can replace it randomly with either a or b. We can replace it by using several terms, we can take values from either 0 or from 1. If a random turn returns 0, we will replace it by a, if it gives 1 we can replace the value by b.
In case of binomial distribution, the missing value can be replaced by either a or b. In binomial distribution, n number of trials with p probability, the probability of getting exactly k successes is $\Pr(X = k) = \binom{n}{k} p^k (1 - p)^{n-k}$. So, we can run it for getting the missing value using binomial distribution. In Δ there are total 10 data, where 4 of them are a and 5 of them are b and one is missing. If the missing value is replaced by a then there will be total 5 a's and if the missing value is replaced by b, then there will be total 6 b's. So lets calculate the random variable for $X = a$ and b using binomial distribution,

$$\Pr(X = a) = \binom{10}{5} \left(\frac{4}{10}\right)^5 \left(1 - \frac{4}{10}\right)^5 = 0.2006$$

$$\Pr(X = b) = \binom{10}{6} \left(\frac{5}{10}\right)^6 \left(1 - \frac{5}{10}\right)^4 = 0.205$$

So the probability of getting b is higher using binomial distribution. So, we will replace the missing value by b.
The significant difference between these three approaches, is that they chose the missing value in different ways. In the case of uniform distribution we are certain that we will replace the missing value by a. But in the case of random distribution, we can replace the missing value by either a or b, which is uncertain. And, in the case of binomial distribution we will replace the missing value by b, which we calculated using the equation of $\beta(p, n)$.

Problem Two: Medical Data

1. The data set Δ is collected from the University of Wisconsin Hospitals, Madison from Dr. William H. Wolberg. The size of Δ is 699. The data set can be used to detect the malignant cancer or it is possible to identify which breast cancers are malignant and which breast cancers are benign. There are 10 attributes for each elements of the data set. And the cancer type actually depends on this attributes. It is possible that some attributes might useful to detect the type of cancer and some are not. We can run some known data mining approach on this data set and using those data mining approach we can learn about the data and then we can use them to identify cancer types. If We use some portion of the data set as a training set and the rest of the sets as the test data set, then we can use the training set to learn about the problem and the test data set can help us to evaluate our approach. And it is possible to achieve more than 90% of accuracy for predicting the type of cancer using some well known data mining algorithms.

   Here the size of Δ is 699.

   (a) No, this is not a large data set. If we compare the dataset with some other real life problem's data sets, we will realize it is pretty small.The problem given in problem 3 has 761K data. If we consider some medical experimental data set, like for the Colon cancer data set it has almost 2000 gene samples the Leukemia data set has almost 7000 gene samples. If we consider some other domains, like for the Network Intrusion detection data set has almost 8 million sample records[1]. So, considering other common data mining problem it has pretty small data set.

   (b) There are 10 attributes excluding the Sample Code Number(SCN) in data sets.This is a pretty large number of attributes considering the data size. The astronomy data set has 6 attributes for 761K data sets. This number of attributes might affect the performance of the data mining algorithms like ID3. For the network intrusion data, though it has 41 attributes but the total number of data set for this problem is 8 million.

2. If we ignore the Sample code number (SCN) there are 10 attributes. They are Clump Thickness,Uniformity of Cell Size, Uniformity of Cell Shape,Marginal Adhesion,Single Epithelial Cell Size,Bare Nuclei,Bland Chromatin,Normal Nucleoli,Mitoses and Class.

3. There are 16 missing values. The following R code will give the SCNs for the missing values and give us Δ^* which is the cleaned data set for Δ where missing values are removed. The missing SCNs are printed at the end of the code and Δ^* is stored in a separate csv file.I observed that the attribute Bare Nuclei only contains the missing value.

```
> cancer_data <- read.table("breast-cancer-wisconsin.data",sep=",",na.strings="?")
> names(cancer_data) <- c("SCN","Clump Thickness","Uniformity of Cell Size",
+ "Uniformity of Cell Shape","Marginal Adhesion","Single Epithelial Cell Size","Bare Nuclei",
+ "Bland Chromatin","Normal Nucleoli","Mitoses","Class")
> write.table(cancer_data,"cancer_data_before_cleaning.csv",sep=",",row.names=FALSE)
> str(cancer_data)
> install.packages("timeSeries")
> library("timeSeries")
> missing_values <- is.na(cancer_data["Bare Nuclei"])
> missing_SCN <- cancer_data[missing_values, c("SCN")]
> cleaned_data <-cancer_data[complete.cases(cancer_data),]
> write.table(cleaned_data,"cancer_data_after_cleaning.csv",sep=",",row.names=FALSE)
> missing_SCN
```

```
[1] 1057013 1096800 1183246 1184840 1193683 1197510 1241232  169356
[9]  432809  563649  606140   61634  704168  733639 1238464 1057067
```

[1]University of edinburgh website(http://www.inf.ed.ac.uk/teaching/courses/dme/html/datasets0405.html)

Moreover, I have found some duplicates. There are 8 duplicate data in the file. I removed it using the following R code. It will give the SCNs of the duplicate data and store the pruned data into a separate csv file. I used that csv file with 675 data in my k-means and ID3 algorithm.

```
> cleaned_data <-cancer_data[complete.cases(cancer_data),]
> cleaned_data <- read.table("cancer_data_after_cleaning.csv",header=TRUE, sep=",")
> v <- duplicated(cleaned_data)

> duplicate <-cleaned_data[v,c("SCN")]
> pruned<-subset(cleaned_data,!duplicated(cleaned_data))
> write.table(pruned,"cancer_data_after_pruning.csv",sep=",", row.names=FALSE)
> summary(v)

   Mode   FALSE    TRUE    NAs
logical     675       8      0

> print(duplicate)

[1] 1218860 1100524 1116116 1198641  320675  704097 1321942  466906
```

The data with SCN 1198641 has three instances where two of them are duplicate and the rest has same SCN but different attribute values. As SCN 1198642 was not assigned, I replaced that SCN with 1198642.

For replacing the missing values I used two approaches. I replaced them using mean,median and random numbers between 1 to 10. I have written the following R code for replacing the missing values using mean,median and random values. The final table is included after the R code.

```
> data <- read.table("cancer_data_before_cleaning.csv", header=TRUE, sep=",")
> cleaned_data <- read.table("cancer_data_after_cleaning.csv", header=TRUE, sep=",")
> data$Bare.Nuclei[is.na(data$Bare.Nuclei)] <- mean(data$Bare.Nuclei,na.rm=TRUE)
> write.table(cancer_data,"cancer_data_after_replace_mean.csv", sep=",", row.names=FALSE)

> data <- read.table("cancer_data_before_cleaning.csv", header=TRUE, sep=",")
> cleaned_data <- read.table("cancer_data_after_cleaning.csv", header=TRUE, sep=",")
> data\Bare.Nuclei[is.na(data$Bare.Nuclei)] <- median(data$Bare.Nuclei,na.rm=TRUE)
> write.table(data,"cancer_data_after_replace_median.csv", sep=",",row.names=FALSE)

> data <- read.table("cancer_data_before_cleaning.csv", header=TRUE, sep=",")
> cleaned_data <- read.table("cancer_data_after_cleaning.csv", header=TRUE, sep=",")
> data$Bare.Nuclei[is.na(data$Bare.Nuclei$)] <- sample(1:10,16,replace=T)
> write.table(data,"cancer_data_after_replace_random.csv", sep=",",row.names=FALSE)
```

The final data is presented in Table 1.

(a) The ratio of the missing data is 0.022. So, from my opinion the amount of missing data is not significant. The removal of missing data from the actual data set will reduce the total no of data values. In data mining algorithms performs better with large data. But, 16 out of 699 is not a large number and may be insignificant. If the ratio is greater than 0.1, then it might get significant.

(b) In my algorithms, I used the data set by removing the tuples with unknown data. I have run it by replacing the values using mean, median and random values. Table 2 gives the result of k-means algorithm where the data set was created by removing the unknown data. It performs better when all the attributes were used and falls gradually when we remove some of the attributes.

Row	SCN	A_i	$Data_{mean}$	$Data_{median}$	$Data_{random}$
24	1057013	Bare Nuclei	3	1	4
41	1096800	Bare Nuclei	3	1	4
140	1183246	Bare Nuclei	3	1	8
146	1184840	Bare Nuclei	3	1	8
159	1193683	Bare Nuclei	3	1	9
165	1197510	Bare Nuclei	3	1	10
236	1241232	Bare Nuclei	3	1	8
250	169356	Bare Nuclei	3	1	3
276	432809	Bare Nuclei	3	1	9
293	563649	Bare Nuclei	3	1	9
295	606140	Bare Nuclei	3	1	10
298	61634	Bare Nuclei	3	1	6
316	704168	Bare Nuclei	3	1	2
412	1238464	Bare Nuclei	3	1	8
618	1057067	Bare Nuclei	3	1	10

Table 1: Final Data set

$C_{km=2}(\Delta^*)$	$PPV_{Euclidean}$	$PPV_{Manhattan}$
$A_1, \ldots, A_9$	0.9600	0.9467
$A_1, \ldots, A_7$	0.9600	0.9556
$A_1, \ldots, A_5$	0.9437	0.9333
$A_1, \ldots, A_3$	0.9407	0.9363
A_1, A_2	0.8593	0.8252

Table 2: PPV values when unknown data is removed

Table 3 is generated by running V-fold cross validation on the same data set. If we look at the results we can found that the PPV is increasing when we are removing some attributes. The weighted PPV $\frac{1}{10}\sum_{i=1}^{10} PPV_i = .9601$ for eucllidean distance and 0.9527 for manhattan distance.

Train	Test	$PPVResult_{Euclidean}$	$PPVResult_{Manhattan}$
$C_{km=2}(D^* - d_1^*)$	$C_{km=2}(d_1^*)$	0.8382	0.8088
$C_{km=2}(D^* - d_2^*)$	$C_{km=2}(d_2^*)$	0.9552	0.9403
$C_{km=2}(D^* - d_3^*)$	$C_{km=2}(d_3^*)$	0.9853	0.9853
$C_{km=2}(D^* - d_4^*)$	$C_{km=2}(d_4^*)$	0.9403	0.9403
$C_{km=2}(D^* - d_5^*)$	$C_{km=2}(d_5^*)$	0.9412	0.9265
$C_{km=2}(D^* - d_6^*)$	$C_{km=2}(d_6^*)$	0.9851	0.9851
$C_{km=2}(D^* - d_7^*)$	$C_{km=2}(d_7^*)$	0.9559	0.9559
$C_{km=2}(D^* - d_8^*)$	$C_{km=2}(d_8^*)$	1.0000	1.0000
$C_{km=2}(D^* - d_9^*)$	$C_{km=2}(d_9^*)$	1.0000	1.0000
$C_{km=2}(D^* - d_{10}^*)$	$C_{km=2}(d_{10}^*)$	1.0000	1.0000

Table 3: V fold cross validation when unknown data is removed

I ran ID3 algorithm with same data set. I used 80% of the data to training set and the rest 20% is used as a test data set. It gave almost 93.3% of accuracy for the same data set.

When we replace the unknown data with mean values, it gave almost same performance as Table 1. Table 4 shows the result for PPV values and Table 5 shows the result of V fold cross validation. In the case of V-fold cross validation, the performance was better with the removed data set. The weighted PPV was 0.9569 for Euclidean distance and 0.9498 for Manhattan distance. In this case

$C_{km=2}(\Delta^*)$	$PPV_{Euclidean}$	$PPV_{Manhattan}$
$A_1, \ldots, A_9$	0.9571	0.9428
$A_1, \ldots, A_7$	0.9571	0.9528
$A_1, \ldots, A_5$	0.9385	0.9313
$A_1, \ldots, A_3$	0.9385	0.9342
A_1, A_2	0.8612	0.8612

Table 4: PPV values when unknown data is replaced by mean

Train	Test	$PPVResult_{Euclidean}$	$PPVResult_{Manhattan}$
$C_{km=2}(D^* - d_1^*)$	$C_{km=2}(d_1^*)$	0.8143	0.7857
$C_{km=2}(D^* - d_2^*)$	$C_{km=2}(d_2^*)$	0.9565	0.9420
$C_{km=2}(D^* - d_3^*)$	$C_{km=2}(d_3^*)$	0.9857	0.9857
$C_{km=2}(D^* - d_4^*)$	$C_{km=2}(d_4^*)$	0.9420	0.9420
$C_{km=2}(D^* - d_5^*)$	$C_{km=2}(d_5^*)$	0.9571	0.9286
$C_{km=2}(D^* - d_6^*)$	$C_{km=2}(d_6^*)$	0.9565	0.9565
$C_{km=2}(D^* - d_7^*)$	$C_{km=2}(d_7^*)$	0.9714	0.9714
$C_{km=2}(D^* - d_8^*)$	$C_{km=2}(d_8^*)$	0.9855	0.9855
$C_{km=2}(D^* - d_9^*)$	$C_{km=2}(d_9^*)$	1.0000	1.0000
$C_{km=2}(D^* - d_{10}^*)$	$C_{km=2}(d_{10}^*)$	1.0000	1.0000

Table 5: V fold cross validation when unknown data is replaced by mean

though the main was 3.5, I took 3. In the ID3 algorithm, I got 93.45% accuracy using this same data set which is also decreasing. They are replaced by same value which can cause performance degradation.

When I replaced the unknown data with the median value, the PPV was same as removed data set. And the V fold cross validation is same as mean value data set. The weighted PPV was 0.9569 for Euclidean distance and 0.9498 for Manhattan distance. In both of these cases data were replaced by same value, that's why there performance is almost same. In the ID3 algorithm, I got better result with this data set. It gave 93.58% accurate classification for the cancer data set. Table 6 shows the PPV values and table 7 shows the result for V fold cross validation.

When I changed the missing values with some random values between 1 to 10. The PPV results after removing attributes and V fold cross validation was almost same as previous cases. The weighted PPV was 0.9583 for Euclidean distance and 0.9498 for Manhattan distance. In the case of ID3 the performance degraded but not much. It can classify almost 93.17% of data correctly. Table 8 and 9 shows the result with the random replacing data set.

So, observing the result we can conclude that the performance is almost same whether we keep or remove the tuples with unknown data. Though, there is some variation with the later data sets, but not much. But replacing the data sometimes degrading the performance. So, there is no sufficient significance for keeping the unknown data. Which means, we can use removed data set for our algorithms.

$C_{km=2}(\Delta^*)$	$PPV_{Euclidean}$	$PPV_{Manhattan}$
$A_1, \ldots, A_9$	0.9571	0.9428
$A_1, \ldots, A_7$	0.9571	0.9514
$A_1, \ldots, A_5$	0.9385	0.9285
$A_1, \ldots, A_3$	0.9385	0.9342
A_1, A_2	0.8584	0.8612

Table 6: PPV when unknown data is replaced by median

Train	Test	$PPV Result_{Euclidean}$	$PPV Result_{Manhattan}$
$C_{km=2}(D^* - d_1^*)$	$C_{km=2}(d_1^*)$	0.8143	0.7857
$C_{km=2}(D^* - d_2^*)$	$C_{km=2}(d_2^*)$	0.9565	0.9420
$C_{km=2}(D^* - d_3^*)$	$C_{km=2}(d_3^*)$	0.9857	0.9857
$C_{km=2}(D^* - d_4^*)$	$C_{km=2}(d_4^*)$	0.9420	0.9420
$C_{km=2}(D^* - d_5^*)$	$C_{km=2}(d_5^*)$	0.9571	0.9286
$C_{km=2}(D^* - d_6^*)$	$C_{km=2}(d_6^*)$	0.9565	0.9565
$C_{km=2}(D^* - d_7^*)$	$C_{km=2}(d_7^*)$	0.9714	0.9714
$C_{km=2}(D^* - d_8^*)$	$C_{km=2}(d_8^*)$	0.9855	0.9855
$C_{km=2}(D^* - d_9^*)$	$C_{km=2}(d_9^*)$	1.0000	1.0000
$C_{km=2}(D^* - d_{10}^*)$	$C_{km=2}(d_{10}^*)$	1.0000	1.0000

Table 7: V fold cross validation when unknown data is replaced by median

$C_{km=2}(\Delta^*)$	$PPV_{Euclidean}$	$PPV_{Manhattan}$
$A_1, \ldots, A_9$	0.9571	0.9428
$A_1, \ldots, A_7$	0.9585	0.9499
$A_1, \ldots, A_5$	0.9385	0.9285
$A_1, \ldots, A_3$	0.9385	0.9342
A_1, A_2	0.8040	0.8584

Table 8: PPV when unknown data is replaced by random values

Train	Test	$PPV Result_{Euclidean}$	$PPV Result_{Manhattan}$
$C_{km=2}(D^* - d_1^*)$	$C_{km=2}(d_1^*)$	0.8286	0.7857
$C_{km=2}(D^* - d_2^*)$	$C_{km=2}(d_2^*)$	0.9565	0.9420
$C_{km=2}(D^* - d_3^*)$	$C_{km=2}(d_3^*)$	0.9857	0.9714
$C_{km=2}(D^* - d_4^*)$	$C_{km=2}(d_4^*)$	0.9420	0.9420
$C_{km=2}(D^* - d_5^*)$	$C_{km=2}(d_5^*)$	0.9571	0.9286
$C_{km=2}(D^* - d_6^*)$	$C_{km=2}(d_6^*)$	0.9565	0.9710
$C_{km=2}(D^* - d_7^*)$	$C_{km=2}(d_7^*)$	0.9714	0.9714
$C_{km=2}(D^* - d_8^*)$	$C_{km=2}(d_8^*)$	0.9855	0.9855
$C_{km=2}(D^* - d_9^*)$	$C_{km=2}(d_9^*)$	1.0000	1.0000
$C_{km=2}(D^* - d_{10}^*)$	$C_{km=2}(d_1 0^*)$	1.0000	1.0000

Table 9: V fold cross validation when unknown data is replaced by random values

4. (a) A_6 or Bare Nuclei has the greatest variance. The following code give us the code for calculating variance and return the maximum variance at the end of the code.

```
> calc_variance <- function(A_i)
+ {
+         name<- names(A_i)
+         datalist <- A_i[,name]
+         nums<- sapply(A_i,as.numeric)
+         mean<-mean(nums)
+         squared_sum <- 0
+         length_Ai <- length(A_i[,name])
+         for(i in 1:length_Ai)
+         {
+                 squared_sum <- squared_sum + ((datalist[i]-mean)^2)
+         }
+         return(squared_sum/(length_Ai-1))
+ }

> get_maxm_variance <- function(dataset)
+ {
+         maximum_variance <- 0
+         index <- 0
+         for(i in 2:10)
+         {
+                 variance <- calc_variance(dataset[i])
+                 if(maximum_variance < variance)
+                 {
+                 maximum_variance <- variance
+                 index <- i
+                 }
+         }
+         return(index)
+ }
> cleaned_data <- read.table("cancer_data_after_cleaning.csv", header=TRUE, sep=",")
> maximum_attr <- get_maxm_variance(cleaned_data)
> print(paste("A_",maximum_attr-1))
```

```
[1] "A_ 6"
```

```
> print(paste("Feature:",names(cleaned_data[maximum_attr])))
```

```
[1] "Feature: Bare.Nuclei"
```

(b) A_6 or Bare Nuclei has the lowest entropy. The following code returns the attribute with lowest entropy.

```
> install.packages("entropy")
> library("entropy")
> get_minimum_entropy <- function(dataset)
+ {
+         minimum_entropy <- 10000000000
+         index <- 0
+         for(i in 2:10)
+         {
+                 data <- dataset[i]
+                 name<- names(data)
+                 datalist <- data[,name]
+                 entropy <- entropy(datalist,unit="log2")
+                 if(minimum_entropy > entropy)
+                 {
+                         minimum_entropy <- entropy
+                         index <- i
+                 }
+         }
+         return(index)
+ }
> cleaned_data <- read.table("cancer_data_after_cleaning.csv", header=TRUE, sep=",")
> min_entrpy <- get_minimum_entropy(cleaned_data)
> print(paste("A_",min_entrpy-1))

[1] "A_ 6"

> print(paste("Feature:",names(cleaned_data[min_entropy])))

[1] "Feature: Bare.Nuclei"
```

(c) The following code gives us the KL distance for attribute pairs. I use seewave package for calculating the KL distance for each pair of attributes.

```
> install.packages("seewave")
> library("seewave")
> get_KL_distance <- function(dataset)
+ {
+ name_list <- c("A_1","A_2","A_3","A_4","A_5","A_6", "A_7", "A_8", "A_9")
+ kl_matrix <- matrix(1:81, ncol=9, nrow=9, dimnames=list(name_list,name_list))
+ for(i in 2:10)
+ {
+ for(j in i:10)
+ {
+ if(i==j)
+ {
+ kl_matrix[i-1,j-1] <- 0
+ }
+ else
+ {
+ kl_dist <- kl.dist(dataset[i],dataset[j])
+ kl_matrix[i-1,j-1] <- round( kl_dist$D1,digits=2)
+ kl_matrix[j-1,i-1] <- round(kl_dist$D2, digits=2)
+ }
+ }
+ }
+ return(kl_matrix)
+ }
> cleaned_data <- read.table("cancer_data_after_pruning.csv", header=TRUE, sep=",")
> kl_matrix <- get_KL_distance(cleaned_data)
> install.packages("xtable")
> library("xtable")
> x=xtable(kl_matrix,align=rep("",ncol(kl_matrix)+1))
> print(x, floating=FALSE,
+ hline.after=NULL, include.rownames=TRUE, include.colnames=TRUE)
```

	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9
A_1	0.00	0.35	0.32	0.46	0.26	0.48	0.29	0.48	0.43
A_2	0.33	0.00	0.09	0.29	0.23	0.34	0.24	0.32	0.53
A_3	0.31	0.09	0.00	0.31	0.24	0.33	0.24	0.33	0.53
A_4	0.45	0.27	0.29	0.00	0.33	0.38	0.31	0.44	0.56
A_5	0.28	0.24	0.25	0.35	0.00	0.44	0.22	0.36	0.34
A_6	0.44	0.30	0.29	0.38	0.40	0.00	0.34	0.50	0.68
A_7	0.30	0.24	0.25	0.31	0.21	0.38	0.00	0.36	0.46
A_8	0.45	0.27	0.27	0.43	0.33	0.47	0.32	0.00	0.59
A_9	0.49	0.48	0.50	0.55	0.34	0.68	0.49	0.54	0.00

5. The following table is generated after running the k-means algorithm with the cleaned data set. There was 675 data in total, 439 of them are benign and 236 of them are malignant data. k-means performs better with large number of attributes. It performed better with 9 and 7 attributes. Using 5 and 3 attributes degrade the performance but not significantly. But using 2 attributes is affecting the performance of k-means algorithm significantly. And after running several iteration, I have come to this conclusion that using 3,5,7 and 9 attributes gives almost same results in every cases. But using two attribute is completely depends on the quality of the initial centroids.Sometimes it gives less than 70% of PPV and sometimes it is greater than 80%. In the case of distance matrix, Euclidean matrix gives slightly better result than Manhattan distance which is obvious. In the ID3 algorithm, I have implemented it by using information gain. And it gave approximately 93.3% of accuracy. The accuracy was calculated by calculating true positive, true negative, false positive and false negative.

$C_{km=2}(\Delta^*)$	$PPV_{Euclidean}$	$PPV_{Manhattan}$
$A_1, \ldots, A_9$	0.9600	0.9467
$A_1, \ldots, A_7$	0.9600	0.9556
$A_1, \ldots, A_5$	0.9437	0.9333
$A_1, \ldots, A_3$	0.9407	0.9363
A_1, A_2	0.8593	0.8252

Table 10: PPV result

6. After running 10 fold cross validation, I came to this conclusion that removing the first $\frac{675}{10} \approx 68$ tuples gives poor result than removing others. And removing the last 3 partitions does not affect the result and gives the best result. The weighted PPV was 0.9601 for Euclidean distance and 0.9527 for Manhattan distance.

Train	Test	$PPV Result_{Euclidean}$	$PPV Result_{Manhattan}$
$C_{km=2}(D^* - d_1^*)$	$C_{km=2}(d_1^*)$	0.8382	0.8088
$C_{km=2}(D^* - d_2^*)$	$C_{km=2}(d_2^*)$	0.9552	0.9403
$C_{km=2}(D^* - d_3^*)$	$C_{km=2}(d_3^*)$	0.9853	0.9853
$C_{km=2}(D^* - d_4^*)$	$C_{km=2}(d_4^*)$	0.9403	0.9403
$C_{km=2}(D^* - d_5^*)$	$C_{km=2}(d_5^*)$	0.9412	0.9265
$C_{km=2}(D^* - d_6^*)$	$C_{km=2}(d_6^*)$	0.9851	0.9851
$C_{km=2}(D^* - d_7^*)$	$C_{km=2}(d_7^*)$	0.9559	0.9559
$C_{km=2}(D^* - d_8^*)$	$C_{km=2}(d_8^*)$	1.0000	1.0000
$C_{km=2}(D^* - d_9^*)$	$C_{km=2}(d_9^*)$	1.0000	1.0000
$C_{km=2}(D^* - d_{10}^*)$	$C_{km=2}(d_{10}^*)$	1.0000	1.0000

Table 11: V fold Cross Validation

Problem Three: Astronomy

1. $R(g,g')$ is a metric. The proof is given below-
Proof of Positivity: $R(g,g') \geq 0$ for all g and g'. It is trivial because -

$$R(g,g') = \left(\left(\frac{z_g - z_{g'}}{0.2596}\right)^2 + \left(\frac{r_g - r_{g'}}{8.6}\right)^2 + \left(\frac{(u_g - rg) - (u_{g'} - r_{g'})}{11.84}\right)^2 \right)^{\frac{1}{2}}$$

is constituted by adding three square terms. The square of any value is always greater than 0. So, R will also be positive. And, R is constructed by three terms. Each term has pair of differences between two rows. So, each term will be equal to zero when the values are equal. For example, when $z_g = z'_g$ the first term will be equal to zero. This is same for the rest of the terms. So $R = 0$, when $g = g'$. So, this proves the proof of positivity.
Proof of Symmetry: This is also a trivial proof.

$$\begin{aligned} R(g,g') &= \left(\left(\frac{z_g - z_{g'}}{0.2596}\right)^2 + \left(\frac{r_g - r_{g'}}{8.6}\right)^2 + \left(\frac{(u_g - rg) - (u_{g'} - r_{g'})}{11.84}\right)^2 \right)^{\frac{1}{2}} \\ &= \left(\left(\frac{z_{g'} - z_g}{0.2596}\right)^2 + \left(\frac{r_{g'} - r_g}{8.6}\right)^2 + \left(\frac{(u_{g'} - r_{g'}) - (u_g - rg)}{11.84}\right)^2 \right)^{\frac{1}{2}} \qquad [(a-b)^2 = (b-a)^2] \\ &= R(g',g) \end{aligned}$$

So, R follows the proof of symmetry.
Proof of Triangle Inequality: To proof the triangle inequality, we can rewrite R in following way-

$$\begin{aligned} R(g,g') &= \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + (a_3 - b_3)^2} \\ &= \sqrt{\left(\sum_{i=1}^{3} (a-b)^2\right)} \\ &= d(a,b) \end{aligned}$$

Where,

$$\begin{aligned} a_1 &= \frac{z_g}{0.2596} \\ b_1 &= \frac{z'_g}{0.2596} \\ a_2 &= \frac{r_g}{8.6} \\ b_2 &= \frac{z'_g}{8.6} \\ a_3 &= \frac{u_g - r_g}{11.84} \\ b_3 &= \frac{u'_g - r'_g}{11.84} \\ a &= a_1, a_2, a_3 \\ b &= b_1, b_2, b_3 \end{aligned}$$

So, we can prove the triangle inequality by proving the follownig equation

$$d(a,c) \;\leq\; d(a,b) + d(b,c) \qquad [d(a,b) = \sqrt{\left(\sum_{i=1}^{3} (a-b)^2\right)}]$$

If we consider a and b as a vector then $d(a,b) = d(\vec{a},\vec{b}) = |\vec{a}-\vec{b}|$. So,

$$\begin{aligned} d(a,c) &= d(\vec{a}-\vec{c}) = |\vec{a}-\vec{c}| \\ &= |\vec{a}-\vec{b}+\vec{b}-\vec{c}| \\ &\leq |\vec{a}-\vec{b}| + |\vec{b}-\vec{c}| \qquad [|x+y| \leq |x|+|y|] \\ &\leq d(a,b) + d(b,c) \end{aligned}$$

So, triangle inequality holds for $R(g,g')$. All the three conditions holds for R, so R is a metric.

2. I ran the control algorithm and get the control list which consists 7246 rows. I compared it with the control_2 subset and got 1093 matching instances. I ran it several times and got almost same number of matching values. I compared the Galaxy Id of control sets with the Galaxy Id of the control_2 sets. I stored my control values in the control.csv file which may be found in the DMHW_P3 directory. And i stored the matched instances in the matched.csv file. The Galaxy Ids can be found in controloutput.txt file. Which may not same as the actual Id's. But, the actual Id's can be found in matched.csv file. At first, the algorithm took almost 15 minutes but after dividing it into threads it gives the result with in 5 minutes.

3. I ran the k-means algorithm on *candidate* data set using *main* as centroids. I divided the candidate data set into three other data set which is candidate_0, candidate_1 and candidate_2 for galaxy types 0,1 and 2. Similarly, I divided the main data set into three other data set main_0 , main_1 and main_2. They stored into csv file. Then I ran the k-means algorithm on these data set. After running k-means algorithm on these data set , I got three blocks of clusters and the centroids were stored in cluster_0_block.csv, cluster_1_block.csv and cluster_2_block.csv. In those files I stored their values and I added one extra column which shows data count. Data count is actually showing number of candidate data associated with the corresponding clusters. The runtime for block 0 and block 1 is pretty small. But, it took almost 6 hours to run the algorithm for block 2.
After running the algorithm several times, I found out that the speed of convergence is dependent on the threshold value. I set 5,2 and 1 as threshold margin. If I use threshold margin 5 for block 0, the algorithm gives the final centroids after 28 iterations. For threshold margin 2, it gives the final result after 44 iterations. And for threshold margin 1, it took 47 iterations. For block 1, if I use threshold margin 5 it needs 94 iterations to give the final centroids. For threshold margin 2, it took 112 iterations and for threshold margin 1, it took 141 iterations. If i use threshold margin 5, it needs almost 284 steps to get the final centroids for block 2.

4. I have run some R codes to perform data analysis on the features of main and candidate. For the candidate data, the galaxy type has three different values 0,1,2. There are 12355 instances with galaxy type 0, 51247 instances with galaxy type 1, 697905 instances with galaxy type 2 values. For the rest of the attributes the mean, median, maximum and minimum values are listed below-

Attribute name	Max Value	Min Value	Mean	Median	Variance	Entropy
right ascension	359.9979	0.0007	182.4604	184.3858	3852.645	13.47414
declination	84.270	-14.071	23.918	22.444	375.7885	13.37053
u-band magnitude	33.05	-9999.00	19.47	19.57	1451.558	13.63387
r-band magnitude	20.792	-1.385	17.091	17.244	0.7836468	13.54169
the spectroscopic redshift	20.792	-0.01011	0.12048	0.10756	0.00512957	13.37489

Table 12: Summary of candidate data

Table 13 represents the maximum value, minimum value, mean, median, variance and entropy for main data. There are 7246 rows in the main data set, in which 949 of them have galaxy type 0, 859 have galaxy type 1 and 5438 have galaxy type 2. Both table 12 and 13 are generated by using some basic R functions like summary, var and entropy.

Attribute name	Max Value	Min Value	Mean	Median	Variance	Entropy
right ascension	359.9950	0.0065	184.5384	184.9200	2828.193	8.840168
declination	64.43	-11.19	25.36	24.16	344.3973	8.6823
u-band magnitude	28.85	15.08	19.87	20.00	1.330857	8.886519
r-band magnitude	20.79	12.19	16.67	16.87	0.7004889	8.886925
the spectroscopic redshift	0.29955	0.01252	0.16303	0.16466	0.003377972	8.821028

Table 13: Summary of main data

If we look at both of the tables we can see that for all of the attributes the values are almost same. But there is significant discrepancy for the u-band magnitude attribute in the candidate data set. The minimum value for u-band magnitude in the candidate data set is significantly small considering other attributes. And, the minimum value for u-band magnitude in the main data set is 15.08. The difference is quite large. And the difference of these values is also causing higher difference in their variance. In the main data set, the variance is near to 1, whether the variance for u-band magnitude in candidate data set is over 1450. Which is reasonably large. So, I wrote another R code to search for counting the negative values in the u-band magnitude attribute on candidate data set. And found that there are 12 rows which have negative values for the u-band magnitude attribute. After removing them I got the minimum value for u-band magnitude is 5.40 and the variance becomes 1.6888. Now, the values of two tables are pretty closed. There might be some discrepancy but that could be negligible considering the difference of the data set.