Predicting Classes via Protein Expression Levels

Yigit Kalyoncu

12/24/2020

OVERVIEW:

The purposes of this project is to produce machine learning algorithms that successfully predict the classification of an instance in data.

The data consists of expression levels of 77 proteins/protein modifications of mice as predictors. There are 3 sub-classes; which are Genotype, Treatment Type & Behaviour, and one main class, called Class, which is the combination of the sub-classes. Ultimately, this is the class that is to be predicted.

Since the computer that is used for this project has limited capability, a data set with fewer instances is chosen in order to make the computations required by some of the models feasible.

It is also important to note that, the project is not conducted with a thorough understanding of the biological aspects of the data used, but purely with a data analysis point of view. Only the basic information provided by the source of the data is taken into account.

The following steps are taken in order to complete the task:

- Data is loaded from an Url
- Preliminary data exploration is performed
- Data is cleaned and prepared based on the inital exploration
- Further data exploration is performed and several vizualizations are created to better understand the data in hand, and to decide what methods are to be used
- A model based on guessing is produced to be used as a benchmark
- A total of 9 models in 3 main groups are produced and analyzed
- A final ensemble model is created using the most successful models
- The results are analyzed and discussed

INTRODUCING THE DATA:

The data is created using the tests performed on 72 mice. 15 measurements per mouse were performed, which yielded a total of 1080 instances. For the purposes of this project, each instance is considered an independent sample.

Upon examining the structure of the data below, it is understood that there are a total of 82 variables.

MouseID is a factor with 1080 levels. It is the a unique Id given to each sample.

Genotype is a factor with 2 levels, Control and Ts65Dn(Trisomic). Treatment is a factor with 2 levels, Memantine and Saline. Behaviour is a factor with 2 levels, C/S (Context-Shock) and S/C (Shock-Context).

Class is a factor with 8 levels, whics is the combination of the 3 factors with 2 levels.

The other 77 variables are numeric variables, representing expression levels of the named proteins.

The 77 numeric variables are used as predictors, and the Class is the main class that is to be predicted. Genotype, Treatment and Behaviour are still used for exploratory purposes.

```
## 'data.frame':    1080 obs. of  82 variables:
##  $ MouseID        : Factor w/ 1080 levels "18899_1","18899_10",..: 46 53 54 55 56 57 58 59 60 47 ...
##  $ DYRK1A_N       : num  0.504 0.515 0.509 0.442 0.435 ...
##  $ ITSN1_N        : num  0.747 0.689 0.73 0.617 0.617 ...
##  $ BDNF_N         : num  0.43 0.412 0.418 0.359 0.359 ...
##  $ NR1_N          : num  2.82 2.79 2.69 2.47 2.37 ...
##  $ NR2A_N         : num  5.99 5.69 5.62 4.98 4.72 ...
##  $ pAKT_N         : num  0.219 0.212 0.209 0.223 0.213 ...
##  $ pBRAF_N        : num  0.178 0.173 0.176 0.176 0.174 ...
##  $ pCAMKII_N      : num  2.37 2.29 2.28 2.15 2.13 ...
##  $ pCREB_N        : num  0.232 0.227 0.23 0.207 0.192 ...
##  $ pELK_N         : num  1.75 1.6 1.56 1.6 1.5 ...
##  $ pERK_N         : num  0.688 0.695 0.677 0.583 0.551 ...
##  $ pJNK_N         : num  0.306 0.299 0.291 0.297 0.287 ...
##  $ PKCA_N         : num  0.403 0.386 0.381 0.377 0.364 ...
##  $ pMEK_N         : num  0.297 0.281 0.282 0.314 0.278 ...
##  $ pNR1_N         : num  1.022 0.957 1.004 0.875 0.865 ...
##  $ pNR2A_N        : num  0.606 0.588 0.602 0.52 0.508 ...
##  $ pNR2B_N        : num  1.88 1.73 1.73 1.57 1.48 ...
##  $ pPKCAB_N       : num  2.31 2.04 2.02 2.13 2.01 ...
##  $ pRSK_N         : num  0.442 0.445 0.468 0.478 0.483 ...
##  $ AKT_N          : num  0.859 0.835 0.814 0.728 0.688 ...
##  $ BRAF_N         : num  0.416 0.4 0.4 0.386 0.368 ...
##  $ CAMKII_N       : num  0.37 0.356 0.368 0.363 0.355 ...
##  $ CREB_N         : num  0.179 0.174 0.174 0.179 0.175 ...
##  $ ELK_N          : num  1.87 1.76 1.77 1.29 1.32 ...
##  $ ERK_N          : num  3.69 3.49 3.57 2.97 2.9 ...
##  $ GSK3B_N        : num  1.54 1.51 1.5 1.42 1.36 ...
##  $ JNK_N          : num  0.265 0.256 0.26 0.26 0.251 ...
##  $ MEK_N          : num  0.32 0.304 0.312 0.279 0.274 ...
##  $ TRKA_N         : num  0.814 0.781 0.785 0.734 0.703 ...
##  $ RSK_N          : num  0.166 0.157 0.161 0.162 0.155 ...
##  $ APP_N          : num  0.454 0.431 0.423 0.411 0.399 ...
##  $ Bcatenin_N     : num  3.04 2.92 2.94 2.5 2.46 ...
##  $ SOD1_N         : num  0.37 0.342 0.344 0.345 0.329 ...
##  $ MTOR_N         : num  0.459 0.424 0.425 0.429 0.409 ...
##  $ P38_N          : num  0.335 0.325 0.325 0.33 0.313 ...
##  $ pMTOR_N        : num  0.825 0.762 0.757 0.747 0.692 ...
##  $ DSCR1_N        : num  0.577 0.545 0.544 0.547 0.537 ...
##  $ AMPKA_N        : num  0.448 0.421 0.405 0.387 0.361 ...
##  $ NR2B_N         : num  0.586 0.545 0.553 0.548 0.513 ...
##  $ pNUMB_N        : num  0.395 0.368 0.364 0.367 0.352 ...
##  $ RAPTOR_N       : num  0.34 0.322 0.313 0.328 0.312 ...
##  $ TIAM1_N        : num  0.483 0.455 0.447 0.443 0.419 ...
##  $ pP70S6_N       : num  0.294 0.276 0.257 0.399 0.393 ...
##  $ NUMB_N         : num  0.182 0.182 0.184 0.162 0.16 ...
##  $ P70S6_N        : num  0.843 0.848 0.856 0.76 0.768 ...
##  $ pGSK3B_N       : num  0.193 0.195 0.201 0.184 0.186 ...
##  $ pPKCG_N        : num  1.44 1.44 1.52 1.61 1.65 ...
##  $ CDK5_N         : num  0.295 0.294 0.302 0.296 0.297 ...
##  $ S6_N           : num  0.355 0.355 0.386 0.291 0.309 ...
```

```
##  $ ADARB1_N       : num  1.34 1.31 1.28 1.2 1.21 ...
##  $ AcetylH3K9_N   : num  0.17 0.171 0.185 0.16 0.165 ...
##  $ RRP1_N         : num  0.159 0.158 0.149 0.166 0.161 ...
##  $ BAX_N          : num  0.189 0.185 0.191 0.185 0.188 ...
##  $ ARC_N          : num  0.106 0.107 0.108 0.103 0.105 ...
##  $ ERBB4_N        : num  0.145 0.15 0.145 0.141 0.142 ...
##  $ nNOS_N         : num  0.177 0.178 0.176 0.164 0.168 ...
##  $ Tau_N          : num  0.125 0.134 0.133 0.123 0.137 ...
##  $ GFAP_N         : num  0.115 0.118 0.118 0.117 0.116 ...
##  $ GluR3_N        : num  0.228 0.238 0.245 0.235 0.256 ...
##  $ GluR4_N        : num  0.143 0.142 0.142 0.145 0.141 ...
##  $ IL1B_N         : num  0.431 0.457 0.51 0.431 0.481 ...
##  $ P3525_N        : num  0.248 0.258 0.255 0.251 0.252 ...
##  $ pCASP9_N       : num  1.6 1.67 1.66 1.48 1.53 ...
##  $ PSD95_N        : num  2.01 2 2.02 1.96 2.01 ...
##  $ SNCA_N         : num  0.108 0.11 0.108 0.12 0.12 ...
##  $ Ubiquitin_N    : num  1.045 1.01 0.997 0.99 0.998 ...
##  $ pGSK3B_Tyr216_N: num  0.832 0.849 0.847 0.833 0.879 ...
##  $ SHH_N          : num  0.189 0.2 0.194 0.192 0.206 ...
##  $ BAD_N          : num  0.123 0.117 0.119 0.133 0.13 ...
##  $ BCL2_N         : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ pS6_N          : num  0.106 0.107 0.108 0.103 0.105 ...
##  $ pCFOS_N        : num  0.108 0.104 0.106 0.111 0.111 ...
##  $ SYP_N          : num  0.427 0.442 0.436 0.392 0.434 ...
##  $ H3AcK18_N      : num  0.115 0.112 0.112 0.13 0.118 ...
##  $ EGR1_N         : num  0.132 0.135 0.133 0.147 0.14 ...
##  $ H3MeK4_N       : num  0.128 0.131 0.127 0.147 0.148 ...
##  $ CaNA_N         : num  1.68 1.74 1.93 1.7 1.84 ...
##  $ Genotype       : Factor w/ 2 levels "Control","Ts65Dn": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Treatment      : Factor w/ 2 levels "Memantine","Saline": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Behavior       : Factor w/ 2 levels "C/S","S/C": 1 1 1 1 1 1 1 1 1 1 ...
##  $ class          : Factor w/ 8 levels "c-CS-m","c-CS-s",..: 1 1 1 1 1 1 1 1 1 1 ...
```

PRELIMINARY DATA EXPLORATION:

Upon examining the proportions of each class, it can be seen that the number of classes are more or less evenly distributed, which means that there is no considerable difference in class prevelance in the sample population.

```
##
##     c-CS-m     c-CS-s     c-SC-m     c-SC-s     t-CS-m     t-CS-s     t-SC-m
## 0.13888889 0.12500000 0.13888889 0.12500000 0.12500000 0.09722222 0.12500000
##     t-SC-s
## 0.12500000
```

It is also seen that some of the predictor variables have a considerable amount of NA's as reported below.

```
##   DYRK1A_N ITSN1_N BDNF_N NR1_N NR2A_N pAKT_N pBRAF_N pCAMKII_N pCREB_N pELK_N
## 1        3       3      3     3      3      3       3         3       3      3
##   pERK_N pJNK_N PKCA_N pMEK_N pNR1_N pNR2A_N pNR2B_N pPKCAB_N pRSK_N AKT_N
## 1      3      3      3      3      3       3       3        3      3     3
##   BRAF_N CAMKII_N CREB_N ELK_N ERK_N GSK3B_N JNK_N MEK_N TRKA_N RSK_N APP_N
```

```
## 1      3        3      3    18     3       3     3     7      3     3     3
##   Bcatenin_N SOD1_N MTOR_N P38_N pMTOR_N DSCR1_N AMPKA_N NR2B_N pNUMB_N
## 1         18      3      3     3       3       3       3      3       3
##   RAPTOR_N TIAM1_N pP70S6_N NUMB_N P70S6_N pGSK3B_N pPKCG_N CDK5_N S6_N
## 1        3       3        3      0       0        0       0      0    0
##   ADARB1_N AcetylH3K9_N RRP1_N BAX_N ARC_N ERBB4_N nNOS_N Tau_N GFAP_N GluR3_N
## 1        0            0      0     0     0       0      0     0      0       0
##   GluR4_N IL1B_N P3525_N pCASP9_N PSD95_N SNCA_N Ubiquitin_N pGSK3B_Tyr216_N
## 1       0      0       0        0       0      0           0               0
##   SHH_N BAD_N BCL2_N pS6_N pCFOS_N SYP_N H3AcK18_N EGR1_N H3MeK4_N CaNA_N
## 1     0   213    285     0      75     0       180    210      270      0
```

Specifically, the below predictors have more than 50 NA's each.

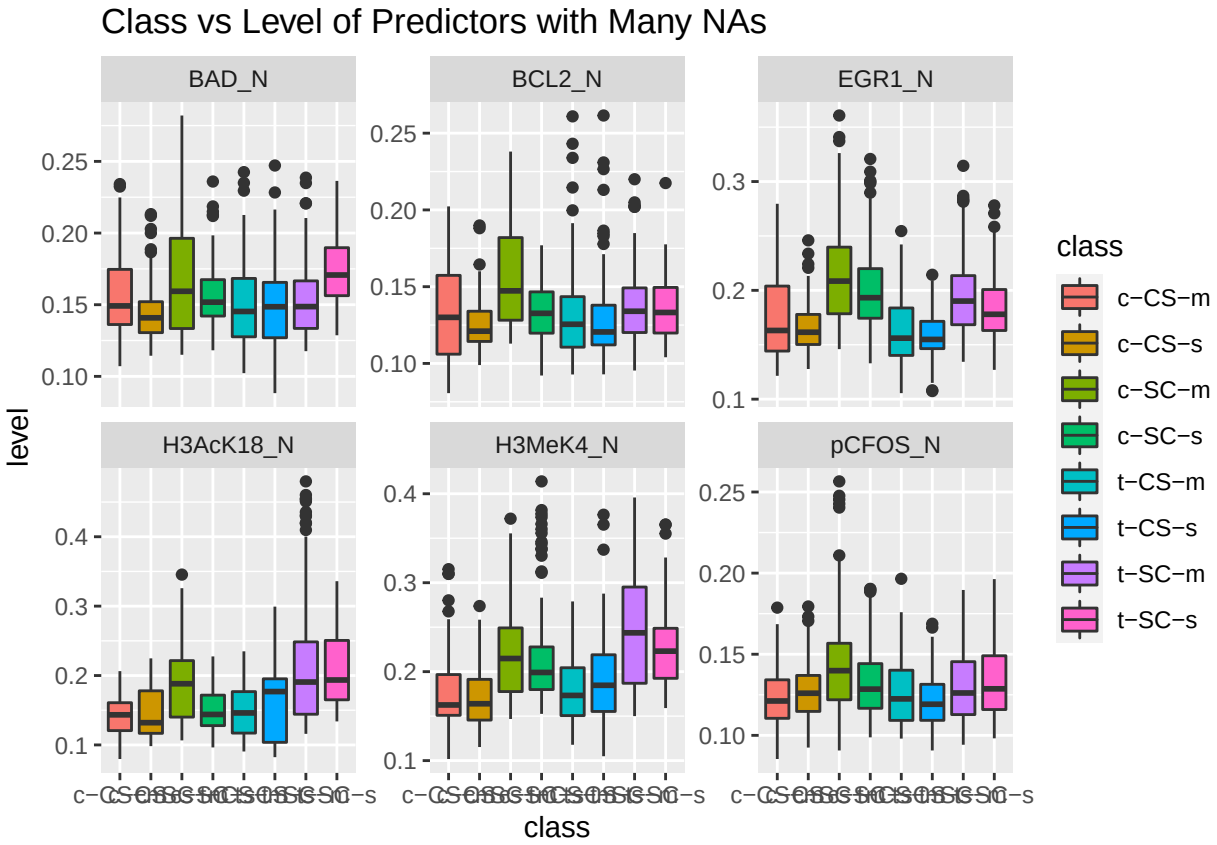
```
## [1] "BAD_N"     "BCL2_N"    "pCFOS_N"   "H3AcK18_N" "EGR1_N"    "H3MeK4_N"
```

DATA CLEANING AND PREPARATION:

Since there are NA's in almost 1/3 of the instances, removing the instances with NA's would be very costly in terms of information. When the relatively small number of instances is also taken into account, this option does not seem very logical.

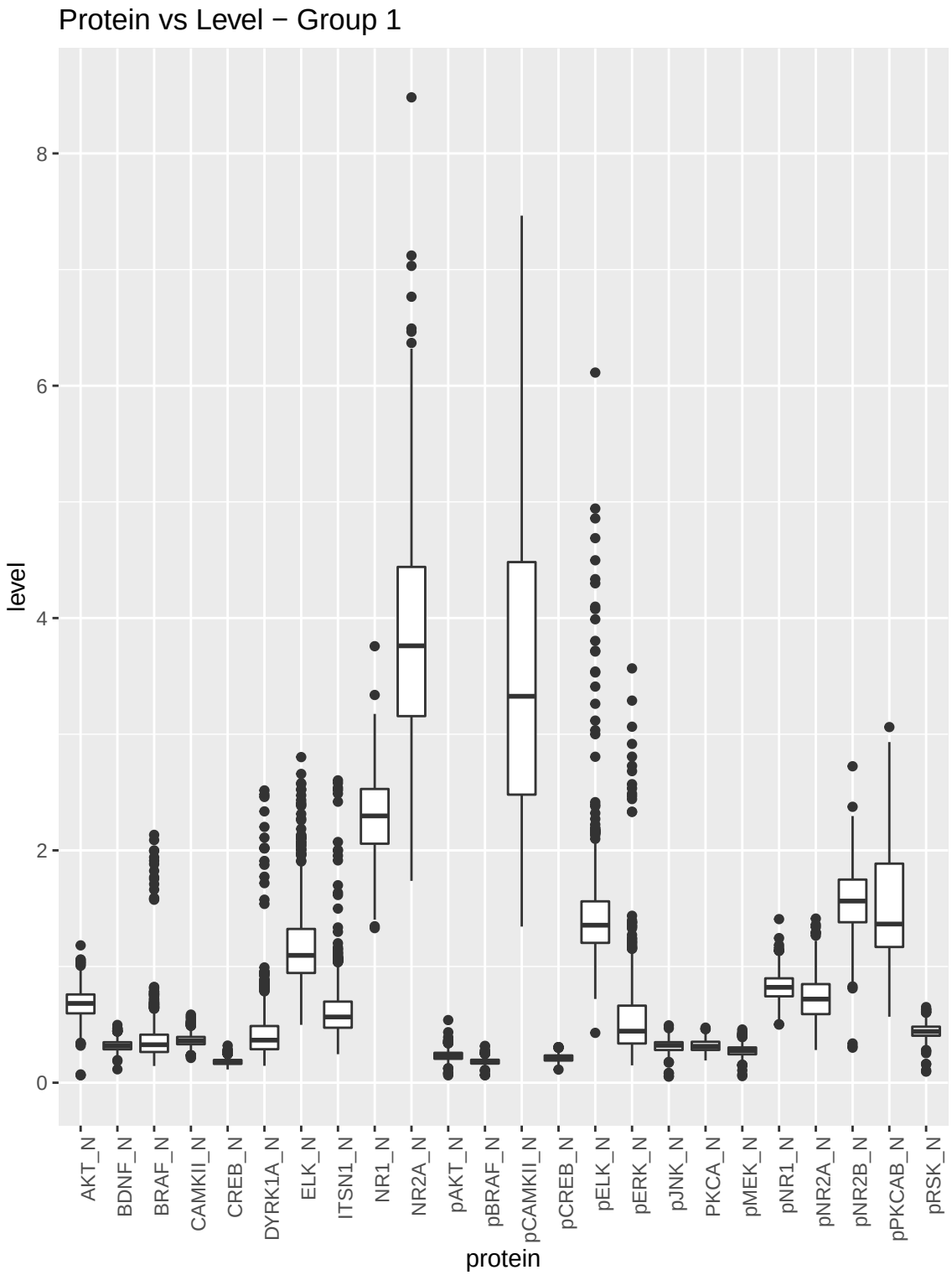
Visualization below reveals that the predictors with more than 50 NA's are not very good at distinguishing between classes. Only ERG1_N seems to partially seperate 2 classes.

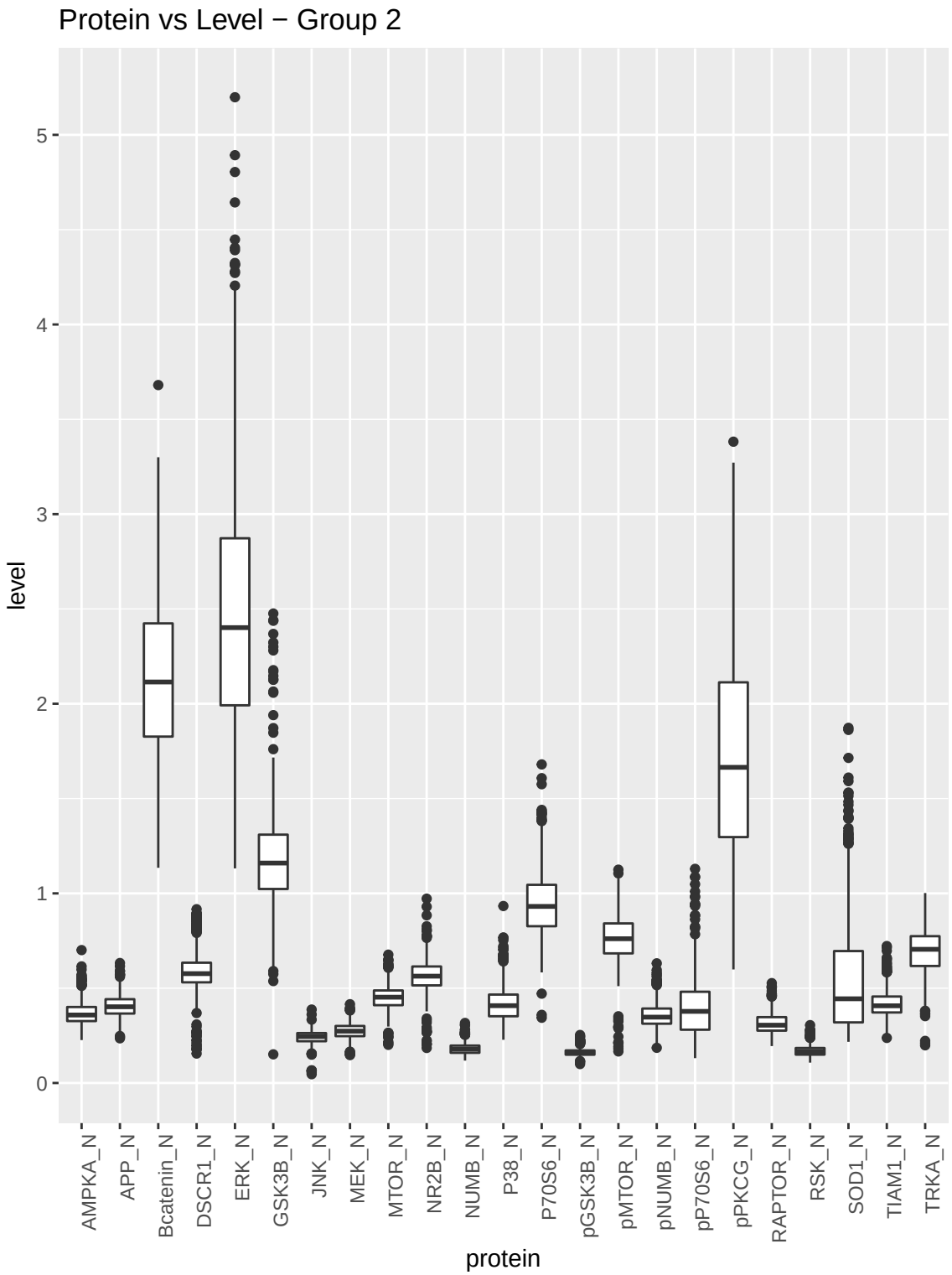
Given this information, it would be reasonable to remove these predictors instead of removing the instances.

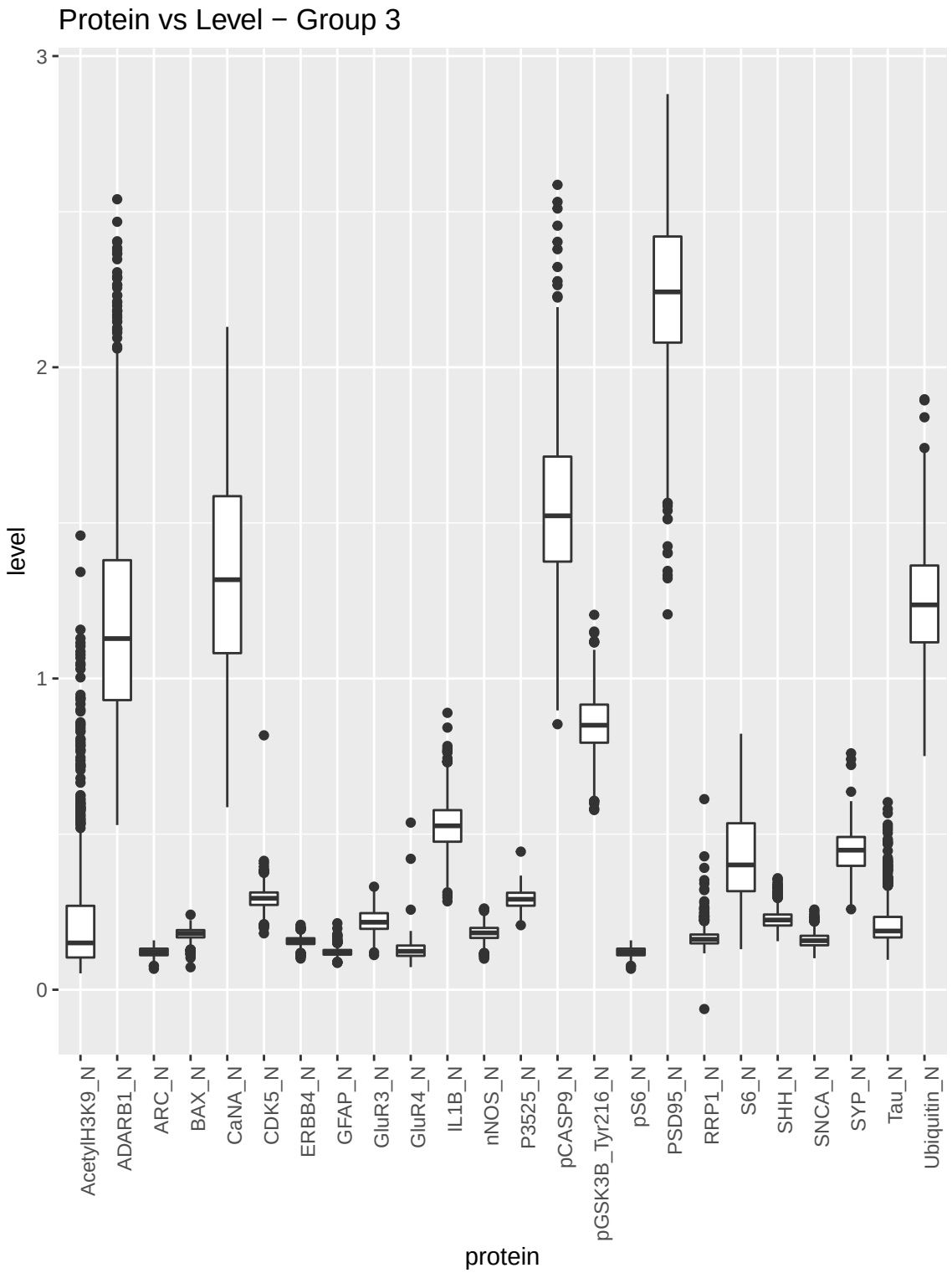


However, after these 6 prredictors are removed, there are still several predictors with fewer NA instances remaining.

In order to reach a decision regarding these NA's, the predictors are portioned into 3 groups, and their expression levels are visualized.


Protein vs Level − Group 1
level
protein
0
2
4
6
8
AKT_N
BDNF_N
BRAF_N
CAMKII_N
CREB_N
DYRK1A_N
ELK_N
ITSN1_N
NR1_N
NR2A_N
pAKT_N
pBRAF_N
pCAMKII_N
pCREB_N
pELK_N
pERK_N
pJNK_N
PKCA_N
pMEK_N
pNR1_N
pNR2A_N
pNR2B_N
pPKCAB_N
pRSK_N


Protein vs Level − Group 2
level
protein
0
1
2
3
4
5
AMPKA_N
APP_N
Bcatenin_N
DSCR1_N
ERK_N
GSK3B_N
JNK_N
MEK_N
MTOR_N
NR2B_N
NUMB_N
P38_N
P70S6_N
pGSK3B_N
pMTOR_N
pNUMB_N
pP70S6_N
pPKCG_N
RAPTOR_N
RSK_N
SOD1_N
TIAM1_N
TRKA_N



Upon examining these visuzalizations, it is seen that there are a large number values that lie far outside of the 2nd and 3rd quartiles for each predictor.

Due to significant number of values that are far from the median, it is sensible to use median imputation to

get rid of the NA's. The mean imputation is not preffered since mean would influenced more by the fringe values, hence could deviate the original data.

```
mice.data.final <- mice.data2 %>% group_by(class) %>%
  dplyr::mutate(across(DYRK1A_N: CaNA_N, ~ ifelse(is.na(.x), median(., na.rm = T), .x))) %>%
  ungroup() %>%
  as.data.frame()
```

After the median imputation is implemented, the data set predictors are checked once again, and it is confirmed that there are no more NA's remaining.

Next, a training set and a test set are created.

Due to the small number of instances in the data set, commonly used 10% to 20% allocation for the test set would not be sufficient. While there would be good amount of data for training, not having adequate data to test the models could cause the models to seem more accurate then they actually are.

Therefore, a division of 60% to 40% is used to create the training and test sets respectively.

In addition, matrices with predictor values for training and test sets are created.

```
set.seed(1986, sample.kind = 'Rounding')

y <- mice.data.final$class

test.index <- createDataPartition(y, times = 1, p = 0.4, list = F)

test.set <- mice.data.final[test.index,]

training.set <- mice.data.final[-test.index,]

test.set.preds <- test.set %>%
  select(-Behavior, -Genotype, -Treatment, -class, -MouseID) %>% as.matrix()

training.set.preds <- training.set %>%
  select(-Behavior, -Genotype, -Treatment, -class, -MouseID) %>% as.matrix()
```

EXPLORATION OF THE CLEANED AND PREPARED TRAINING SET DATA:

To gain a more in depth idea about the data in hand, which would be helpful in deciding on the methods that are to be used, more exploration is necessary.

Initially the standard deviations and variances of the predictors are calculated and examined.

```
min(sds)
```

```
## [1] 0.01317316
```

```
max(sds)
```

```
## [1] 1.283553
```

```
min(variances)
```

```
## [1] 0.0001735321
```

```
max(variances)
```

```
## [1] 1.647508
```

As seen above, there is high variability among standard deviation and variance values of predictors.

Under these circumstances, applying standardization would be wise if principal component anaylsis is to be performed. Otherwise principal components could be dominated by certain predictors.

Next, it is checked whether there a predictors with near zero variance. If there are, it would be advisable to remove them since they would not be helpful in predicting classes.

```
nzv <- nearZeroVar(training.set.preds)

nzv
```

```
## integer(0)
```

It appears there are not predictors with near zero variance. Based on this result, it is not possible to remove predictors at this point.

To investiage if there are any predictors that are correlated, a correlation matrix is produced.

The diagonal and lower triangle of the matrix are assigned zero values in order to avoid overestimating the number of highly correlated predictors.

Then the percentage of predictor pairs whose correlation coefficients are higher than 0.90, 0.70 and 0.50 are calculated. Both negative and positive correlation is taken into account.

	x
Very High Correlation %	0.0015870
High Correlation %	0.0208292
Moderate Correlation %	0.0702242

It is seen that the highly correlated predictor pairs amount to about 2% of total pairs, and moderately correlated pairs amount to about %7.

This indicates that multicollinearity is not a significant issue, and predictors are independent for the most part.

In order to check if the variables are multivariate normal, MVN package is utilized Doornik-Hansen tests are implemented. The results below show that variables are not multivariate normal.

Test	E	df	p value	MVN
Doornik-Hansen	18259.7	142	0	NO

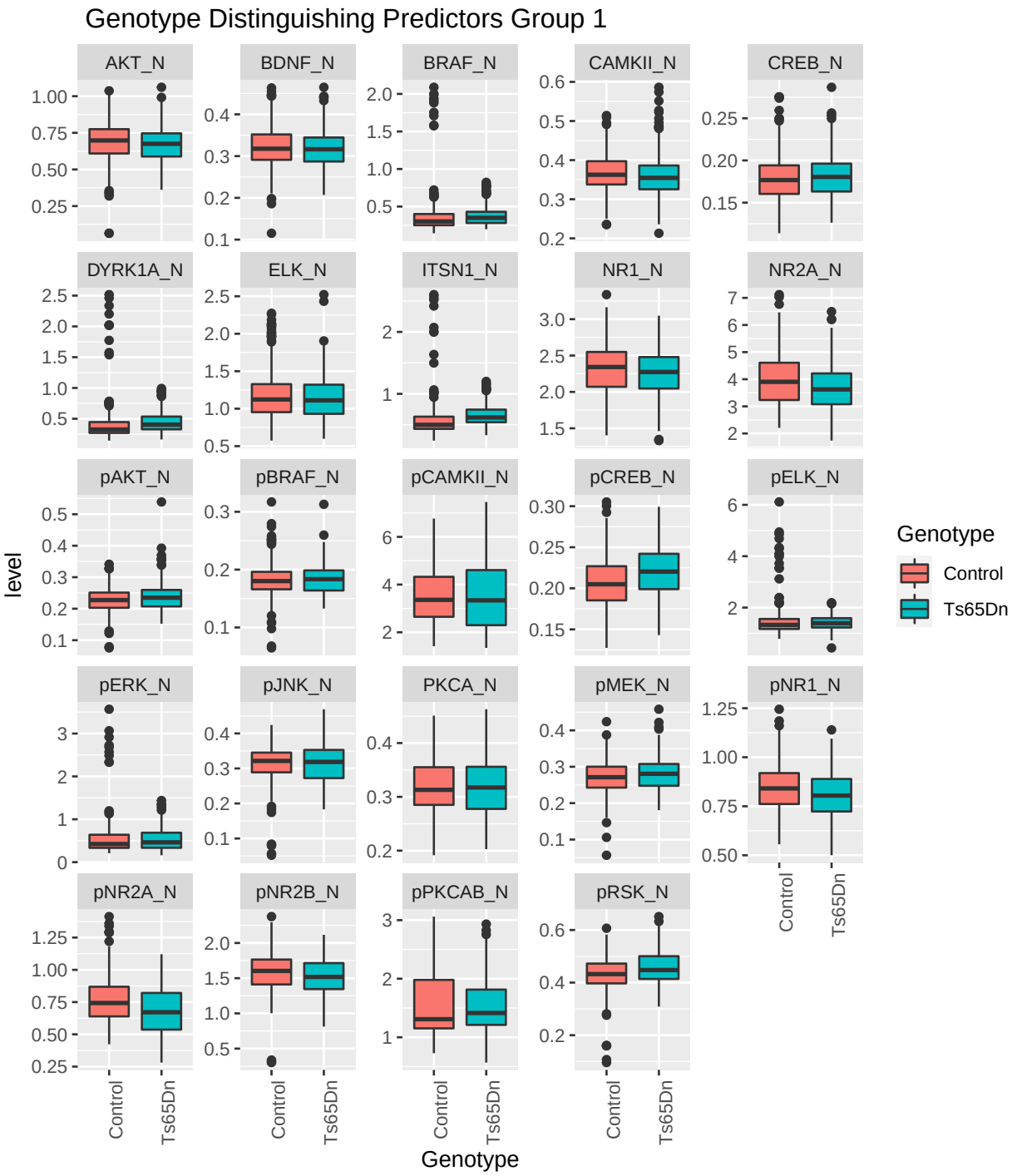
To visualize the prediction capability of predictors, a tidy data frame is created.

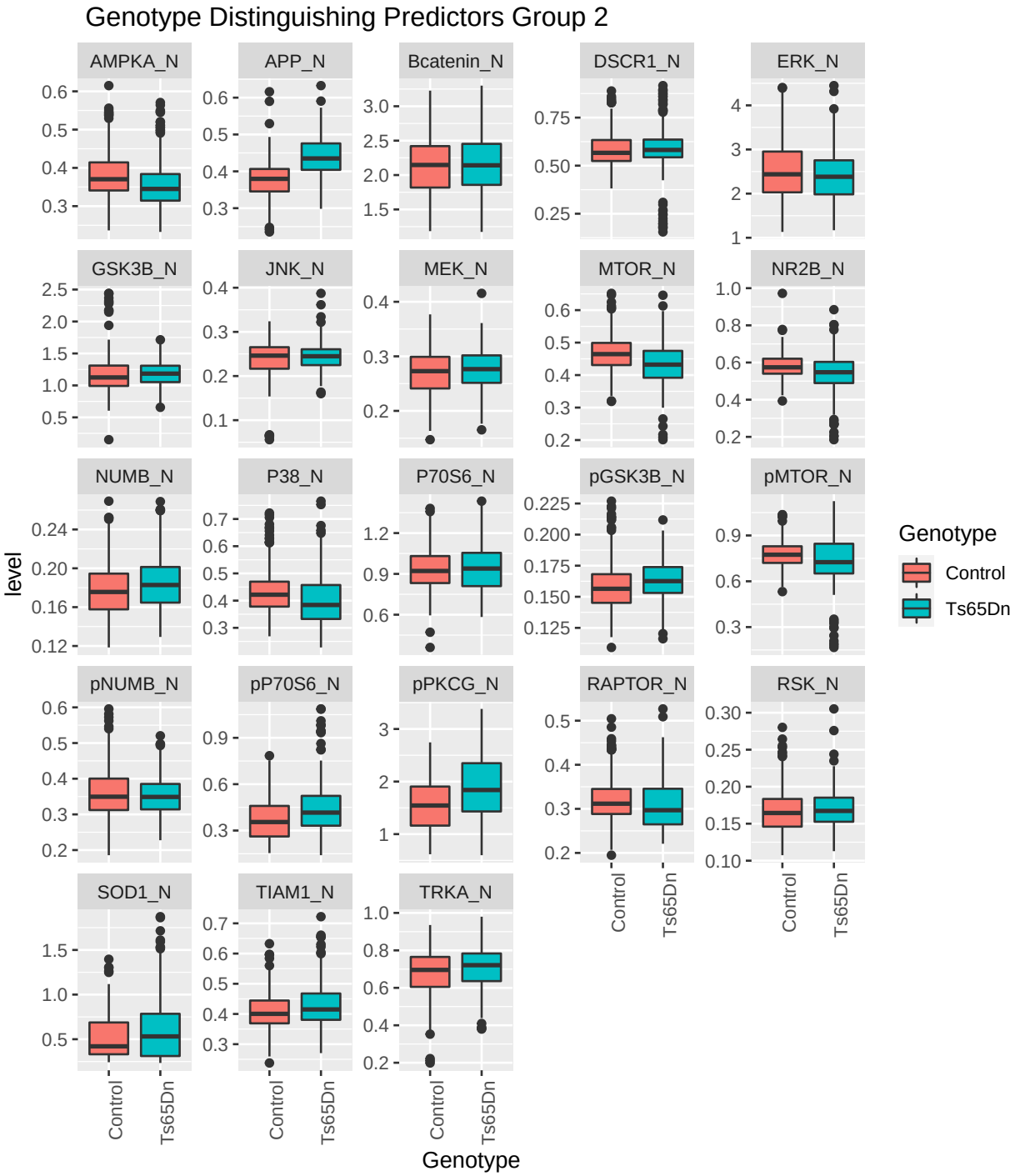
```
#gathering the predictors for analysis
proteins.gather <- training.set %>%
  gather(protein, level, "DYRK1A_N" :"CaNA_N")
```

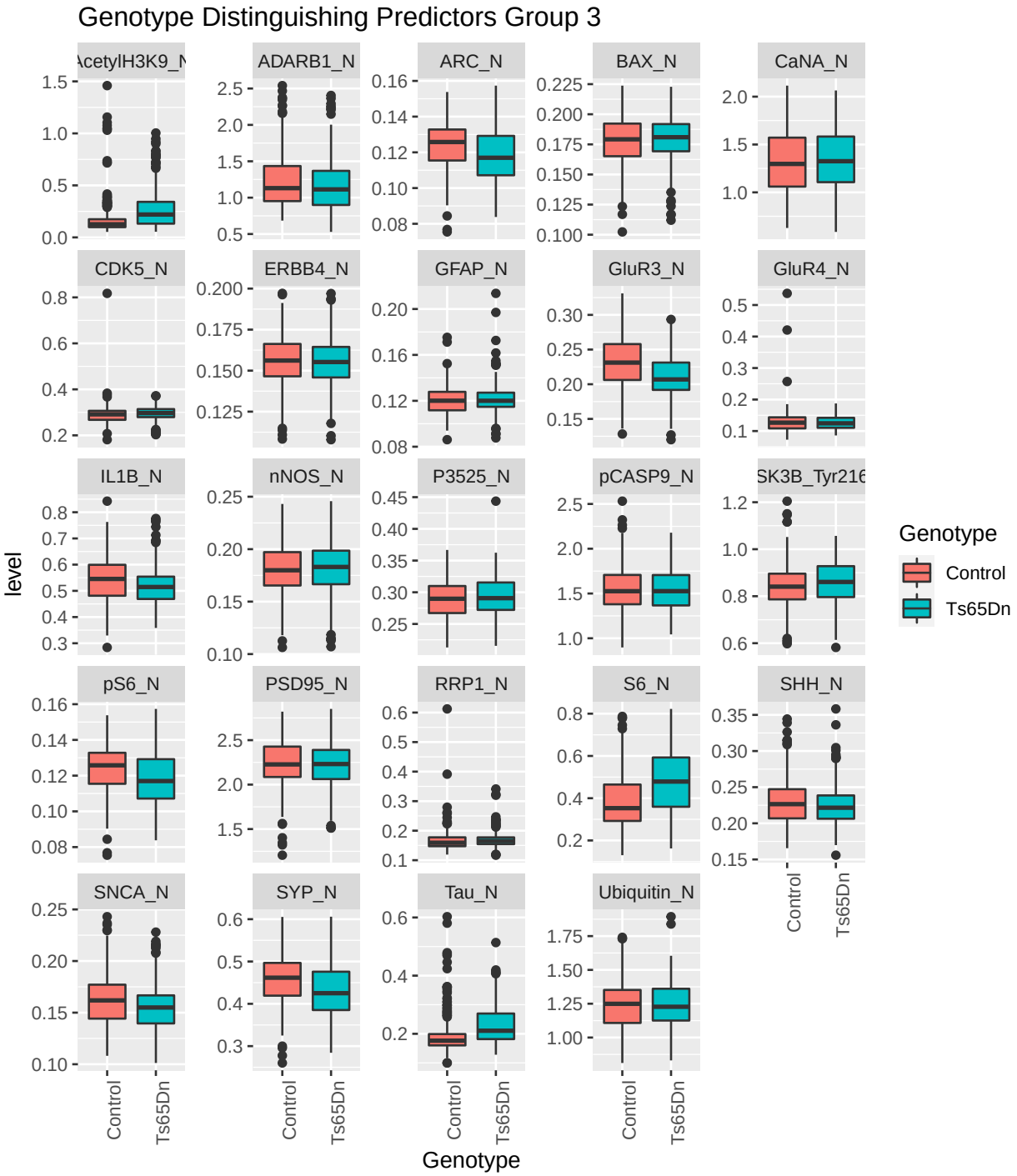
The aim of the following plots is to see whether any of the predictors distinguish between classes. Since class "Class" has 8 factors and consists of the combination of 3 classes that have 2 factors each, these sub-classes are utilized in the plots to simplify the analysis.

Furthermore, instead of visualizing all predictors and once, the 3 groups of predictors that were created before are used once again for simplification and readability purposes.

The first 3 plots are created to see if any predictors distinguish between 2 Genotypes are below.

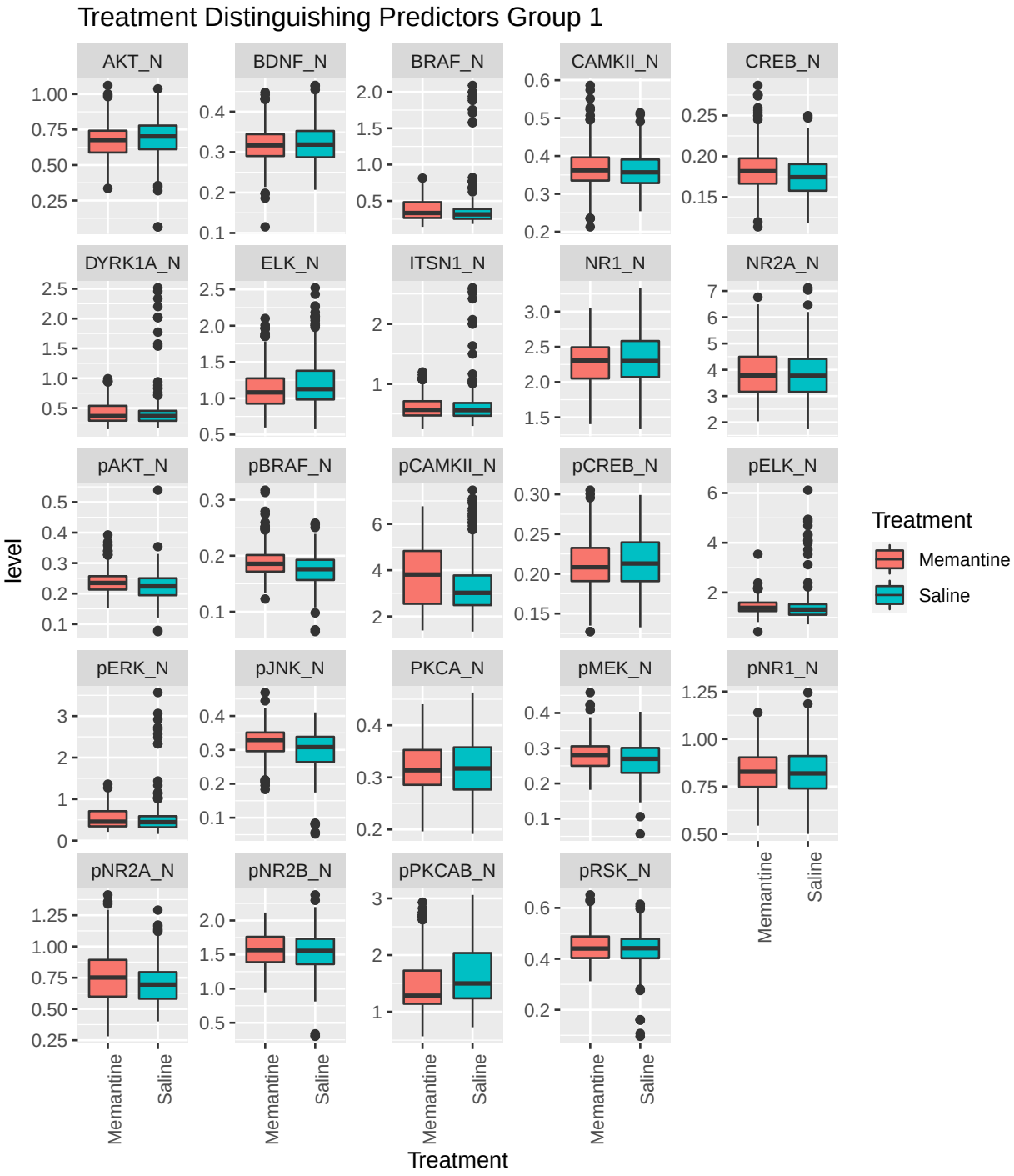

Genotype Distinguishing Predictors Group 1
AKT_N
BDNF_N
BRAF_N
CAMKII_N
CREB_N
DYRK1A_N
ELK_N
ITSN1_N
NR1_N
NR2A_N
pAKT_N
pBRAF_N
pCAMKII_N
pCREB_N
pELK_N
pERK_N
pJNK_N
PKCA_N
pMEK_N
pNR1_N
pNR2A_N
pNR2B_N
pPKCAB_N
pRSK_N
level
Genotype
Control
Ts65Dn

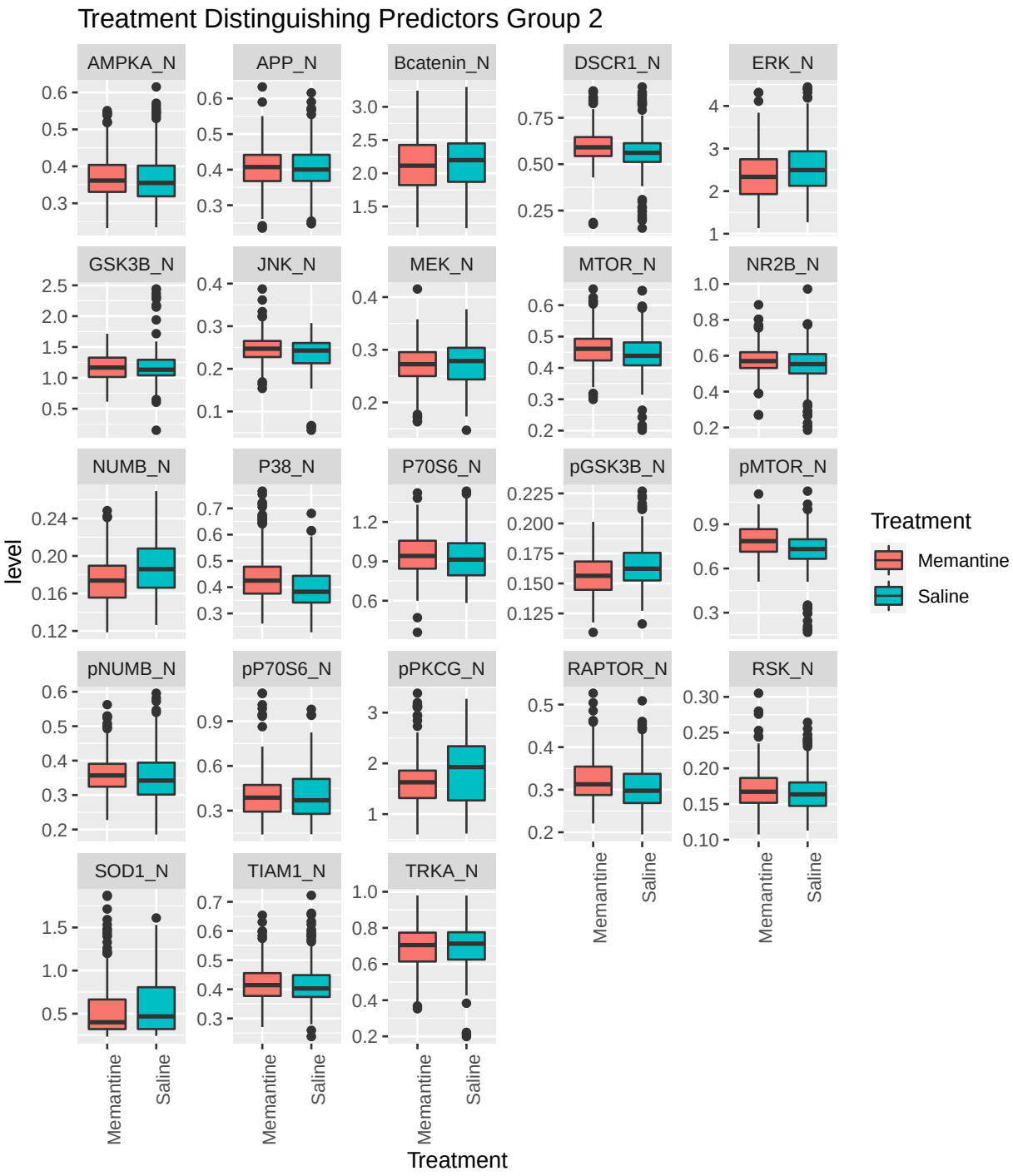

Genotype Distinguishing Predictors Group 2
AMPKA_N
APP_N
Bcatenin_N
DSCR1_N
ERK_N
GSK3B_N
JNK_N
MEK_N
MTOR_N
NR2B_N
NUMB_N
P38_N
P70S6_N
pGSK3B_N
pMTOR_N
pNUMB_N
pP70S6_N
pPKCG_N
RAPTOR_N
RSK_N
SOD1_N
TIAM1_N
TRKA_N
level
Control
Ts65Dn
Genotype

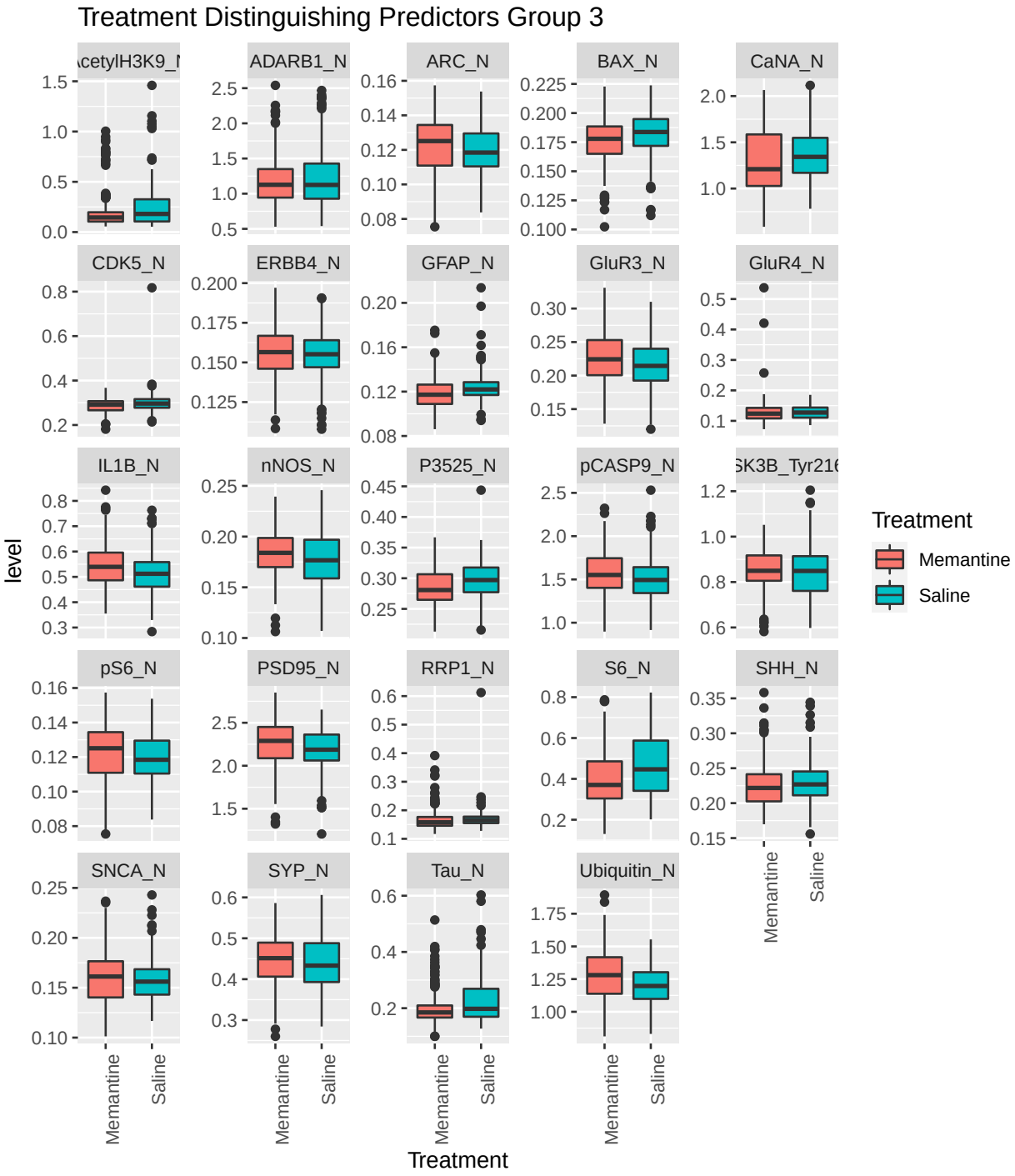


It appears BRAF_N , pELK_N and p_ERK_N in group 1 & APP_N, DSCR1_N and MTOR_N, PMTOR_N in group 2 could be useful in differentiating Genotype.

The next 3 plots are created to see if any predictors distinguish between 2 types of Treatment are below.

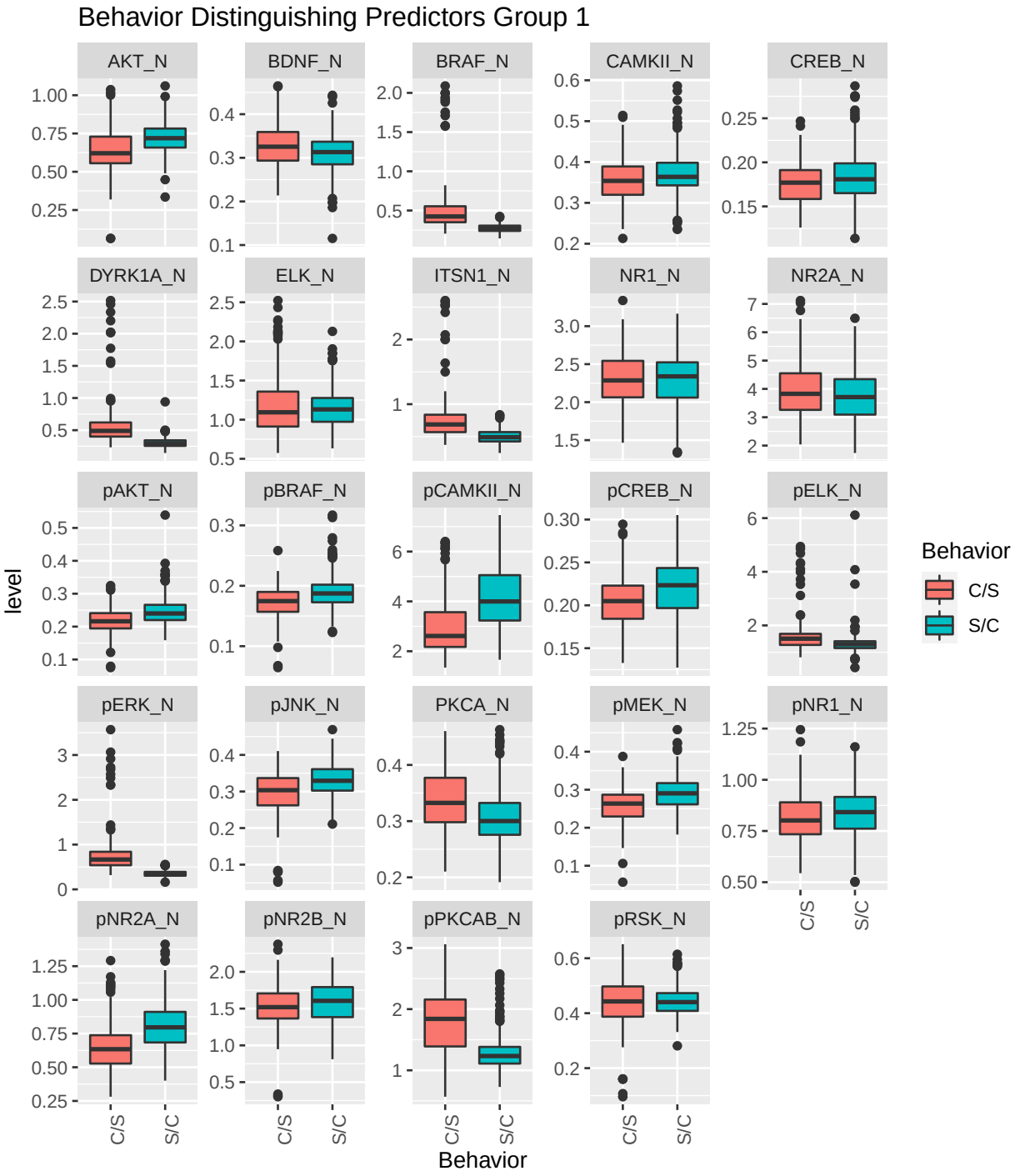

Treatment Distinguishing Predictors Group 1
AKT_N
BDNF_N
BRAF_N
CAMKII_N
CREB_N
DYRK1A_N
ELK_N
ITSN1_N
NR1_N
NR2A_N
pAKT_N
pBRAF_N
pCAMKII_N
pCREB_N
pELK_N
pERK_N
pJNK_N
PKCA_N
pMEK_N
pNR1_N
pNR2A_N
pNR2B_N
pPKCAB_N
pRSK_N
level
Memantine
Saline
Treatment

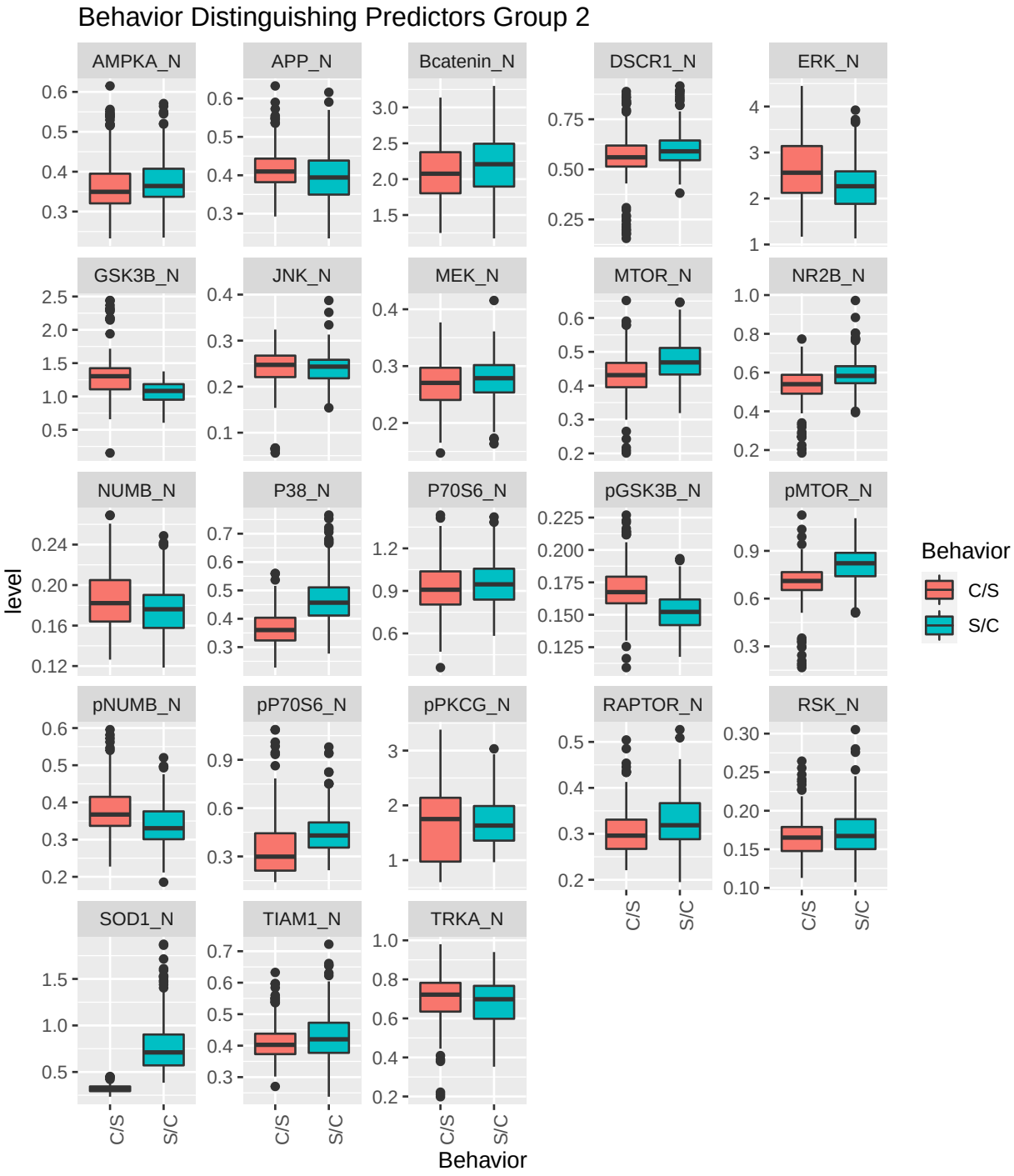

Treatment Distinguishing Predictors Group 2
AMPKA_N
APP_N
Bcatenin_N
DSCR1_N
ERK_N
GSK3B_N
JNK_N
MEK_N
MTOR_N
NR2B_N
NUMB_N
P38_N
P70S6_N
pGSK3B_N
pMTOR_N
pNUMB_N
pP70S6_N
pPKCG_N
RAPTOR_N
RSK_N
SOD1_N
TIAM1_N
TRKA_N
level
Memantine
Saline
Treatment

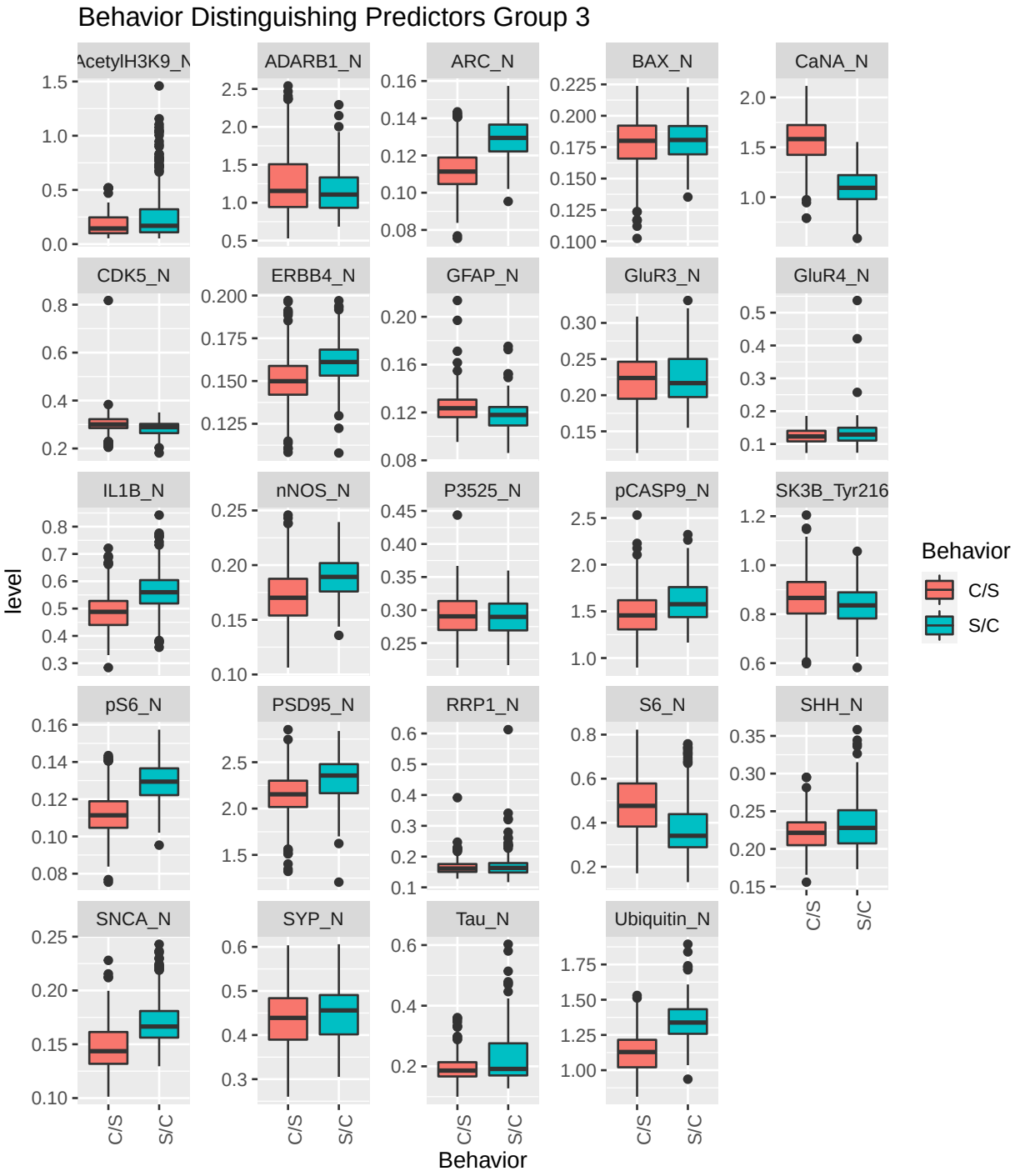


It appears again that BRAF_N , pELK_N and p_ERK_N in group 1 & pMTOR_N and pPKCG_N in group 2 could be useful in differentiating Treatment type.

The final set of plots are created to see if any predictors distinguish between 2 types of Behavior are below.


Behavior Distinguishing Predictors Group 1
AKT_N
BDNF_N
BRAF_N
CAMKII_N
CREB_N
DYRK1A_N
ELK_N
ITSN1_N
NR1_N
NR2A_N
pAKT_N
pBRAF_N
pCAMKII_N
pCREB_N
pELK_N
pERK_N
pJNK_N
PKCA_N
pMEK_N
pNR1_N
pNR2A_N
pNR2B_N
pPKCAB_N
pRSK_N
level
Behavior
C/S
S/C
Behavior


Behavior Distinguishing Predictors Group 2
AMPKA_N
APP_N
Bcatenin_N
DSCR1_N
ERK_N
GSK3B_N
JNK_N
MEK_N
MTOR_N
NR2B_N
NUMB_N
P38_N
P70S6_N
pGSK3B_N
pMTOR_N
pNUMB_N
pP70S6_N
pPKCG_N
RAPTOR_N
RSK_N
SOD1_N
TIAM1_N
TRKA_N
level
C/S
S/C
Behavior
Behavior
C/S
S/C



It is seen again that pERK_N, BRAF_N, DYRK1A_N in group 1, SOD1_N in group 2 & C aNA_N in group 3 could be useful in differentiating Treatment type.

Overall, via these plots, it became evident that there a few common predictors that appear to successfully separate 3 sub-classes. In addition, there are other predictors that at least partially distinguish between the sub-classes. Hence, with a proper model, it should be possible to predict the main class with a relatively high level of accuracy.

METHOD & MODELS:

Since this is a multiclass classification problem, the algorithms that are utilized are Classification Tree, KNN and Random Forest.

QDA is not be used for 2 reasons. First, the test results indicated that the variables are not multivariate normal. Second, due to few observations and large number of predictors, the parameter amount used for QDA would reach unpractical levels quickly.

5 main approaches are considered, yielding a total of 11 models.

1) Initially, a model that simple guesses is produced to be used as a benchmark.
2) Classification Tree, KNN and Random forest is implemented using manually chosen predictors that are deemed most useful upon analyzing the plots created previously.
3) Classification Tree, KNN and Random forest is implemented using all predictors in training set.
4) Dimension reduction via principal component analysis is performed, and using the principal components, Classification Tree, KNN and Random forest is implemented.

APPROACH NO 1 - SIMPLY GUESSING:

A simplistic model that predicts the classes purely by guessing is implemented. The result from guessing model is considered a benchmark to evaluate the other models.

Sampling with replacement is used for the model mentioned, and the accuracy is reported below.

```
## [1] 0.1087963
```

Given that there are 8 classes to choose from, the resulting accuracy by guessing is reasonable.

APPROACH NO 2 - MANUAL SELECTION OF PREDICTORS:

Upon investigating the plots that were produced to see if any predictors distinguish between sub-classes, some of the predictors were identified by eye to be more useful than others.

For this approach, only these predictors are used to create the 3 models listed previously.

The parameters for each model are tuned accordingly.

The code is shown below.

```
#MODEL 2: CLASSIFICATION TREE WITH MANUAL PREDICTOR SELECTION


chosen.predictors <- c('BRAF_N', 'pELK_N', 'pERK_N', 'APP_N', 'DSCR1_N', 'MTOR_N', 'pMTOR_N',
                       'DYRK1A_N', 'SOD1_N', 'CaNA_N')

chosen.index <- which(colnames(training.set.preds) %in% chosen.predictors)

training.chosen.predictors <- training.set.preds[,chosen.index]

test.chosen.predictors <- test.set.preds[,chosen.index]

set.seed(1986, sample.kind = 'Rounding')

fit.manual.rpart <- train(training.chosen.predictors, training.set$class,
```

```
                      method = 'rpart',
                      tuneGrid = data.frame(cp = seq(0.0, 0.05, len = 40)))

accuracy.manual.rpart <- mean(predict(fit.manual.rpart, test.chosen.predictors) == test.set$class)


#MODEL 3:  K-NEAREST NEIGHBORS WITH MANUAL PREDICTOR SELECTION

set.seed(1986, sample.kind = 'Rounding')

fit.manual.knn <- train(training.chosen.predictors, training.set$class,
                  method = 'knn',
                  tuneGrid = data.frame(k = seq(2, 20, 1)))

accuracy.manual.knn <- mean(predict(fit.manual.knn, test.chosen.predictors) == test.set$class)

#MODEL 4: RANDOM FOREST WITH MANUAL PREDICTOR SELECTION

set.seed(1986, sample.kind = 'Rounding')

fit.manual.rf <-train(training.chosen.predictors, training.set$class,
                method = 'rf',
                tuneGrid = data.frame(mtry = seq(1, 8, 1)), importance = T, ntree = 500)

accuracy.manual.rf <- mean(predict(fit.manual.rf, test.chosen.predictors) == test.set$class)
```

APPROACH NO 3 - USING ALL PREDICTORS:

In this straightforward approach, all predictors are used to create the 3 models. The parameters for each model are tuned the same way they were tuned for the 2nd approach.

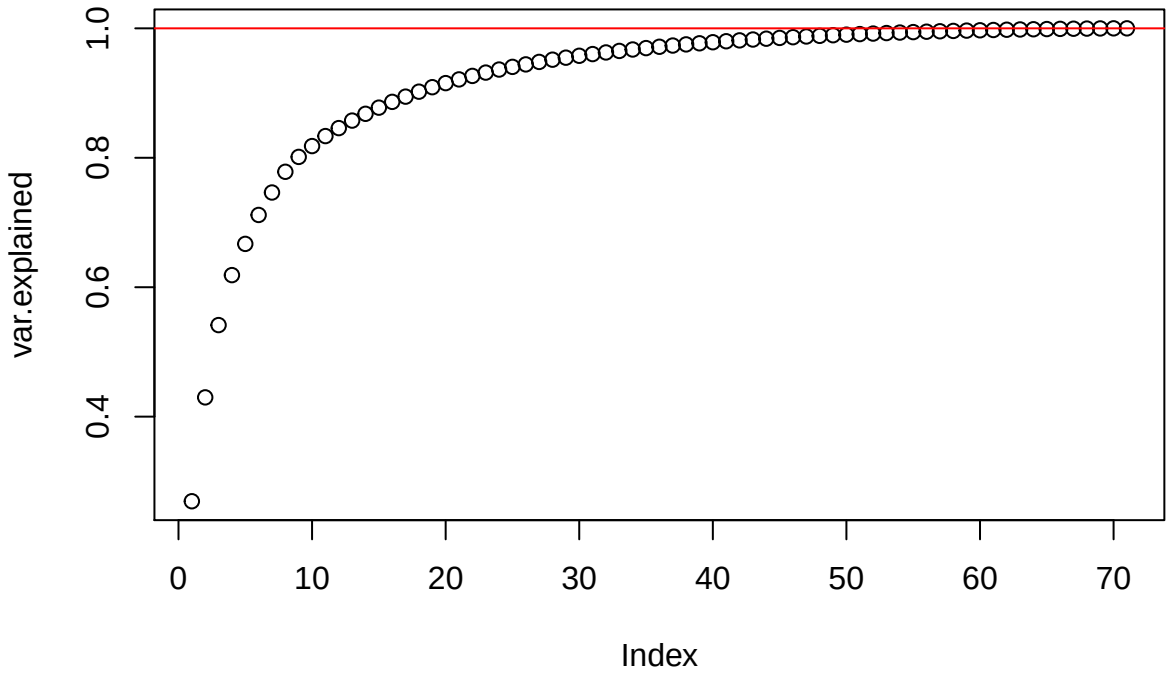
APPROACH NO 4 - USING PRINCIPAL COMPONENTS AS PREDICTORS:

For this approach, the predictor matrix of training set is scaled and PCA is performed using prcomp function. To keep consistency of transformation, the test set is also scaled according to the columnn means and standard deviations of the training set. The scaled test set values will be used during prediction stage.

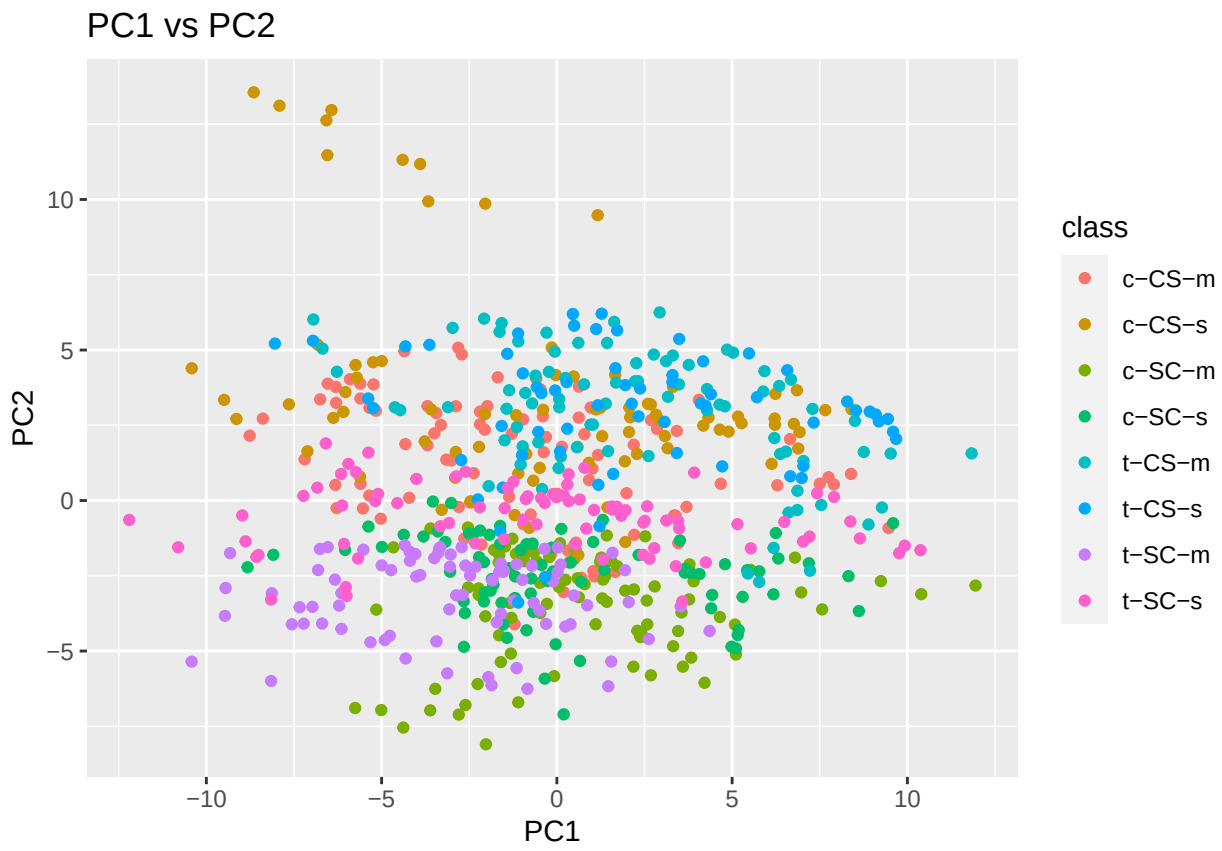
The summary and the plot below shows that: 80% of variability is explained at 9th principal component, 90% of variability is explained at 18th principal component, 99% of variability is explained at 50th principal component.

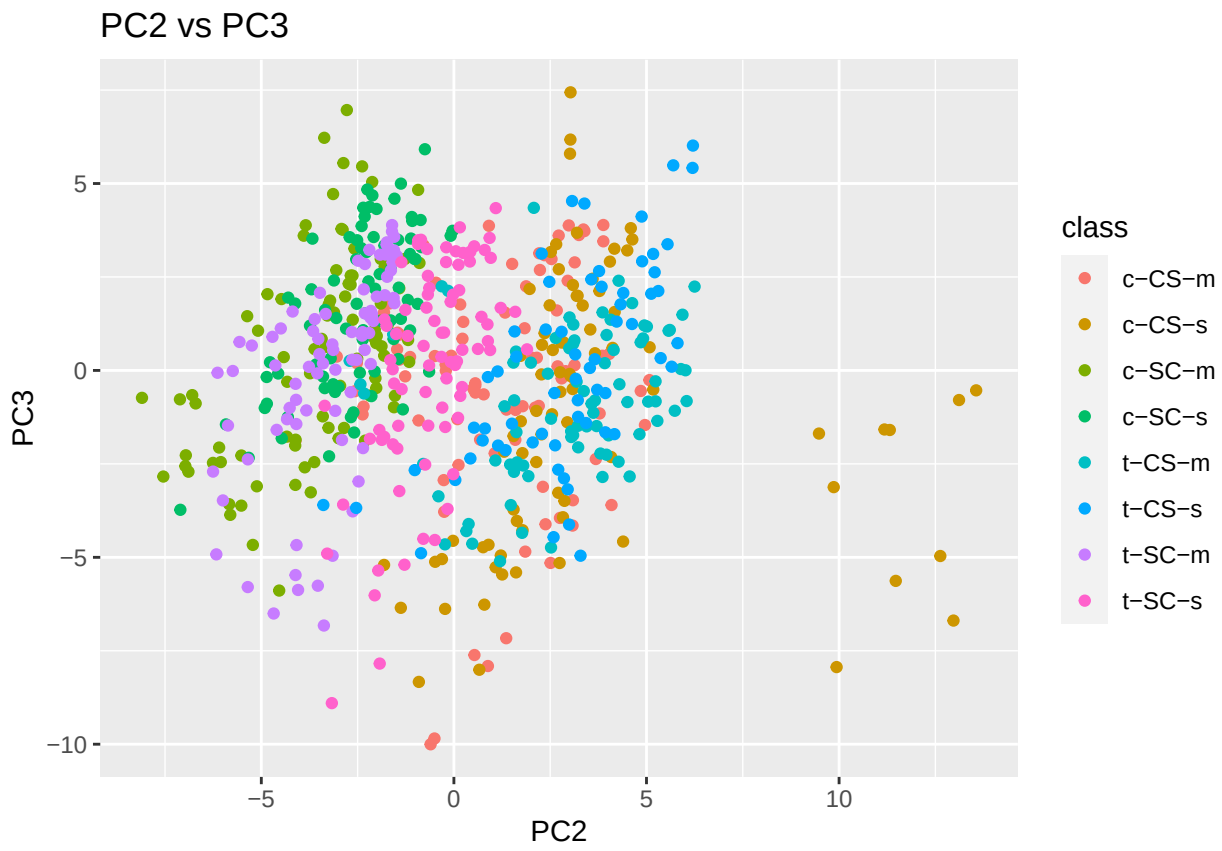
```
## Importance of components:
##                           PC1    PC2    PC3    PC4     PC5     PC6    PC7
## Standard deviation     4.3724 3.3757 2.8163 2.3412 1.85072 1.78166 1.5697
## Proportion of Variance 0.2693 0.1605 0.1117 0.0772 0.04824 0.04471 0.0347
## Cumulative Proportion  0.2693 0.4298 0.5415 0.6187 0.66692 0.71163 0.7463
##                           PC8     PC9    PC10    PC11    PC12    PC13    PC14
## Standard deviation     1.50790 1.27772 1.08960 1.04729 0.93586 0.90870 0.86169
## Proportion of Variance 0.03203 0.02299 0.01672 0.01545 0.01234 0.01163 0.01046
## Cumulative Proportion  0.77836 0.80135 0.81808 0.83352 0.84586 0.85749 0.86795
```

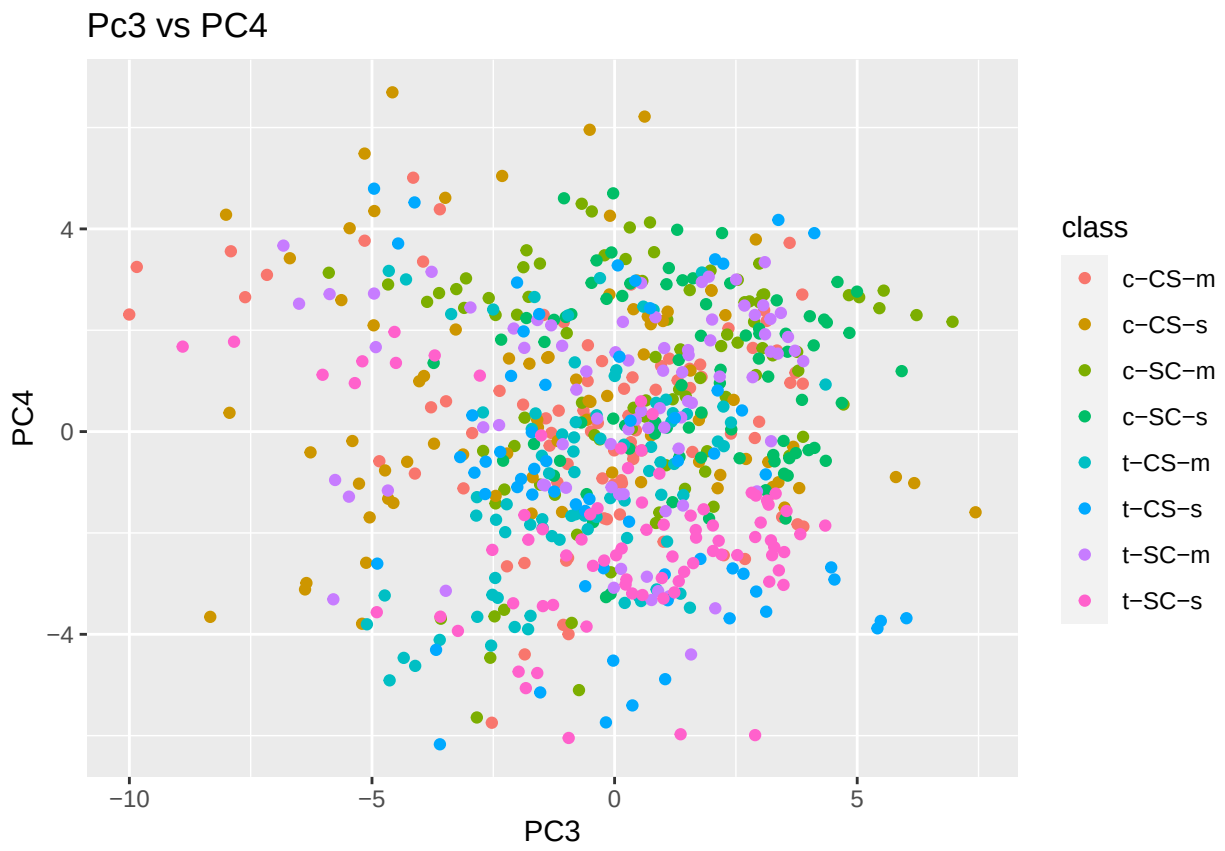
```
##                            PC15    PC16    PC17    PC18    PC19   PC20    PC21
## Standard deviation     0.82282 0.79565 0.75751 0.73785 0.70172 0.6690 0.63717
## Proportion of Variance 0.00954 0.00892 0.00808 0.00767 0.00694 0.0063 0.00572
## Cumulative Proportion  0.87748 0.88640 0.89448 0.90215 0.90908 0.9154 0.92111
##                            PC22    PC23    PC24    PC25    PC26    PC27    PC28
## Standard deviation     0.61969 0.60289 0.57814 0.54104 0.52243 0.52026 0.49661
## Proportion of Variance 0.00541 0.00512 0.00471 0.00412 0.00384 0.00381 0.00347
## Cumulative Proportion  0.92652 0.93164 0.93634 0.94047 0.94431 0.94812 0.95160
##                            PC29   PC30    PC31    PC32    PC33    PC34    PC35
## Standard deviation     0.46960 0.4540 0.43233 0.42369 0.40762 0.39974 0.38992
## Proportion of Variance 0.00311 0.0029 0.00263 0.00253 0.00234 0.00225 0.00214
## Cumulative Proportion  0.95470 0.9576 0.96024 0.96277 0.96511 0.96736 0.96950
##                            PC36    PC37    PC38    PC39    PC40    PC41    PC42
## Standard deviation     0.38089 0.36644 0.35598 0.34583 0.34224 0.32712 0.31128
## Proportion of Variance 0.00204 0.00189 0.00178 0.00168 0.00165 0.00151 0.00136
## Cumulative Proportion  0.97154 0.97343 0.97522 0.97690 0.97855 0.98006 0.98142
##                            PC43    PC44   PC45    PC46    PC47   PC48    PC49
## Standard deviation     0.30791 0.29958 0.2914 0.28379 0.27839 0.2671 0.25791
## Proportion of Variance 0.00134 0.00126 0.0012 0.00113 0.00109 0.0010 0.00094
## Cumulative Proportion  0.98276 0.98402 0.9852 0.98635 0.98744 0.9885 0.98939
##                            PC50    PC51    PC52    PC53    PC54    PC55    PC56
## Standard deviation     0.24906 0.24660 0.24253 0.23681 0.23270 0.22180 0.21351
## Proportion of Variance 0.00087 0.00086 0.00083 0.00079 0.00076 0.00069 0.00064
## Cumulative Proportion  0.99026 0.99112 0.99194 0.99273 0.99350 0.99419 0.99483
##                            PC57    PC58    PC59    PC60    PC61    PC62    PC63
## Standard deviation     0.21139 0.19593 0.19178 0.18403 0.18115 0.17394 0.16602
## Proportion of Variance 0.00063 0.00054 0.00052 0.00048 0.00046 0.00043 0.00039
## Cumulative Proportion  0.99546 0.99600 0.99652 0.99700 0.99746 0.99788 0.99827
##                            PC64    PC65    PC66    PC67    PC68   PC69    PC70
## Standard deviation     0.15318 0.14939 0.14026 0.12875 0.12594 0.1206 0.10093
## Proportion of Variance 0.00033 0.00031 0.00028 0.00023 0.00022 0.0002 0.00014
## Cumulative Proportion  0.99860 0.99892 0.99919 0.99943 0.99965 0.9999 1.00000
##                              PC71
## Standard deviation     1.942e-15
## Proportion of Variance 0.000e+00
## Cumulative Proportion  1.000e+00
```



When the first 4 principal components are plotted against each other, they seem to partially separate between classes, but there are still a lot of overlapping points from different classes. This makes sense,since the first 4 PC's only explain approximately 60% of variability cumulatively.


PC1 vs PC2
PC2
PC1
10
5
0
−5
−10
−5
0
5
10
class
c−CS−m
c−CS−s
c−SC−m
c−SC−s
t−CS−m
t−CS−s
t−SC−m
t−SC−s


PC2 vs PC3
PC3
PC2
5
0
−5
−10
−5
0
5
10
class
c−CS−m
c−CS−s
c−SC−m
c−SC−s
t−CS−m
t−CS−s
t−SC−m
t−SC−s



Although from previous analysis of PC's, an understanding is obtained regarding how many PC's should be used for the models, number of PC's to be included is treated like a tuning parameter, and the number that yields the best accuracy is chosen to create the models.

Tuning of the model specific parameters are done the same way they were done in previous approaches.

The code for 3 models created using princical components is shown below.

```
pcas.included <- seq(2,71,1)

set.seed(1986, sample.kind = 'Rounding')

rpart.tune.pc <- function(pcn) {
  fit.pca.rpart <- train(pca$x[, 1:pcn], training.set$class,
                         method = 'rpart',
                         tuneGrid = data.frame(cp = seq(0, 0.05, length = 40)))
  return(caretEnsemble:: getMetric(fit.pca.rpart))
}

set.seed(1986, sample.kind = 'Rounding')

rpart.pc.options <- sapply(X = pcas.included , FUN =  rpart.tune.pc)

rpart.best.pc <- pcas.included[which.max(rpart.pc.options)]

set.seed(1986, sample.kind = 'Rounding')
```

```
fit.pca.rpart <- train(pca$x[, 1:rpart.best.pc], training.set$class,
                       method = 'rpart',
                       tuneGrid = data.frame(cp = seq(0, 0.05, length = 20)))

accuracy.pca.rpart <- mean(predict(fit.pca.rpart,
                                   predict(pca, standard.test.pred)) == test.set$class)

#MODEL 9 K-NEAREST NEIGHBORS WITH PCA

set.seed(1986, sample.kind = 'Rounding')

knn.tune.pc <- function(pcn) {
  fit.pca.knn <- train(pca$x[, 1:pcn], training.set$class,
                       method = 'knn',
                       tuneGrid = data.frame(k = seq(2, 20, 2)))
  return(caretEnsemble:: getMetric(fit.pca.knn))
}


set.seed(1986, sample.kind = 'Rounding')

knn.pc.options <- sapply(X = pcas.included , FUN =  knn.tune.pc)

knn.best.pc <- pcas.included[which.max(knn.pc.options)]

set.seed(1986, sample.kind = 'Rounding')

fit.pca.knn <- train(pca$x[, 1:knn.best.pc], training.set$class,
                       method = 'knn',
                       tuneGrid = data.frame(k = seq(2, 20, 2)))


accuracy.pca.knn <- mean(predict(fit.pca.knn,
                                 predict(pca, standard.test.pred)) == test.set$class)

#MODEL 10 RANDOM FOREST WITH PCA


rf.tune.pc <- function(pcn) {
  fit.pca.rf <- train(pca$x[, 1:pcn], training.set$class,
                       method = 'rf',
                       tuneGrid = data.frame(mtry = seq(1, 8, 1)), ntree = 100)
  return(caretEnsemble:: getMetric(fit.pca.rf))
}

set.seed(1986, sample.kind = 'Rounding')

rf.pc.options <- sapply(X = pcas.included , FUN =  rf.tune.pc)

rf.best.pc <- pcas.included[which.max(rf.pc.options)]

set.seed(1986, sample.kind = 'Rounding')
```

```
fit.pca.rf <- train(pca$x[, 1:rf.best.pc], training.set$class,
                    method = 'rf',
                    tuneGrid = data.frame(mtry = seq(1, 8, 1)),
                   importance = T, ntree = 500)


accuracy.pca.rf <- mean(predict(fit.pca.rf, predict(pca, standard.test.pred)) == test.set$class)
```

ANALYZING RESULTS & THE 5TH APPROACH - ENSEMBLE:

For all classification models, the complexity parameter chosen by the function was 0. This indicates that for best accuracy, the tree needed to branch out all the way

For all KNN models, 2 was chosen as k, the number of neighbors. This was the smallest option in tuning data frame provided to the function. This is likely to result in overfitting. If additinal data were to be provided and the predictions using this data was required, the accuracy of KNN models would most likely decrease.

For Random Forest models, the mtry parameters chosen were 2, 6 and 2 in that order.

When the variable importances of the random forest model created with manually chosen predictors, and the one created with all predictors are compared, it is seen that only 6 of the predictors that were chosen manually are also in top 20 most important variables in the other model.

This indicates that choices for the 1st approach were not as good as they should've been. This is normal since human error is factor in the creating of models using a manual approach.

```
## rf variable importance
##
##   variables are sorted by maximum importance across the classes
##          c-CS-m c-CS-s c-SC-m  c-SC-s t-CS-m t-CS-s t-SC-m t-SC-s
## SOD1_N    56.11  49.55 48.942 100.000  44.69  47.21 66.439  72.79
## APP_N     24.58  37.37 67.575  69.007  42.72  47.98 68.712  61.59
## BRAF_N    27.24  40.08 26.259  29.785  68.58  29.25 15.588  24.91
## CaNA_N    52.34  45.19 65.950  18.141  27.30  18.18 47.389  38.81
## pMTOR_N   41.39  32.22 37.456  31.440  56.76  25.48 59.484  55.53
## DYRK1A_N  22.27  32.51 52.593  41.296  40.07  27.69 27.613  30.38
## MTOR_N    44.11  12.32  7.603  33.269  43.59  49.58  9.696  36.04
## pERK_N    44.50  37.03 34.572  46.893  38.70  29.89 47.966  45.52
## pELK_N    27.05  39.98 14.595   6.694   0.00  28.55 25.024  41.72
## DSCR1_N   29.97  19.97  3.456  22.115  16.52  28.24 37.283  29.68
```

```
## rf variable importance
##
##   variables are sorted by maximum importance across the classes
##   only 20 most important variables shown (out of 71)
##
##              c-CS-m c-CS-s c-SC-m c-SC-s t-CS-m t-CS-s t-SC-m t-SC-s
## SOD1_N        87.87  80.13  76.89  97.46  77.97  75.30  81.75 100.00
## pPKCG_N       59.68  38.72  69.52  76.24  48.27  45.86  76.35  88.42
## APP_N         46.33  51.46  77.79  65.41  55.39  52.36  86.61  64.04
## BRAF_N        58.70  59.04  48.51  46.58  80.85  48.94  47.06  58.17
## ITSN1_N       49.86  42.25  80.02  66.87  52.78  54.34  63.48  49.42
## pERK_N        77.73  59.58  64.25  69.61  77.80  62.61  66.38  70.17
## pCAMKII_N     49.23  42.62  72.91  62.86  44.52  52.71  53.83  57.65
## pP70S6_N      53.95  36.26  51.86  46.62  36.16  48.43  48.32  72.20
## P38_N         46.86  50.06  42.69  41.96  44.67  72.00  39.66  46.34
```

```
## CaNA_N        66.73  64.69  70.40  50.30  62.50  48.10  69.11  57.66
## Tau_N         68.08  68.97  46.54  38.66  53.06  51.36  36.89  57.46
## DYRK1A_N      57.36  52.65  67.22  60.35  68.45  55.70  48.95  57.63
## pNUMB_N       51.89  48.41  34.31  67.12  39.08  39.45  53.25  38.45
## pPKCAB_N      41.33  61.47  53.78  48.34  50.93  45.03  59.13  66.44
## Ubiquitin_N   55.70  65.96  62.04  43.52  51.78  30.49  54.12  49.38
## S6_N          44.23  45.09  53.57  44.40  42.19  63.26  37.85  49.40
## AKT_N         33.53  36.86  55.32  60.99  50.60  26.75  40.52  39.61
## ARC_N         44.96  46.75  45.93  60.64  58.65  40.05  57.83  40.36
## AcetylH3K9_N  56.08  50.56  47.39  52.59  40.67  53.15  47.59  60.56
## pS6_N         41.77  50.36  47.46  59.55  54.13  41.46  56.98  46.59
```

The accuracy for each model is showm below.

model	accuracy
PCA Random Forest Model	0.9907407
Regular Random Forest Model	0.9884259
PCA KNN Model	0.9699074
Regular KNN Model	0.9074074
Manual Random Forest Model	0.8981481
Manual KNN Model	0.8402778
Regular Cls. Tree Model	0.7337963
Manual Cls. Tree Model	0.7106481
PCA Cls. Tree Model	0.6898148
Guessing Model	0.1087963

It is evident that the random forest models have the highest accuracy. The number of trees that were used for each forest was 500. The number was chosen to shorten the computing time. If higher number of trees were used, the accuracy of these models may have improved.

The table also shows that KNN models have highest accuracy after Random Forest. As noted before, since the number of neighbors were chosen as 2, the models are likely overfitted.

When we combine this with the limited number of instances in the test set, the accuracy of the KNN models is most likely an overestimate.

In fact, low number of instances most likely caused a somewhat higher accuracy to be achieved for all models. This is why it must be noted that the resulting accuracies should be evaluated with some scepticism. Had there been a higher number of instances avaiable in the data set, the accuracies yielded could have been accepted more confidently.

Nevertheless to improve the results further, an ensemble of 3 most accurate models are used. The prediction is selected by popular vote.

The accuracy obtained by the ensemble model is shown below.

```
## [1] 0.9976852
```

The ensemble by popular vote yields an almost perfect accuracy, which is higher that all of the individual models used to create the ensemble. This shows the power of ensembles, and how utlizing several models can eliminate mistakes made by them individually.

However, this result too should not be accepted with full confidence since the accuracy may be lower if the model is tested on data with larger amount of observations.

CONCLUSION:

In order to accomplish the task of predicting classes via the remaining 71 predictors in the training data set, several different approaches were employed and a total of 11 models were created.

The methodology employed in this work takes into account the limiations of computing power available, as well as the limitations posed by the relatively small number of instances.

Both of these limiations are clearly reflected on the models produced.

Reliability of all models tested could have been improved with the availability of more computing power and a data set with higher number of observations. This could have been achieved with usage of better cross validation, tuning and number of trees used in related models, as well as through better training due to increased number of observations.

Regardless of the limitations, if the final model was to be tested using new observations, it is believed that the it would yield a more than satisfactory result, even if the accuracy would be not be as high as what is indicated by the tests conducted in this work.

The results of this project could also be used to limit the number of protein levels tested, since it provides a concrete result regaring which of the protein expressions could be used to identify among classes, and which ones would not be helpful at all. This could possibly save both resources and time and render prospective laboratory test in this subject more productive.

REFERENCES:

Higuera C, Gardiner KJ, Cios KJ (2015) Self-Organizing Feature Maps Identify Proteins Critical to Learning in a Mouse Model of Down Syndrome. PLoS ONE 10(6): e0129126

Ahmed MM, Dhanasekaran AR, Block A, Tong S, Costa ACS, Stasko M, et al. (2015) Protein Dynamics Associated with Failed and Rescued Learning in the Ts65Dn Mouse Model of Down Syndrome. PLoS ONE 10(3): e0119491.