Yeast Data Analysis

Kerry Chu

Data Exploration

Read Data

```
#get working directory
wd<-getwd()
#set working directory
setwd(wd)
#read yeast data from working directory
yeast <-read.table("./yeast.data", sep = "", header = FALSE)
#Change the column name of the dataset according to the reference list
names(yeast) <- c("Sequence.Name", "mcg","gvh","alm","mit","erl","pox","vac","nuc","Localisation.Site")
```

70/30

Split Data

```
#Generate random data
set.seed(1234)
#split data into 70% 30%
sep1 <- sample(2, nrow(yeast), replace=TRUE, prob=c(0.7, 0.3))
train_data1 <- yeast[sep1==1,]
test_data1 <- yeast[sep1==2,]
```

Prediction

```
#Load party, the pacakge for partitioning
library(party)
```

```
## Loading required package: grid
## Loading required package: mvtnorm
## Loading required package: modeltools
## Loading required package: stats4
## Loading required package: strucchange
## Loading required package: zoo
##
## Attaching package: 'zoo'
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
## Loading required package: sandwich
```

```
#set the formula
formula <-Localisation.Site~mcg+gvh+alm+mit+erl+pox+vac+nuc
#set training data
yeast_ctree1<-ctree(formula, data=train_data1)
#prediction
table(predict(yeast_ctree1, newdata = test_data1), test_data1$Localisation.Site)
```

```
##
##       CYT ERL EXC ME1 ME2 ME3 MIT NUC POX VAC
##   CYT  69   0   0   0   2   1  16  23   2   3
##   ERL   0   0   0   0   0   0   0   0   0   0
##   EXC   1   3   9   1   2   0   1   0   1   0
##   ME1   0   0   1   7   2   0   1   0   0   0
##   ME2   0   1   0   0   7   0   3   0   0   0
##   ME3   2   0   0   1   2  42   7   4   0   4
##   MIT  10   0   0   0   0   0  52  10   1   0
##   NUC  56   1   1   0   0   3   8  67   0   2
##   POX   0   0   0   0   0   0   0   0   3   0
##   VAC   0   0   0   0   0   0   0   0   0   0
```

Confusion Matrix

```
#load caret, the package for classification and regression training
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
#confusion matrix
con1<-confusionMatrix(predict(yeast_ctree1, newdata = test_data1), test_data1$Localisation.Site)
con1
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction CYT ERL EXC ME1 ME2 ME3 MIT NUC POX VAC
##        CYT  69   0   0   0   2   1  16  23   2   3
##        ERL   0   0   0   0   0   0   0   0   0   0
##        EXC   1   3   9   1   2   0   1   0   1   0
##        ME1   0   0   1   7   2   0   1   0   0   0
##        ME2   0   1   0   0   7   0   3   0   0   0
##        ME3   2   0   0   1   2  42   7   4   0   4
##        MIT  10   0   0   0   0   0  52  10   1   0
##        NUC  56   1   1   0   0   3   8  67   0   2
##        POX   0   0   0   0   0   0   0   0   3   0
##        VAC   0   0   0   0   0   0   0   0   0   0
##
## Overall Statistics
##
##                Accuracy : 0.5926
##                  95% CI : (0.5446, 0.6393)
##     No Information Rate : 0.3194
```

```
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.481
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: CYT Class: ERL Class: EXC Class: ME1
## Sensitivity              0.5000    0.00000    0.81818    0.77778
## Specificity              0.8401    1.00000    0.97862    0.99054
## Pos Pred Value           0.5948        NaN    0.50000    0.63636
## Neg Pred Value           0.7816    0.98843    0.99517    0.99525
## Prevalence               0.3194    0.01157    0.02546    0.02083
## Detection Rate           0.1597    0.00000    0.02083    0.01620
## Detection Prevalence     0.2685    0.00000    0.04167    0.02546
## Balanced Accuracy        0.6701    0.50000    0.89840    0.88416
##                      Class: ME2 Class: ME3 Class: MIT Class: NUC
## Sensitivity             0.46667    0.91304     0.5909     0.6442
## Specificity             0.99041    0.94819     0.9390     0.7835
## Pos Pred Value          0.63636    0.67742     0.7123     0.4855
## Neg Pred Value          0.98100    0.98919     0.8997     0.8741
## Prevalence              0.03472    0.10648     0.2037     0.2407
## Detection Rate          0.01620    0.09722     0.1204     0.1551
## Detection Prevalence    0.02546    0.14352     0.1690     0.3194
## Balanced Accuracy       0.72854    0.93062     0.7649     0.7139
##                      Class: POX Class: VAC
## Sensitivity            0.428571    0.00000
## Specificity            1.000000    1.00000
## Pos Pred Value         1.000000        NaN
## Neg Pred Value         0.990676    0.97917
## Prevalence             0.016204    0.02083
## Detection Rate         0.006944    0.00000
## Detection Prevalence   0.006944    0.00000
## Balanced Accuracy      0.714286    0.50000
```

80/20

```
#generate random data
set.seed(1234)
#split data into 80% 20%
sep2 <- sample(2, nrow(yeast), replace=TRUE, prob=c(0.8, 0.2))
train_data2 <- yeast[sep2==1,]
test_data2 <- yeast[sep2==2,]
```

Prediction

```
#set training data
yeast_ctree2<-ctree(formula, data=train_data2)
#prediction
table(predict(yeast_ctree2, newdata = test_data2), test_data2$Localisation.Site)
```

```
##
```

```
##        CYT ERL EXC ME1 ME2 ME3 MIT NUC POX VAC
##   CYT  46   0   1   0   1   1  23  12   1   1
##   ERL   0   0   0   0   0   0   0   0   0   0
##   EXC   1   2   4   1   1   0   0   0   1   0
##   ME1   0   0   1   5   2   0   0   0   0   0
##   ME2   0   0   0   0   7   0   2   0   0   0
##   ME3   0   0   0   0   1  31   5   1   0   1
##   MIT   3   0   0   0   0   0  29   2   0   0
##   NUC  44   1   0   0   0   1   9  49   1   2
##   POX   0   0   0   0   0   0   0   0   2   0
##   VAC   0   0   2   0   1   0   1   0   0   0
```

Confusion Matrix

```
#confusion matrix
con2<-confusionMatrix(predict(yeast_ctree2, newdata = test_data2), test_data2$Localisation.Site)
con2
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction CYT ERL EXC ME1 ME2 ME3 MIT NUC POX VAC
##        CYT  46   0   1   0   1   1  23  12   1   1
##        ERL   0   0   0   0   0   0   0   0   0   0
##        EXC   1   2   4   1   1   0   0   0   1   0
##        ME1   0   0   1   5   2   0   0   0   0   0
##        ME2   0   0   0   0   7   0   2   0   0   0
##        ME3   0   0   0   0   1  31   5   1   0   1
##        MIT   3   0   0   0   0   0  29   2   0   0
##        NUC  44   1   0   0   0   1   9  49   1   2
##        POX   0   0   0   0   0   0   0   0   2   0
##        VAC   0   0   2   0   1   0   1   0   0   0
##
## Overall Statistics
##
##                Accuracy : 0.5786
##                  95% CI : (0.5204, 0.6352)
##     No Information Rate : 0.3144
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4661
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: CYT Class: ERL Class: EXC Class: ME1
## Sensitivity              0.4894    0.00000    0.50000    0.83333
## Specificity              0.8049    1.00000    0.97938    0.98976
## Pos Pred Value           0.5349        NaN    0.40000    0.62500
## Neg Pred Value           0.7746    0.98997    0.98616    0.99656
## Prevalence               0.3144    0.01003    0.02676    0.02007
## Detection Rate           0.1538    0.00000    0.01338    0.01672
## Detection Prevalence     0.2876    0.00000    0.03344    0.02676
```

```
## Balanced Accuracy         0.6471    0.50000    0.73969    0.91155
##                       Class: ME2 Class: ME3 Class: MIT Class: NUC
## Sensitivity              0.53846     0.9394    0.42029     0.7656
## Specificity              0.99301     0.9699    0.97826     0.7532
## Pos Pred Value           0.77778     0.7949    0.85294     0.4579
## Neg Pred Value           0.97931     0.9923    0.84906     0.9219
## Prevalence               0.04348     0.1104    0.23077     0.2140
## Detection Rate           0.02341     0.1037    0.09699     0.1639
## Detection Prevalence     0.03010     0.1304    0.11371     0.3579
## Balanced Accuracy        0.76573     0.9547    0.69928     0.7594
##                       Class: POX Class: VAC
## Sensitivity             0.400000    0.00000
## Specificity             1.000000    0.98644
## Pos Pred Value          1.000000    0.00000
## Neg Pred Value          0.989899    0.98644
## Prevalence              0.016722    0.01338
## Detection Rate          0.006689    0.00000
## Detection Prevalence    0.006689    0.01338
## Balanced Accuracy       0.700000    0.49322
```

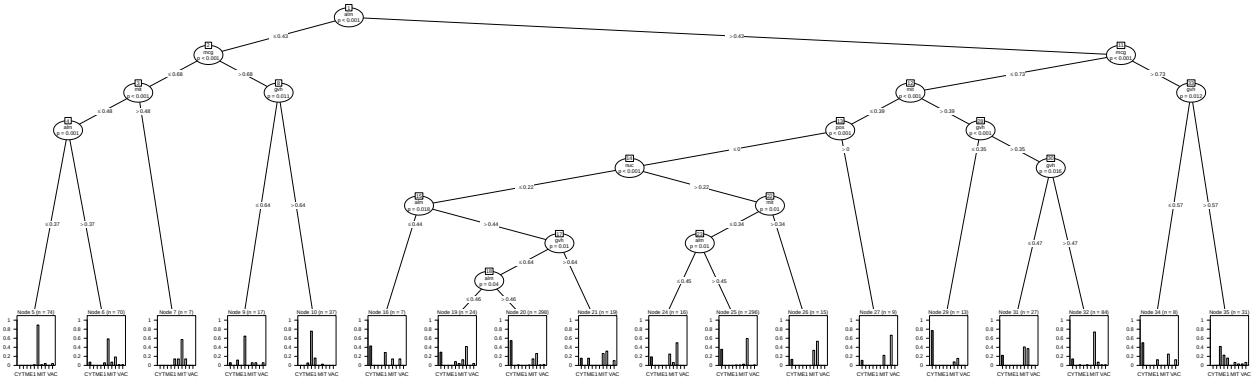
It can be seen that the 80/20 split's accuarcy is lower than that of 70/30.

(For detailed explanation please see Task 2 and Task 3 Q1 & Q3)

Visualisation

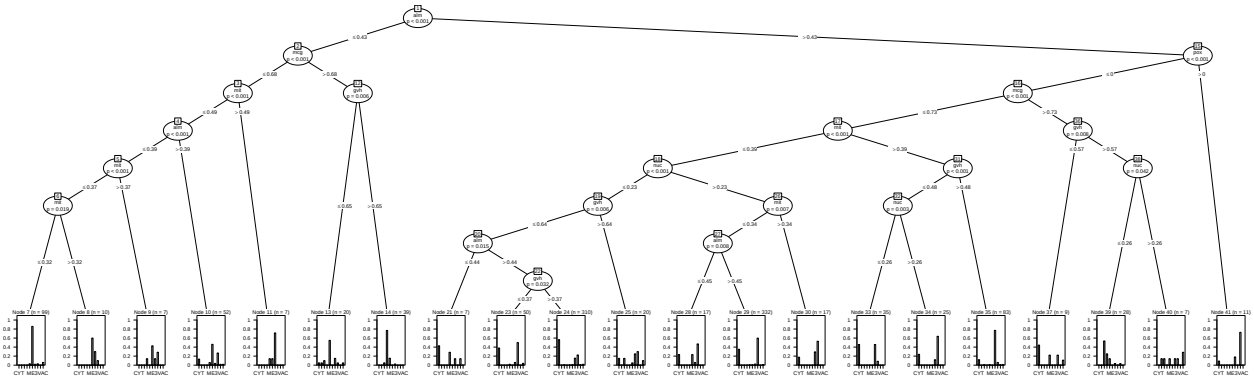
Plot(70/30)

```
plot(yeast_ctree1)
```



##Plot(80/20)

```
plot(yeast_ctree2)
```

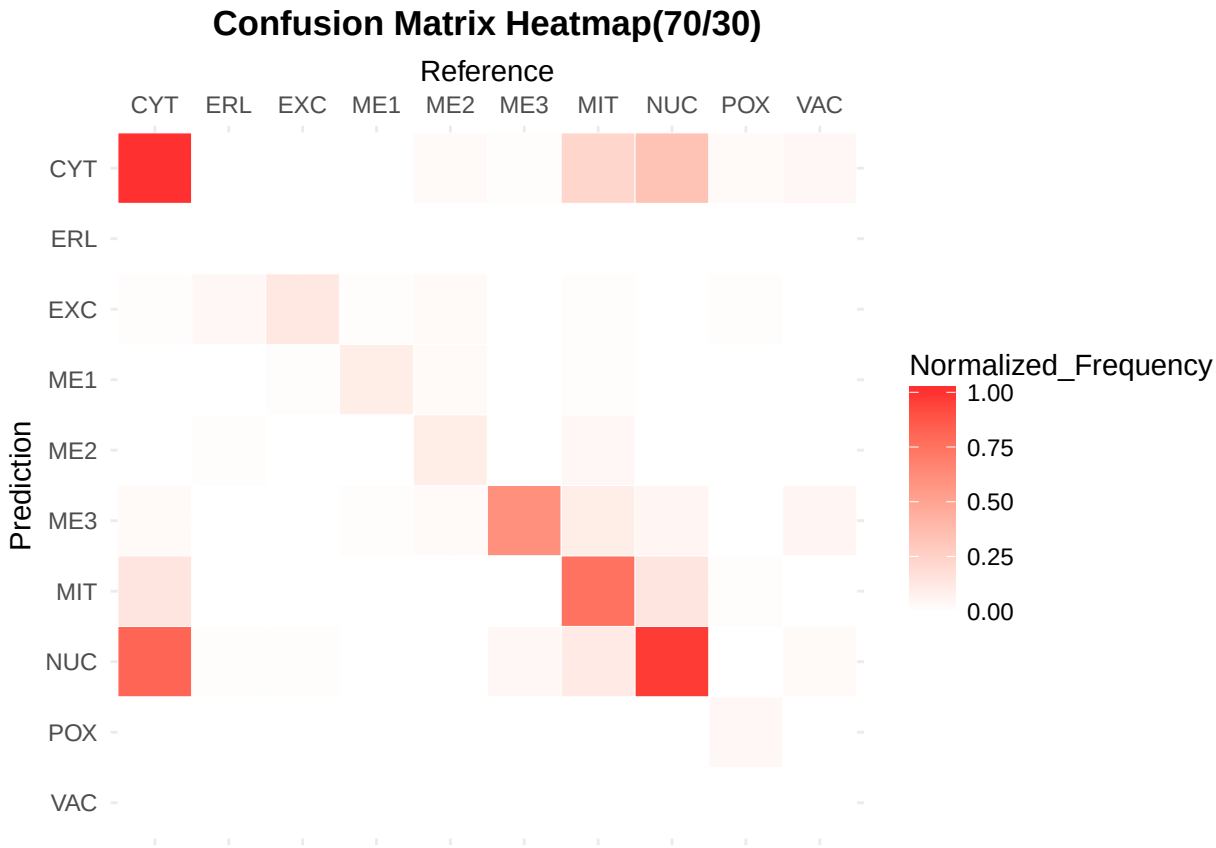


In general, the 80/20 split, which as 20 nodes, has more nodes than 70/30 split, which only 17 nodes. The difference in node numbers unavoidably leads to the difference of final leaf numbers. Moreover, some values in the nodes vary from each other in the 70/30 split and 80/20 split. The difference is a reflection of the statistics in the confusion matrix which can be caused by various factors. *(Please see Task 3 Question 1&3 for details)*

Heatmap(70/30)

```
#convert result of confusion matrix (class) into data frame
n1<-as.data.frame(con1$table)
#normalize the Frequency of n1 to between 0 and 1 by using min-max normalization
n1$Normalized_Frequency<-(n1$Freq-min(n1$Freq))/(max(n1$Freq)-min(n1$Freq))*(1-0)+0
#rename the column
names(n1)<-c("Prediction","Reference","Frequency","Normalized_Frequency")
```

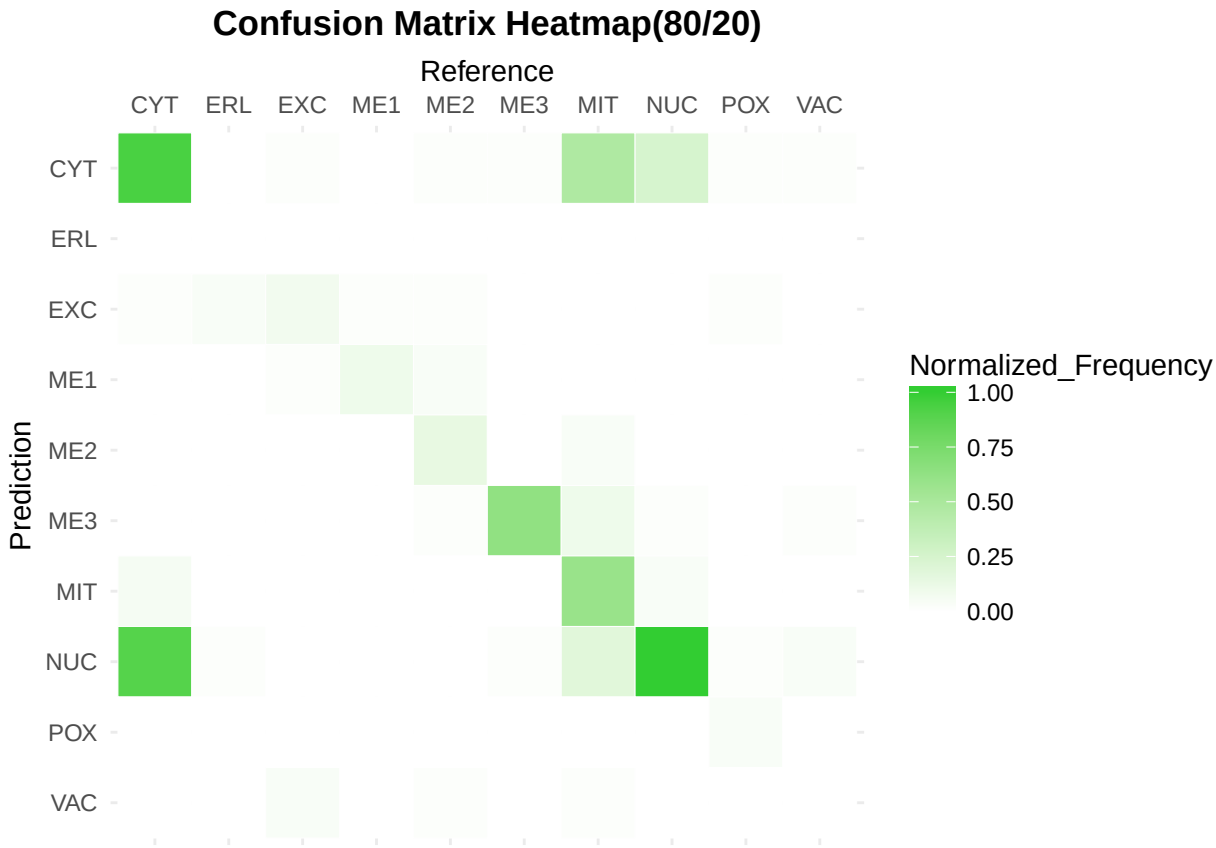
```
library(ggplot2)
library(reshape2)
#reverse the order of y axis (Prediction)
n1$Prediction<-with(n1,factor(Prediction,levels = rev(levels(Prediction))))
#Plot the data
ggplot(aes(Reference, Prediction), data=n1)+geom_tile(aes(fill=Normalized_Frequency), color="White",data
```



Heatmap(80/20)

```
#convert result of confusion matrix (class) into data frame
n2<-as.data.frame(con2$table)
#normalize the Frequency of n2 to between 0 and 1 by using min-max normalization
n2$Normalized_Frequency<-(n2$Freq-min(n2$Freq))/(max(n2$Freq)-min(n2$Freq))*(1-0)+0
#rename the column
names(n2)<-c("Prediction","Reference","Frequency","Normalized_Frequency")
```

```
#reverse the order of y axis (Prediction)
n2$Prediction<-with(n2,factor(Prediction,levels = rev(levels(Prediction))))
#Plot the data
ggplot(aes(Reference, Prediction), data=n2)+geom_tile(aes(fill=Normalized_Frequency), color="White",data
```



Data Analysis

Confusion Matrix Interpretation

Confusion Matrix 70/30

```
#save the confusion matrix table as a data frame but keep the orginal format for the convenience of ana
cm1<-as.data.frame.matrix(con1$table)
#count the total of FP and TP
cm1$Total1 <-rowSums(cm1)
#Use for loop to calculate the incidence of TP and FP
for (i in 1:10)
{
  cm1$TP1[i]<-cm1[i,i]
  cm1$FP1[i]<-cm1$Total1[i]-cm1[i,i]
}
#Use for loop to calculate the precision
for (i in 1:10)
{
  if(cm1$Total1[i]==0)
  {
    cm1$Precision1[i]=0
  }
  else
  {
```

```
      cm1$Precision1[i]<-cm1$TP1[i]/cm1$Total1[i]
  }
}
#subset useful columns for later comparision
acm1<-subset(cm1,select=c(TP1,FP1,Total1,Precision1))
```

Confusion Matrix 80/20

```
#save the confusion matrix table as a data frame but keep the orginal format for the convenience of ana
cm2<-as.data.frame.matrix(con2$table)
#count the total of FP and TP
cm2$Total2 <-rowSums(cm2)
#Use for loop to calculate the incidence of TP and FP
for (i in 1:10)
{
  cm2$TP2[i]<-cm2[i,i]
  cm2$FP2[i]<-cm2$Total2[i]-cm2[i,i]
}
#Use for loop to calculate the precision
for (i in 1:10)
{
  if(cm2$Total2[i]==0)
  {
    cm2$Precision2[i]=0
  }
  else
  {
      cm2$Precision2[i]<-cm2$TP2[i]/cm2$Total2[i]
  }
}
#subset useful columns for later comparision
acm2<-subset(cm2,select=c(TP2,FP2,Total2,Precision2))
```

```
#show comparison results
compare_con<-cbind(acm1,acm2)
compare_con
```

```
##     TP1 FP1 Total1 Precision1 TP2 FP2 Total2 Precision2
## CYT  69  47    116  0.5948276  46  40     86  0.5348837
## ERL   0   0      0  0.0000000   0   0      0  0.0000000
## EXC   9   9     18  0.5000000   4   6     10  0.4000000
## ME1   7   4     11  0.6363636   5   3      8  0.6250000
## ME2   7   4     11  0.6363636   7   2      9  0.7777778
## ME3  42  20     62  0.6774194  31   8     39  0.7948718
## MIT  52  21     73  0.7123288  29   5     34  0.8529412
## NUC  67  71    138  0.4855072  49  58    107  0.4579439
## POX   3   0      3  1.0000000   2   0      2  1.0000000
## VAC   0   0      0  0.0000000   0   4      4  0.0000000
```

Generally, the **Accuracy** of 70/30 split(0.5926) is higher than 80/20 split (0.5786) as it is shown in Task 1. The table above shows a detailed comparison between the two splits.

The total number of TP and FP of each independent variables varies between the two splits because of the difference of the test data. Also because of the size of test data, the number of FP and TP of 70/30 is significantly higher than 80/20. That's why the author thinks that a comparison of precision is needed. Overall, most of the precisions of independent classes of 70/30 are slightly higher than those of 80/20.

However, the 80/20 precisions of some independent variables are actually higher than 70/30, for example, ME2, ME3, etc.

(Please see task 3 Q3 to see the explanation of the cause.)

Classifier Effectiveness Analysis

When looking at the yeast dataset for the first time, only two columns are worth of considering: the first column—**sequence number** and the last column—**Localisation Site**. The rest of columns are the result of different signal measurements of the cell or scores of discriminant analysis of different proteins.

The choice between the two potential classifiers is obvious even for those who do not have biology domain knowledge. The idea of classification is to group data into different sets. **Sequence Name** is very unique, since almost every row has its own unique sequence name. *(See below)*

```
summary(yeast$Sequence.Name)
```

```
## EF1A_YEAST   H3_YEAST   H4_YEAST IF4A_YEAST MAT2_YEAST  MTC_YEAST
##          2          2          2          2          2          2
## RL12_YEAST RL15_YEAST RL18_YEAST RL19_YEAST RL1A_YEAST  RL2_YEAST
##          2          2          2          2          2          2
## RL35_YEAST RL41_YEAST RL44_YEAST RLUB_YEAST RS22_YEAST RS24_YEAST
##          2          2          2          2          2          2
## RS28_YEAST RS41_YEAST RS4E_YEAST  RS8_YEAST 6P2K_YEAST 6PGD_YEAST
##          2          2          2          2          1          1
## AAR2_YEAST AATC_YEAST AATM_YEAST ABC1_YEAST ABF2_YEAST ABP1_YEAST
##          1          1          1          1          1          1
## ACE1_YEAST ACE2_YEAST ACEA_YEAST ACH1_YEAST ACO1_YEAST ACON_YEAST
##          1          1          1          1          1          1
## ACOX_YEAST  ACP_YEAST ACR1_YEAST  ACT_YEAST ACT2_YEAST ACT3_YEAST
##          1          1          1          1          1          1
## ACT5_YEAST ADA2_YEAST ADA3_YEAST ADB1_YEAST ADB2_YEAST ADH1_YEAST
##          1          1          1          1          1          1
## ADH2_YEAST ADH3_YEAST ADH4_YEAST ADP1_YEAST ADR1_YEAST ADR6_YEAST
##          1          1          1          1          1          1
## ADT1_YEAST ADT2_YEAST ADT3_YEAST AFG3_YEAST AFR1_YEAST AGA1_YEAST
##          1          1          1          1          1          1
## AGA2_YEAST AGAL_YEAST  ALF_YEAST ALG1_YEAST ALG5_YEAST ALG8_YEAST
##          1          1          1          1          1          1
## ALP1_YEAST AMPL_YEAST AMPM_YEAST AMYG_YEAST AMYH_YEAST ANC1_YEAST
##          1          1          1          1          1          1
## ANP1_YEAST AP17_YEAST AP19_YEAST AP54_YEAST APE2_YEAST APE3_YEAST
##          1          1          1          1          1          1
## APN1_YEAST AR56_YEAST ARD1_YEAST ARF1_YEAST ARF2_YEAST ARG1_YEAST
##          1          1          1          1          1          1
## ARG2_YEAST ARG3_YEAST ARGD_YEAST ARGI_YEAST ARLY_YEAST ARO1_YEAST
##          1          1          1          1          1          1
## AROC_YEAST AROF_YEAST AROG_YEAST ASG2_YEAST ASSY_YEAST AST1_YEAST
##          1          1          1          1          1          1
## ATC1_YEAST ATC2_YEAST ATC3_YEAST    (Other)
##          1          1          1       1363
```

On the contrary, **Localization Site** only has a certain number of unique elements in the column and each represent multiple rows.*(Also can be seen from the summary below)*

```r
summary(yeast$Localisation.Site)
```

```
## CYT ERL EXC ME1 ME2 ME3 MIT NUC POX VAC
## 463   5  35  44  51 163 244 429  20  30
```

However, in the spirit of science, some researches have been done by the author. This project is actually doing **Protein Subsecelluar Localization Prediction** whose definition is stated as follows:

"Protein subcellular localization prediction (or just protein localization prediction) involves the prediction of where a protein resides in a cell, its subcellular localization.In general, prediction tools take as input information about a protein, such as a protein sequence of amino acids, and produce a predicted location within the cell as output, such as the nucleus, Endoplasmic reticulum, Golgi apparatus, extracellular space, or other organelles. The aim is to build tools that can accurately predict the outcome of protein targeting in cells."

From the paragraph above, it is clear that the columns in between the first and last column work as input information to build the predicting model to predict the location within the cells. But what is **Accession Number**?

"In libraries, art galleries, museums and archives, initial control of an acquisition is usually achieved through assignment of a unique identifier. When used for this purpose, such an identifier is denoted an accession number. Assignment of accession numbers typically occurs at the point of accessioning or cataloging. The term is something of a misnomer, because the form accession numbers take is often alpha-numeric."

All of these prove that **Localization Site** is the best choice to be the classifier. The classifier is definitely effective even though the prediction accuracy is not very satisfiable due to the distribution of class variables. *(See the next section below)*

Prediction Analysis

The prediction is made by: 1st, generate random data by using set.seed() 2nd, split data to 2 sample size 3rd, create formula 4th, use ctree to train data 5th, predict data by using test data

However, as it is said in ctree plot comments and Task 3 Q1, the prediction results are slightly different but both with relatively low accuracy and precision. The initial assumption of the author about which factors lead to the result and difference include:

1. dataset size
2. set.seed() function
3. distribution of classes

Not set.seed() First, the author removed the second assumption set.seed() by exploring the functionality of set.seed() in R, it is just used to generate fixed random data so that other researchers could reproduce the result.

Not Dataset Size Originally, the author thought may be it is because that the dataset is too small and therefore there's not enough data to train the machine to be more accurate.

However, this is proved to be wrong.

```r
nrow(iris)
```

```
## [1] 150
```

```r
set.seed(1234)
ind <- sample(2, nrow(iris), replace=TRUE, prob=c(0.7, 0.3))
itrain_data <- iris[ind==1,]
itest_data <- iris[ind==2,]
i_formula <- Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width
```

```
iris_ctree <- ctree(i_formula, data = itrain_data)
confusionMatrix(predict(iris_ctree, newdata = itest_data), itest_data$Species)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   setosa versicolor virginica
##   setosa         10          0         0
##   versicolor      0         12         2
##   virginica       0          0        14
##
## Overall Statistics
##
##                Accuracy : 0.9474
##                  95% CI : (0.8225, 0.9936)
##     No Information Rate : 0.4211
##     P-Value [Acc > NIR] : 7.335e-12
##
##                   Kappa : 0.9202
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: setosa Class: versicolor Class: virginica
## Sensitivity                 1.0000            1.0000           0.8750
## Specificity                 1.0000            0.9231           1.0000
## Pos Pred Value              1.0000            0.8571           1.0000
## Neg Pred Value              1.0000            1.0000           0.9167
## Prevalence                  0.2632            0.3158           0.4211
## Detection Rate              0.2632            0.3158           0.3684
## Detection Prevalence        0.2632            0.3684           0.3684
## Balanced Accuracy           1.0000            0.9615           0.9375
```

As it can be seen, iris dataset only has 150 rows which is much smaller than the yeast dataset which has 1484 rows. However, the accuracy of the iris is much higher.Therefore, the size of dataset is not a factor that influence the Accuracy.

Yes, Class Distribution In order to test the difference of two split, the author also split the iris dataset into 80/20

```
set.seed(1234)
ind2 <- sample(2, nrow(iris), replace=TRUE, prob=c(0.8, 0.2))
itrain_data2 <- iris[ind2==1,]
itest_data2 <- iris[ind2==2,]
iris_ctree2 <- ctree(i_formula, data = itrain_data2)
confusionMatrix(predict(iris_ctree, newdata = itest_data2), itest_data2$Species)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   setosa versicolor virginica
##   setosa          8          0         0
##   versicolor      0          7         1
##   virginica       0          0        11
##
```

```
## Overall Statistics
##
##                Accuracy : 0.963
##                  95% CI : (0.8103, 0.9991)
##     No Information Rate : 0.4444
##     P-Value [Acc > NIR] : 1.077e-08
##
##                   Kappa : 0.9434
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: setosa Class: versicolor Class: virginica
## Sensitivity                 1.0000            1.0000           0.9167
## Specificity                 1.0000            0.9500           1.0000
## Pos Pred Value              1.0000            0.8750           1.0000
## Neg Pred Value              1.0000            1.0000           0.9375
## Prevalence                  0.2963            0.2593           0.4444
## Detection Rate              0.2963            0.2593           0.4074
## Detection Prevalence        0.2963            0.2963           0.4074
## Balanced Accuracy           1.0000            0.9750           0.9583
```

As it is shown above, the 80/20 split has higher accuracy, which is close to intuitive thinking: with more training data, the predictive model should be more accurate. But why is the case of yeast data so counter-intuitive? (The 80/20 split has lower accuracy than 70/30). Let's see the class distribution of iris

```
summary(iris$Species)
```

```
##     setosa versicolor  virginica
##         50         50         50
```

Compare with yeast class distribution

```
summary(yeast$Localisation.Site)
```

```
## CYT ERL EXC ME1 ME2 ME3 MIT NUC POX VAC
## 463   5  35  44  51 163 244 429  20  30
```

As it is shown above, the iris distribution is very balanced with each species has 50 data points. On the contrary, the yeast class distribution is very unbalanced with some classes have as many as hundreds of data points and some only has 5.

This unbalance is reflected in the ctree plot in Task 2 and TP FP comparison. ERL does not even have any prediction since it only has 5 data points and in 70/30 split VAC don't have any prediction at all but in 80/20 it has 4 FP, POX has 3 TP in 70/30 then the number dropped to only 2 in 80/20.

In conclusion, class distribution is the most significant factor that affects the accuracy, precision and different result in 70/30 and 80/20 split.