HW2: Cluster Analysis and Decision Tree Induction

Eunmi(Ellie) Jeong

2020 10 10

Section 1. Data preparation

1. Check the structure of dataset (training dataset)

```
data <- read.csv("Weather Forecast Training.csv")
summary(data)
str(data) #51978 obs. 16 vars
```

2. Data cleaning to remove data quality issues (missing data, duplicate data, biased data, outliers/noises, attributes with no or low variance)

- Missing data
  - Delete attributes which include too many missing values (more than 5% of total number of instances 51987*0.05 = 2599)
  - Deleted attributes: Evaporation, Sunshine, WindGustSpeed, WindGustDir Pressure, Cloud
  - na_interpolation: if the number of NAs is less than 2599
  - Remove blanks

```
sum(is.na(data))
data2 <- subset(data,select=-c(Evaporation, Sunshine, WindGustSpeed, WindGustDir, Pressure, Cloud ))
data2$MinTemp <- na_interpolation(data2$MinTemp)
data2$MaxTemp <- na_interpolation(data2$MaxTemp)
data2$Rainfall <- na_interpolation(data2$Rainfall)
data2$WindSpeed <- na_interpolation(data2$WindSpeed)
data2$Humidity <- na_interpolation(data2$Humidity)
data2$Temp <- na_interpolation(data2$Temp)
levels(data2$RainToday) #""=747
data2 <- data2[!(data2$RainToday==""),]
data2$RainToday <- as.character(data2$RainToday)
data2$RainToday <- as.factor(data2$RainToday)
sum(!complete.cases(data2)) #no missing data
```

- Check duplicate data (none)

```
nrow(data2[!duplicated(data2), ]) #51231 (no duplicates)
```

```
[1] 51231
```

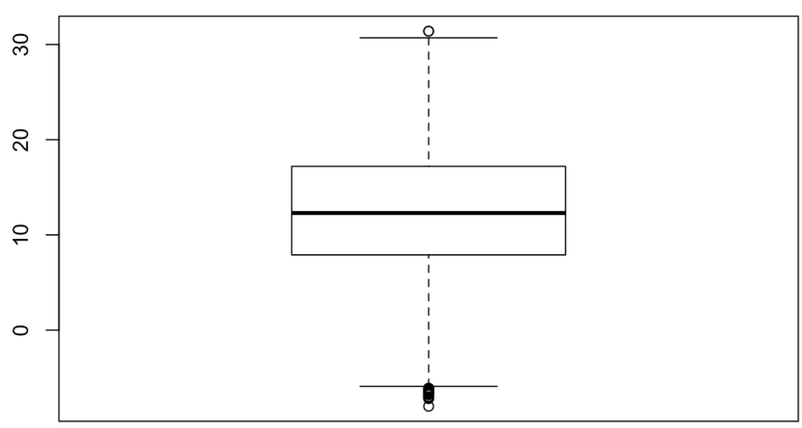
- Remove low or no variance (none)

```
caret::nearZeroVar(data2, saveMetrics = T)
```

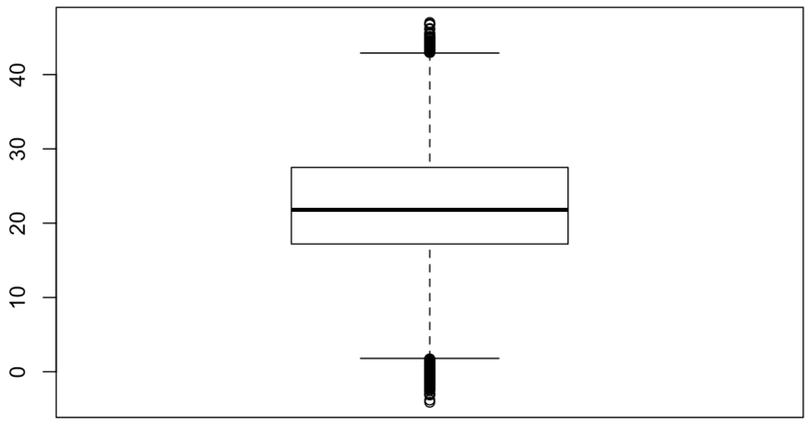
```
             freqRatio percentUnique zeroVar   nzv
Location      1.013354   0.095645215   FALSE FALSE
MinTemp       1.066869   0.882278308   FALSE FALSE
MaxTemp       1.019231   1.044289590   FALSE FALSE
Rainfall      9.106314   1.169213952   FALSE FALSE
WindDir       1.014552   0.033183034   FALSE FALSE
WindSpeed     1.015552   0.228377350   FALSE FALSE
Humidity      1.000000   0.411860007   FALSE FALSE
Temp          1.021875   1.346840780   FALSE FALSE
RainToday     2.265825   0.003903886   FALSE FALSE
RainTomorrow  1.052031   0.003903886   FALSE FALSE
```

- Identify&remove noises & outliers

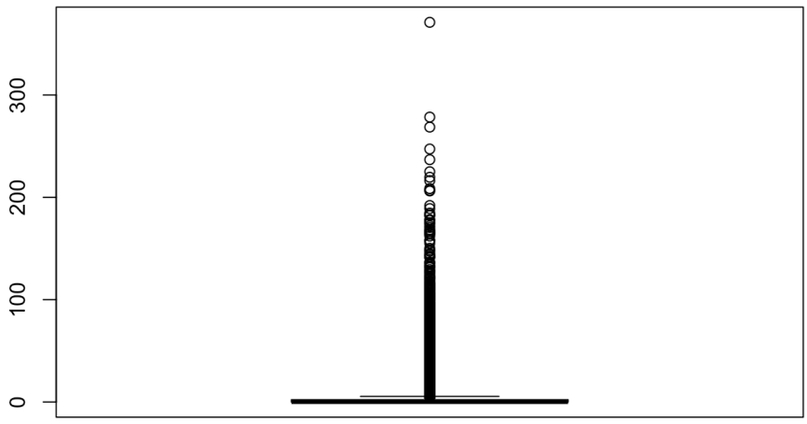
```
boxplot(data2$MinTemp)
```



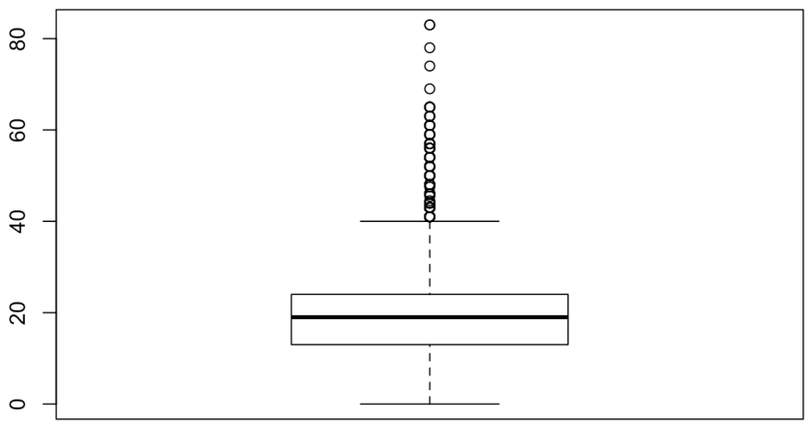
```
data2$MinTemp[data2$MinTemp %in% boxplot.stats(data2$MinTemp)$out] <- median(data2$MinTemp, na.rm = T)
boxplot(data2$MaxTemp)
```



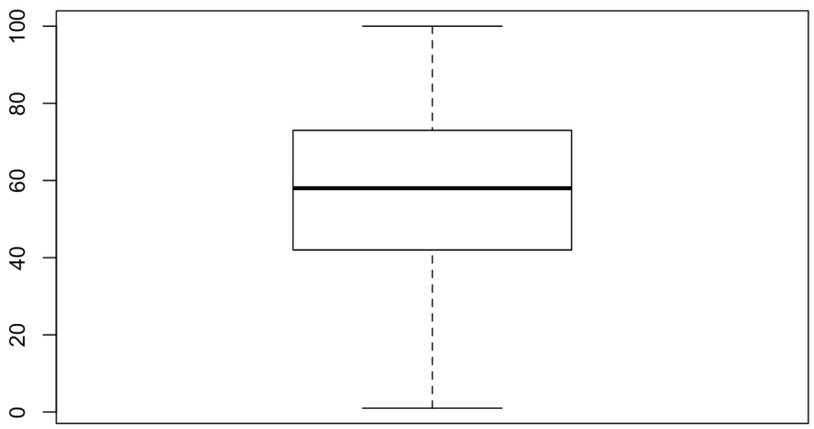

```
data2$MaxTemp[data2$MaxTemp %in% boxplot.stats(data2$MaxTemp)$out] <- median(data2$MaxTemp, na.rm = T)
boxplot(data2$Rainfall)
```



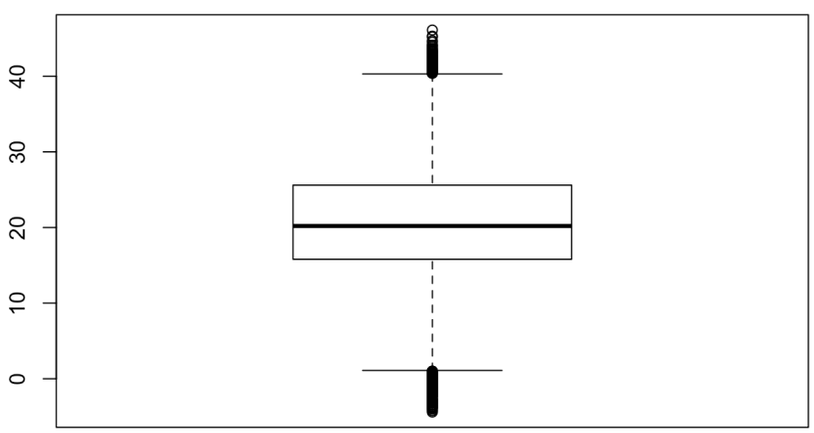

```
data2$Rainfall[data2$Rainfall %in% boxplot.stats(data2$Rainfall)$out] <- median(data2$MinTemp, na.rm = T)
boxplot(data2$WindSpeed)
```



```
data2$WindSpeed[data2$WindSpeed %in% boxplot.stats(data2$WindSpeed)$out] <- median(data2$WindSpeed, na.rm = T)
boxplot(data2$Humidity)
```




```
boxplot(data2$Temp)
```




```
data2$Temp[data2$Temp %in% boxplot.stats(data2$Temp)$out] <- median(data2$Temp, na.rm = T)
```

3. Discretization (for 'Decision Tree' modeling which work with both categorical and numerical attributes)

```
data2$MinTemp_grp <- ifelse(data2$MinTemp > 0,"aboveZero", "belowZero")
data2 %>%
  group_by(MinTemp_grp) %>%
  summarise(avg = mean(MinTemp), count = n(), min = min(MinTemp), max = max(MinTemp))
```

```
# A tibble: 2 x 5
  MinTemp_grp   avg count   min   max
  <chr>       <dbl> <int> <dbl> <dbl>
1 aboveZero   12.9  50194   0.1  30.7
2 belowZero   -1.71  1037  -5.9   0
```

```
  data2$MaxTemp_grp <- ifelse(data2$MaxTemp >= 30,"high", ifelse(data2$MaxTemp >= 15, "middle", "low"))
data2 %>%
  group_by(MaxTemp_grp) %>%
  summarise(avg = mean(MaxTemp), count = n(), min = min(MaxTemp), max = max(MaxTemp))
```

```
# A tibble: 3 x 5
  MaxTemp_grp   avg count   min   max
  <chr>       <dbl> <int> <dbl> <dbl>
1 high         33.6  8545  30    42.9
2 low          12.6  7207   1.8  14.9
3 middle       21.9 35479  15    29.9
```

```
  data2$Rainfall_grp <- ifelse(data2$Rainfall > 0,"Yes", "No")
data2$WS_grp <- ifelse(data2$WindSpeed >= 25,"high", ifelse(data2$WindSpeed >= 15, "middle", "low"))
data2 %>%
  group_by(WS_grp) %>%
  summarise(avg = mean(WindSpeed), count = n(), min = min(WindSpeed), max = max(WindSpeed))
```

```
# A tibble: 3 x 5
  WS_grp   avg count   min   max
  <chr>  <dbl> <int> <dbl> <dbl>
1 high   30.2  11061    25  40
2 low     9.35 16044     0  14.7
3 middle 19.2  24126    15  24.5
```

```
  data2$H_grp <- ifelse(data2$Humidity >= 70,"high", ifelse(data2$Humidity < 40, "low", "middle"))
data2 %>%
  group_by(H_grp) %>%
  summarise(avg = mean(Humidity), count = n(), min = min(Humidity), max = max(Humidity))
```

```
# A tibble: 3 x 5
  H_grp    avg count   min   max
  <chr>  <dbl> <int> <dbl> <dbl>
1 high    82.7 15441    70 100
2 low     26.8 11125     1  39.5
3 middle  55.4 24665    40  69.7
```

```
  data2$Temp_grp <- ifelse(data2$Temp >= 30,"high", ifelse(data2$Temp >= 15, "middle", "low"))
data2 %>%
  group_by(Temp_grp) %>%
  summarise(avg = mean(Temp), count = n(), min = min(Temp), max = max(Temp))
```

```
# A tibble: 3 x 5
  Temp_grp   avg count   min   max
  <chr>    <dbl> <int> <dbl> <dbl>
1 high      33.1  5558  30    40.3
2 low       12.2 10540   1.1  15.0
3 middle    21.5 35133  15    29.9
```

```
data2$MinTemp_grp <- as.factor(data2$MinTemp_grp)
data2$MaxTemp_grp <- as.factor(data2$MaxTemp_grp)
data2$Rainfall_grp <- as.factor(data2$Rainfall_grp)
data2$WS_grp <- as.factor(data2$WS_grp)
data2$H_grp <- as.factor(data2$H_grp)
data2$Temp_grp <- as.factor(data2$Temp_grp)
```

4. Scaling

```
data3 <- scale(data2[complete.cases(data2), sapply(data2, is.numeric)], center = T, scale = T)
```

5. Check the structure of dataset (testing dataset)

```
test <- read.csv("Weather Forecast Testing.csv")
str(test) #12994 obs, 16 vars
```

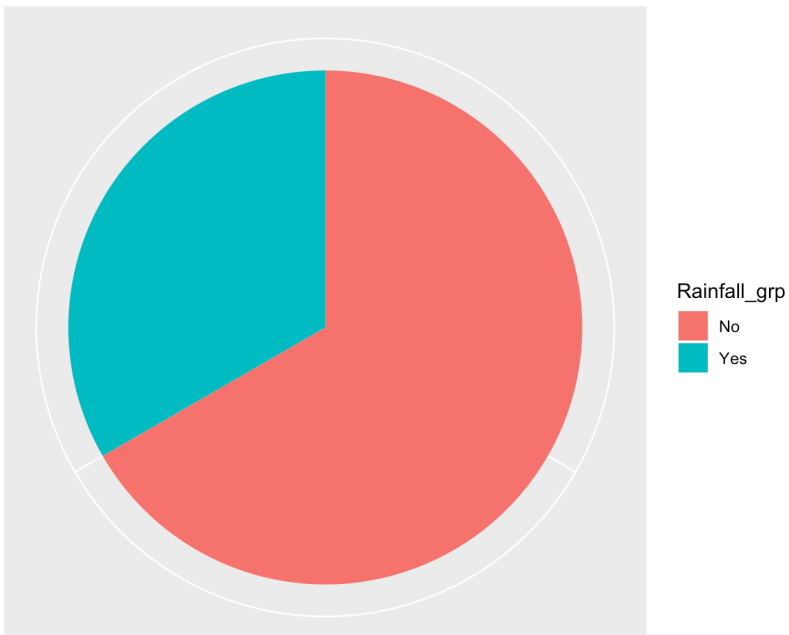
6. Data preprocessing (cleaning, na interpolation, discretization, scaling)

```
test2 <- subset(test,select=-c(Evaporation, Sunshine, WindGustSpeed, WindGustDir, Pressure, Cloud ))
test2$MinTemp <- na_interpolation(test2$MinTemp)
test2$MaxTemp <- na_interpolation(test2$MaxTemp)
test2$Rainfall <- na_interpolation(test2$Rainfall)
test2$WindSpeed <- na_interpolation(test2$WindSpeed)
test2$Humidity <- na_interpolation(test2$Humidity)
test2$Temp <- na_interpolation(test2$Temp)
levels(test2$RainToday) #""=161
test2 <- test2[!(test2$RainToday==""),]
test2$RainToday <- as.character(test2$RainToday)
test2$RainToday <- as.factor(test2$RainToday)
summary(test2$RainToday)
sum(!complete.cases(test2))
test2$ID <- as.character(test2$ID)

test2_dt <- subset(test2,select=c(1:10))
test2_dt$MinTemp_grp <- ifelse(test2_dt$MinTemp > 0,"aboveZero", "belowZero")
test2_dt$MaxTemp_grp <- ifelse(test2_dt$MaxTemp >= 30,"high", ifelse(test2_dt$MaxTemp >= 15, "middle", "low"))
test2_dt$Rainfall_grp <- ifelse(test2_dt$Rainfall > 0,"Yes", "No")
test2_dt$WS_grp <- ifelse(test2_dt$WindSpeed >= 25,"high", ifelse(test2_dt$WindSpeed >= 15, "middle", "low"))
test2_dt$H_grp <- ifelse(test2_dt$Humidity >= 70,"high", ifelse(test2_dt$Humidity < 40, "low", "middle"))
test2_dt$Temp_grp <- ifelse(test2_dt$Temp >= 30,"high", ifelse(test2_dt$Temp >= 15, "middle", "low"))

test2_dt$MinTemp_grp <- as.factor(test2_dt$MinTemp_grp)
test2_dt$MaxTemp_grp <- as.factor(test2_dt$MaxTemp_grp)
test2_dt$Rainfall_grp <- as.factor(test2_dt$Rainfall_grp)
test2_dt$WS_grp <- as.factor(test2_dt$WS_grp)
test2_dt$H_grp <- as.factor(test2_dt$H_grp)
test2_dt$Temp_grp <- as.factor(test2_dt$Temp_grp)
test3 <- scale(test2[complete.cases(test2), sapply(test2, is.numeric)], center = T, scale = T)
```

7. Exploratory Data Analysis (EDA) for training data

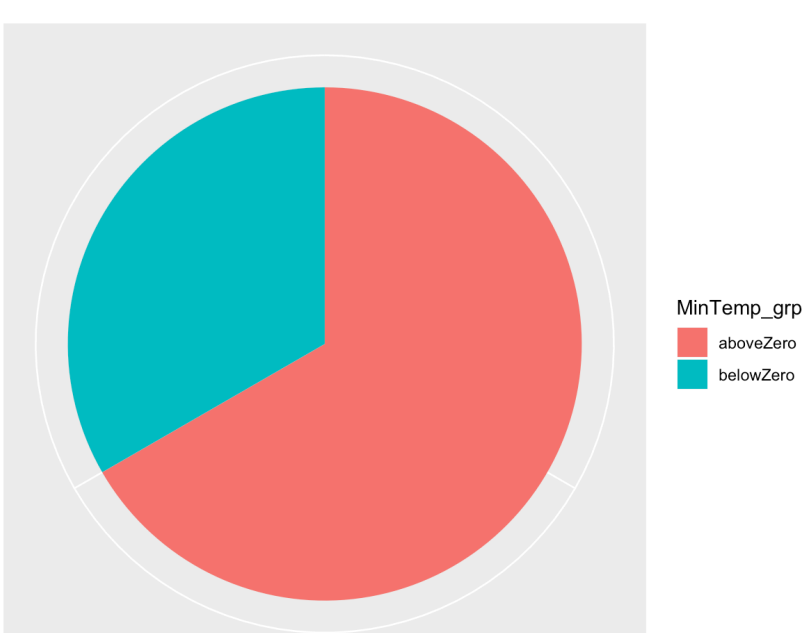
- Pie-chart: Rainfall group

```
data2 %>%  group_by(Rainfall_grp) %>%
  summarise(pct = percent(n()/nrow(data2))) %>%
  ggplot(aes(x = factor(1), y = pct, fill = Rainfall_grp)) +
  geom_bar(stat = "identity") +
  coord_polar(theta = "y") +
  theme(axis.ticks = element_blank(),
        axis.text = element_blank(),
        axis.title = element_blank())
```



- Pie-chart: Minimum temperature group

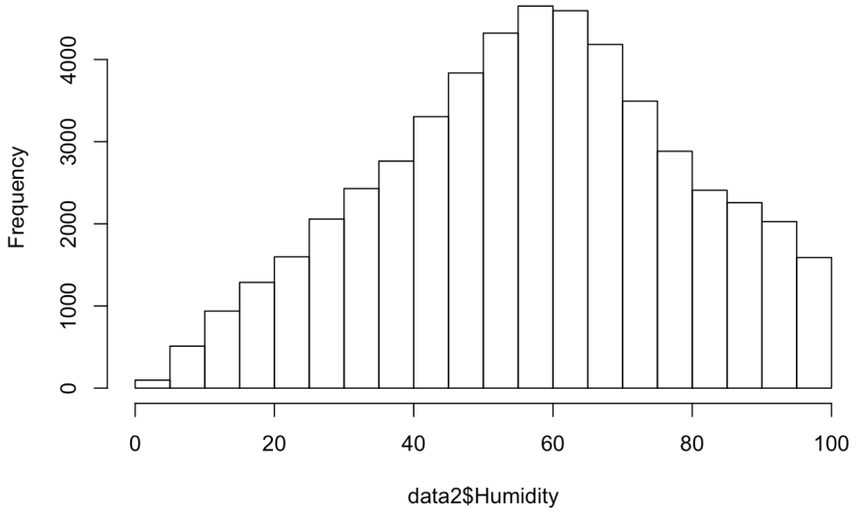
```
data2 %>%  group_by(MinTemp_grp) %>%
  summarise(pct = percent(n()/nrow(data2))) %>%
  ggplot(aes(x = factor(1), y = pct, fill = MinTemp_grp)) +
  geom_bar(stat = "identity") +
  coord_polar(theta = "y") +
  theme(axis.ticks = element_blank(),
        axis.text = element_blank(),
        axis.title = element_blank())
```



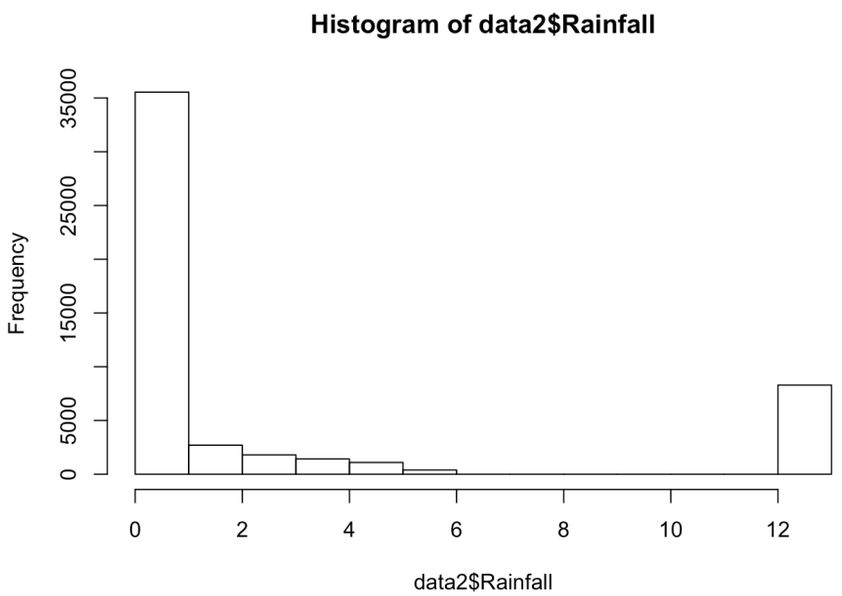
- Histogram: Humidity, rainfall, temperature

```
hist(data2$Humidity)
```

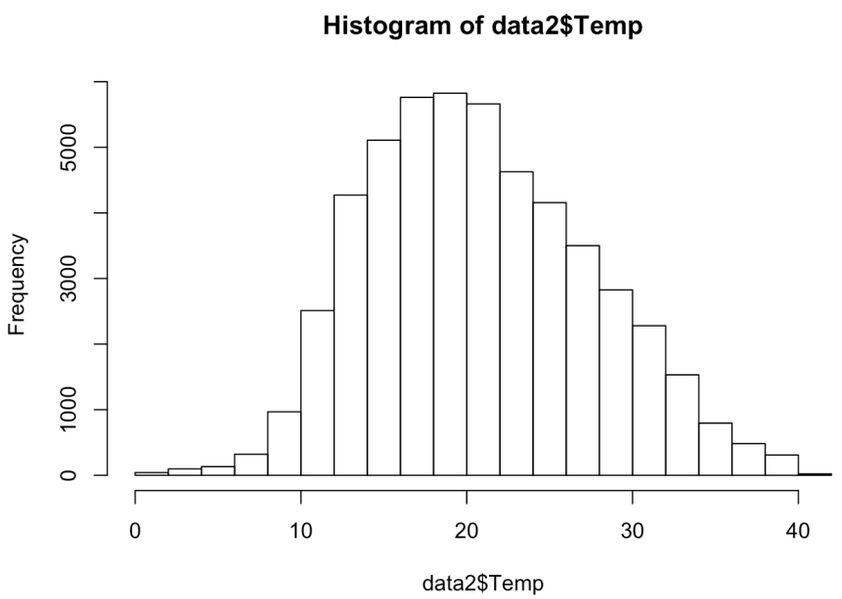
Histogram of data2$Humidity




```
hist(data2$Rainfall)
```

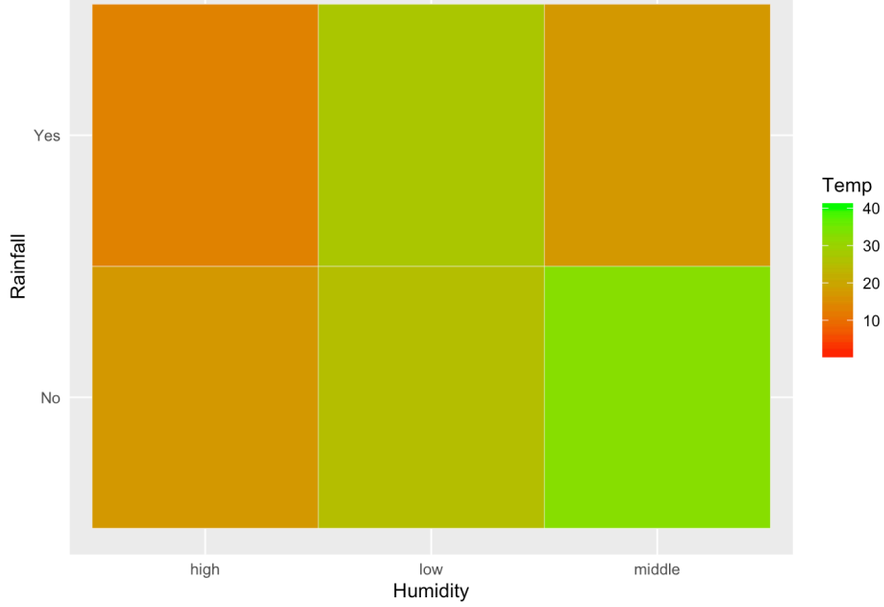



```
hist(data2$Temp)
```



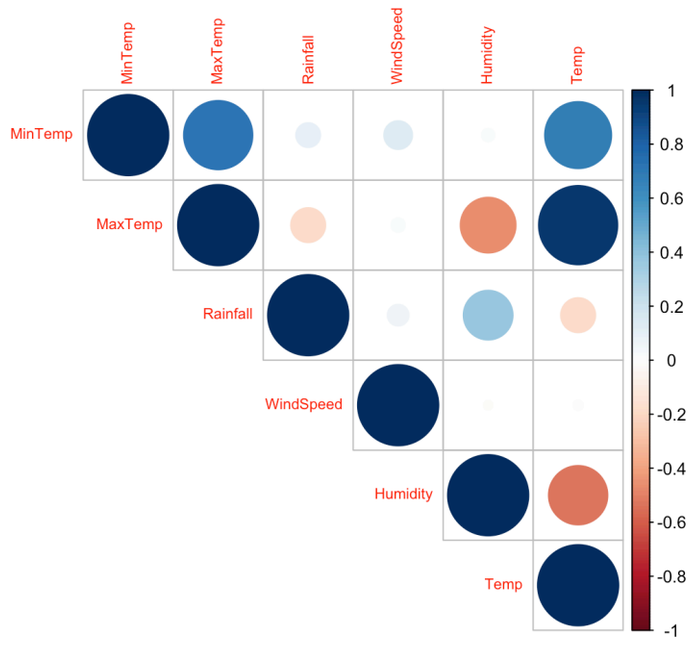
- Heatmap: Rainfall group-Humidity group-Temperature

```
ggplot(data2, aes(x = H_grp, y = Rainfall_grp)) +
  geom_tile(aes(fill = Temp), color = "white") +
  scale_fill_gradient(low = "red", high = "green") +
  theme(axis.ticks = element_blank()) +
  labs(x = "Humidity", y = "Rainfall")
```



- Attribute's Correlation
  - Positive relationship: MinTemp-MaxTemp, Min/MaxTemp-Temp, Rainfall-Humidity/MinTemp, WindSpeed-Rainfall/MinTemp,
  - Negative relationship: Rainfall-MaxTemp/Temp, Humidity-MaxTemp

```
cor_matrix <- cor(data2[complete.cases(data2), sapply(data2, is.numeric)], method = "pearson")
corrplot(cor_matrix, type="upper", tl.cex = 0.7 )
```

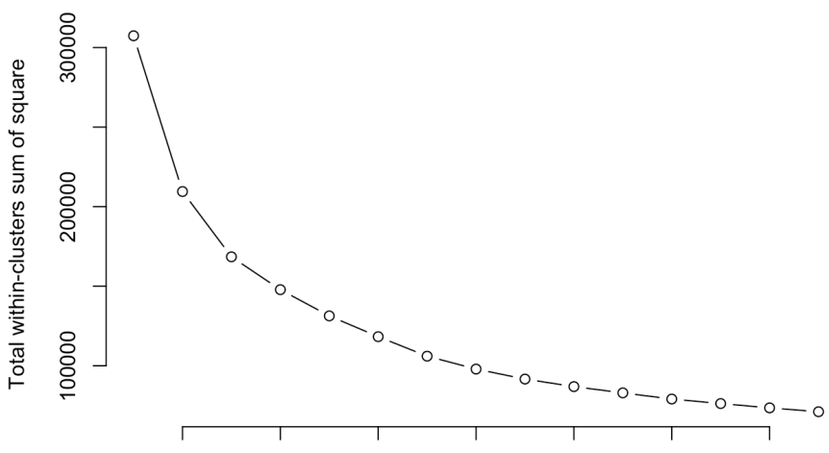


Section 2: Build, tune and evaluate cluster analysis and decision tree models

Section 3. Prediction and interpretation

1. Clustering analysis - kmeans

- Elbow method to decide optimal number of clusters: optimized number of clusters -> 3



2 4 6 8 10 12 14

Number of clusters K

- kmeans function

```
km_output <- kmeans(data3, centers = 2, nstart = 25, iter.max = 100, algorithm = "Hartigan-Wong")
str(km_output)
```

```
List of 9
 $ cluster     : Named int [1:51231] 2 1 1 1 1 1 1 1 2 1 ...
  ..- attr(*, "names")= chr [1:51231] "1" "2" "3" "4" ...
 $ centers     : num [1:2, 1:6] -0.564 0.728 -0.713 0.919 0.163 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:2] "1" "2"
  .. ..$ : chr [1:6] "MinTemp" "MaxTemp" "Rainfall" "WindSpeed" ...
 $ totss       : num 307380
 $ withinss    : num [1:2] 119110 90385
 $ tot.withinss: num 209495
 $ betweenss   : num 97885
 $ size        : int [1:2] 28854 22377
 $ iter        : int 1
 $ ifault      : int 0
 - attr(*, "class")= chr "kmeans"
```

- Hyperparameters' tuning
  - centers: number of clusters
  - iter.max: number of repeats
- Performance evaluation metrics
  - withinss: within cluster sum of squares (the lower, the better)
  - betweenss: between clusters sum of squares (the higher, the better)

```
km1 <- kmeans(data3, centers = 2, nstart = 25, iter.max = 1000, algorithm = "Hartigan-Wong") #no change
str(km1)
```

```
List of 9
 $ cluster     : Named int [1:51231] 2 1 1 1 1 1 1 1 2 1 ...
  ..- attr(*, "names")= chr [1:51231] "1" "2" "3" "4" ...
 $ centers     : num [1:2, 1:6] -0.564 0.728 -0.713 0.919 0.163 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:2] "1" "2"
  .. ..$ : chr [1:6] "MinTemp" "MaxTemp" "Rainfall" "WindSpeed" ...
 $ totss       : num 307380
 $ withinss    : num [1:2] 119110 90385
 $ tot.withinss: num 209495
 $ betweenss   : num 97885
 $ size        : int [1:2] 28854 22377
 $ iter        : int 1
 $ ifault      : int 0
 - attr(*, "class")= chr "kmeans"
```

- km2: increase centers, iter.max -> lower withinss(more condensed), higher betweenss(different clusters are more separated)

```
km2 <- kmeans(data3, centers = 3, nstart = 25, iter.max = 1000, algorithm = "Hartigan-Wong")
str(km2)
```

```
List of 9
 $ cluster     : Named int [1:51231] 2 3 1 3 3 1 3 1 2 3 ...
  ..- attr(*, "names")= chr [1:51231] "1" "2" "3" "4" ...
 $ centers     : num [1:3, 1:6] 0.172 0.734 -0.639 -0.388 0.995 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:3] "1" "2" "3"
  .. ..$ : chr [1:6] "MinTemp" "MaxTemp" "Rainfall" "WindSpeed" ...
 $ totss       : num 307380
 $ withinss    : num [1:3] 31942 65089 71440
 $ tot.withinss: num 168471
 $ betweenss   : num 138909
 $ size        : int [1:3] 8156 19027 24048
 $ iter        : int 4
 $ ifault      : int 0
 - attr(*, "class")= chr "kmeans"
```

- kmeans with PCA : importance of components:PC1, PC2 (low variance)

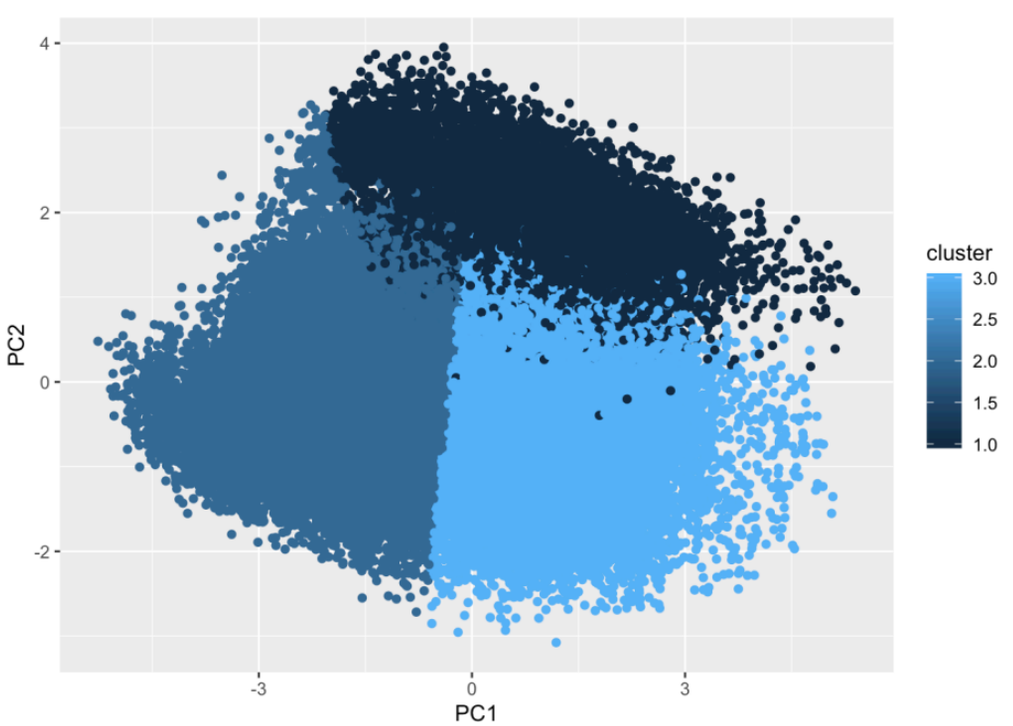
```
pca <- prcomp(data3, scale. = T, center = T)
print(pca)
```

```
Standard deviations (1, .., p=6):
[1] 1.6823815 1.1720558 0.9873376 0.7958376 0.3774660 0.2126133

Rotation (n x k) = (6 x 6):
                  PC1        PC2         PC3          PC4         PC5
MinTemp   -0.43308900 0.48714780 -0.11646418  0.319164253  0.67705028
MaxTemp   -0.57690954 0.07626727 -0.07228799 -0.007604832 -0.39518987
Rainfall   0.16268142 0.64008356 -0.13244488 -0.738064263 -0.02999111
WindSpeed -0.03930512 0.27015067  0.95675660  0.056975412 -0.08226527
Humidity   0.34379846 0.52267148 -0.20914992  0.583847922 -0.47021549
Temp      -0.57740955 0.03156749 -0.06739442 -0.095983830 -0.39580075
                   PC6
MinTemp    0.035974065
MaxTemp   -0.707036254
Rainfall  -0.025452477
WindSpeed  0.008760869
Humidity   0.053781516
Temp       0.703696361
```

```
summary(pca)
```

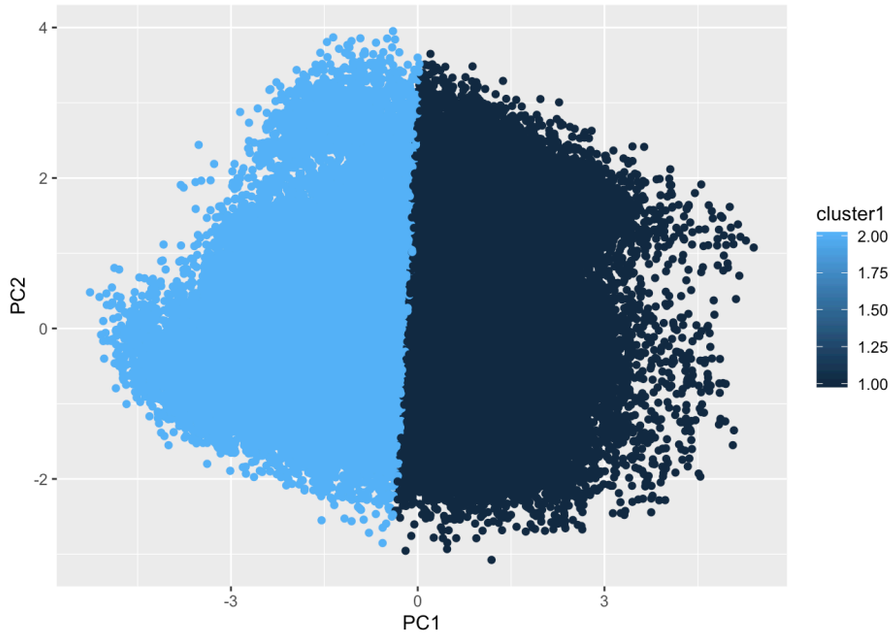
```
Importance of components:
                          PC1    PC2    PC3    PC4      PC5     PC6
Standard deviation     1.6824 1.1721 0.9873 0.7958 0.37747 0.21261
Proportion of Variance 0.4717 0.2289 0.1625 0.1056 0.02375 0.00753
Cumulative Proportion  0.4717 0.7007 0.8632 0.9687 0.99247 1.00000
```

```
pcs <- as.data.frame(predict(pca, newdata = data3))
cluster <- km2$cluster
pcs$cluster <- cluster[match(rownames(pcs), names(cluster))]
qplot(PC1, PC2, colour = cluster, data = pcs)
```



- Repurpose clustering analysis for classification - kmeans
  - The purpose of clustering analysis is grouping unlabeled data. Therefore it does not pay attention to the prediction functionality.
  - However, we can classify sub-datasets of clusters that we defined through clustering analysis.
  - From trainig model, I induced that cluster 1 is related to raining, and cluster 2 is related to not raining.

```
pcs1 <- as.data.frame(predict(pca, newdata = data3))
cluster1 <- km1$cluster
pcs1$cluster1 <- cluster1[match(rownames(pcs1), names(cluster1))]
qplot(PC1, PC2, colour = cluster1, data = pcs1)
```




```
training <- data2[complete.cases(data2), sapply(data2, is.numeric)]
training$cluster <- cluster1
training$RainTomorrow <- data2$RainTomorrow
table(cluster=training$cluster, RainTomorrow=training$RainTomorrow)
```

```
       RainTomorrow
cluster    No   Yes
      1 12439 16415
      2 13826  8551
```

```
training$Rain <- ifelse(training$RainTomorrow=="Yes","1", "2")
training$Rain <- as.numeric(training$Rain)
```

*Based on this assumption, I figured out values for 'RainTomorrow' attribute of training dataset. (km1 model was used since it's a binary question)

```
km_test <- kmeans(test3, centers = 2, nstart = 25, iter.max = 1000, algorithm = "Hartigan-Wong")
```

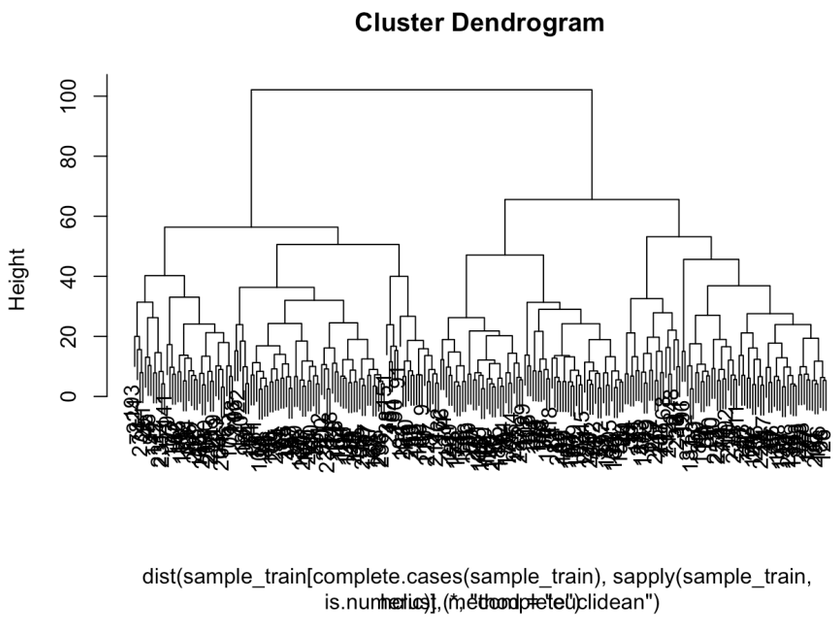
2. Clustering analysis - HAC

- I used random sampling due to repeated errors caused by big size of dataset)

```
sample_train <- data2 %>%
  sample_frac(size = 0.005, replace = FALSE)
```

- Modeling

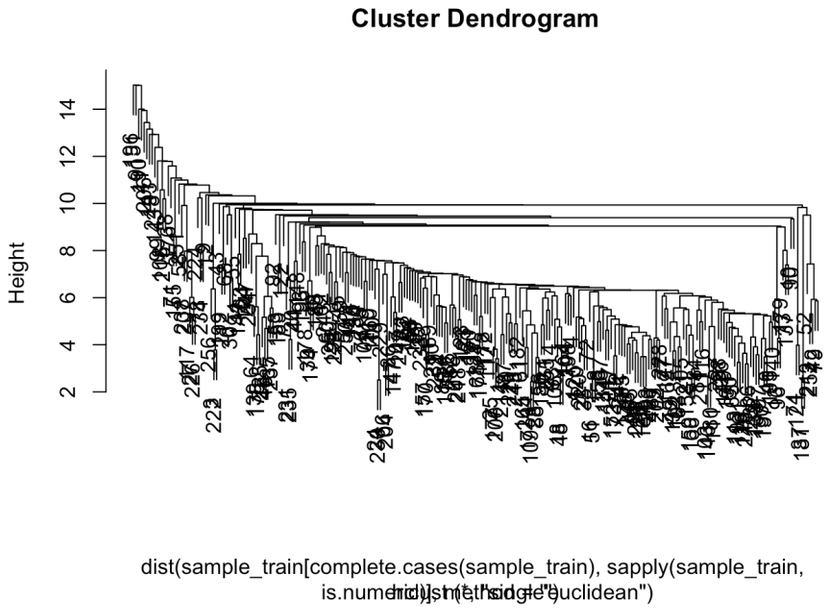
```
hac_output <- hclust(dist(sample_train[complete.cases(sample_train), sapply(sample_train, is.numeric)], method =
"euclidean"), method = "complete")
plot(hac_output)
```



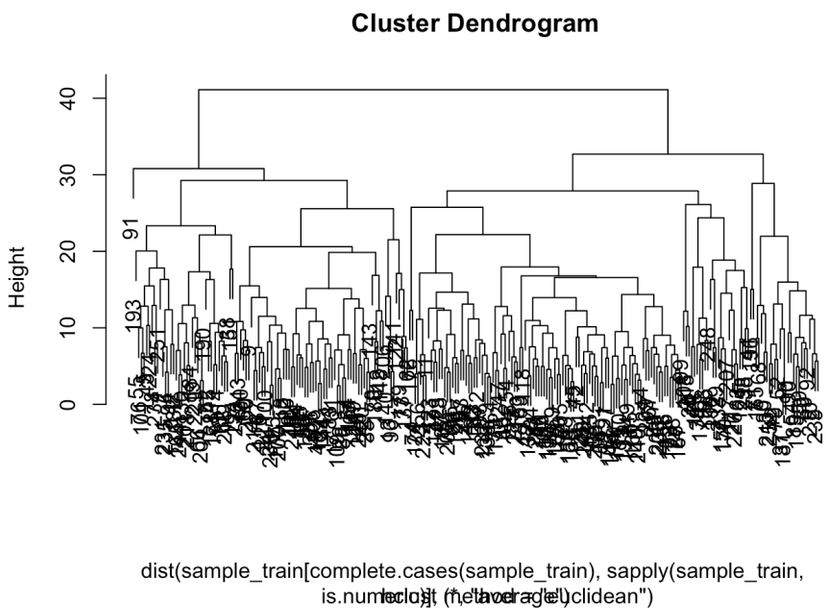
- Hyperparameters' tuning
  - method(intracluster): distance function between data points
  - method(intercluster): definition of intercluster distance
- Performance evaluation metrics
  - single linkage: useful for clusters of different size

- complete linkage: useful for clusters of similar size
- average, centroid linkage: resistant to outliers

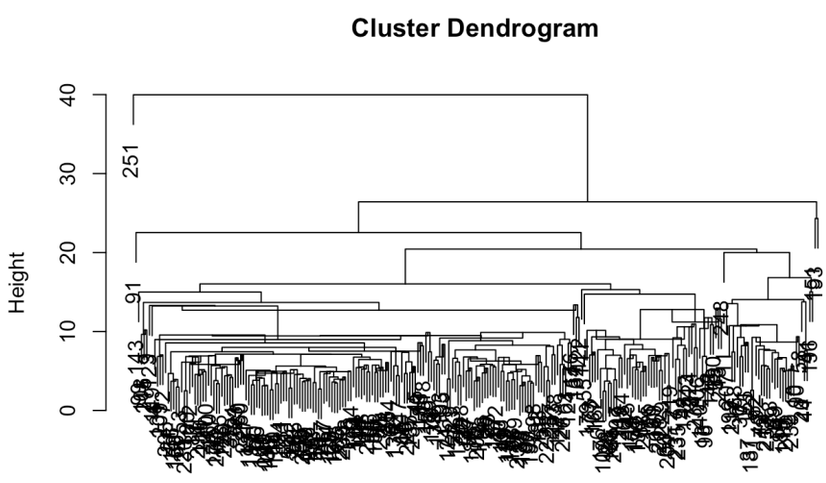
```
hac1 <- hclust(dist(sample_train[complete.cases(sample_train), sapply(sample_train, is.numeric)], method = "eucli
dean"), method = "single")
plot(hac1)
```

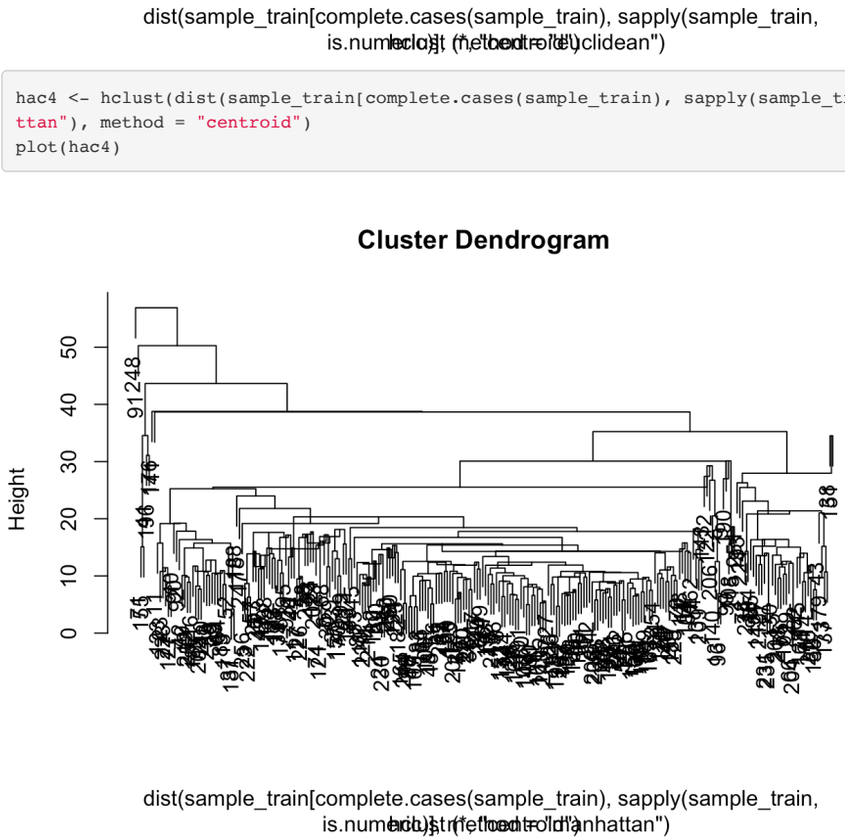



```
hac2 <- hclust(dist(sample_train[complete.cases(sample_train), sapply(sample_train, is.numeric)], method = "eucli
dean"), method = "average")
plot(hac2)
```




```
hac3 <- hclust(dist(sample_train[complete.cases(sample_train), sapply(sample_train, is.numeric)], method = "eucli
dean"), method = "centroid")
plot(hac3)
```

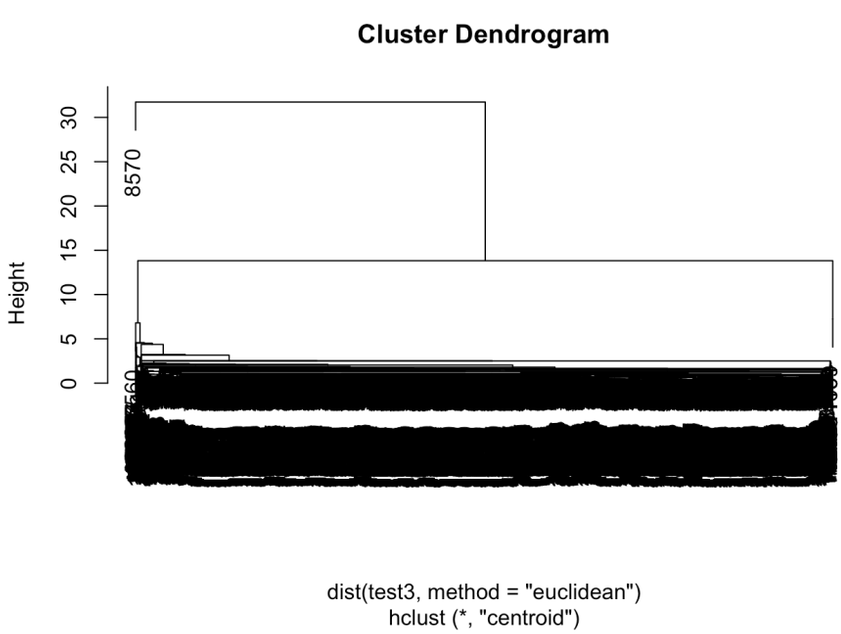





```
hac4 <- hclust(dist(sample_train[complete.cases(sample_train), sapply(sample_train, is.numeric)], method = "manha
ttan"), method = "centroid")
plot(hac4)
```


- Cutting 2 clusters (k=2)
  - Share the same features with kmeans regarding prediction
  - Apply same technique for testing dataset classification
  - From training model, I induced that cluster 1 is related to raining, and cluster 2 is related to not raining.
- Identify subset of the clusters
  - cluster 1: Yes(rain)
  - cluster 2: No(no rain)
- Repurpose clustering analysis for classification - HAC

```
hac_test <- hclust(dist(test3, method = "euclidean"), method = "centroid")
plot(hac_test)
```




```
tree2 <- cutree(hac_test, k=2)
```

```
[1] "c..1....1L...2....1L...3....1L...4....1L...5....1L...6....1L.."
```

3. Decision Tree

- Modeling
  - Modeling with numerical + categorical attributes

```
dt_model <- train(RainTomorrow ~ ., data = data2, metric = "Accuracy", method = "rpart")
print(dt_model)
```

```
CART

51231 samples
   15 predictor
    2 classes: 'No', 'Yes'

No pre-processing
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 51231, 51231, 51231, 51231, 51231, 51231, ...
Resampling results across tuning parameters:

  cp           Accuracy   Kappa
  0.004526156  0.7347512  0.4686600
  0.012917568  0.7294927  0.4572232
  0.429704398  0.6373621  0.2636795

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was cp = 0.004526156.
```

```
print(dt_model$finalModel)
```

```
n= 51231

node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 51231 24966 No (0.5126779 0.4873221)
   2) Humidity< 64.75 31541  9757 No (0.6906566 0.3093434)
     4) Rainfall< 0.35 23829  5910 No (0.7519829 0.2480171) *
     5) Rainfall>=0.35 7712  3847 No (0.5011670 0.4988330)
      10) Humidity< 48.16667 2391   864 No (0.6386449 0.3613551) *
      11) Humidity>=48.16667 5321  2338 Yes (0.4393911 0.5606089) *
   3) Humidity>=64.75 19690  4481 Yes (0.2275775 0.7724225) *
```

- Modeling with categorical attributes only

```
category_train <- data2[complete.cases(data2), sapply(data2, is.factor)]
dt_model2 <- train(RainTomorrow ~ ., data = category_train, metric = "Accuracy", method = "rpart")
print(dt_model2)
```

```
CART

51231 samples
    9 predictor
    2 classes: 'No', 'Yes'

No pre-processing
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 51231, 51231, 51231, 51231, 51231, 51231, ...
Resampling results across tuning parameters:

  cp          Accuracy   Kappa
  0.01619536  0.7231416  0.4454688
  0.03747096  0.7090149  0.4174651
  0.32872707  0.5637326  0.1095065

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was cp = 0.01619536.
```

- Hyperparameters' tuning
  - minsplit: the minimum number of observations that must exist in a node
  - minbucket: the minimum number of observations in any terminal node
  - maxdepth: maximum depth of any any node of the final tree
  - cp: model complexity
  - To make unbiased and low-variance model: minsplit, minbucket,cp should be high, and maxdepth should be low
  - However, bias and complexity have tradeoffs. For example, as cp(complexity) gets higher, accuracy and kappa(bias) gets lower. Therefore, we need to find the balanced model.
- Performance evaluation method
  - Confusion matrix, Cross-validation, ROC curve, Bootstrapping
- Performance evaluation metrics
  - accuracy, kappa, precision, recall, F-measure, AUC(area under curve) : high in good models

```
dt_model3 <- train(RainTomorrow ~ ., data = data2, method = "rpart",
                   metric = "Accuracy",
                   tuneLength = 8,
                   control = rpart.control(minsplit = 10, minbucket = 10, maxdepth = 3))
dt_model3
```

```
CART

51231 samples
   15 predictor
    2 classes: 'No', 'Yes'

No pre-processing
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 51231, 51231, 51231, 51231, 51231, 51231, ...
Resampling results across tuning parameters:

  cp            Accuracy   Kappa
  0.0008277925  0.7333955  0.4661131
  0.0008411440  0.7333955  0.4661131
  0.0009212529  0.7333955  0.4661131
  0.0010213891  0.7333955  0.4661131
  0.0043258832  0.7333955  0.4661131
  0.0045261556  0.7333955  0.4661131
  0.0129175679  0.7276679  0.4532538
  0.4297043980  0.6880993  0.3696306

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was cp = 0.004526156.
```

```
tr_control <- trainControl(method = "cv", number = 3)
dt_model_cv <- train(RainTomorrow ~ .,
                     data = data2, method = "rpart", metric = "Accuracy",
                     control = rpart.control(minsplit = 50, minbucket = 20,
                                             maxdepth = 5, cp = 0.01),
                     trControl = tr_control, na.action = na.omit)
print(dt_model_cv)
```

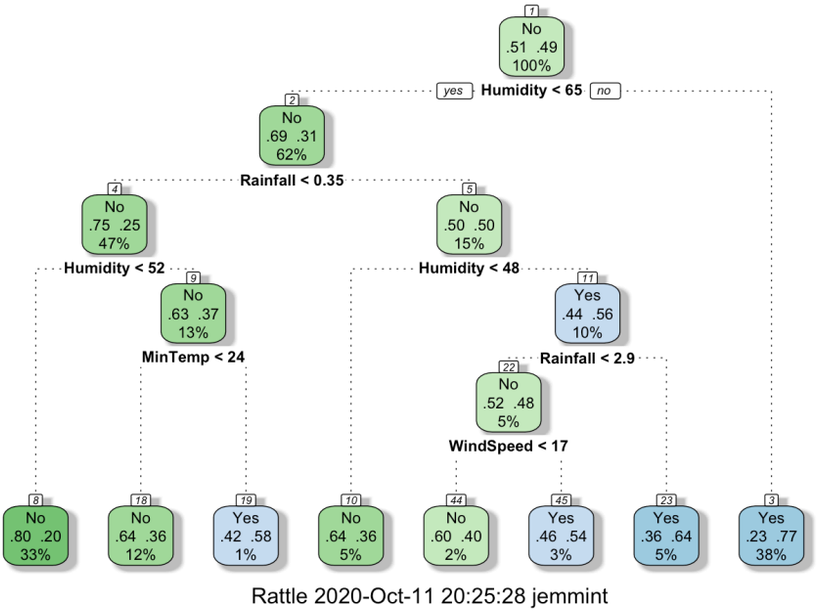
```
CART

51231 samples
   15 predictor
    2 classes: 'No', 'Yes'

No pre-processing
Resampling: Cross-Validated (3 fold)
Summary of sample sizes: 34154, 34154, 34154
Resampling results across tuning parameters:

  cp           Accuracy  Kappa
  0.004526156  0.735785  0.4700742
  0.012917568  0.730788  0.4595382
  0.429704398  0.650973  0.2913318

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was cp = 0.004526156.
```

- Decision Tree model visualization

```
dt_model_roc <- train(RainTomorrow ~ ., data = data2, method = "rpart", metric = "ROC",
                     trControl = trainControl(method = "cv", number = 5, classProbs = T,
                                              summaryFunction = twoClassSummary),
                     tuneGrid = expand.grid(cp = seq(0, 0.01, 0.001)),
                     control = rpart.control(minsplit = 3, minbucket = 1))
print(dt_model_roc)
```

```
CART

51231 samples
   15 predictor
    2 classes: 'No', 'Yes'

No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 40985, 40984, 40985, 40985, 40985
Resampling results across tuning parameters:

  cp     ROC        Sens       Spec
  0.000  0.6838866  0.7202361  0.6818468
  0.001  0.7842282  0.7893394  0.6887765
  0.002  0.7564536  0.7685132  0.7051192
  0.003  0.7562698  0.7676375  0.7037966
  0.004  0.7559405  0.7586141  0.7097249
  0.005  0.7557571  0.7543118  0.7127692
  0.006  0.7556113  0.7442223  0.7232639
  0.007  0.7556113  0.7442223  0.7232639
  0.008  0.7556113  0.7442223  0.7232639
  0.009  0.7556113  0.7442223  0.7232639
  0.010  0.7556113  0.7442223  0.7232639

ROC was used to select the optimal model using the largest value.
The final value used for the model was cp = 0.001.
```

```
fancyRpartPlot(dt_model_roc$finalModel)
```



- Decision Tree model interpretation
  - Humidity, rainfall, and wind speed are major features related to predict whether it will rain or not tomorrow.
  - It is more likely to rain when humidity is higher than 65%.
  - Wind speed higher than 17km/hr, and the amount of rainfall for the day over 2.9mm contribute to rain for tomorrow.
  - If the amount of rainfall for the day is less than 0.35mm, humidity is lower than 52mm, and minimum temperature is lower than 24C, it is less likely to rain tomorrow.
- Prediction on testing data

```
dt_predict <- predict(dt_model_roc, newdata = test2_dt, type = "raw")
```

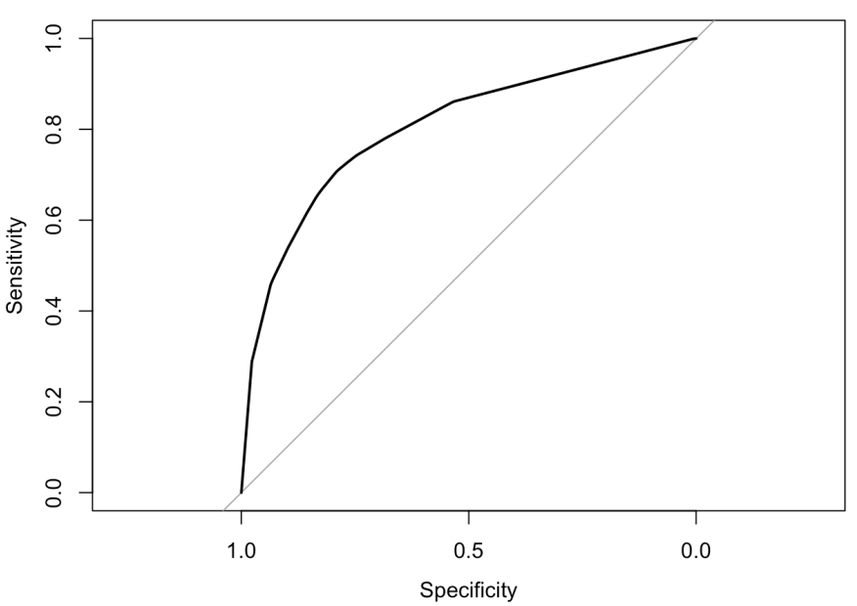
```
[1] "structure.c.2L..1L..1L..1L..1L..2L..1L..2L..1L..1L..2L..1L..2L.."
```

- Performance evaluation
  - ROC curve and AUC (ability of a classifier to distinguish between classes-Yes/No)
  - AUC(training):0.805
  - AUC(testing) 1 (As the training dataset does not have actual value to compare, the predicted values themselves are used for validation)

```
dt_plot <- train(RainTomorrow ~ ., data = data2, method = "rpart", tuneLength = 10)
dt_pred_prob0 <- predict(dt_plot, newdata = data2, type = "prob")
head(dt_pred_prob0, n = 5)
```

```
          No       Yes
1 0.66078431 0.3392157
2 0.80040828 0.1995917
3 0.07774078 0.9222592
4 0.66078431 0.3392157
5 0.80040828 0.1995917
```

```
roc_curve0 <- roc(data2$RainTomorrow,dt_pred_prob0$Yes)
plot(roc_curve0)
```



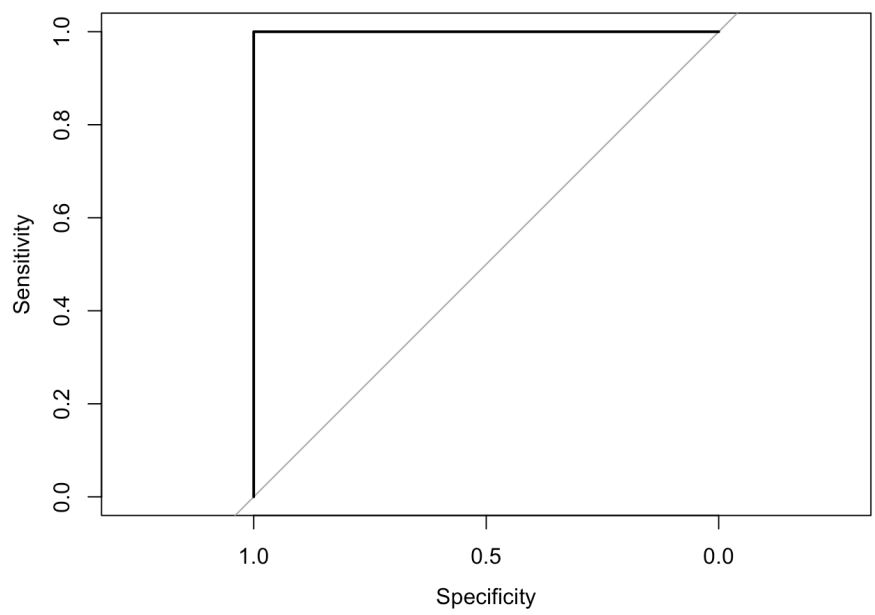

```
auc(roc_curve0)
```

```
Area under the curve: 0.805
```

```
dt_pred_prob <- predict(dt_model_roc, newdata = test2_dt, type = "prob")
head(dt_pred_prob, n = 5)
```

```
         No       Yes
1 0.2275775 0.7724225
2 0.6386449 0.3613551
3 0.8004083 0.1995917
4 0.8004083 0.1995917
5 0.8004083 0.1995917
```

```
roc_curve <- roc(test2_dt$RainTomorrow,dt_pred_prob$Yes)
plot(roc_curve)
```




```
auc(roc_curve)
```

```
Area under the curve: 1
```

- Confusion matrix

```
prop.table(table(dt_predict, test2_dt$RainTomorrow), margin = NULL)
```

```
dt_predict        No       Yes
       No  0.5250526 0.0000000
       Yes 0.0000000 0.4749474
```

```
confusionMatrix(dt_predict, test2_dt$RainTomorrow)
```

```
Confusion Matrix and Statistics

          Reference
Prediction   No  Yes
       No  6738    0
       Yes    0 6095

               Accuracy : 1
                 95% CI : (0.9997, 1)
    No Information Rate : 0.5251
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 1

 Mcnemar's Test P-Value : NA

            Sensitivity : 1.0000
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 1.0000
             Prevalence : 0.5251
         Detection Rate : 0.5251
   Detection Prevalence : 0.5251
      Balanced Accuracy : 1.0000

       'Positive' Class : No
```

- Precision, recall, F-measure

```
precision <- posPredValue(dt_predict, test2_dt$RainTomorrow, positive = "yes")
recall <- sensitivity(dt_predict, test2_dt$RainTomorrow, positive = "yes")
f <- 2 * precision * recall / (precision + recall)
sprintf("Precision is %.2f; recall is %.2f; F measure if %.2f",
        precision, recall, f)
```

```
[1] "Precision is NA; recall is NA; F measure if NA"
```

4. Best models & parameters

Type	Model ID	Model specifications
kmeans	km2	centers=3, nstart=25, iter.max=1000, low withinss(condensed), high betweenss(clusters are separated)
HAC	hac1	method(intracluster)=euclidean, method(intercluster)=single, useful for clusters of different size
DT	dt_model_cv	minsplit=50, minbucket=10, maxdepth=5, cp=0.01. highest accuracy and kappa