CS – GY 6923 Machine Learning

Professor: Dr. Raman Kannan

HW2: Classification Models

Avinash Authipudi

Section 1 - Review of the Dataset

In Exploratory Data Analysis, we removed features which were constant throughout the dataset with a zero value. These features must be removed since they add no value to the dataset and its analysis. The original dataset had 112 features out of which one variable "phishing" is the dependent variable. After removing the constant features, we had 98 features and 1 dependent variable.

We tried eliminating the correlated features in the dataset by setting the upper triangle matrix of the correlated features to 0 and then remove them since the highly correlated features would take a spot in the upper triangular matrix in the heatmap. This yielded us a dataset with 78 predictors and 1 dependent variable.

We now review the features of the newly obtained dataset with the help of VIF. From the below output of 'vif' command, the features have a VIF value of more than 10 which indicates that the features are still correlated.

```
> dim(ncpdata.new)
[1] 58645    79
> model <- glm(phishing~., data = ncpdata.new, family = "binomial")
Warning message:
glm.fit: fitted probabilities numerically 0 or 1 occurred
> vif(model)
Error in vif(model) : could not find function "vif"
> require(car)
Loading required package: car
Loading required package: carData
```

For analysis, we retain the current dataset which contains features that are still correlated but not as much as the previously highly correlated features just for the sake of analysis. We draw conclusions later to see if retaining some correlated features while running classification algorithms would play a part in its performance since making the dataset completely correlated means removing many features which could potentially add more accuracy in prediction.

The more features the dataset has, the more variance it has. In contrast, the less features a dataset has, the more bias. Hence, we run our classification algorithms on this variant of the dataset as well.

```
> vif(model)
            qty_dot_url           qty_hyphen_url       qty_underline_url           qty_slash_url
           1.010079e+03            4.407647e+02            1.096662e+02            6.886857e+02
   qty_questionmark_url           qty_equal_url             qty_at_url             qty_and_url
           3.653031e+00            1.558771e+02            5.625363e+05            1.065678e+02
    qty_exclamation_url           qty_space_url           qty_tilde_url           qty_comma_url
           1.171604e+00            1.000003e+00            1.334498e+08            1.218612e+00
           qty_plus_url        qty_asterisk_url         qty_hashtag_url          qty_dollar_url
           3.532844e+01            7.248481e+00            1.000000e+00            4.743476e+01
        qty_percent_url             qty_tld_url               length_url          qty_dot_domain
           4.833316e+02            1.608530e+00            4.505303e+02            6.530518e+02
      qty_hyphen_domain    qty_underline_domain           qty_at_domain       qty_vowels_domain
           4.856283e+01            1.000000e+00            1.005659e+00            4.748106e+00
          domain_length             domain_in_ip    server_client_domain       qty_dot_directory
           7.556641e+01            1.154633e+00            1.013267e+00            4.580154e+02
   qty_hyphen_directory qty_underline_directory     qty_slash_directory     qty_equal_directory
           4.490672e+02            1.645732e+02            9.426993e+02            3.328217e+02
       qty_at_directory       qty_and_directory     qty_tilde_directory  qty_asterisk_directory
           3.501592e+08            6.339475e+02            6.335883e+09            9.010698e+07
   qty_dollar_directory   qty_percent_directory        directory_length             qty_dot_file
           7.336805e+08            6.006958e+02            3.208280e+02            2.343028e+01
        qty_hyphen_file      qty_underline_file           qty_plus_file       qty_asterisk_file
           5.858463e+00            1.025392e+01            4.148467e+02            9.936097e+07
        qty_dollar_file        qty_percent_file             file_length          qty_dot_params
           6.719112e+09            2.552583e+01            8.755871e+00            1.479081e+02
      qty_hyphen_params    qty_underline_params         qty_slash_params qty_questionmark_params
           2.818620e+01            4.401639e+01            3.917771e+01            5.504639e+02
       qty_equal_params           qty_at_params          qty_and_params  qty_exclamation_params
           3.798479e+02            1.347043e+08            2.250520e+02            8.369311e+07
       qty_comma_params          qty_plus_params       qty_dollar_params      qty_percent_params
           2.201065e+07            1.419219e+02            2.404089e+08            2.019507e+02
          params_length      tld_present_params             qty_params               email_in_url
           1.048302e+02            4.984974e+01            7.106150e+01            8.968851e+00
          time_response              domain_spf                  asn_ip  time_domain_activation
           1.044845e+00            1.026658e+00            1.035417e+00            1.536067e+00

 time_domain_expiration         qty_ip_resolved         qty_nameservers           qty_mx_servers
           1.248476e+00            1.178548e+00            1.112519e+00            1.209572e+00
           ttl_hostname     tls_ssl_certificate           qty_redirects         url_google_index
           1.089416e+00            1.222067e+00            1.166888e+00            1.260863e+00
    domain_google_index            url_shortened
           1.266898e+00            1.022938e+00
```

In order to remove the correlated features completely from the dataset, we run the below code which removes features with a correlation index more than 0.5. This makes the dataset uncorrelated at the expense of removing most of the features which could be good predictors in the dataset. We retain only 1 out of the 2 correlated features.

This leaves us with a dataset which has 36 predictors and 1 dependent variable.

```
> cordata=cor(ncpdata.new[,1:78])
> print(ifelse(any(abs(cordata[cordata!=1])>0.5),"Correlated Predictors Exist","
+ No Correlated Predictors"))
[1] "Correlated Predictors Exist"
> cor_index=which(abs(cordata)>0.5 & abs(cordata)!=1, arr.ind = T)
> cor_index=cor_index[!duplicated(cbind(pmax(cor_index[,1], cor_index[,2]), pmin(
+     cor_index[,1], cor_index[,2]))),]
> tbl_cor_index=table(cor_index[,1])
> cor_attributes=as.numeric(names(tbl_cor_index))
> ncpdata_uncor=ncpdata.new[,-cor_attributes]
> dim(ncpdata_uncor)
[1] 58645    37
```

```
> cordata=cor(ncpdata_uncor[,1:36])
> print(ifelse(any(abs(cordata[cordata!=1])>0.5),"Correlated Predictors Exist","
+ + No Correlated Predictors"))
[1] "\n+ No Correlated Predictors"
```

Running vif on the dataset yields the below result.

```
> model <- glm(phishing~., data = ncpdata_uncor, family = "binomial")
Warning message:
glm.fit: fitted probabilities numerically 0 or 1 occurred
> vif(model)
           qty_dot_url          qty_hyphen_url      qty_underline_url          qty_slash_url
              5.004030                1.246269                1.061003                1.375014
  qty_questionmark_url             qty_at_url    qty_exclamation_url          qty_space_url
              1.012643                1.033126                1.030031                1.000000
         qty_tilde_url          qty_comma_url           qty_plus_url       qty_asterisk_url
              1.006196                1.089673                1.003158                1.000000
       qty_hashtag_url        qty_percent_url            qty_tld_url         qty_dot_domain
              1.000000                1.058293                1.224848                5.131383
     qty_hyphen_domain   qty_underline_domain          qty_at_domain      qty_vowels_domain
              1.234197                1.000000                1.000000                1.221426
          domain_in_ip   server_client_domain          time_response             domain_spf
              1.155954                1.007248                1.046651                1.023811
                asn_ip time_domain_activation time_domain_expiration        qty_ip_resolved
              1.024231                1.497011                1.234289                1.197851
       qty_nameservers         qty_mx_servers           ttl_hostname    tls_ssl_certificate
              1.090542                1.145526                1.078433                1.220895
         qty_redirects       url_google_index    domain_google_index          url_shortened
              1.153666                1.248849                1.250735                1.018854
```

Running variable importance using random forest yields the below result.

```
> require(rpart)
Loading required package: rpart
> tree.data <- rpart(phishing~., data=ncpdata_uncor)
> tree.data$variable.importance
         qty_slash_url time_domain_activation time_domain_expiration          url_shortened
          13970.614195            3048.245245             843.107480             443.435999
       qty_ip_resolved          time_response         qty_dot_domain           ttl_hostname
            438.594159             387.547008             304.039482             302.069464
        qty_mx_servers      qty_vowels_domain             domain_spf        qty_nameservers
            219.792649             147.999065             118.319036              29.966347
                asn_ip   qty_questionmark_url          qty_redirects       url_google_index
             20.009562               2.371113               2.371113               2.195750
```

```
> ncpdata_uncor$phishing <- as.factor(ncpdata_uncor$phishing)
> require(randomForest)
Loading required package: randomForest
randomForest 4.6-14
Type rfNews() to see new features/changes/bug fixes.
> rfmodel <- randomForest(phishing~., ncpdata_uncor)
> rfmodel_features = varImp(rfmodel)
```

```
> rfmodel_features = varImp(rfmodel)
> rfmodel_features = rfmodel_features[order(rfmodel_features, decreasing = TRUE),,drop = FALSE]
```

```
> top16features = row.names(rfmodel_features[1:16,,drop = FALSE])
> top16features
 [1] "qty_slash_url"          "time_domain_activation" "ttl_hostname"
 [4] "qty_dot_domain"         "asn_ip"                 "time_response"
 [7] "time_domain_expiration" "qty_hyphen_url"         "qty_vowels_domain"
[10] "qty_dot_url"            "qty_mx_servers"         "qty_nameservers"
[13] "qty_underline_url"      "qty_ip_resolved"        "qty_redirects"
[16] "qty_tld_url"
```

We now have 2 datasets ‘ncpdata.new’ which has more features and ‘ncpdata_uncor’ which has lesser uncorrelated features. For simplicity, we shall call the larger dataset as ‘Dataset1’ and the smaller dataset as ‘Dataset2’. We continue with running the below classification algorithms on our datasets.

1. Logistic Regression
2. Decision Trees
3. SVM
4. KNN
5. Naïve Bayes

At the end, we contrast the performance metrics of the various classifiers on both the datasets and summarize the results.

Section 2 - Logistic Regression on 2 datasets

We run logistic regression on both the datasets. First, we split the dataset into train and test datasets in the ratio 70:30. 'Traindata' and 'Testdata' reflect partitions on the larger dataset while 'Traindata_uncor' and 'Testdata_uncor' reflect partitions on the smaller dataset.

```
> dim(ncpdata.new)
[1] 58645    79
> dim(ncpdata_uncor)
[1] 58645    37
> set.seed(43)
> tridx1 <- sample(1:nrow(ncpdata.new), 0.7*nrow(ncpdata.new), replace = F)
> traindata <- ncpdata.new[tridx1,]
> testdata <- ncpdata.new[-tridx1,]
>
> set.seed(43)
> tridx2 <- sample(1:nrow(ncpdata_uncor), 0.7*nrow(ncpdata_uncor), replace = F)
> traindata_uncor <- ncpdata_uncor[tridx2,]
> testdata_uncor <- ncpdata_uncor[-tridx2,]
>
> table(traindata$phishing)

    0     1
19718 21333
> table(testdata$phishing)

   0    1
8280 9314
> table(traindata_uncor$phishing)

    0     1
19718 21333
> table(testdata_uncor$phishing)

   0    1
8280 9314
>
```

We check if the train and test datasets are balanced. We now train the model using the train dataset and predict on the train and test datasets.

Logistic Regression on Dataset1:

```
> details <- glm(phishing~.,data=traindata,family="binomial")
Warning message:
glm.fit: fitted probabilities numerically 0 or 1 occurred
> summary(details)

Call:
glm(formula = phishing ~ ., family = "binomial", data = traindata)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-4.0531  -0.2738   0.0000   0.3218   5.0215

Coefficients:
                          Estimate Std. Error z value Pr(>|z|)
(Intercept)              1.929e+00  3.164e-01   6.096 1.09e-09 ***
qty_dot_url             -9.023e-01  6.609e-01  -1.365 0.172137
qty_hyphen_url          -8.061e-02  2.794e-01  -0.289 0.772913
qty_underline_url       -4.600e-02  3.636e-01  -0.126 0.899340
qty_slash_url            5.951e-01  4.090e-01   1.455 0.145691
qty_questionmark_url    -1.402e+00  1.921e+00  -0.730 0.465338
qty_equal_url            4.172e-01  2.452e-01   1.702 0.088818 .
qty_at_url               6.268e+01  9.769e+02   0.064 0.948838
qty_and_url             -5.592e-01  3.096e-01  -1.806 0.070922 .
qty_exclamation_url      6.200e-01  5.465e-01   1.134 0.256609
qty_space_url            1.437e+01  7.202e+02   0.020 0.984077
qty_tilde_url            1.851e+01  4.381e+03   0.004 0.996629
qty_comma_url           -3.857e+00  4.331e-01  -8.906  < 2e-16 ***
qty_plus_url            -1.342e+00  1.047e+00  -1.283 0.199628
qty_asterisk_url         1.229e+01  3.756e+02   0.033 0.973897
qty_hashtag_url          1.284e+01  1.060e+03   0.012 0.990339
qty_dollar_url           1.245e-01  4.418e+02   0.000 0.999775
qty_percent_url          2.887e-01  1.232e+00   0.234 0.814711
qty_tld_url              5.473e-01  1.518e-01   3.605 0.000312 ***
length_url               2.296e-02  1.981e-02   1.159 0.246522

qty_and_params           1.605e+00  4.230e-01   3.794 0.000148 ***
qty_exclamation_params   1.102e+01  6.209e+02   0.018 0.985846
qty_comma_params         1.005e+01  3.816e+02   0.026 0.978998
qty_plus_params          5.449e-01  1.069e+00   0.510 0.610225
qty_dollar_params       -1.558e+00  5.225e+02  -0.003 0.997620
qty_percent_params      -8.168e-01  1.235e+00  -0.662 0.508229
params_length            7.088e-02  2.202e-02   3.219 0.001287 **
tld_present_params       2.723e+00  5.347e-01   5.092 3.54e-07 ***
qty_params              -1.588e+00  2.521e-01  -6.301 2.95e-10 ***
email_in_url            -2.800e+01  3.343e+02  -0.084 0.933251
time_response            6.622e-02  1.141e-02   5.802 6.55e-09 ***
domain_spf              -2.673e-02  3.244e-02  -0.824 0.409980
asn_ip                   2.698e-06  3.750e-07   7.194 6.30e-13 ***
time_domain_activation  -4.471e-04  9.254e-06 -48.314  < 2e-16 ***
time_domain_expiration  -1.744e-04  4.116e-05  -4.237 2.27e-05 ***
qty_ip_resolved          1.242e-01  2.046e-02   6.068 1.29e-09 ***
qty_nameservers         -2.565e-01  1.505e-02 -17.041  < 2e-16 ***
qty_mx_servers           4.574e-02  1.105e-02   4.138 3.50e-05 ***
ttl_hostname             4.612e-05  2.506e-06  18.403  < 2e-16 ***
tls_ssl_certificate     -5.675e-01  3.940e-02 -14.403  < 2e-16 ***
qty_redirects            1.101e-01  2.381e-02   4.625 3.74e-06 ***
url_google_index         3.637e-01  5.305e-01   0.686 0.493003
domain_google_index     -7.885e-01  4.458e-01  -1.769 0.076942 .
url_shortened            5.929e+00  4.445e-01  13.340  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 56845  on 41050  degrees of freedom
Residual deviance: 20579  on 40972  degrees of freedom
AIC: 20737

Number of Fisher Scoring iterations: 18
```

Learning Phase Confusion Matrix on Dataset1 using Logistic Regression:

```
> sglmlog.pred <- predict(details,traindata,family="binomial")
> sglmlog.class<-ifelse(sglmlog.pred<0.5,0,1)
> scfmlog<-table(traindata[,79], sglmlog.class)
> cmlog <- confusionMatrix(scfmlog)
> cmlog
Confusion Matrix and Statistics

   sglmlog.class
       0     1
  0 18185  1533
  1  2808 18525

               Accuracy : 0.8943
                 95% CI : (0.8912, 0.8972)
    No Information Rate : 0.5114
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7887

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.8662
            Specificity : 0.9236
         Pos Pred Value : 0.9223
         Neg Pred Value : 0.8684
             Prevalence : 0.5114
         Detection Rate : 0.4430
   Detection Prevalence : 0.4803
      Balanced Accuracy : 0.8949

       'Positive' Class : 0
```

Testing Phase Confusion Matrix on Dataset1 using Logistic Regression:

```
> sglmlogtest.pred <- predict(details,testdata,family="binomial")
> sglmlogtest.class<-ifelse(sglmlogtest.pred<0.5,0,1)
> scfmlogtest<-table(testdata[,79], sglmlogtest.class)
> cmlogtest <- confusionMatrix(scfmlogtest)
> cmlogtest
Confusion Matrix and Statistics

   sglmlogtest.class
      0    1
  0 7677  603
  1 1312 8002

               Accuracy : 0.8912
                 95% CI : (0.8865, 0.8957)
    No Information Rate : 0.5109
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7826

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.8540
            Specificity : 0.9299
         Pos Pred Value : 0.9272
         Neg Pred Value : 0.8591
             Prevalence : 0.5109
         Detection Rate : 0.4363
   Detection Prevalence : 0.4706
      Balanced Accuracy : 0.8920

       'Positive' Class : 0
```

Logistic Regression on Dataset2:

```
> details_uncor <- glm(phishing~.,data=traindata_uncor,family="binomial")
Warning message:
glm.fit: fitted probabilities numerically 0 or 1 occurred
> summary(details_uncor)

Call:
glm(formula = phishing ~ ., family = "binomial", data = traindata_uncor)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-4.9114  -0.3993   0.0004   0.3651   5.1676

Coefficients:
                          Estimate Std. Error z value Pr(>|z|)
(Intercept)            -9.615e-01  1.465e-01  -6.564 5.24e-11 ***
qty_dot_url             1.287e+00  4.290e-02  29.996  < 2e-16 ***
qty_hyphen_url         -2.486e-01  1.455e-02 -17.085  < 2e-16 ***
qty_underline_url       1.681e-01  3.107e-02   5.411 6.26e-08 ***
qty_slash_url           1.288e+00  1.659e-02  77.654  < 2e-16 ***
qty_questionmark_url    8.179e-01  5.448e-01   1.501  0.13327
qty_at_url              8.325e+00  1.401e+00   5.943 2.80e-09 ***
qty_exclamation_url     1.500e+00  3.136e-01   4.783 1.73e-06 ***
qty_space_url           1.101e+01  2.589e+02   0.043  0.96608
qty_tilde_url          -6.556e-01  2.091e-01  -3.136  0.00171 **
qty_comma_url          -2.964e+00  2.345e-01 -12.642  < 2e-16 ***
qty_plus_url           -3.017e-01  1.396e-01  -2.161  0.03067 *
qty_asterisk_url        1.135e+01  8.095e+01   0.140  0.88848
qty_hashtag_url         1.137e+01  4.222e+02   0.027  0.97851
qty_percent_url        -9.393e-02  1.211e-02  -7.754 8.91e-15 ***
qty_tld_url             7.956e-01  1.319e-01   6.034 1.60e-09 ***
qty_dot_domain         -2.178e+00  5.214e-02 -41.775  < 2e-16 ***
qty_hyphen_domain       1.017e+00  4.467e-02  22.773  < 2e-16 ***
qty_underline_domain    1.582e+01  9.799e+02   0.016  0.98712
.....................................................................
qty_percent_url        -9.393e-02  1.211e-02  -7.754 8.91e-15 ***
qty_tld_url             7.956e-01  1.319e-01   6.034 1.60e-09 ***
qty_dot_domain         -2.178e+00  5.214e-02 -41.775  < 2e-16 ***
qty_hyphen_domain       1.017e+00  4.467e-02  22.773  < 2e-16 ***
qty_underline_domain    1.582e+01  9.799e+02   0.016  0.98712
qty_at_domain           8.239e+00  3.956e+03   0.002  0.99834
qty_vowels_domain       1.169e-01  6.692e-03  17.469  < 2e-16 ***
domain_in_ip            4.957e+00  4.347e-01  11.403  < 2e-16 ***
server_client_domain    7.912e-01  2.706e-01   2.924  0.00345 **
time_response           6.615e-02  1.064e-02   6.216 5.11e-10 ***
domain_spf              2.893e-02  3.033e-02   0.954  0.34020
asn_ip                  2.459e-06  3.340e-07   7.363 1.80e-13 ***
time_domain_activation -4.506e-04  8.571e-06 -52.570  < 2e-16 ***
time_domain_expiration -1.964e-04  4.057e-05  -4.840 1.30e-06 ***
qty_ip_resolved         1.792e-01  1.849e-02   9.693  < 2e-16 ***
qty_nameservers        -2.361e-01  1.367e-02 -17.278  < 2e-16 ***
qty_mx_servers          4.214e-02  1.003e-02   4.200 2.67e-05 ***
ttl_hostname            4.090e-05  2.301e-06  17.777  < 2e-16 ***
tls_ssl_certificate    -4.284e-01  3.635e-02 -11.787  < 2e-16 ***
qty_redirects           5.261e-02  2.155e-02   2.441  0.01464 *
url_google_index        2.193e-01  4.909e-01   0.447  0.65503
domain_google_index    -5.308e-01  4.141e-01  -1.282  0.19989
url_shortened           6.400e+00  4.214e-01  15.188  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 56845  on 41050  degrees of freedom
Residual deviance: 23700  on 41014  degrees of freedom
AIC: 23774

Number of Fisher Scoring iterations: 16
```

Learning Phase Confusion Matrix on Dataset2 using Logistic Regression:

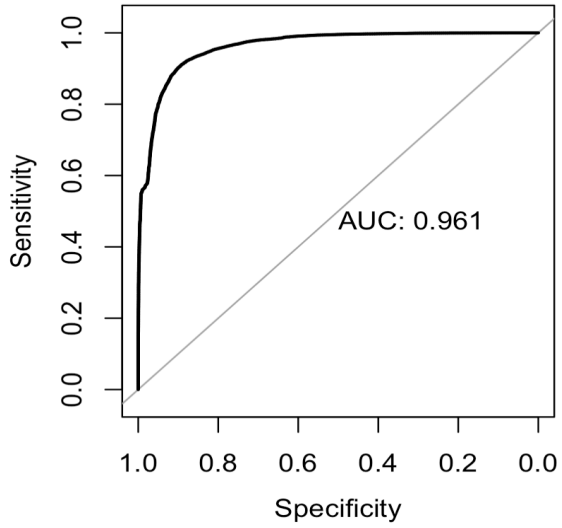
```
> sglmlog_uncor.pred <- predict(details_uncor,traindata_uncor,family="binomial")
> sglmlog_uncor.class<-ifelse(sglmlog_uncor.pred<0.5,0,1)
> scfmlog_uncor<-table(traindata_uncor[,37], sglmlog_uncor.class)
> cmlog_uncor <- confusionMatrix(scfmlog_uncor)
> cmlog_uncor
Confusion Matrix and Statistics

   sglmlog_uncor.class
       0     1
  0 18095  1623
  1  3831 17502

               Accuracy : 0.8671
                 95% CI : (0.8638, 0.8704)
    No Information Rate : 0.5341
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.735

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.8253
            Specificity : 0.9151
         Pos Pred Value : 0.9177
         Neg Pred Value : 0.8204
             Prevalence : 0.5341
         Detection Rate : 0.4408
   Detection Prevalence : 0.4803
      Balanced Accuracy : 0.8702

       'Positive' Class : 0
```

Testing Phase Confusion Matrix on Dataset2 using Logistic Regression:

```
> sglmlogtest_uncor.pred <- predict(details_uncor,testdata_uncor,family="binomial")
> sglmlogtest_uncor.class<-ifelse(sglmlogtest_uncor.pred<0.5,0,1)
> scfmlogtest_uncor<-table(testdata_uncor[,37], sglmlogtest_uncor.class)
> cmlogtest_uncor <- confusionMatrix(scfmlogtest_uncor)
> cmlogtest_uncor
Confusion Matrix and Statistics

   sglmlogtest_uncor.class
      0    1
  0 7632  648
  1 1724 7590

               Accuracy : 0.8652
                 95% CI : (0.86, 0.8702)
    No Information Rate : 0.5318
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7314

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.8157
            Specificity : 0.9213
         Pos Pred Value : 0.9217
         Neg Pred Value : 0.8149
             Prevalence : 0.5318
         Detection Rate : 0.4338
   Detection Prevalence : 0.4706
      Balanced Accuracy : 0.8685

       'Positive' Class : 0
```

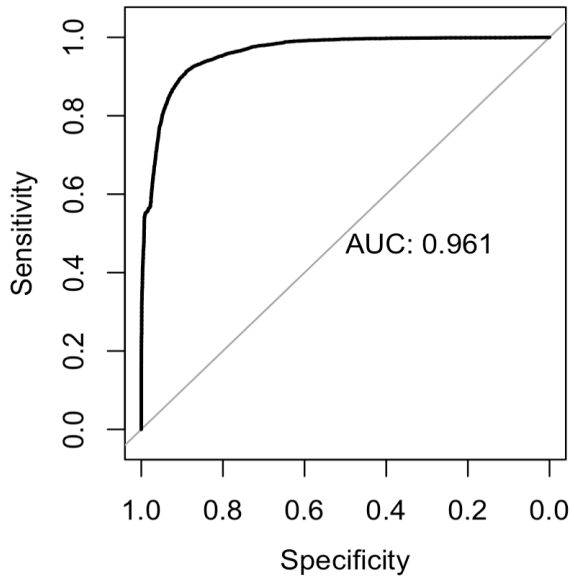
ROC Curve with AUC for the Learning Phase of Dataset1.

```
> pred_train <- predict(details, traindata, type = "response")
> roc_train <- roc(traindata$phishing~pred_train, plot = TRUE, print.auc = TRUE)
Setting levels: control = 0, case = 1
Setting direction: controls < cases
```



ROC Curve with AUC for the Testing Phase of Dataset1:

```
> pred_test <- predict(details, testdata, type = "response")
> roc_test <- roc(testdata$phishing~pred_test, plot = TRUE, print.auc = TRUE)
Setting levels: control = 0, case = 1
Setting direction: controls < cases
```




```
> as.numeric(roc_train$auc)
[1] 0.9614993
> as.numeric(roc_test$auc)
[1] 0.9607186
```

ROC Curve with AUC for the Training Phase of Dataset2:

```
> pred.uncor_train <- predict(details_uncor, traindata_uncor, type = "response")
> roc.uncor_train <- roc(traindata_uncor$phishing~pred.uncor_train, plot = TRUE, print.auc = TRUE)
Setting levels: control = 0, case = 1
Setting direction: controls < cases
> pred.uncor_test <- predict(details_uncor, testdata_uncor, type = "response")
> roc.uncor_test <- roc(testdata_uncor$phishing~pred.uncor_test, plot = TRUE, print.auc = TRUE)
Setting levels: control = 0, case = 1
Setting direction: controls < cases
> as.numeric(roc.uncor_train$auc)
[1] 0.9505184
> as.numeric(roc.uncor_test$auc)
[1] 0.9506103
```

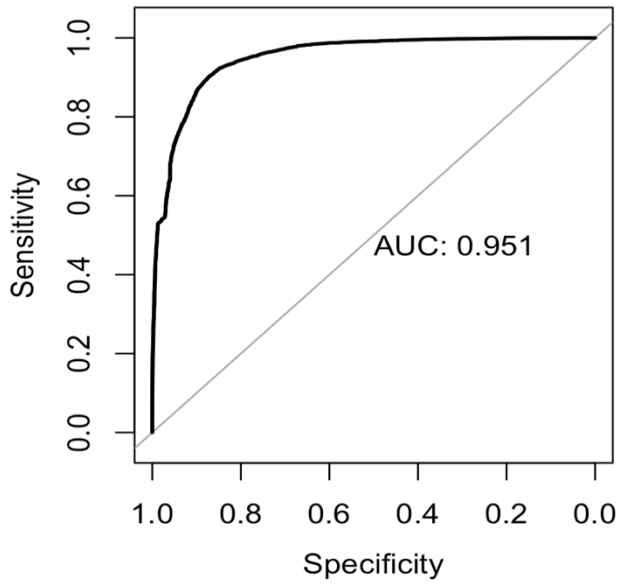


Fig: Training Phase

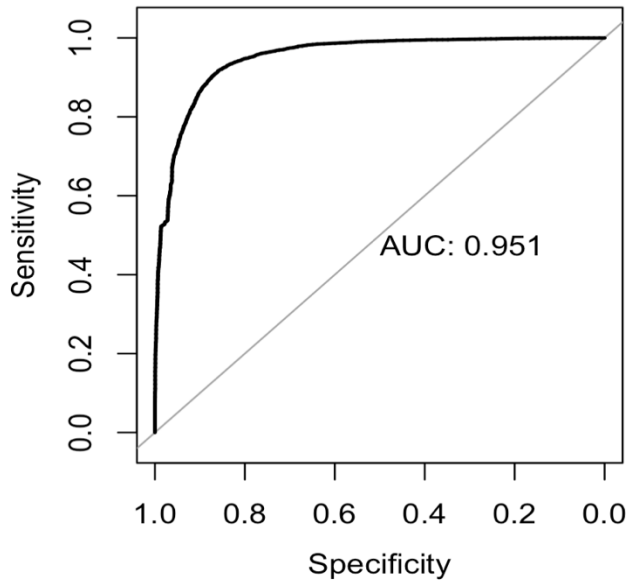


Fig: Testing Phase

Performance Metrics of the Datasets:

Performance Metrics	Dataset 1		Dataset2	
	Train	Test	Train	Test
Accuracy	89.43	89.12	86.71	86.52
Sensitivity	86.62	85.40	82.53	81.57
Specificity	92.36	92.99	91.51	92.13
Balanced Accuracy	89.49	89.20	87.02	86.85

The above table indicates the performance metrics obtained by running logistic regression classifier on the 2 datasets. The results are summarized by running confusion matrices on the models obtained from training the datasets.

We can see that higher accuracy is obtained for the larger version of the dataset. Since it has more features than the smaller dataset, many contributing features could add to the accuracy of the model.

Also, the percentage decrease between the training and testing models is not significant (> 25%) which means that none of the models run above are overfitting. This means that the models do not have high variance.

A point to note here is the Null and Residual Deviance resulting from the models run. The Chi-Square values of the 2 models run above are as follows:
X^2 = (Null Deviance – Residual Deviance) = 36266 (Dataset1)
Degrees of freedom = 78

X^2 = (Null Deviance – Residual Deviance) = 33145 (Dataset2)
Degrees of freedom = 36

Converting the Chi-Square scores to p-values indicate that the p-values for both the models is less than 0.01 which indicates a good fit of the models.

Section 3 - Decision Tree on 2 Datasets

We now run decision tree classifier on both the datasets. The following code below indicates training and testing of the classifier models on the datasets.

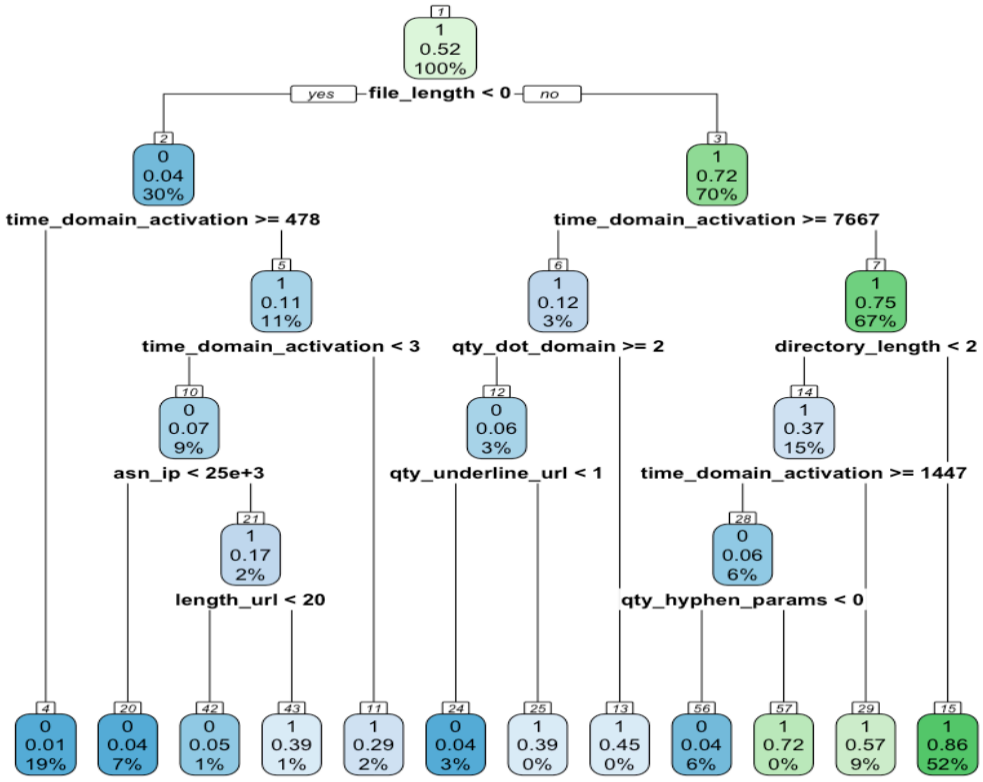
Decision tree classifier on Dataset1

Learning Phase Confusion Matrix on Dataset1 using Decision trees:

```
> penalty.matrix <- matrix(c(0,1,10,0), byrow=TRUE, nrow=2)
> tree <- rpart(phishing~., data=traindata, parms = list(loss = penalty.matrix),method = "class")
> pred <- predict(tree, newdata = traindata[-79], type ='class')
> t <- table(traindata$phishing, pred)
> confusionMatrix(t)
Confusion Matrix and Statistics

   pred
       0     1
  0 14248  5470
  1   310 21023

               Accuracy : 0.8592
                 95% CI : (0.8558, 0.8626)
    No Information Rate : 0.6454
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7151

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9787
            Specificity : 0.7935
         Pos Pred Value : 0.7226
         Neg Pred Value : 0.9855
             Prevalence : 0.3546
         Detection Rate : 0.3471
   Detection Prevalence : 0.4803
      Balanced Accuracy : 0.8861

       'Positive' Class : 0
```

Plotting the decision tree plot for Dataset1:

```
> rpart.plot(tree, nn=TRUE)
```

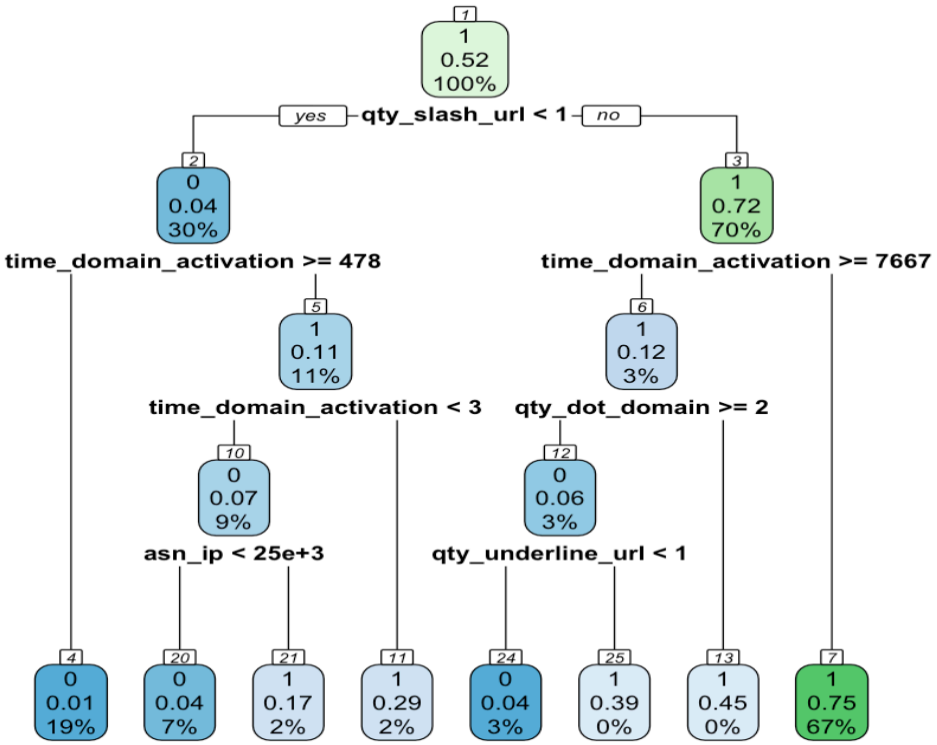


Testing Phase Confusion Matrix on Dataset1 using Decision trees:

```
> pred_test <- predict(tree, newdata = testdata[-79], type ='class')
> t_test<- table(testdata$phishing, pred_test)
> confusionMatrix(t_test)
Confusion Matrix and Statistics

   pred_test
      0    1
  0 6089 2191
  1  158 9156

               Accuracy : 0.8665
                 95% CI : (0.8614, 0.8715)
    No Information Rate : 0.6449
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7283

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9747
            Specificity : 0.8069
         Pos Pred Value : 0.7354
         Neg Pred Value : 0.9830
             Prevalence : 0.3551
         Detection Rate : 0.3461
   Detection Prevalence : 0.4706
      Balanced Accuracy : 0.8908

       'Positive' Class : 0

>
```

Decision tree classifier on Dataset2
Plotting the decision tree for Dataset2

```
> tree_uncor <- rpart(phishing~., data=traindata_uncor, parms = list(loss = penalty.matrix),method = "cl
ass")
> rpart.plot(tree_uncor, nn=TRUE)
```



Learning Phase Confusion Matrix on Dataset2 using Decision trees:

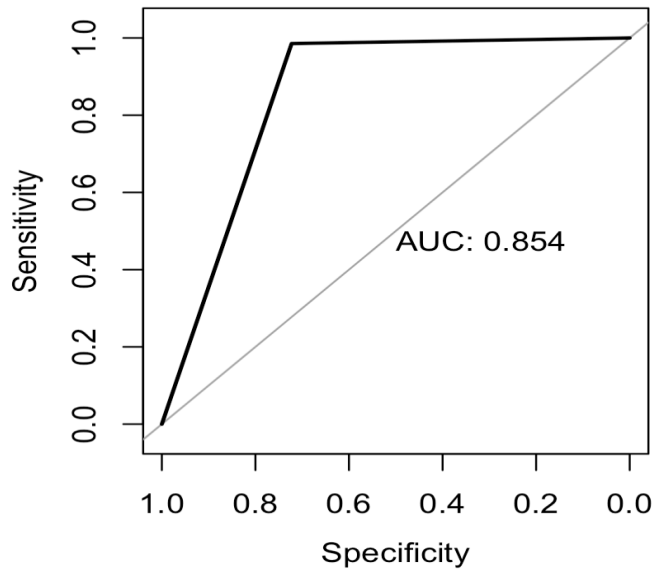
```
> pred_uncor <- predict(tree_uncor, newdata = traindata_uncor[-37], type ='class')
> t_uncor <- table(traindata_uncor$phishing, pred_uncor)
> confusionMatrix(t_uncor)
Confusion Matrix and Statistics

   pred_uncor
       0     1
  0 11524  8194
  1   183 21150

               Accuracy : 0.7959
                 95% CI : (0.792, 0.7998)
    No Information Rate : 0.7148
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.5849

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9844
            Specificity : 0.7208
         Pos Pred Value : 0.5844
         Neg Pred Value : 0.9914
             Prevalence : 0.2852
         Detection Rate : 0.2807
   Detection Prevalence : 0.4803
      Balanced Accuracy : 0.8526

       'Positive' Class : 0
```

Testing Phase Confusion Matrix on Dataset2 using Decision trees:

```
> pred_uncor_test <- predict(tree_uncor, newdata = testdata_uncor[-37], type ='class')
> t_uncor_test <- table(testdata_uncor$phishing, pred_uncor_test)
> confusionMatrix(t_uncor_test)
Confusion Matrix and Statistics

   pred_uncor_test
      0    1
  0 4965 3315
  1   96 9218

               Accuracy : 0.8061
                 95% CI : (0.8002, 0.8119)
    No Information Rate : 0.7123
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.6023

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9810
            Specificity : 0.7355
         Pos Pred Value : 0.5996
         Neg Pred Value : 0.9897
             Prevalence : 0.2877
         Detection Rate : 0.2822
   Detection Prevalence : 0.4706
      Balanced Accuracy : 0.8583

       'Positive' Class : 0
```

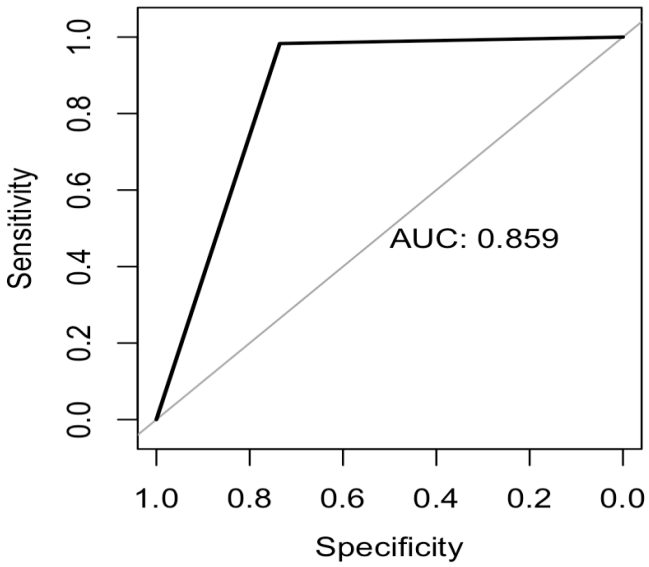
ROC Curve with AUC for the Training Phase of Dataset1:

```
> dt_AUC_pred = roc(traindata[, 79], as.ordered(pred))
Setting levels: control = 0, case = 1
Setting direction: controls < cases
> plot.roc(dt_AUC_pred, plot = TRUE, print.auc = TRUE)
```



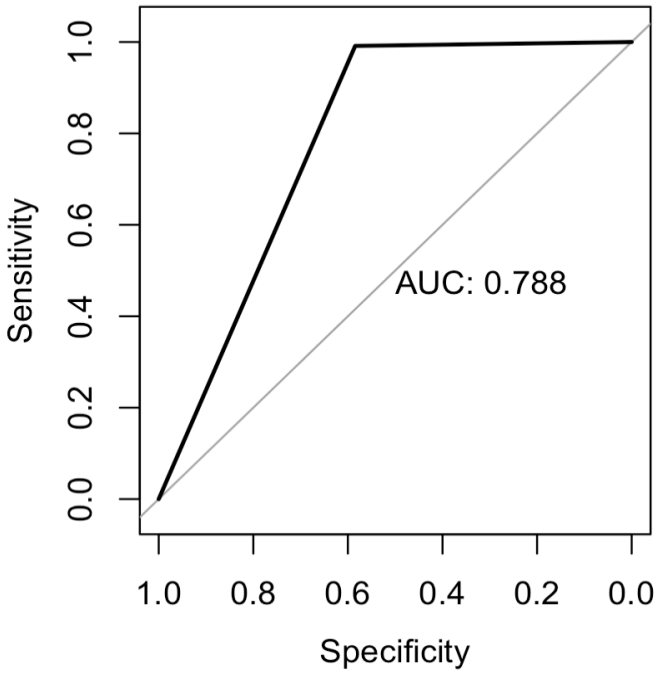
ROC Curve with AUC for the Testing Phase of Dataset1:

```
> dt_AUC_pred_test = roc(testdata[, 79], as.ordered(pred_test))
Setting levels: control = 0, case = 1
Setting direction: controls < cases
> plot.roc(dt_AUC_pred_test, plot = TRUE, print.auc = TRUE)
```



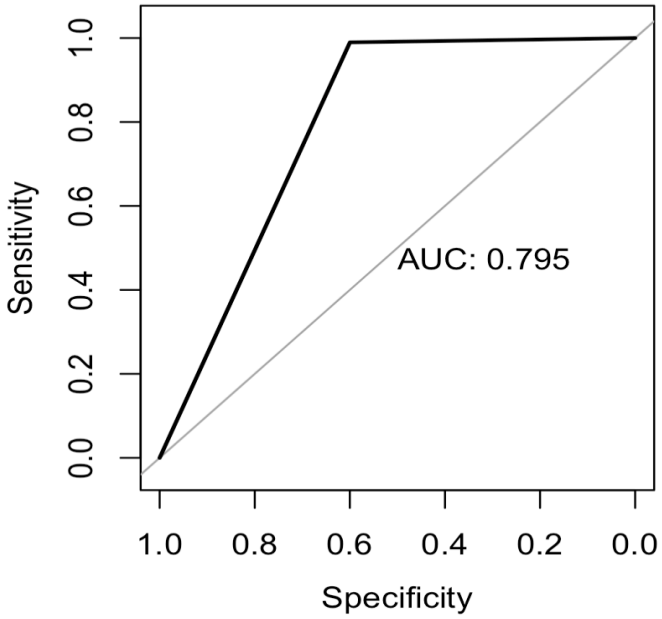
ROC Curve with AUC for the Training Phase of Dataset2:

```
> dt_AUC_pred_uncor = roc(traindata_uncor[, 37], as.ordered(pred_uncor))
Setting levels: control = 0, case = 1
Setting direction: controls < cases
> plot.roc(dt_AUC_pred_uncor, plot = TRUE, print.auc = TRUE)
```



ROC Curve with AUC for the Testing Phase of Dataset2:

```
> dt_AUC_pred_uncor_test = roc(testdata_uncor[, 37], as.ordered(pred_uncor_test))
Setting levels: control = 0, case = 1
Setting direction: controls < cases
> plot.roc(dt_AUC_pred_uncor_test, plot = TRUE, print.auc = TRUE)
```



Performance Metrics of the Datasets:

Performance Metrics	Dataset 1		Dataset2	
	Train	Test	Train	Test
Accuracy	85.92	86.65	79.59	80.61
Sensitivity	97.89	97.47	98.44	98.10
Specificity	79.35	80.69	72.08	73.55
Balanced Accuracy	88.61	89.08	85.26	85.83

The above table indicates the performance metrics obtained by running decision tree classifier on the 2 datasets. The results are summarized by running confusion matrices on the models obtained from training and testing the datasets.

We can see that higher accuracy is obtained for the larger version of the dataset. A similar result is seen from the result of the logistic regression models as well.

Also, the percentage decrease between the training and testing models is not significant (> 25%) which means that none of the models run above are overfitting. This means that the models do not have high variance.

Section 4 - SVM on 2 Datasets

We now proceed with the next classifier that is SVM. The below code indicates the implementation of the SVM classifier on both the datasets.

SVM classifier on Dataset1

Learning Phase Confusion Matrix on Dataset1 using SVM:

```
> traindata_svm = traindata
> testdata_svm = testdata
> traindata_svm$phishing=as.factor(traindata_svm$phishing)
> testdata_svm$phishing=as.factor(testdata_svm$phishing)
> svm_model=svm(phishing~.,data=traindata_svm, probability=TRUE)
> svm_pred_train = predict(svm_model,traindata[, -79], type="class")
> cfm_train_svm=confusionMatrix(table(traindata[,79], svm_pred_train))
> cfm_train_svm
Confusion Matrix and Statistics

   svm_pred_train
       0     1
  0 17990  1728
  1  1469 19864

               Accuracy : 0.9221
                 95% CI : (0.9195, 0.9247)
    No Information Rate : 0.526
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8439

 Mcnemar's Test P-Value : 5.043e-06

            Sensitivity : 0.9245
            Specificity : 0.9200
         Pos Pred Value : 0.9124
         Neg Pred Value : 0.9311
             Prevalence : 0.4740
         Detection Rate : 0.4382
   Detection Prevalence : 0.4803
      Balanced Accuracy : 0.9222

       'Positive' Class : 0
```

Testing Phase Confusion Matrix on Dataset1 using SVM:

```
> svm_pred_test = predict(svm_model,testdata[, -79], type="class")
> cfm_test_svm=confusionMatrix(table(testdata[,79], svm_pred_test))
> cfm_test_svm
Confusion Matrix and Statistics

   svm_pred_test
      0    1
  0 7560  720
  1  716 8598

               Accuracy : 0.9184
                 95% CI : (0.9142, 0.9224)
    No Information Rate : 0.5296
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.8362

 Mcnemar's Test P-Value : 0.9369

            Sensitivity : 0.9135
            Specificity : 0.9227
         Pos Pred Value : 0.9130
         Neg Pred Value : 0.9231
             Prevalence : 0.4704
         Detection Rate : 0.4297
   Detection Prevalence : 0.4706
      Balanced Accuracy : 0.9181

       'Positive' Class : 0
```

Learning Phase Confusion Matrix on Dataset2 using SVM:

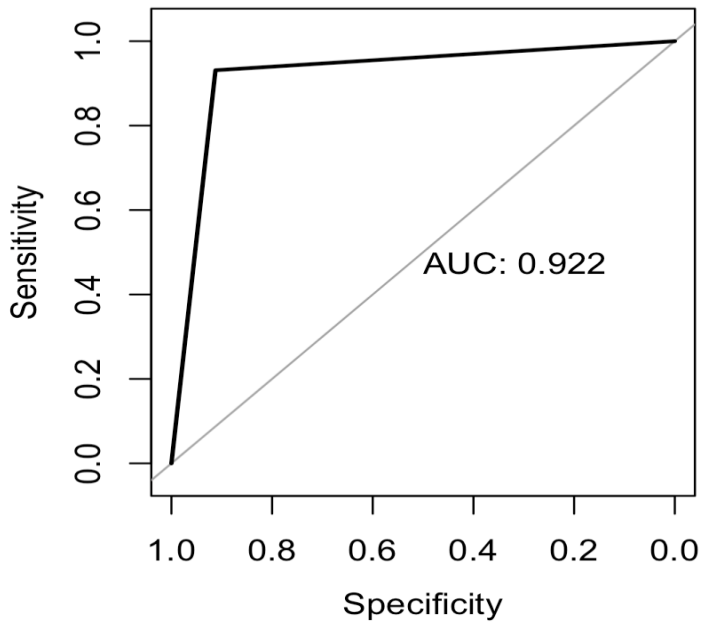
```
> traindata_svm_uncor = traindata_uncor
> testdata_svm_uncor = testdata_uncor
> traindata_svm_uncor$phishing=as.factor(traindata_svm_uncor$phishing)
> testdata_svm_uncor$phishing=as.factor(testdata_svm_uncor$phishing)
> svm_model_uncor=svm(phishing~.,data=traindata_svm_uncor, probability=TRUE)
> svm_pred_train_uncor = predict(svm_model_uncor,traindata_uncor[, -37], type="class")
> cfm_train_svm_uncor=confusionMatrix(table(traindata_uncor[,37], svm_pred_train_uncor))
> cfm_train_svm_uncor
Confusion Matrix and Statistics

   svm_pred_train_uncor
       0     1
  0 18132  1586
  1  1575 19758

               Accuracy : 0.923
                 95% CI : (0.9204, 0.9256)
    No Information Rate : 0.5199
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.8458

 Mcnemar's Test P-Value : 0.8588

            Sensitivity : 0.9201
            Specificity : 0.9257
         Pos Pred Value : 0.9196
         Neg Pred Value : 0.9262
             Prevalence : 0.4801
         Detection Rate : 0.4417
   Detection Prevalence : 0.4803
      Balanced Accuracy : 0.9229

       'Positive' Class : 0
```

Testing Phase Confusion Matrix on Dataset2 using SVM:

```
> svm_pred_test_uncor = predict(svm_model_uncor,testdata_uncor[, -37], type="class")
> cfm_test_svm_uncor=confusionMatrix(table(testdata_uncor[,37], svm_pred_test_uncor))
> cfm_test_svm_uncor
Confusion Matrix and Statistics

   svm_pred_test_uncor
       0    1
  0 7589  691
  1  755 8559

               Accuracy : 0.9178
                 95% CI : (0.9137, 0.9218)
    No Information Rate : 0.5257
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.8351

 Mcnemar's Test P-Value : 0.09757

            Sensitivity : 0.9095
            Specificity : 0.9253
         Pos Pred Value : 0.9165
         Neg Pred Value : 0.9189
             Prevalence : 0.4743
         Detection Rate : 0.4313
   Detection Prevalence : 0.4706
      Balanced Accuracy : 0.9174

       'Positive' Class : 0
```

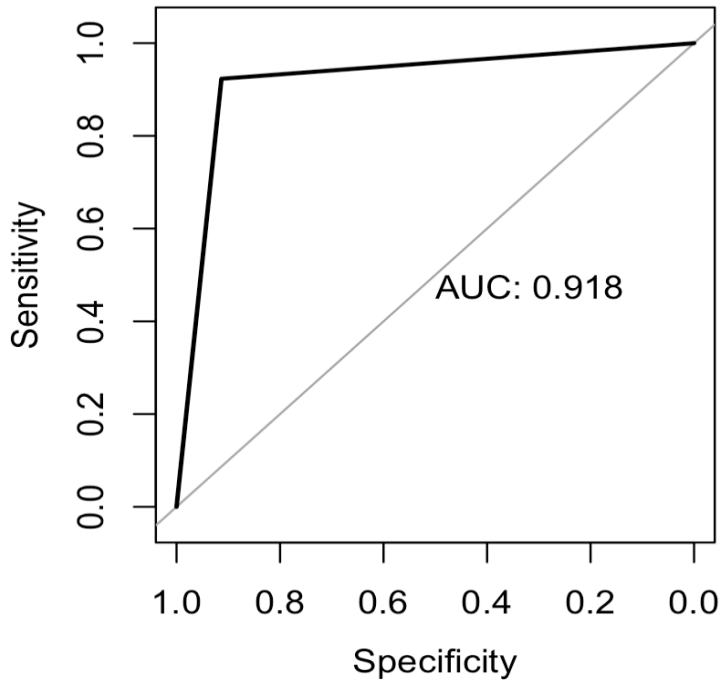
ROC Curve with AUC for the Training Phase of Dataset1:

```
> svm_AUC_pred = roc(traindata[, 79], as.ordered(svm_pred_train))
Setting levels: control = 0, case = 1
Setting direction: controls < cases
There were 16 warnings (use warnings() to see them)
> plot.roc(svm_AUC_pred, plot = TRUE, print.auc = TRUE)
```



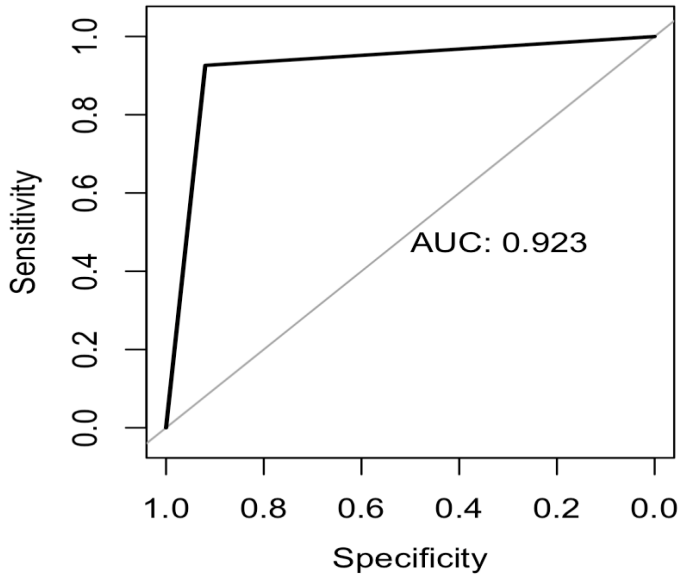
ROC Curve with AUC for the Testing Phase of Dataset1:

```
> svm_AUC_pred_test = roc(testdata[, 79], as.ordered(svm_pred_test))
Setting levels: control = 0, case = 1
Setting direction: controls < cases
> plot.roc(svm_AUC_pred_test, plot = TRUE, print.auc = TRUE)
>
```



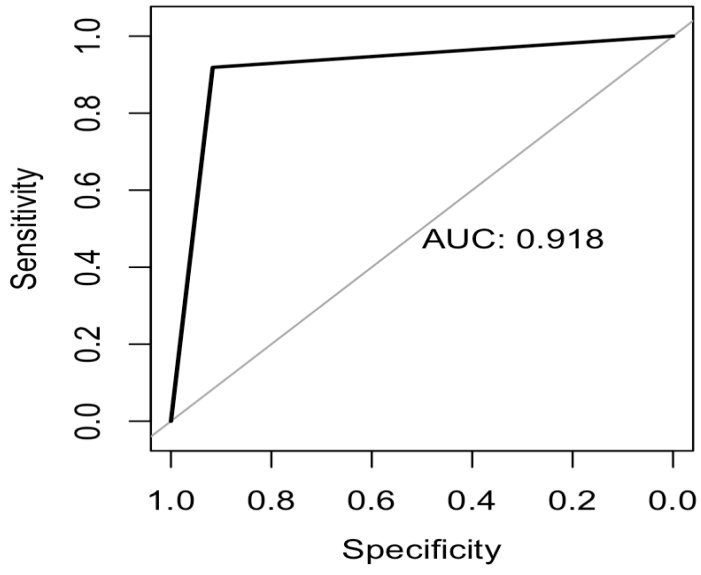
ROC Curve with AUC for the Training Phase of Dataset2:

```
> svm_AUC_pred_uncor = roc(traindata_uncor[, 37], as.ordered(svm_pred_train_uncor))
Setting levels: control = 0, case = 1
Setting direction: controls < cases
> plot.roc(svm_AUC_pred_uncor, plot = TRUE, print.auc = TRUE)
>
```



ROC Curve with AUC for the Testing Phase of Dataset2:

```
> svm_AUC_pred_uncor_test = roc(testdata_uncor[, 37], as.ordered(svm_pred_test_uncor))
Setting levels: control = 0, case = 1
Setting direction: controls < cases
> plot.roc(svm_AUC_pred_uncor_test, plot = TRUE, print.auc = TRUE)
>
```



Performance Metrics of the Datasets:

Performance Metrics	Dataset 1		Dataset2	
	Train	Test	Train	Test
Accuracy	92.21	91.84	92.30	91.78
Sensitivity	92.45	91.35	92.01	90.95
Specificity	92.00	92.27	92.57	92.53
Balanced Accuracy	92.22	91.81	92.29	91.74

The above table indicates the performance metrics obtained by running decision tree classifier on the 2 datasets. The results are summarized by running confusion matrices on the models obtained from training and testing the datasets.

We can see that higher accuracy is obtained for the larger version of the dataset. The same pattern of results continue with this classifier as well.

Also, the percentage decrease between the training and testing models is not significant (> 25%) which means that none of the models run above are overfitting. This means that the models do not have high variance.

Section 5 – KNN on 2 Datasets

The next classifier is KNN which is run on the 2 datasets. The below shows the implementation of KNN.

KNN classifier on Dataset1

Testing Phase Confusion Matrix on Dataset1 using KNN:

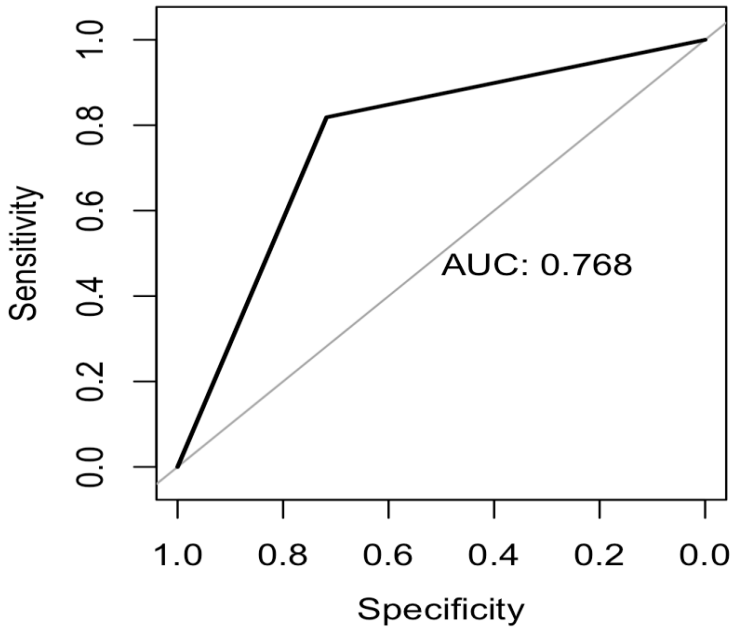
```
> traindata_knn=traindata[, -which(names(traindata)=="phishing")]
> testdata_knn=testdata[, -which(names(testdata)=="phishing")]
> trainclass_knn=factor(traindata[, which(names(traindata)=="phishing")])
> testclass_knn=factor(testdata[, which(names(testdata)=="phishing")])
> knn_pred_test=knn(traindata_knn,testdata_knn,trainclass_knn, k = 243, prob=TRUE)
> knn_cfm_tst=confusionMatrix(table(testclass_knn,knn_pred_test))
> knn_cfm_tst
Confusion Matrix and Statistics

             knn_pred_test
testclass_knn    0    1
            0 5941 2339
            1 1690 7624

               Accuracy : 0.771
                 95% CI : (0.7647, 0.7772)
    No Information Rate : 0.5663
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.5384

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.7785
            Specificity : 0.7652
         Pos Pred Value : 0.7175
         Neg Pred Value : 0.8186
             Prevalence : 0.4337
         Detection Rate : 0.3377
   Detection Prevalence : 0.4706
      Balanced Accuracy : 0.7719

       'Positive' Class : 0
```

Testing Phase Confusion Matrix on Dataset2 using KNN:

```
> traindata_knn_uncor=traindata_uncor[, -which(names(traindata_uncor)=="phishing")]
> testdata_knn_uncor=testdata_uncor[, -which(names(testdata_uncor)=="phishing")]
> trainclass_knn_uncor=factor(traindata_uncor[, which(names(traindata_uncor)=="phishing")])
> testclass_knn_uncor=factor(testdata_uncor[, which(names(testdata_uncor)=="phishing")])
> knn_pred_test_uncor=knn(traindata_knn_uncor,testdata_knn_uncor,trainclass_knn_uncor, k = 243, prob=TRU
E)
> knn_cfm_tst_uncor=confusionMatrix(table(testclass_knn_uncor,knn_pred_test_uncor))
> knn_cfm_tst_uncor
Confusion Matrix and Statistics

                   knn_pred_test_uncor
testclass_knn_uncor    0    1
                  0 5669 2611
                  1 1810 7504

               Accuracy : 0.7487
                 95% CI : (0.7422, 0.7551)
    No Information Rate : 0.5749
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.493

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.7580
            Specificity : 0.7419
         Pos Pred Value : 0.6847
         Neg Pred Value : 0.8057
             Prevalence : 0.4251
         Detection Rate : 0.3222
   Detection Prevalence : 0.4706
      Balanced Accuracy : 0.7499

       'Positive' Class : 0
```

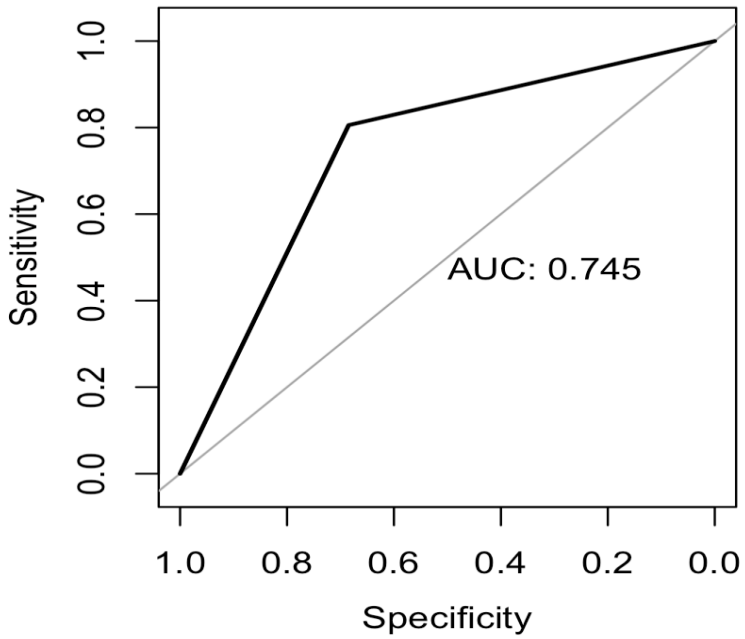
ROC Curve with AUC for the Testing Phase of Dataset1:

```
> knn_AUC_tst = roc(testclass_knn, as.ordered(knn_pred_test))
Setting levels: control = 0, case = 1
Setting direction: controls < cases
> knn_AUC_tst

Call:
roc.default(response = testclass_knn, predictor = as.ordered(knn_pred_test))

Data: as.ordered(knn_pred_test) in 8280 controls (testclass_knn 0) < 9314 cases (testclass_knn 1).
Area under the curve: 0.768
> plot.roc(knn_AUC_tst, plot = TRUE, print.auc = TRUE)
```



ROC Curve with AUC for the Testing Phase of Dataset2:

```
> plot.roc(knn_AUC_tst, plot = TRUE, print.auc = TRUE)
> knn_AUC_tst_uncor = roc(testclass_knn_uncor, as.ordered(knn_pred_test_uncor))
> plot.roc(knn_AUC_tst_uncor, plot = TRUE, print.auc = TRUE)
```



Performance Metrics of the Datasets:

Performance Metrics	Dataset1	Dataset2
Accuracy	77.10	74.87
Sensitivity	77.85	75.80
Specificity	76.52	74.19
Balanced Accuracy	77.19	74.99

The above table indicates the performance metrics obtained by running decision tree classifier on the 2 datasets. The results are summarized by running confusion matrices on the models obtained from training and testing the datasets.

We can see that higher accuracy is obtained for the larger version of the dataset. The results are the same as that predicted from the other models where the larger dataset with more features has higher accuracy.

Also, the percentage decrease between the training and testing models is not significant (> 25%) which means that none of the models run above are overfitting. This means that the models do not have high variance.

Section 6 – NB on 2 Datasets

The next classifier we run is Naïve Bayes. The below implementation shows the implementation of NB classifier on the 2 datasets.

NB classifier on Dataset1

Training Phase Confusion Matrix on Dataset1 using NB:

```
> require(e1071)
> traindata_nb = traindata
> testdata_nb = testdata
> traindata_nb$phishing=as.factor(traindata_nb$phishing)
> dim(traindata_nb)
[1] 41051    79
> nb_model<-naiveBayes(phishing~.,data=traindata_nb)

> nbtr.trpred<-predict(nb_model,traindata_nb[,-c(79)],type='raw')
> nbtr.trclass<-unlist(apply(round(nbtr.trpred),1,which.max))-1
> nbtr.trtbl<-table(nbtr.trclass,traindata_nb[[79]])
> tr.cfm<-caret::confusionMatrix(nbtr.trtbl)
> tr.cfm
Confusion Matrix and Statistics


nbtr.trclass     0     1
           0 19126 15323
           1   592  6010

               Accuracy : 0.6123
                 95% CI : (0.6076, 0.617)
    No Information Rate : 0.5197
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.2448

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9700
            Specificity : 0.2817
         Pos Pred Value : 0.5552
         Neg Pred Value : 0.9103
             Prevalence : 0.4803
         Detection Rate : 0.4659
   Detection Prevalence : 0.8392
      Balanced Accuracy : 0.6258

       'Positive' Class : 0
```

Testing Phase Confusion Matrix on Dataset1 using NB:

```
> nbtr.tspred<-predict(nb_model,testdata_nb[,-c(79)],type='raw')
> nbtr.tsclass<-unlist(apply(round(nbtr.tspred),1,which.max))-1
> nbtr.tstbl<-table(nbtr.tsclass,testdata_nb[[79]])
> tst.cfm<-caret::confusionMatrix(nbtr.tstbl)
> tst.cfm
Confusion Matrix and Statistics


nbtr.tsclass    0    1
           0 8011 6727
           1  269 2587

               Accuracy : 0.6024
                 95% CI : (0.5951, 0.6096)
    No Information Rate : 0.5294
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.2351

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9675
            Specificity : 0.2778
         Pos Pred Value : 0.5436
         Neg Pred Value : 0.9058
             Prevalence : 0.4706
         Detection Rate : 0.4553
   Detection Prevalence : 0.8377
      Balanced Accuracy : 0.6226

       'Positive' Class : 0
```

Training Phase Confusion Matrix on Dataset2 using NB:

```
> traindata_nb_uncor = traindata_uncor
> testdata_nb_uncor = testdata_uncor
> traindata_nb_uncor$phishing=as.factor(traindata_nb_uncor$phishing)
> dim(traindata_nb_uncor)
[1] 41051    37
> nb_model_uncor<-naiveBayes(phishing~.,data=traindata_nb_uncor)
> nbtr.trpred_uncor<-predict(nb_model_uncor,traindata_nb_uncor[,-c(37)],type='raw')
> nbtr.trclass_uncor<-unlist(apply(round(nbtr.trpred_uncor),1,which.max))-1
```

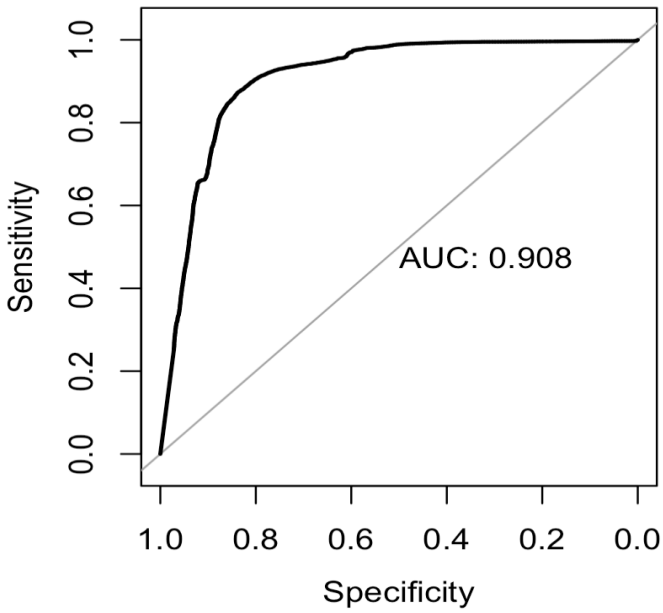
```
> nbtr.trtbl_uncor<-table(nbtr.trclass_uncor,traindata_nb_uncor[[37]])
> tr.cfm_uncor<-caret::confusionMatrix(nbtr.trtbl_uncor)
> tr.cfm_uncor
Confusion Matrix and Statistics


nbtr.trclass_uncor     0     1
                 0 19373 17853
                 1   345  3480

               Accuracy : 0.5567
                 95% CI : (0.5519, 0.5615)
    No Information Rate : 0.5197
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.1409

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9825
            Specificity : 0.1631
         Pos Pred Value : 0.5204
         Neg Pred Value : 0.9098
             Prevalence : 0.4803
         Detection Rate : 0.4719
   Detection Prevalence : 0.9068
      Balanced Accuracy : 0.5728

       'Positive' Class : 0
```

Testing Phase Confusion Matrix on Dataset2 using NB:

```
> nbtr.tspred_uncor<-predict(nb_model_uncor,testdata_nb_uncor[,-c(37)],type='raw')
> nbtr.tsclass_uncor<-unlist(apply(round(nbtr.tspred_uncor),1,which.max))-1
> nbtr.tstbl_uncor<-table(nbtr.tsclass_uncor,testdata_nb_uncor[[37]])
> tst.cfm_uncor<-caret::confusionMatrix(nbtr.tstbl_uncor)
> tst.cfm_uncor
Confusion Matrix and Statistics


nbtr.tsclass_uncor    0    1
                 0 8136 7847
                 1  144 1467

               Accuracy : 0.5458
                 95% CI : (0.5384, 0.5532)
    No Information Rate : 0.5294
    P-Value [Acc > NIR] : 6.485e-06

                  Kappa : 0.1332

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9826
            Specificity : 0.1575
         Pos Pred Value : 0.5090
         Neg Pred Value : 0.9106
             Prevalence : 0.4706
         Detection Rate : 0.4624
   Detection Prevalence : 0.9084
      Balanced Accuracy : 0.5701

       'Positive' Class : 0
```

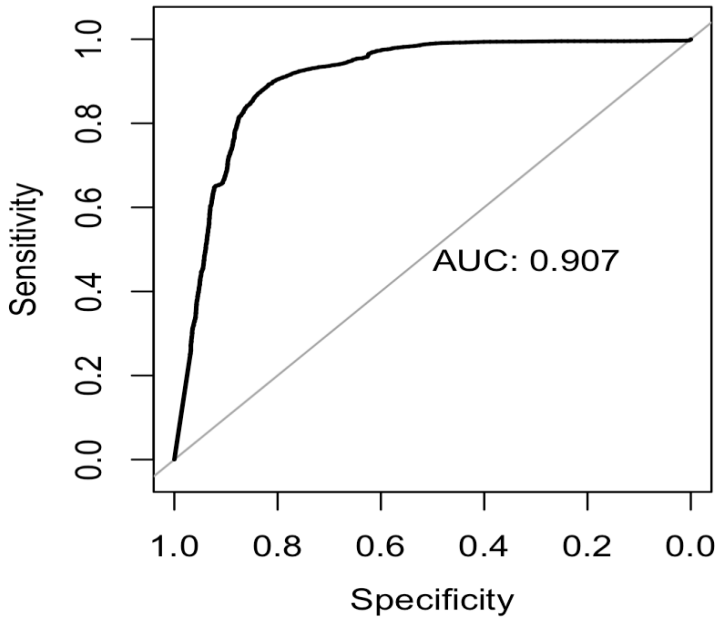
ROC Curve with AUC for the Training Phase of Dataset1:

```
> roc.nbtr.trpred<-nbtr.trpred[,2]
> auc_nb_train = roc(response = traindata_nb[,79], predictor = roc.nbtr.trpred)
Setting levels: control = 0, case = 1
Setting direction: controls < cases
> plot.roc(auc_nb_train, plot = TRUE, print.auc = TRUE)
```



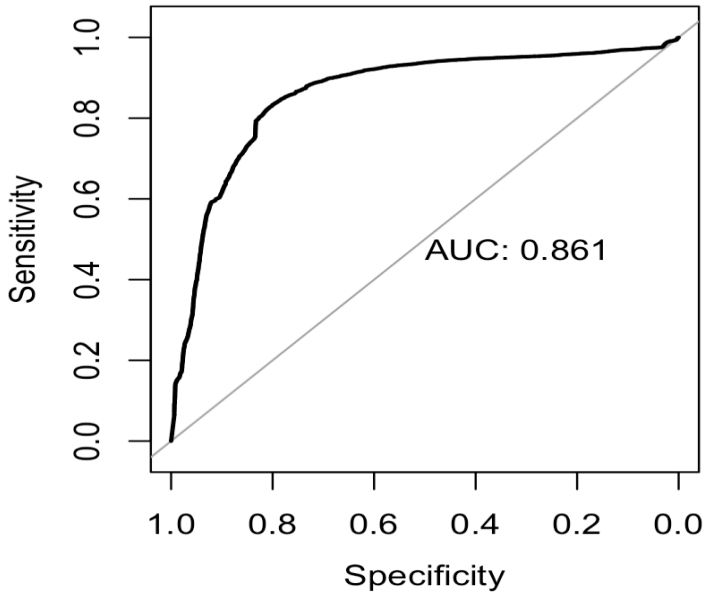
ROC Curve with AUC for the Testing Phase of Dataset1:

```
> roc.nbtr.tspred<-nbtr.tspred[,2]
> auc_nb_test = roc(response = testdata_nb[,79], predictor = roc.nbtr.tspred)
Setting levels: control = 0, case = 1
Setting direction: controls < cases
> plot.roc(auc_nb_test, plot = TRUE, print.auc = TRUE)
```



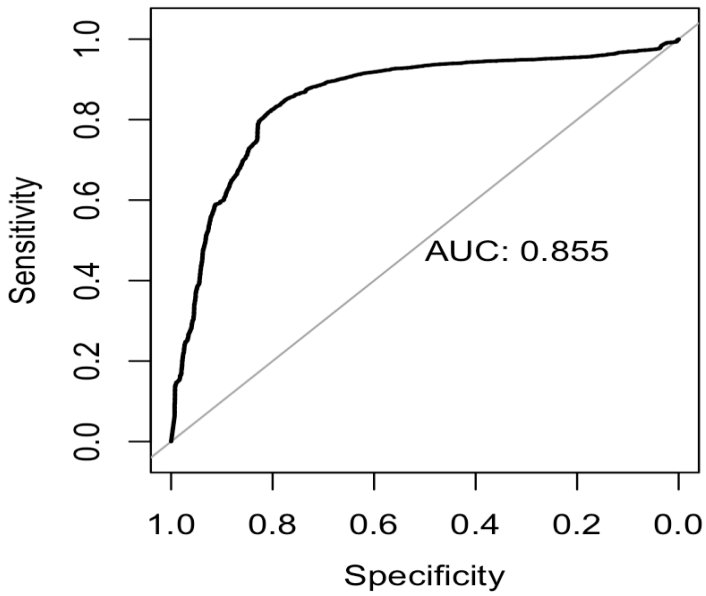
ROC Curve with AUC for the Training Phase of Dataset2:

```
> roc.nbtr.trpred_uncor<-nbtr.trpred_uncor[,2]
> auc_nb_train_uncor = roc(response = traindata_nb_uncor[,37], predictor = roc.nbtr.trpred_uncor)
Setting levels: control = 0, case = 1
Setting direction: controls < cases
> plot.roc(auc_nb_train_uncor, plot = TRUE, print.auc = TRUE)
```



ROC Curve with AUC for the Testing Phase of Dataset2:

```
> roc.nbtr.tspred_uncor<-nbtr.tspred_uncor[,2]
> auc_nb_test_uncor = roc(response = testdata_nb_uncor[,37], predictor = roc.nbtr.tspred_uncor)
Setting levels: control = 0, case = 1
Setting direction: controls < cases
> plot.roc(auc_nb_test_uncor, plot = TRUE, print.auc = TRUE)
```



Performance Metrics of the Datasets:

Performance Metrics	Dataset 1		Dataset2	
	Train	Test	Train	Test
Accuracy	61.23	60.24	55.67	54.58
Sensitivity	97.00	96.75	98.25	98.26
Specificity	28.17	27.78	16.31	15.75
Balanced Accuracy	62.58	62.26	57.28	57.01

The above table indicates the performance metrics obtained by running decision tree classifier on the 2 datasets. The results are summarized by running confusion matrices on the models obtained from training and testing the datasets.

We can see that higher accuracy is obtained for the larger version of the dataset. The same pattern of results continue with this classifier as well. This model does not produce good accuracy since it performs better for multiclass classification models.

Also, the percentage decrease between the training and testing models is not significant (> 25%) which means that none of the models run above are overfitting. This means that the models do not have high variance.

Section 7 – Summary of all Classifiers

We have run a total of 5 classification models on 2 variants of our Dataset. Testing Phase accuracies are tabulated below:

Model	Dataset1	Dataset2
	Testing Accuracy	Testing Accuracy
Logistic Regression	89.12	86.52
Decision Trees	86.65	80.61
SVM	91.84	91.78
KNN	77.10	74.87
Naïve Bayes	60.24	54.58

From the above results, we conclude that the model that is best suited for our dataset is the SVM Model with testing accuracies of 91.84 and 91.78 on datasets 1 and 2 respectively.

Also, a constant observation here is that the Dataset with a greater number of features performed better for every classification model we ran. This raises a question on how large datasets with many features have to be studied and analyzed as removing all correlated features basing it correlation index greater than 0.5 might not be an ideal solution always. Some classification algorithms perform better when there are a few correlated features while some don't.

It is not always ideal to get rid of correlated features in a dataset. If we require the dataset to be improved with respect to performance, then removing correlated features would be a good choice.

Also, we must also observe that none of the models we ran were overfit which means that the models do not have high variance.

We continue with the analysis of the 2 Datasets in HW3 where we run Ensemble Methods to improve the accuracy and performance of our classification models.