HarvardX: PH125.9x - Data Science Own Project Report

Aparajithan Rajendran

February 17, 2019

Introduction

This is a capstone 'Own' project report generated as part of the course 'Data Science' (HarvardX: PH125.9x) at Harvard University in collaboration with edx.

The data set, obtained from UC Irvine Machine Learning repository, consists of the expression levels of 77 proteins/protein modifications that produced detectable signals in the nuclear fraction of cortex. There are 38 control mice and 34 trisomic mice (Down syndrome), for a total of 72 mice. In the experiments, 15 measurements were registered of each protein per sample/mouse. Therefore, for control mice, there are 38x15, or 570 measurements, and for trisomic mice, there are 34x15, or 510 measurements. The dataset contains a total of 1080 measurements per protein. Each measurement can be considered as an independent sample/mouse. The eight classes of mice are described based on features such as genotype, behavior and treatment. According to genotype, mice can be control or trisomic. According to behavior, some mice have been stimulated to learn (context-shock) and others have not (shock-context) and in order to assess the effect of the drug memantine in recovering the ability to learn in trisomic mice, some mice have been injected with the drug and others have not.

Goal

This is a multivariate logistic regression or multinomial classification problem. There are 8 classes of mice observed:

4 types of control mice c-CS-s: control mice, stimulated to learn, injected with saline (9 mice) c-CS-m: control mice, stimulated to learn, injected with memantine (10 mice) c-SC-s: control mice, not stimulated to learn, injected with saline (9 mice) c-SC-m: control mice, not stimulated to learn, injected with memantine (10 mice)

4 types of trisomic mice t-CS-s: trisomy mice, stimulated to learn, injected with saline (7 mice) t-CS-m: trisomy mice, stimulated to learn, injected with memantine (9 mice) t-SC-s: trisomy mice, not stimulated to learn, injected with saline (9 mice) t-SC-m: trisomy mice, not stimulated to learn, injected with memantine (9 mice)

The ultimate aim is to identify subsets of proteins out of 77 proteins/protein modifications that are discriminant between the classes.

Key Steps

There are four key steps performed in order to build recommendation system that could predict ratings on the validation set.

Step #1: Initial Data Setup

Step #2: Apply RandomForest algorithm and Variable Importance method to identify discriminant features

Step #3: Apply RandomForest algorithm and Principle Component Analysis for dimensionality reduction

Step #4: Cross verify the discriminant features to conclude the results

---

Methodology

The above stated steps are explained in a detail manner below.

Step #1: Initial Data Setup

The following steps are performed as part of initial/one-time data setup

- Task #1: Downloading if the cleaned data set is not found in the working directory

- Task #2: Data Exploration
- Task #3: Data Cleaning
- Task #4: Restoring Cleaned Data for future reuse

```
if (file.exists("edx_cleaned.Rda")) {
    load("edx_cleaned.Rda")
} else {
    # The following code was provided by the instructors:

    ############################################################
    # Create edx set
    ############################################################

    # Note: this process could take a couple of minutes

    if(!require(randomForest)) install.packages("randomForest", repos = "http://cran.us.r-project.org")
    if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
    if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
    if(!require(readxl)) install.packages("readxl", repos = "http://cran.us.r-project.org")

    # Mice Cortex Nuclear 1080 dataset:
    # https://archive.ics.uci.edu/ml/machine-learning-databases/00342/Data_Cortex_Nuclear.xls

    # Task #1: Downloading if the cleaned data set is not found in the working directory
    URL <- "https://archive.ics.uci.edu/ml/machine-learning-databases/00342/Data_Cortex_Nuclear.xls"
    fileName <- "/Data_Cortex_Nuclear.xls"
    download.file(URL, paste(getwd(),fileName, sep = ""), mode="wb")

    # Please place the file directly under your documents directory or wherever the getwd() is showing below
    print("The XLS file is placed under directory:")
    print(paste(getwd(),fileName, sep = ""))

    library(readxl)
    edx <- read_excel(paste(getwd(),fileName, sep = ""))

    # Task #2: Data Exploration
    print(dim(edx))

    print(names(edx))

    # Task #3: Data Cleaning
    edx_cleaned <- edx %>% mutate(MouseID = as.numeric(as.factor(MouseID)), Genotype = as.numeric(as.factor(Genotype)), Tre
atment = as.numeric(as.factor(Treatment)), Behavior = as.numeric(as.factor(Behavior)), class = as.factor(class))

    for(i in 2:ncol(edx_cleaned)-1){
        edx_cleaned[is.na(edx_cleaned[,i]), i] <- 0
    }

    # To make sure that there is zero NA values
    sum(is.na(edx_cleaned))

    # Task #4: Restoring Cleaned Data for future reuse
    ## The next line was introduced by me to avoid creating the datasets every time
    save(edx_cleaned, file = "edx_cleaned.Rda")
}
```

```
## Loading required package: randomForest
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
## Loading required package: tidyverse
```

```
## -- Attaching packages ---------------------------------------------------------------------------------------------
------- tidyverse 1.2.1 --
```

```
## v ggplot2 3.1.0     v purrr   0.2.5
## v tibble  2.0.1     v dplyr   0.7.8
## v tidyr   0.8.2     v stringr 1.3.1
## v readr   1.3.1     v forcats 0.3.0
```

```
## -- Conflicts ------------------------------------------------------------------------------------------------------
- tidyverse_conflicts() --
## x dplyr::combine()  masks randomForest::combine()
## x dplyr::filter()   masks stats::filter()
## x dplyr::lag()      masks stats::lag()
## x ggplot2::margin() masks randomForest::margin()
```

```
## Loading required package: caret
```

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

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
## Loading required package: readxl
```

```
## [1] "The XLS file is placed under directory:"
## [1] "C:/Users/apar rajendran/Documents/1-Apar/2-Personal/Classes/DS_Harvard/Data_Cortex_Nuclear.xls"
## [1] 1080   82
##  [1] "MouseID"         "DYRK1A_N"        "ITSN1_N"
##  [4] "BDNF_N"          "NR1_N"           "NR2A_N"
##  [7] "pAKT_N"          "pBRAF_N"         "pCAMKII_N"
## [10] "pCREB_N"         "pELK_N"          "pERK_N"
## [13] "pJNK_N"          "PKCA_N"          "pMEK_N"
## [16] "pNR1_N"          "pNR2A_N"         "pNR2B_N"
## [19] "pPKCAB_N"        "pRSK_N"          "AKT_N"
## [22] "BRAF_N"          "CAMKII_N"        "CREB_N"
## [25] "ELK_N"           "ERK_N"           "GSK3B_N"
## [28] "JNK_N"           "MEK_N"           "TRKA_N"
## [31] "RSK_N"           "APP_N"           "Bcatenin_N"
## [34] "SOD1_N"          "MTOR_N"          "P38_N"
## [37] "pMTOR_N"         "DSCR1_N"         "AMPKA_N"
## [40] "NR2B_N"          "pNUMB_N"         "RAPTOR_N"
## [43] "TIAM1_N"         "pP70S6_N"        "NUMB_N"
## [46] "P70S6_N"         "pGSK3B_N"        "pPKCG_N"
## [49] "CDK5_N"          "S6_N"            "ADARB1_N"
## [52] "AcetylH3K9_N"    "RRP1_N"          "BAX_N"
## [55] "ARC_N"           "ERBB4_N"         "nNOS_N"
## [58] "Tau_N"           "GFAP_N"          "GluR3_N"
## [61] "GluR4_N"         "IL1B_N"          "P3525_N"
## [64] "pCASP9_N"        "PSD95_N"         "SNCA_N"
## [67] "Ubiquitin_N"     "pGSK3B_Tyr216_N" "SHH_N"
## [70] "BAD_N"           "BCL2_N"          "pS6_N"
## [73] "pCFOS_N"         "SYP_N"           "H3AcK18_N"
## [76] "EGR1_N"          "H3MeK4_N"        "CaNA_N"
## [79] "Genotype"        "Treatment"       "Behavior"
## [82] "class"
```

```
### Loading packages
library(randomForest)
library(tidyverse)
library(caret)
```

- Task #5: Data Partioning into Training (70%) & Test (30%) datasets

```
# Step #5: Data Partioning into Training (70%) & Test (30%) datasets
ind <- sample(2, nrow(edx_cleaned), replace=TRUE, prob=c(0.7,0.3))
trainData_x <- edx_cleaned[ind == 1,] %>% dplyr::select(-MouseID, -Genotype, -Treatment, -Behavior)
trainData_y <- trainData_x$class
trainData_x <- trainData_x %>% dplyr::select(-class)
testData_x <- edx_cleaned[ind == 2,] %>% dplyr::select(-MouseID, -Genotype, -Treatment, -Behavior)
testData_y <- testData_x$class
testData_x <- testData_x %>% dplyr::select(-class)
```

Step #2: Apply RandomForest algorithm and Variable Importance method to identify discriminant features

The following steps are performed as part of model training & feature selection

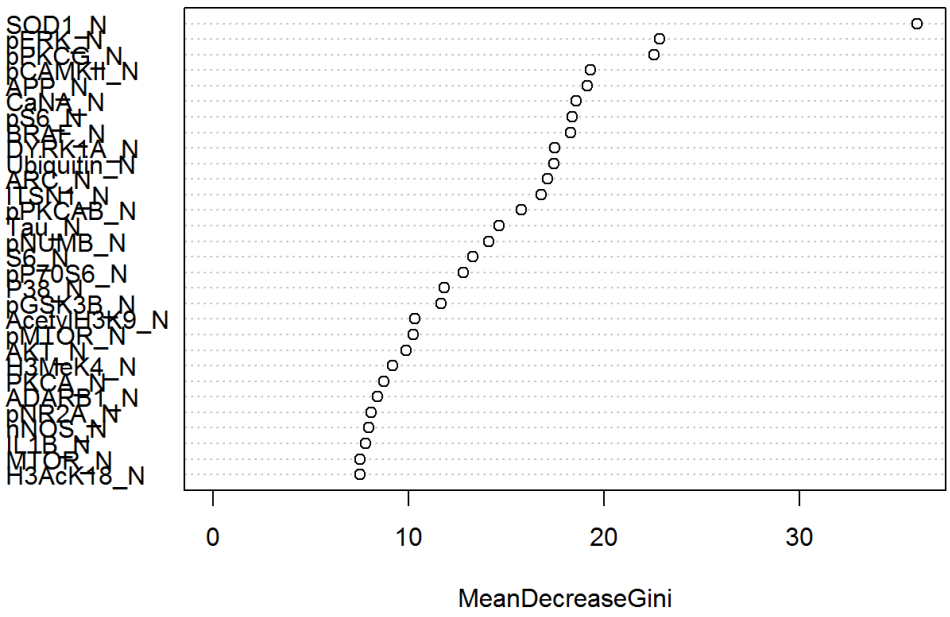
- Task #6: Train ML model for the entire set of features using RandomForest algorithm

```
# Task #6: Train ML model for the entire set of features using RandomForest algorithm
model_rf <- randomForest(trainData_y ~. , data = trainData_x, proximity=TRUE)
confusionMatrix(predict(model_rf), trainData_y)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction c-CS-m c-CS-s c-SC-m c-SC-s t-CS-m t-CS-s t-SC-m t-SC-s
##     c-CS-m     95      2      0      0      0      0      0      0
##     c-CS-s      1     93      0      0      1      0      0      0
##     c-SC-m      0      0     94      0      0      0      1      0
##     c-SC-s      0      0      0     98      0      0      0      0
##     t-CS-m      0      0      0      0    100      1      0      0
##     t-CS-s      0      1      0      0      0     81      0      0
##     t-SC-m      0      0      2      0      0      0     96      0
##     t-SC-s      0      0      0      0      0      0      0     94
##
## Overall Statistics
##
##                Accuracy : 0.9882
##                  95% CI : (0.9776, 0.9946)
##     No Information Rate : 0.1329
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9865
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: c-CS-m Class: c-CS-s Class: c-SC-m
## Sensitivity                 0.9896        0.9688        0.9792
## Specificity                 0.9970        0.9970        0.9985
## Pos Pred Value              0.9794        0.9789        0.9895
## Neg Pred Value              0.9985        0.9955        0.9970
## Prevalence                  0.1263        0.1263        0.1263
## Detection Rate              0.1250        0.1224        0.1237
## Detection Prevalence        0.1276        0.1250        0.1250
## Balanced Accuracy           0.9933        0.9829        0.9888
##                      Class: c-SC-s Class: t-CS-m Class: t-CS-s
## Sensitivity                 1.0000        0.9901        0.9878
## Specificity                 1.0000        0.9985        0.9985
## Pos Pred Value              1.0000        0.9901        0.9878
## Neg Pred Value              1.0000        0.9985        0.9985
## Prevalence                  0.1289        0.1329        0.1079
## Detection Rate              0.1289        0.1316        0.1066
## Detection Prevalence        0.1289        0.1329        0.1079
## Balanced Accuracy           1.0000        0.9943        0.9932
##                      Class: t-SC-m Class: t-SC-s
## Sensitivity                 0.9897        1.0000
## Specificity                 0.9970        1.0000
## Pos Pred Value              0.9796        1.0000
## Neg Pred Value              0.9985        1.0000
## Prevalence                  0.1276        0.1237
## Detection Rate              0.1263        0.1237
## Detection Prevalence        0.1289        0.1237
## Balanced Accuracy           0.9933        1.0000
```

- Task #7: Identify the subset of proteins which are discriminant by variable importance > 10

```
# Task #7: Identify the subset of proteins which are discriminant by variable importance > 10
varImpPlot(model_rf, main='Variable Importance Plot: Base Model')
```




```
imp <- importance(model_rf)
protein_imp <- data.frame(protein = rownames(imp), meanVal = imp)
protein_iv_10 <- protein_imp[which(protein_imp[order(-protein_imp$MeanDecreaseGini),]$MeanDecreaseGini > 10),]$protein
iv.used_10 <- length(protein_iv_10)
```

- Task #8: Train ML model only for the subset of proteins which are discriminant

```
# Task #8: Train ML model only for the subset of proteins which are discriminant
trainData_x_iv_10 <- trainData_x %>% dplyr::select(protein_iv_10)
model_rf_iv_10 <- randomForest(trainData_y ~., data=trainData_x_iv_10, proximity=TRUE)
confusionMatrix(predict(model_rf_iv_10), trainData_y)
```

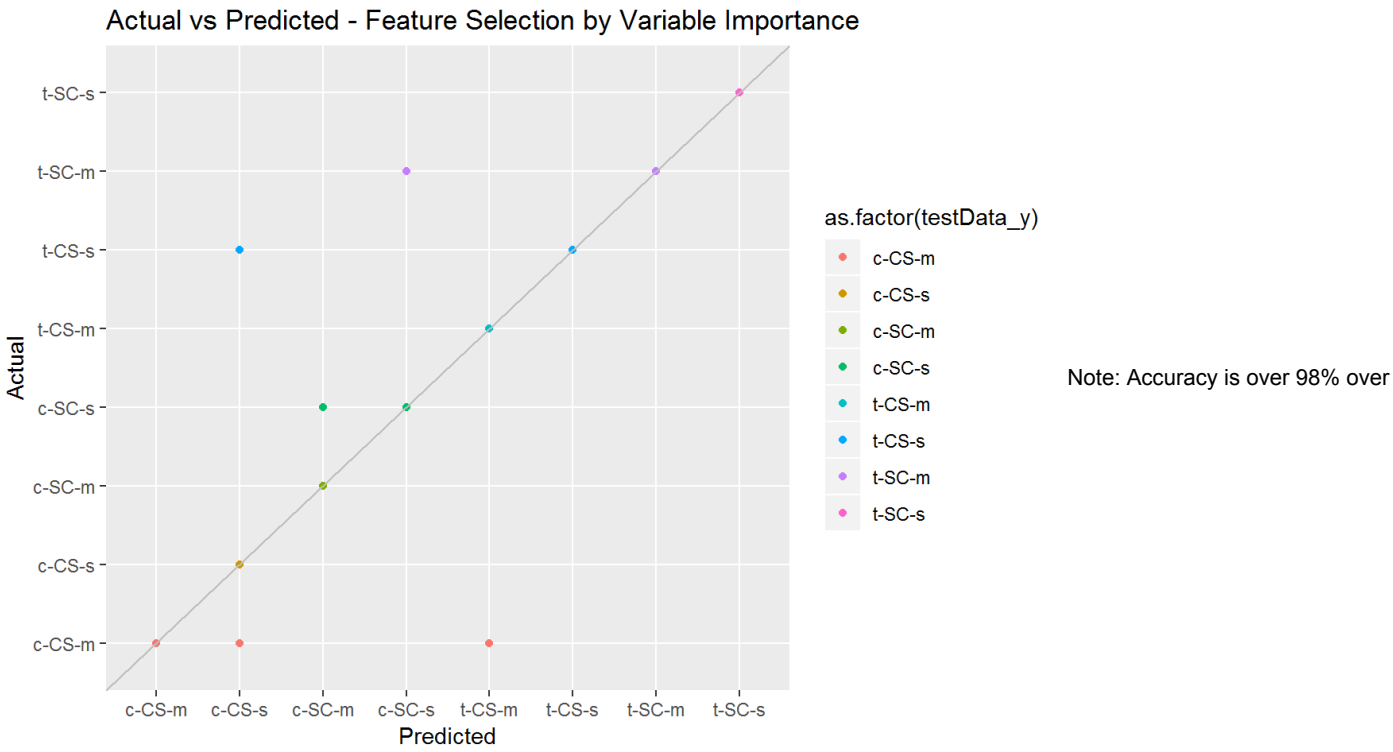
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction c-CS-m c-CS-s c-SC-m c-SC-s t-CS-m t-CS-s t-SC-m t-SC-s
##     c-CS-m     94      0      0      0      0      0      0      0
##     c-CS-s      1     93      0      0      3      0      0      0
##     c-SC-m      0      0     92      1      0      0      3      0
##     c-SC-s      0      0      0     95      0      0      1      0
##     t-CS-m      1      0      0      0     97      1      0      0
##     t-CS-s      0      2      0      0      0     81      0      0
##     t-SC-m      0      0      4      2      0      0     93      0
##     t-SC-s      0      1      0      0      1      0      0     94
##
## Overall Statistics
##
##                Accuracy : 0.9724
##                  95% CI : (0.9581, 0.9828)
##     No Information Rate : 0.1329
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9684
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: c-CS-m Class: c-CS-s Class: c-SC-m
## Sensitivity                 0.9792        0.9688        0.9583
## Specificity                 1.0000        0.9940        0.9940
## Pos Pred Value              1.0000        0.9588        0.9583
## Neg Pred Value              0.9970        0.9955        0.9940
## Prevalence                  0.1263        0.1263        0.1263
## Detection Rate              0.1237        0.1224        0.1211
## Detection Prevalence        0.1237        0.1276        0.1263
## Balanced Accuracy           0.9896        0.9814        0.9762
##                      Class: c-SC-s Class: t-CS-m Class: t-CS-s
## Sensitivity                 0.9694        0.9604        0.9878
## Specificity                 0.9985        0.9970        0.9971
## Pos Pred Value              0.9896        0.9798        0.9759
## Neg Pred Value              0.9955        0.9939        0.9985
## Prevalence                  0.1289        0.1329        0.1079
## Detection Rate              0.1250        0.1276        0.1066
## Detection Prevalence        0.1263        0.1303        0.1092
## Balanced Accuracy           0.9839        0.9787        0.9924
##                      Class: t-SC-m Class: t-SC-s
## Sensitivity                 0.9588        1.0000
## Specificity                 0.9910        0.9970
## Pos Pred Value              0.9394        0.9792
## Neg Pred Value              0.9939        1.0000
## Prevalence                  0.1276        0.1237
## Detection Rate              0.1224        0.1237
## Detection Prevalence        0.1303        0.1263
## Balanced Accuracy           0.9749        0.9985
```

- Step #9: Run predictions on the test data using the trained model

```
# Task #9: Run predictions on the test data using the trained model
testData_x_iv_10 <- testData_x %>% dplyr::select(protein_iv_10)
predictions_test_y_iv_10 <- predict(model_rf_iv_10, testData_x_iv_10)
confusionMatrix(predictions_test_y_iv_10, testData_y)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction c-CS-m c-CS-s c-SC-m c-SC-s t-CS-m t-CS-s t-SC-m t-SC-s
##     c-CS-m     52      0      0      0      0      0      0      0
##     c-CS-s      1     39      0      0      0      1      0      0
##     c-SC-m      0      0     54      1      0      0      0      0
##     c-SC-s      0      0      0     36      0      0      1      0
##     t-CS-m      1      0      0      0     34      0      0      0
##     t-CS-s      0      0      0      0      0     22      0      0
##     t-SC-m      0      0      0      0      0      0     37      0
##     t-SC-s      0      0      0      0      0      0      0     41
##
## Overall Statistics
##
##                Accuracy : 0.9844
##                  95% CI : (0.9639, 0.9949)
##     No Information Rate : 0.1688
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.982
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: c-CS-m Class: c-CS-s Class: c-SC-m
## Sensitivity                 0.9630        1.0000        1.0000
## Specificity                 1.0000        0.9929        0.9962
## Pos Pred Value              1.0000        0.9512        0.9818
## Neg Pred Value              0.9925        1.0000        1.0000
## Prevalence                  0.1688        0.1219        0.1688
## Detection Rate              0.1625        0.1219        0.1688
## Detection Prevalence        0.1625        0.1281        0.1719
## Balanced Accuracy           0.9815        0.9964        0.9981
##                      Class: c-SC-s Class: t-CS-m Class: t-CS-s
## Sensitivity                 0.9730        1.0000       0.95652
## Specificity                 0.9965        0.9965       1.00000
## Pos Pred Value              0.9730        0.9714       1.00000
## Neg Pred Value              0.9965        1.0000       0.99664
## Prevalence                  0.1156        0.1062       0.07187
## Detection Rate              0.1125        0.1062       0.06875
## Detection Prevalence        0.1156        0.1094       0.06875
## Balanced Accuracy           0.9847        0.9983       0.97826
##                      Class: t-SC-m Class: t-SC-s
## Sensitivity                 0.9737        1.0000
## Specificity                 1.0000        1.0000
## Pos Pred Value              1.0000        1.0000
## Neg Pred Value              0.9965        1.0000
## Prevalence                  0.1187        0.1281
## Detection Rate              0.1156        0.1281
## Detection Prevalence        0.1156        0.1281
## Balanced Accuracy           0.9868        1.0000
```

```
ggplot(testData_x_iv_10)+
    geom_point(aes(y=testData_y, x=predictions_test_y_iv_10, color=as.factor(testData_y)))+
    ylab('Actual')+
    xlab('Predicted')+
    ggtitle('Actual vs Predicted - Feature Selection by Variable Importance')+
    geom_abline(colour="grey")
```



Note: Accuracy is over 98% over test dataset with 21 protein

Step #3: Apply RandomForest algorithm and Principle Component Analysis for dimensionality reduction

- Task #10: Applying PCA to cross verify the selection of the subset of proteins which are discriminant

```
# Task #10: Applying PCA to cross verify the selection of the subset of proteins which are discriminant
trainData_x_pca <- prcomp(trainData_x, center = TRUE, scale. = TRUE)
```

- Task #11: Exploring PCA and Variance Explained

```
# Task #11: Exploring PCA and Variance Explained
summary(trainData_x_pca)
```

```
## Importance of components:
##                            PC1    PC2    PC3     PC4     PC5     PC6
## Standard deviation     4.5179 3.3454 2.8321 2.34687 1.92918 1.85790
## Proportion of Variance 0.2651 0.1453 0.1042 0.07153 0.04833 0.04483
## Cumulative Proportion  0.2651 0.4104 0.5146 0.58612 0.63445 0.67928
##                            PC7     PC8     PC9    PC10    PC11    PC12
## Standard deviation     1.61053 1.55806 1.34034 1.25839 1.14551 1.04712
## Proportion of Variance 0.03369 0.03153 0.02333 0.02057 0.01704 0.01424
## Cumulative Proportion  0.71297 0.74449 0.76783 0.78839 0.80543 0.81967
##                           PC13    PC14   PC15    PC16    PC17    PC18
## Standard deviation     1.0158 0.96098 0.8991 0.86656 0.81464 0.80493
## Proportion of Variance 0.0134 0.01199 0.0105 0.00975 0.00862 0.00841
## Cumulative Proportion  0.8331 0.84506 0.8556 0.86531 0.87393 0.88235
##                            PC19    PC20    PC21    PC22    PC23    PC24
## Standard deviation     0.73802 0.73633 0.71926 0.68596 0.67715 0.64715
## Proportion of Variance 0.00707 0.00704 0.00672 0.00611 0.00595 0.00544
## Cumulative Proportion  0.88942 0.89646 0.90318 0.90929 0.91525 0.92069
##                            PC25    PC26   PC27    PC28    PC29    PC30
## Standard deviation     0.61985 0.61172 0.5952 0.57299 0.54516 0.53311
## Proportion of Variance 0.00499 0.00486 0.0046 0.00426 0.00386 0.00369
## Cumulative Proportion  0.92568 0.93054 0.9351 0.93940 0.94326 0.94695
##                           PC31    PC32    PC33    PC34    PC35    PC36
## Standard deviation     0.5116 0.50479 0.47344 0.46207 0.43730 0.42837
## Proportion of Variance 0.0034 0.00331 0.00291 0.00277 0.00248 0.00238
## Cumulative Proportion  0.9504 0.95366 0.95657 0.95934 0.96183 0.96421
##                            PC37    PC38    PC39    PC40    PC41    PC42
## Standard deviation     0.41260 0.39610 0.37899 0.36996 0.36400 0.35737
## Proportion of Variance 0.00221 0.00204 0.00187 0.00178 0.00172 0.00166
## Cumulative Proportion  0.96642 0.96846 0.97032 0.97210 0.97382 0.97548
##                            PC43    PC44    PC45    PC46    PC47    PC48
## Standard deviation     0.34699 0.34043 0.33107 0.31551 0.31222 0.30637
## Proportion of Variance 0.00156 0.00151 0.00142 0.00129 0.00127 0.00122
## Cumulative Proportion  0.97704 0.97855 0.97997 0.98127 0.98253 0.98375
##                            PC49    PC50    PC51    PC52    PC53    PC54
## Standard deviation     0.30058 0.28645 0.28497 0.27592 0.26552 0.25685
## Proportion of Variance 0.00117 0.00107 0.00105 0.00099 0.00092 0.00086
## Cumulative Proportion  0.98492 0.98599 0.98704 0.98803 0.98895 0.98981
##                            PC55    PC56    PC57    PC58    PC59    PC60
## Standard deviation     0.24996 0.24574 0.24161 0.23632 0.22476 0.21849
## Proportion of Variance 0.00081 0.00078 0.00076 0.00073 0.00066 0.00062
## Cumulative Proportion  0.99062 0.99140 0.99216 0.99288 0.99354 0.99416
##                           PC61    PC62    PC63    PC64    PC65    PC66
## Standard deviation     0.2152 0.21122 0.20235 0.19304 0.18911 0.18603
## Proportion of Variance 0.0006 0.00058 0.00053 0.00048 0.00046 0.00045
## Cumulative Proportion  0.9948 0.99534 0.99587 0.99636 0.99682 0.99727
##                            PC67   PC68    PC69   PC70    PC71    PC72
## Standard deviation     0.17750 0.1754 0.17108 0.1521 0.14870 0.14466
## Proportion of Variance 0.00041 0.0004 0.00038 0.0003 0.00029 0.00027
## Cumulative Proportion  0.99768 0.9981 0.99846 0.9988 0.99905 0.99932
##                            PC73    PC74    PC75    PC76      PC77
## Standard deviation     0.12729 0.11901 0.11127 0.09834 3.003e-16
## Proportion of Variance 0.00021 0.00018 0.00016 0.00013 0.000e+00
## Cumulative Proportion  0.99953 0.99971 0.99987 1.00000 1.000e+00
```

Note: The summary shows that, by PC21, the proportion of the variance explained is more than 90%

```
str(trainData_x_pca)
```

```
## List of 5
##  $ sdev    : num [1:77] 4.52 3.35 2.83 2.35 1.93 ...
##  $ rotation: num [1:77, 1:77] -0.0415 -0.0679 -0.1907 -0.1888 -0.1698 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:77] "DYRK1A_N" "ITSN1_N" "BDNF_N" "NR1_N" ...
##   .. ..$ : chr [1:77] "PC1" "PC2" "PC3" "PC4" ...
##  $ center  : Named num [1:77] 0.426 0.619 0.319 2.289 3.819 ...
##   ..- attr(*, "names")= chr [1:77] "DYRK1A_N" "ITSN1_N" "BDNF_N" "NR1_N" ...
##  $ scale   : Named num [1:77] 0.246 0.2571 0.0524 0.3642 0.9421 ...
##   ..- attr(*, "names")= chr [1:77] "DYRK1A_N" "ITSN1_N" "BDNF_N" "NR1_N" ...
##  $ x       : num [1:760, 1:77] -2.456 -2.531 -0.476 0.62 0.797 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr [1:77] "PC1" "PC2" "PC3" "PC4" ...
##  - attr(*, "class")= chr "prcomp"
```

```
var_explained <- cumsum(trainData_x_pca$sdev^2/sum(trainData_x_pca$sdev^2))
```

Note: str function gives the glimpse of the features such as “DYRK1A_N” “ITSN1_N” “BDNF_N” “NR1_N” which are the top four of 20 features that explain the 90% variances

- Task #12: Identify the subset of proteins which are discriminant by PCA with variance explained > 90%

```
# Task #12: Identify the subset of proteins which are discriminant by PCA with variance explained > 90%
pc.used_90 <- which(var_explained>=0.90)[1]
trainData_x_pc_90 <- trainData_x[,1:pc.used_90]
protein_pc_90 <- colnames(trainData_x_pc_90)
```

- Task #13: Train ML model for the subset of proteins (identified by PCA) using RandomForest algorithm

```
# Task #13: Train ML model for the subset of proteins (identified by PCA) using RandomForest algorithm
model_rf_pc_90 <- randomForest(trainData_y ~., data=trainData_x_pc_90, proximity=TRUE)
confusionMatrix(predict(model_rf_pc_90), trainData_y)
```

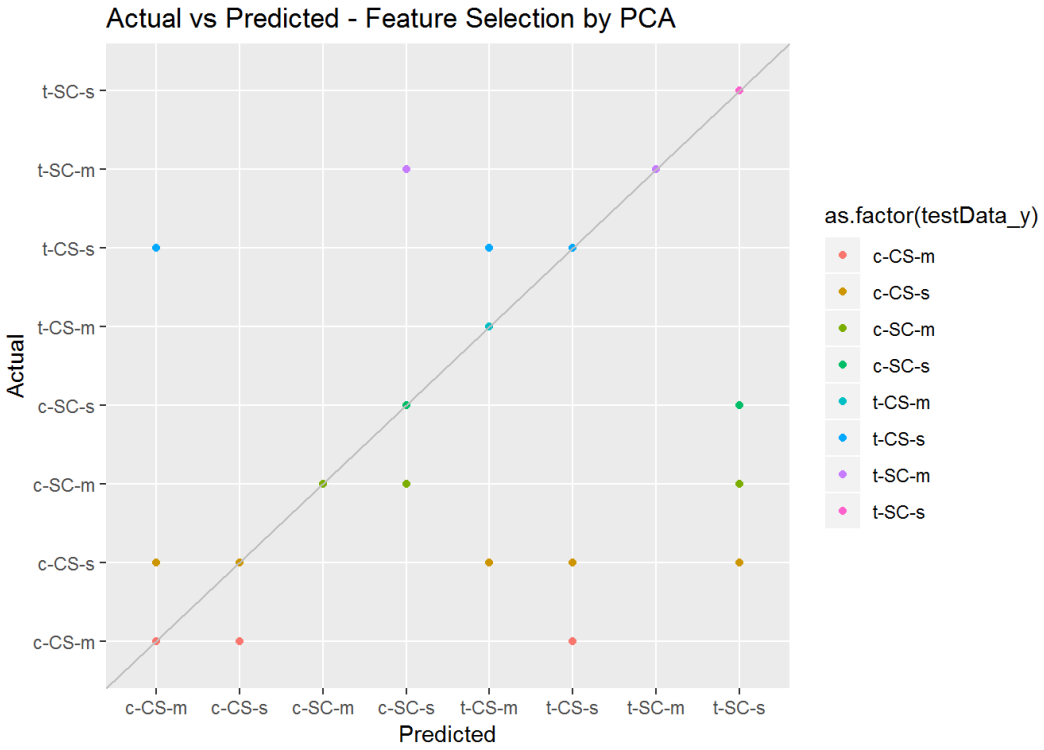
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction c-CS-m c-CS-s c-SC-m c-SC-s t-CS-m t-CS-s t-SC-m t-SC-s
##     c-CS-m     87      7      0      0      2      1      0      0
##     c-CS-s      6     86      0      0      1      2      0      0
##     c-SC-m      0      0     89      0      0      0      3      2
##     c-SC-s      0      0      0     96      0      0      2      0
##     t-CS-m      0      0      0      0     96      2      0      0
##     t-CS-s      3      3      0      0      2     77      0      0
##     t-SC-m      0      0      6      0      0      0     92      0
##     t-SC-s      0      0      1      2      0      0      0     92
##
## Overall Statistics
##
##                Accuracy : 0.9408
##                  95% CI : (0.9216, 0.9565)
##     No Information Rate : 0.1329
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9323
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: c-CS-m Class: c-CS-s Class: c-SC-m
## Sensitivity                 0.9062        0.8958        0.9271
## Specificity                 0.9849        0.9864        0.9925
## Pos Pred Value              0.8969        0.9053        0.9468
## Neg Pred Value              0.9864        0.9850        0.9895
## Prevalence                  0.1263        0.1263        0.1263
## Detection Rate              0.1145        0.1132        0.1171
## Detection Prevalence        0.1276        0.1250        0.1237
## Balanced Accuracy           0.9456        0.9411        0.9598
##                      Class: c-SC-s Class: t-CS-m Class: t-CS-s
## Sensitivity                 0.9796        0.9505        0.9390
## Specificity                 0.9970        0.9970        0.9882
## Pos Pred Value              0.9796        0.9796        0.9059
## Neg Pred Value              0.9970        0.9924        0.9926
## Prevalence                  0.1289        0.1329        0.1079
## Detection Rate              0.1263        0.1263        0.1013
## Detection Prevalence        0.1289        0.1289        0.1118
## Balanced Accuracy           0.9883        0.9737        0.9636
##                      Class: t-SC-m Class: t-SC-s
## Sensitivity                 0.9485        0.9787
## Specificity                 0.9910        0.9955
## Pos Pred Value              0.9388        0.9684
## Neg Pred Value              0.9924        0.9970
## Prevalence                  0.1276        0.1237
## Detection Rate              0.1211        0.1211
## Detection Prevalence        0.1289        0.1250
## Balanced Accuracy           0.9697        0.9871
```

- Task #14: Run predictions on the test data using the trained model

```
# Task #14: Run predictions on the test data using the trained model
testData_x_pc_90 <- testData_x[,1:pc.used_90]
predictions_test_y_pc_90 <- predict(model_rf_pc_90, testData_x_pc_90)
confusionMatrix(predictions_test_y_pc_90, testData_y)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction c-CS-m c-CS-s c-SC-m c-SC-s t-CS-m t-CS-s t-SC-m t-SC-s
##     c-CS-m     52      2      0      0      0      1      0      0
##     c-CS-s      1     33      0      0      0      0      0      0
##     c-SC-m      0      0     51      0      0      0      0      0
##     c-SC-s      0      0      2     36      0      0      2      0
##     t-CS-m      0      1      0      0     34      2      0      0
##     t-CS-s      1      2      0      0      0     20      0      0
##     t-SC-m      0      0      0      0      0      0     36      0
##     t-SC-s      0      1      1      1      0      0      0     41
##
## Overall Statistics
##
##                Accuracy : 0.9469
##                  95% CI : (0.9163, 0.9688)
##     No Information Rate : 0.1688
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9388
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: c-CS-m Class: c-CS-s Class: c-SC-m
## Sensitivity                 0.9630        0.8462        0.9444
## Specificity                 0.9887        0.9964        1.0000
## Pos Pred Value              0.9455        0.9706        1.0000
## Neg Pred Value              0.9925        0.9790        0.9888
## Prevalence                  0.1688        0.1219        0.1688
## Detection Rate              0.1625        0.1031        0.1594
## Detection Prevalence        0.1719        0.1062        0.1594
## Balanced Accuracy           0.9758        0.9213        0.9722
##                      Class: c-SC-s Class: t-CS-m Class: t-CS-s
## Sensitivity                 0.9730        1.0000       0.86957
## Specificity                 0.9859        0.9895       0.98990
## Pos Pred Value              0.9000        0.9189       0.86957
## Neg Pred Value              0.9964        1.0000       0.98990
## Prevalence                  0.1156        0.1062       0.07187
## Detection Rate              0.1125        0.1062       0.06250
## Detection Prevalence        0.1250        0.1156       0.07187
## Balanced Accuracy           0.9794        0.9948       0.92973
##                      Class: t-SC-m Class: t-SC-s
## Sensitivity                 0.9474        1.0000
## Specificity                 1.0000        0.9892
## Pos Pred Value              1.0000        0.9318
## Neg Pred Value              0.9930        1.0000
## Prevalence                  0.1187        0.1281
## Detection Rate              0.1125        0.1281
## Detection Prevalence        0.1125        0.1375
## Balanced Accuracy           0.9737        0.9946
```

```
ggplot(testData_x_pc_90)+
    geom_point(aes(y=testData_y, x=predictions_test_y_pc_90, color=as.factor(testData_y)))+
    ylab('Actual')+
    xlab('Predicted')+
    ggtitle('Actual vs Predicted - Feature Selection by PCA')+
    geom_abline(colour="grey")
```



Results

Step #4: Cross verify the discriminant features to conclude the results

- Task #15: Obtain results by comparing the discriminant proteins identified by both processes

```
# Task #15: Obtain results by comparing the top 20 discriminant proteins identified by both processes
sum(head(protein_iv_10, 20) == head(protein_pc_90, 20))
```

```
## [1] 20
```

Note: Matching number of Proteins/Protein modifications 20

Conclusion

Feature selection by Variable Importance seems to yield little bit better accuracy compared to the one by PCA but that is obvious outcome due to consideration of the additional features by the former method. However, both methods identify the same set of top 20 discriminant proteins/protein modifications out of 77 successfully. Here is the list:

```
head(protein_pc_90, 20)
```

```
##  [1] "DYRK1A_N"  "ITSN1_N"   "BDNF_N"    "NR1_N"     "NR2A_N"
##  [6] "pAKT_N"    "pBRAF_N"   "pCAMKII_N" "pCREB_N"   "pELK_N"
## [11] "pERK_N"    "pJNK_N"    "PKCA_N"    "pMEK_N"    "pNR1_N"
## [16] "pNR2A_N"   "pNR2B_N"   "pPKCAB_N"  "pRSK_N"    "AKT_N"
```

End