MATH 495 - Project #1

Richard Groenewald - 301263762

November 15th, 2018

```
load("druguse.Rdata")
suppressMessages(require(dplyr))
suppressMessages(require(rpart))
suppressMessages(require(rpart.plot))
```

```
## Warning: package 'rpart.plot' was built under R version 3.3.3
```

```
suppressMessages(require(ggplot2))
suppressMessages(require(randomForest))
```

```
## Warning: package 'randomForest' was built under R version 3.3.3
```

```
suppressMessages(require(caret))
```

```
## Warning: package 'caret' was built under R version 3.3.3
```

```
suppressMessages(require(mlbench))
```

```
## Warning: package 'mlbench' was built under R version 3.3.3
```

```
suppressMessages(require(e1071))
```

```
## Warning: package 'e1071' was built under R version 3.3.3
```

```
suppressMessages(require(class))
suppressMessages(require(neuralnet))
```

```
## Warning: package 'neuralnet' was built under R version 3.3.3
```

```
suppressMessages(require(GGally))
```

```
## Warning: package 'GGally' was built under R version 3.3.3
```

```
suppressMessages(require(corrplot))
```

```
## Warning: package 'corrplot' was built under R version 3.3.3
```

Question 1

We explore the relationship between ethnicity and UseLevel:

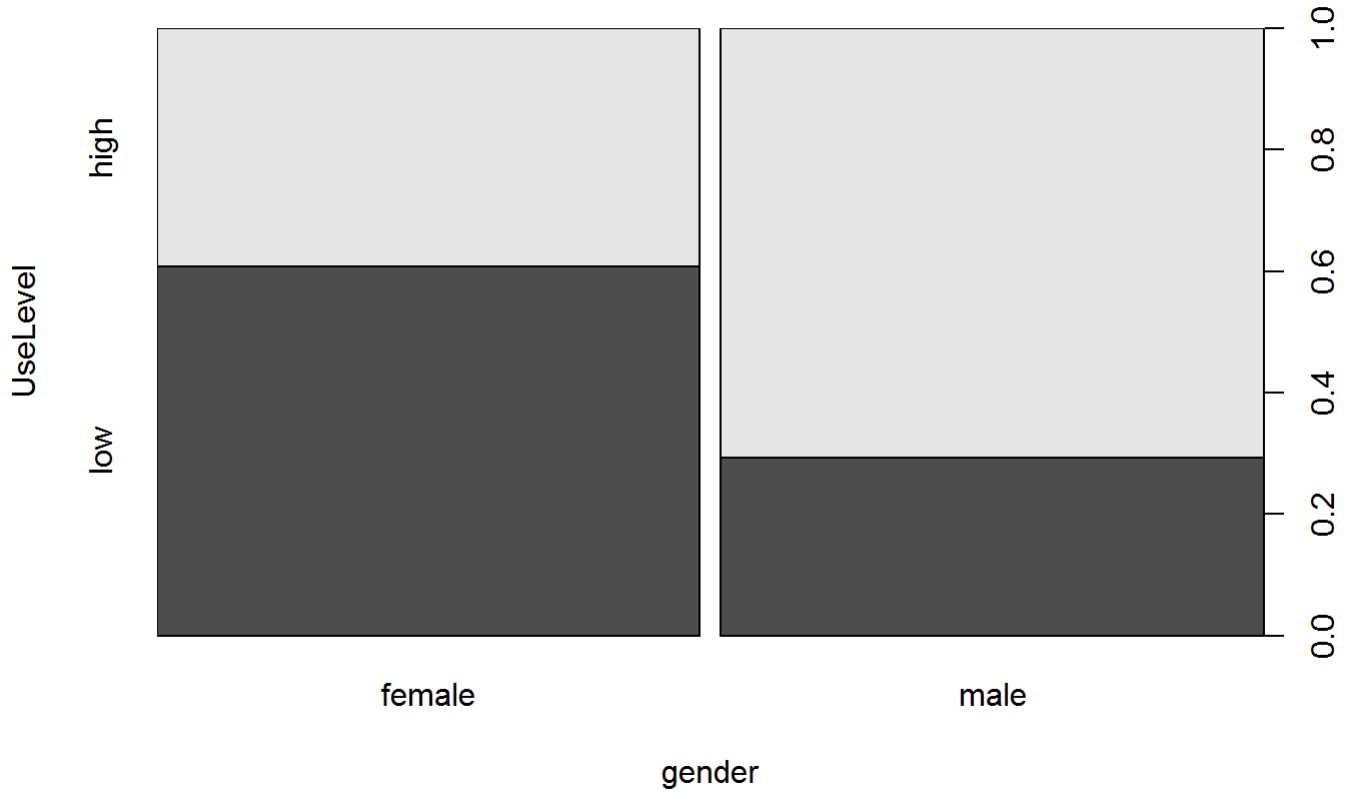
```
druguse %>% group_by(ethnicity,UseLevel) %>% tally()
```

```
## Source: local data frame [13 x 3]
## Groups: ethnicity [?]
##
##            ethnicity UseLevel     n
##               <fctr>   <fctr> <int>
## 1              Asian      low    19
## 2              Asian     high     7
## 3              Black      low    26
## 4              Black     high     7
## 5  Mixed-Black/Asian     high     3
## 6  Mixed-White/Asian      low     8
## 7  Mixed-White/Asian     high    12
## 8  Mixed-White/Black      low    10
## 9  Mixed-White/Black     high    10
## 10             Other      low    19
## 11             Other     high    44
## 12             White      low   767
## 13             White     high   953
```

We observe that individuals with *ethnicity* = white OR other have high use level signficantly more common than low use level.

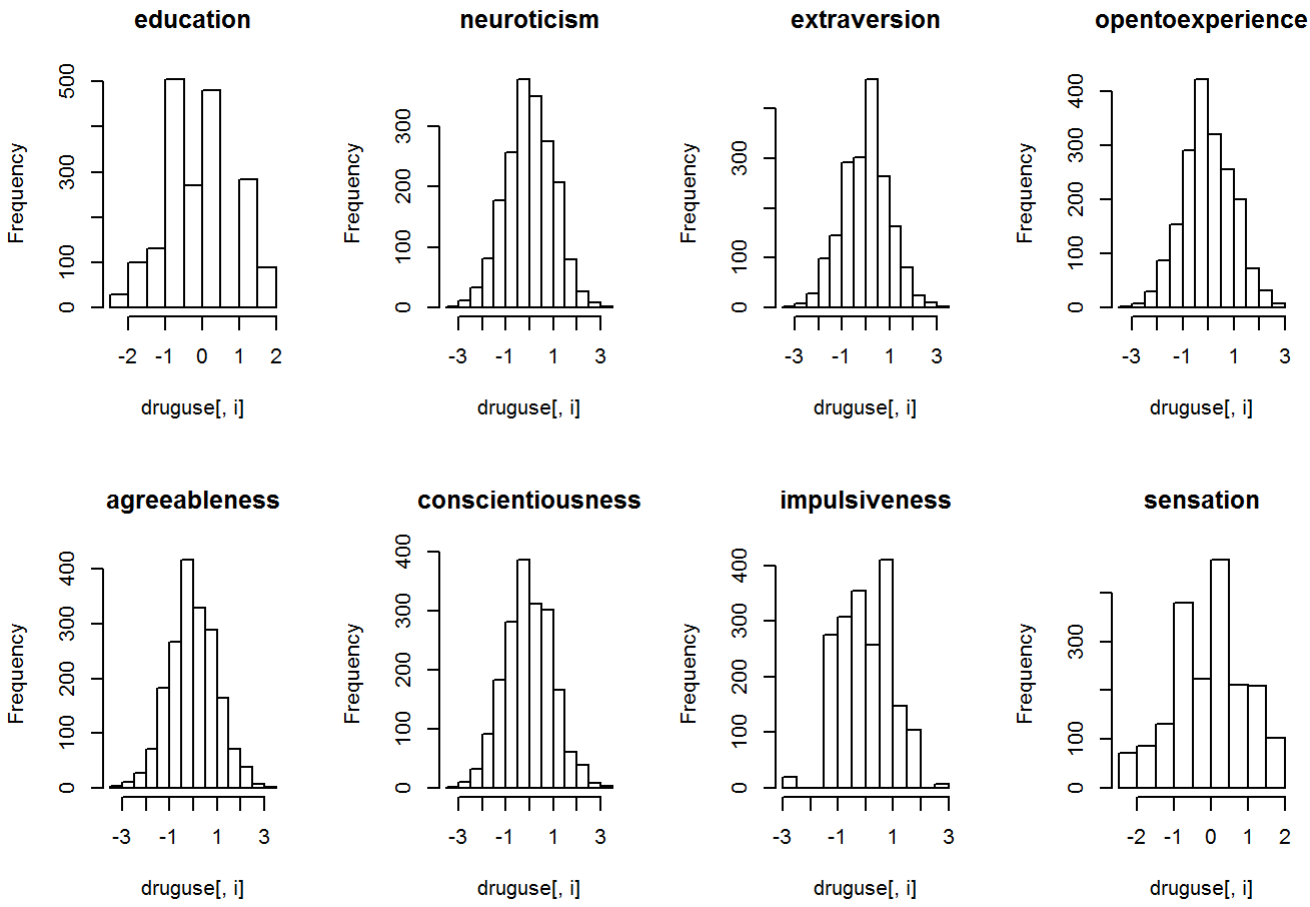
And similarly for gender, noting that men appear to have a much larger proportion of high level drug users than women:

```
plot(druguse$gender, druguse$UseLevel, xlab="gender", ylab="UseLevel")
```



We plot histograms of the the eight “continuous” variables to test graphically for normality (this is particularly important for applying a naive Bayes classifier below which assumes a Gaussian distribution for continuous variables). There seem to be some slight departures from this assumption in the cases of *education*, *impulsiveness* and *sensation*, but the remaining variables seem to satisfy it:

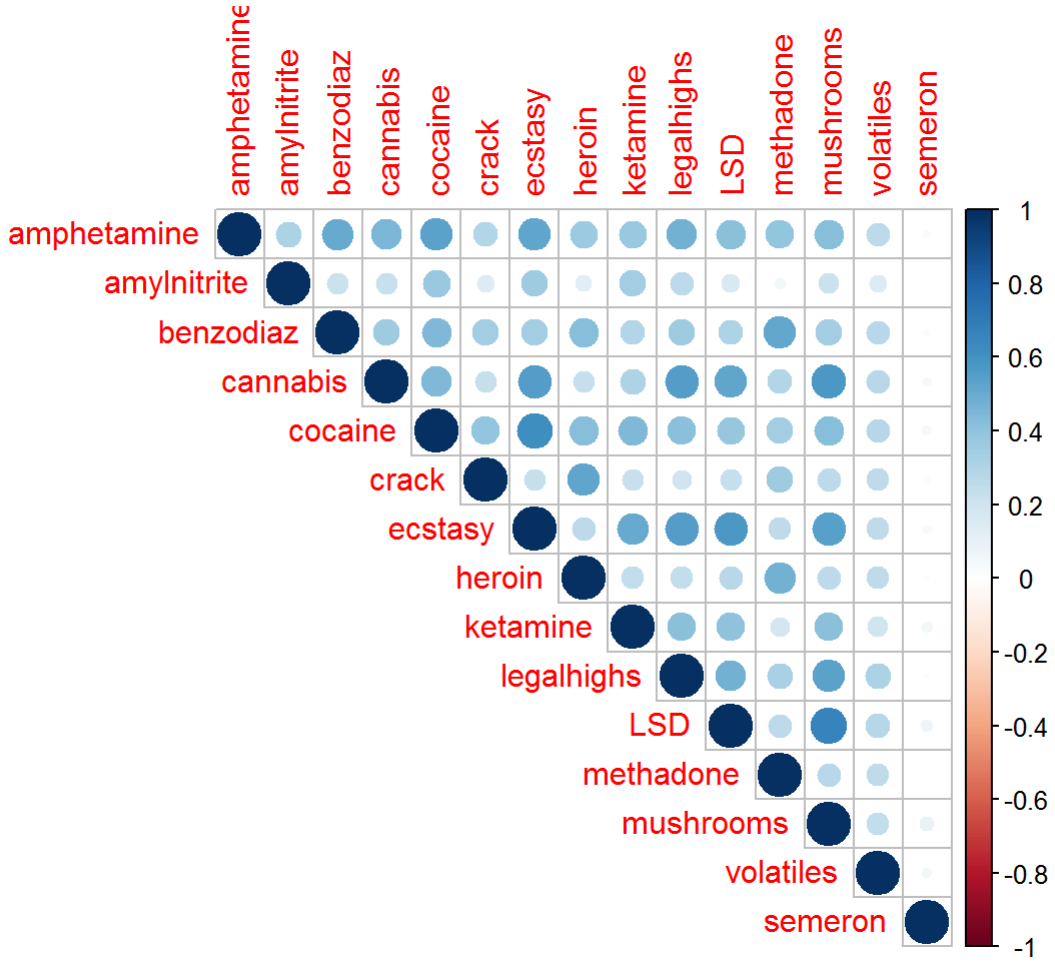
```
par(mfrow = c(2,4))
indices = c(3, 6:12)
for(i in indices){
  hist(druguse[,i], main = colnames(druguse)[i])
}
```




```
par(mfrow = c(1,1))
```

We produce a correplation plot between the illegal drugs to identify patterns that may be helpful in predicting one from several of the others (particularly for ideas in question 4):

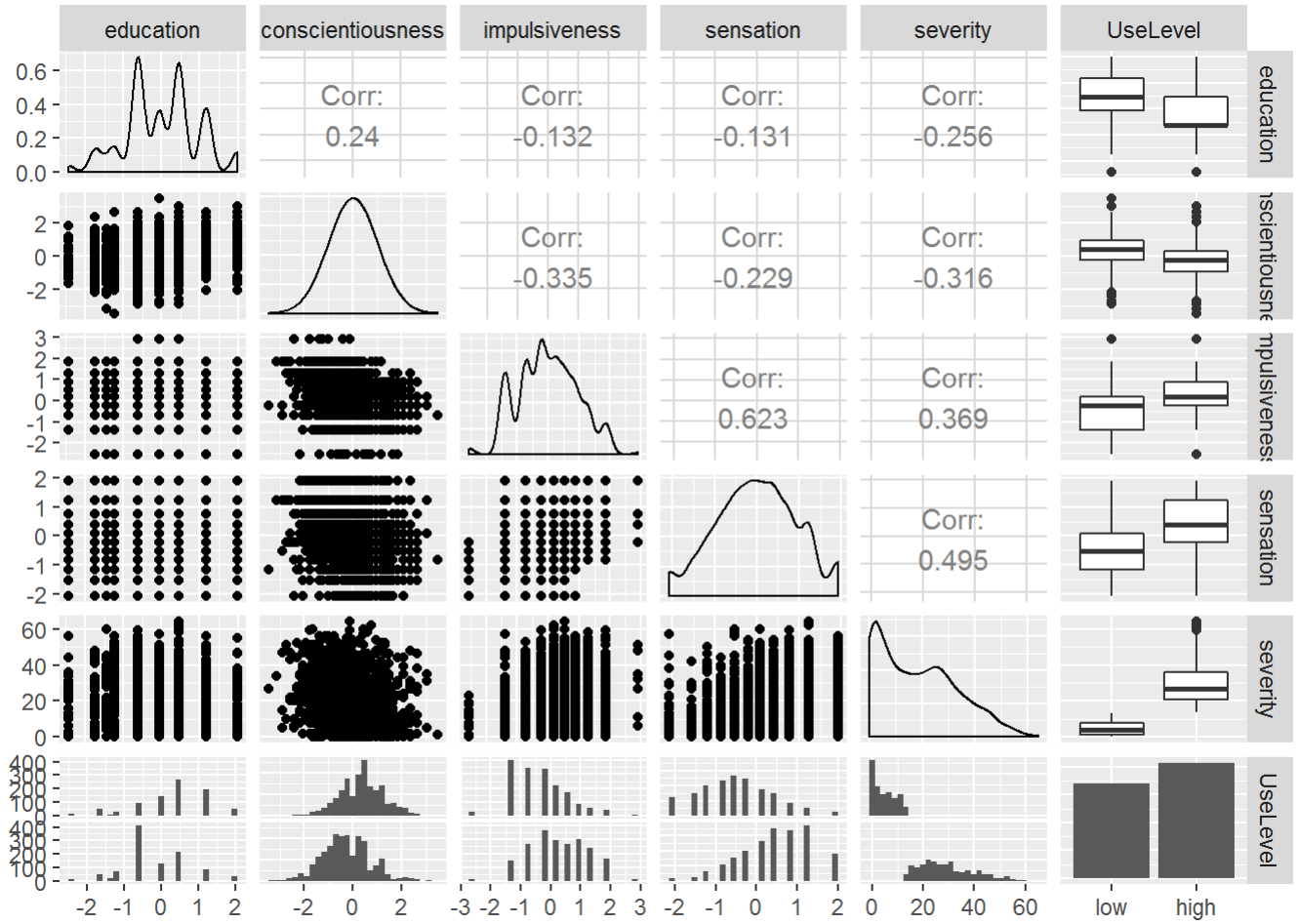
```
corrs = cor(druguse[,17:31])
corrplot(corrs, type ="upper")
```



We note somewhat high correlation (≈ 0.6) between heroin and crack and heroin and benzodiaz. It is worth noting that there eappear to be no negative correlations among the different types of illegal drugs.

```
ggpairs(druguse[,c(3,10:12,33,34)])
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



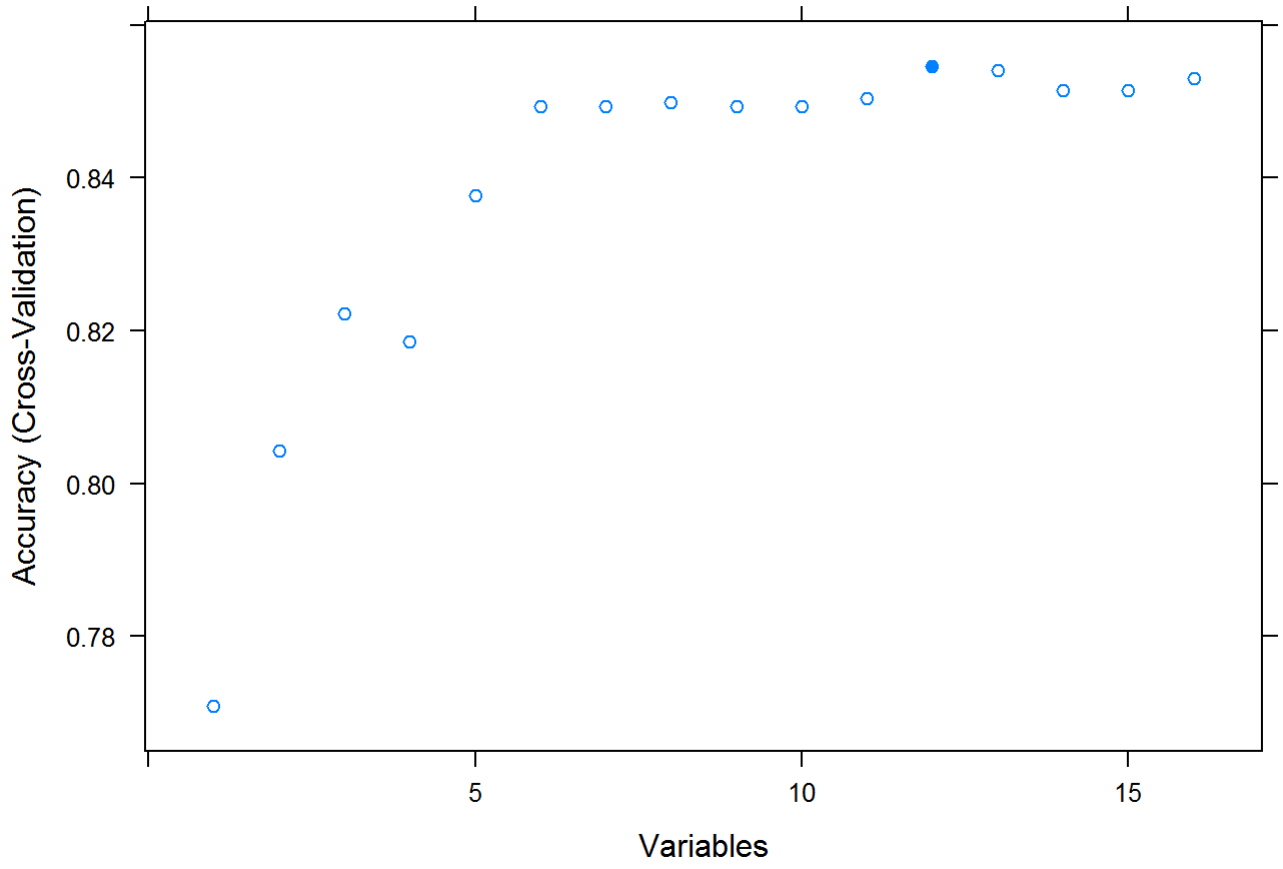
We note that based on the barplots in the above figure, UseLevel may be separated somewhat by *sensation*, *impulsiveness* and *education*. We also see mild/small correlations between all of these predictors and severity, which is surprising given their apparent relationship with *UseLevel*.

Looking ahead to problems 2 and 3, we first desire to limit the model space to a reasonable size (since the full space has 2^{16} total models due to our 16 possible predictors - we do not consider interaction terms). While limiting the space will optimistically bias an estimate of test error later, we need to do so due to computational time. Particularly for problem 3, we determine **sets** of candidate variables using the *caret* package and stepwise logistic regression - choosing the sets with the lowest cross validated test error from the *caret* package and lowest Bayesian Information Criterion values from stepwise logistic regression, after which we will compare the test error of different learning methods on EACH set of variables. Choosing many sets helps reduce the bias.

```
predictors = druguse[, 1:16] #extract relevant predictors/response
response = druguse[, 34]
full = druguse[,c(1:16,34)]
```

In the *caret* package, using the Recursive Feature Elimination (RFE) function:

```
set.seed(12345)
#5 fold CV with random forest method
controlset = rfeControl(functions = rfFuncs, method = "cv", number = 5)
 #compute estimated test error for each size of model - 1 variable to 16
res = rfe(predictors, response, size = 1:16, rfeControl = controlset)
plot(res)
```




```
res$optVariables
```

```
##  [1] "nicotine"          "country"           "sensation"
##  [4] "opentoexperience"  "conscientiousness" "gender"
##  [7] "agegroup"          "education"         "impulsiveness"
## [10] "agreeableness"     "neuroticism"       "extraversion"
```

```
#define matrix with TRUE/FALSE entries determining variable inclusion/exclusion
modelspace = matrix(ncol = 16, nrow = 13); colnames(modelspace) = colnames(predictors)
#extract variables from RFE
varmtx = res$variables[res$variables[,6]=="Fold1",4:5]
#fill space
for (i in 6:16){
  modelspace[i-5,] = (colnames(modelspace) %in% varmtx$var[varmtx$Variables==i])
}
```

We note from the plot of the estimated test error for each of the model sizes that models with the number of predictors larger than 5 appear to have high estimated test accuracy. We save the variables from models with 6 or more predictors for later use, and note the “best” model’s variables above.

We add two more sets of variables to the model space using the Akaike/Bayesian Information Criterion (AIC/BIC) (stepwise from intercept only model to reduce the number of fitted variables). $AIC(m) = -2log(L_m) + 2k$ and $BIC(m) = -2log(L_m) + k\log(n)$, where L_m is the realized likelihood of an arbitrary model m which fits k

parameters with n observed data points. We attempt to choose the model with minmimum AIC or BIC (selecting the minimum in a stepwise procedure).

```
nullmod = glm(UseLevel~1, family=binomial(link="logit"), data=full)
#determine "bounds" for model variables
fullmod = glm(UseLevel~., family=binomial(link="logit"), data=full)

#fit with criteria
AICmod = step(nullmod, scope = list(lower=nullmod, upper=fullmod), direction="both", k=2, trace=
FALSE)
BICmod = step(nullmod, scope = list(lower=nullmod, upper=fullmod), direction="both", k=log(nrow(
full)), trace=FALSE)

#extract variable names
AICmodvars = all.vars(AICmod$formula)[-1]
BICmodvars = all.vars(BICmod$formula)[-1]

#add to model space for Q3
modelspace[12,] = colnames(modelspace) %in% AICmodvars
modelspace[13,] = colnames(modelspace) %in% BICmodvars
```

Question 2

2.1

Based on the output from the *rfe* function above, and based on the desirability for interpretability in our initial approach, we fit a logistic regression model using the “best” subset of variables discovered via the test accuracy estimates in *rfe*.

```
train = full[1:1500,]
test = full[1501:1885,]
form = as.formula(paste("UseLevel ~", paste(res$optVariables, collapse = " + ")))
form
```

```
## UseLevel ~ nicotine + country + sensation + opentoexperience +
##     conscientiousness + gender + agegroup + education + impulsiveness +
##     agreeableness + neuroticism + extraversion
```

```
logregmod_q2 = glm(form, family = binomial(link="logit"), data=train) #fitted model
summary(logregmod_q2)
```

```
##
## Call:
## glm(formula = form, family = binomial(link = "logit"), data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.4803  -0.4097   0.1208   0.4270   2.8849
##
## Coefficients:
##                            Estimate Std. Error z value Pr(>|z|)
## (Intercept)               -0.006226   0.530913  -0.012 0.990643
## nicotine                   0.487909   0.037297  13.082  < 2e-16 ***
## countryCanada             -0.857596   0.617299  -1.389 0.164750
## countryNewZealand         -0.638589   1.489233  -0.429 0.668066
## countryOther              -1.335886   0.597736  -2.235 0.025423 *
## countryRepublicofIreland  -2.005239   0.881236  -2.275 0.022877 *
## countryUK                 -2.328460   0.508081  -4.583 4.59e-06 ***
## countryUSA                -0.145895   0.532432  -0.274 0.784072
## sensation                  0.702890   0.124375   5.651 1.59e-08 ***
## opentoexperience           0.585545   0.103403   5.663 1.49e-08 ***
## conscientiousness         -0.336936   0.100315  -3.359 0.000783 ***
## gendermale                 0.991588   0.177479   5.587 2.31e-08 ***
## agegroup25-34              0.321533   0.235950   1.363 0.172973
## agegroup35-44             -0.132670   0.243178  -0.546 0.585363
## agegroup45-54             -0.620625   0.261134  -2.377 0.017471 *
## agegroup55-64             -0.967433   0.389875  -2.481 0.013087 *
## agegroup65+              -16.749192 483.016139  -0.035 0.972338
## education                  -0.268986   0.093779  -2.868 0.004127 **
## impulsiveness              -0.005360   0.109902  -0.049 0.961101
## agreeableness              -0.075178   0.094050  -0.799 0.424093
## neuroticism                -0.061117   0.101642  -0.601 0.547644
## extraversion               -0.108384   0.105472  -1.028 0.304134
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2061.47  on 1499  degrees of freedom
## Residual deviance:  965.43  on 1478  degrees of freedom
## AIC: 1009.4
##
## Number of Fisher Scoring iterations: 15
```

Based on this output, we see that (based on the limiting distributions of maximum likelihood estimates) that *nicotine*, *country*, *sensation*, *opentoexperience*, *conscientiousness*, *gender*, *age* and *education* have statistically significant coefficients, while the remaining coefficients have large standard errors.

Our fitted probability (that an individual will have *high* usage) given covariates x is:

$$\hat{P}(x) = \frac{exp(\hat{\beta}^T x)}{1 + exp(\hat{\beta}^T x)}$$

where $\hat{\beta}$ is the vector of above coefficients. We predict that an individual will have a high UseLevel if this fitted probability is greater than 0.5:

2.2

```
#predictions
predLR = predict(logregmod_q2, test, type="response")
predLR = factor(ifelse(predLR < .5, "low", "high"))

#confusion matrix
xtabs(~ predLR + test$UseLevel)
```

```
##       test$UseLevel
## predLR low high
##   high  25  175
##   low  156   29
```

```
#accuracy
sum(predLR == test$UseLevel)/nrow(test)
```

```
## [1] 0.8597403
```

The above table shows that 25 *low* usage individuals are misclassified as *high* and 29 *high* usage individuals are misclassified as being *low*.

Our accuracy is $\frac{156+175}{385} = 0.8597$, or 85.97%

Question 3

3.1

We perform ten fold cross validation to select an optimal model from the model space determined during our exploratory data analysis (question 1) using different statistical learning methods. Since the number of observations is 1885, the first nine folds have 188 observations and the last has 193. For the sake of conciseness, we include only the analysis for the optimal model type, allowing the reader to make minor variations to the commented code below to view output for alternative methods (CTRL + SHIFT + C to remove/restore comments to a highlighted section).

For methods such as support vector machines, neural networks and K-nearest-neighbours which require numeric data (or a distance metric), we have encoded our categorical data to be numeric and standardized it prior to fitting the models. For our naive Bayes predictions, we use our own defined function as shown below (both of these choices are further discussed in a *very brief* Mathematical Appendix):

```
#NAIVE BAYES------------------------
naive_bayes_predictor = function(predictors, response, xnew){
  resp = as.numeric(response) - 1
  priorhigh = sum(resp)/nrow(predictors)
  priorlow = 1-priorhigh

  #defining helper functions to estimate conditional probabilities/densities
          helper = function(col, x){
            if(is.numeric(col) & length(unique(col)) > 7){ #this condition is very specific for
 our dataset
              x = as.numeric(x)
              #continuous case
              meanhigh = mean(col[resp==1])
              sdhigh = sd(col[resp==1])

              meanlow = mean(col[resp==0])
              sdlow = sd(col[resp==0])

              outvec = c(dnorm(x, meanhigh, sdhigh), dnorm(x, meanlow, sdlow))
              names(outvec) = c("high", "low")

              return(outvec)
            } else {
              probhigh = sum(col == x & resp == 1)/sum(resp == 1)

              problow = sum(col == x & resp == 0)/sum(resp == 0)

              outvec = c(probhigh, problow)
              names(outvec) = c("high", "low")
              return(outvec)
            }
          }

  probsmtx = matrix(nrow=2, ncol = length(xnew))
  rownames(probsmtx) = c("high","low")
  for (i in 1:length(xnew)){
    probsmtx[,i] = helper(predictors[,i], xnew[i])
  }

  highprob = prod(probsmtx[1,])*priorhigh

  lowprob = prod(probsmtx[2,])*priorlow

  return(factor(ifelse(highprob>lowprob, "high", "low")))

}
```

```
set.seed(1)
permutation_inds = sample(nrow(full))
resultsmtx = data.frame(modelspace, Error = rep(NA,nrow(modelspace)), Accuracy = rep(NA, nrow(mo
delspace)))

for (j in 1:nrow(modelspace)){
  loop_predictors = predictors[, modelspace[j,]]
  loop_full = full[, c(modelspace[j,],TRUE)]
  testerrvec = NULL
  accuracyvec = NULL
  for (i in 1:10){
    inds = (188*(i-1)+1):(188*i)
    if (i ==10){
      inds = (188*(i-1)+1):1885
    }

    trainpreds = loop_predictors[permutation_inds[-inds],]
    trainresp = response[permutation_inds[-inds]]
    trainset = loop_full[permutation_inds[-inds],]

    testpreds = loop_predictors[permutation_inds[inds],]
    testresp = response[permutation_inds[inds]]
    testset = loop_full[permutation_inds[inds],]

    #LOGISTIC REGRESSION:
        # logregmod = glm(UseLevel ~ ., family = binomial(link="logit"), data = trainset)
        # predprobs = predict(logregmod, newdata = testpreds, type = "response")
        # preds = factor(ifelse(predprobs >.5, "high", "low"))

    #NAIVE BAYES:
        # preds = apply(testpreds, 1, naive_bayes_predictor, predictors = trainpreds, response =
 trainresp)

    #DECISION TREE:
        # dectreemod = rpart(UseLevel ~ ., data=trainset)
        # preds = predict(dectreemod, newdata = testpreds, type="class")

    #RANDOM FOREST:
        rfmod = randomForest(UseLevel ~., data =trainset, ntree = 550)
        preds = predict(rfmod, newdata = testpreds)

    #SUPPORT VECTOR MACHINES:
        # indx <- sapply(trainset, is.factor)
        # indx[length(indx)] = FALSE
        # svmtrainset = trainset
        # svmtrainset[indx] <- lapply(svmtrainset[indx], function(x) as.numeric(x))
        # svmtrainset[1:(ncol(svmtrainset)-1)] = lapply(svmtrainset[1:(ncol(svmtrainset)-1)], sc
ale)
        # svmtestset = testset
        # svmtestset[indx] <- lapply(svmtestset[indx], function(x) as.numeric(x))
        # svmtestset[1:(ncol(svmtestset)-1)] = lapply(svmtestset[1:(ncol(svmtestset)-1)], scale)
        #
        # svmmod = svm(UseLevel ~ ., data = svmtrainset, kernel = "linear")
```

```
        # preds = predict(svmmod, svmtestset,kernel="linear")

    #NEURAL NETWORK:
        # indx <- sapply(trainset, is.factor)
        # indx[length(indx)] = FALSE
        # nntrainset = trainset
        # nntrainset[indx] <- lapply(nntrainset[indx], function(x) as.numeric(x))
        # nntrainset[1:(ncol(nntrainset)-1)] = lapply(nntrainset[1:(ncol(nntrainset)-1)], scale)
        # nntrainset$UseLevel = ifelse(nntrainset$UseLevel=="high", 1,0)
        # nntestset = testset
        # nntestset[indx] <- lapply(nntestset[indx], function(x) as.numeric(x))
        # nntestset[1:(ncol(nntestset)-1)] = lapply(nntestset[1:(ncol(nntestset)-1)], scale)
        # nntestset$UseLevel = ifelse(nntestset$UseLevel=="high", 1,0)
        #
        # #taken from practical
        # n = names(nntrainset)
        # form = as.formula(paste("UseLevel ~", paste(n[!n %in% "UseLevel"], collapse = " + ")))
        # nnmod = neuralnet(form, nntrainset, linear.output = FALSE)
        # predprobs = compute(nnmod, nntestset[,-ncol(nntestset)]); predprobs = predprobs$net.re
sult
        # preds = ifelse( predprobs >.5, 1, 0)
        # testresp = ifelse(response[permutation_inds[inds]]=="high",1,0)


    #K-NEAREST NEIGHBOURS:
        # indx <- sapply(trainset, is.factor)
        # indx[length(indx)] = FALSE
        # knntrainset = trainset
        # knntrainset[indx] <- lapply(knntrainset[indx], function(x) as.numeric(x))
        # knntrainset[1:(ncol(knntrainset)-1)] = lapply(knntrainset[1:(ncol(knntrainset)-1)], sc
ale)
        # knntestset = testset
        # knntestset[indx] <- lapply(knntestset[indx], function(x) as.numeric(x))
        # knntestset[1:(ncol(knntestset)-1)] = lapply(knntestset[1:(ncol(knntestset)-1)], scale)
        #
        # preds = knn(knntrainset[, -ncol(knntrainset)], knntestset[,-ncol(knntestset)], trainre
sp, k = 35)
        # #manually (innefficiently) tested different k - anything from 25 to 40 seemed approxim
ately satisfactory

    testerror = sum(preds != testresp)/nrow(testset)
    accuracy = sum(preds == testresp)/nrow(testset)

    testerrvec = c(testerrvec, testerror)
    accuracyvec = c(accuracyvec, accuracy)

  }
  resultsmtx$Error[j] = mean(testerrvec)
  resultsmtx$Accuracy[j] = mean(accuracyvec)
}
```

The minimum estimated test error via cross validation over all features in the model space in the methods that are commented out was:

1. logistic regression: 0.140109
2. naive Bayes: 0.1523
3. decision tree: 0.17775
4. support vector machines: 0.1586
5. neural net: 0.16073
6. k-nearest-neighbours: 0.15548

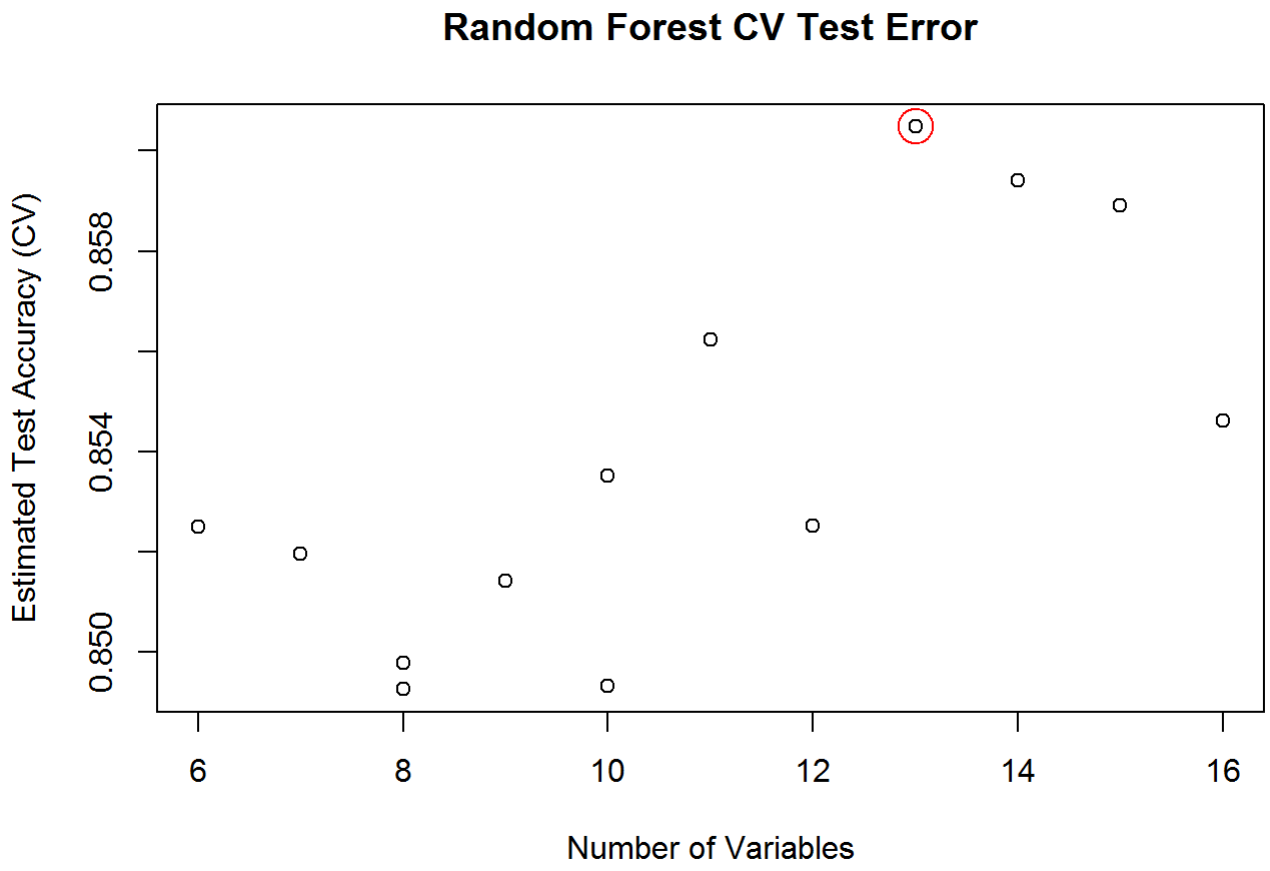
Our chosen model is a RANDOM FOREST with the following predictors:

```
varindicator = unlist(resultsmtx[which.min(resultsmtx$Error), 1:16])
finalmodvars = colnames(resultsmtx[,1:16])[varindicator]
finalmodvars
```

```
##  [1] "agegroup"          "gender"            "education"
##  [4] "country"           "neuroticism"       "extraversion"
##  [7] "opentoexperience"  "agreeableness"     "conscientiousness"
## [10] "impulsiveness"     "sensation"         "nicotine"
## [13] "alcohol"
```

We plot the accuracy against the number of predictors (note that some of the model sizes have more than one model fitted with different variables):

```
varnumber = rowSums(as.matrix(resultsmtx[,1:16]))
plot(varnumber, resultsmtx$Accuracy, xlab = "Number of Variables", ylab = "Estimated Test Accura
cy (CV)", main = "Random Forest CV Test Error")
points(varnumber[which.max(resultsmtx$Accuracy)], resultsmtx$Accuracy[which.max(resultsmtx$Accur
acy)], col="red", cex=2.5)
```




```
#Note maximum at 13 variables, our optimal model found above
```

3.2

Our criterion for choosing this model was to maximize prediction accuracy, so we have already made this estimation. Based on the cross validation above, we would expect our model to have accuracy in prediction of approximately 86.05% (implying prediction error of 13.95%)

```
#estimated accuracy of optimal model:
resultsmtx$Accuracy[which.max(resultsmtx$Accuracy)]
```

```
## [1] 0.8604784
```

```
#estimated error rate:
resultsmtx$Error[which.min(resultsmtx$Error)]
```

```
## [1] 0.1395216
```

It is worth noting that logistic regression yields a NARROWLY greater predicted test error in far less computing time. It may be worth using the logistic regression predictor alongside the random forest because the performance is so similar. However, we report our minimum as our final model. To summarize, this model is:

```
finalmoddf = druguse[,c(finalmodvars, "UseLevel")]
final_mod = randomForest(UseLevel ~ ., data=finalmoddf, ntree=550)
final_mod
```

```
##
## Call:
##  randomForest(formula = UseLevel ~ ., data = finalmoddf, ntree = 550)
##                Type of random forest: classification
##                      Number of trees: 550
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 14.54%
## Confusion matrix:
##      low high class.error
## low  703  146   0.1719670
## high 128  908   0.1235521
```

Question 4

4.1

```
druguse$heroin_binary = factor(ifelse(druguse$heroin > 0, "yes", "no"))
```

The variables *any*, *UseLevel* and *severity* should **NOT** be used for the following reasons:

1. *any* - we know all of the users' drug specifications other than heroin. If any of those specifications are nonzero, then *any* will be TRUE, and as such it is redundant (completely determined by other available data). If all other specifications are ZERO, then the value of *any* tells us if the subject in quesiton uses heroin - and this is unrealistic because we are assuming we do not know the variable we desire to predict.

(2)/(3) *UseLevel* and *severity* were determined entirely based on the set of drugs and other variables, and if used (and are correlated with the response variable), will be confounded with the effects of the other predictors.

Additionally, *heroin* is of course omitted from our set of predictors since it completely determines our binary response. We construct analysis based on all other variables:

```
excl = which(names(druguse) %in% c("any", "UseLevel", "severity", "heroin"))
full = druguse[, -excl]
predictors = full[,-ncol(full)]
response = full$heroin_binary
```

We desire to select variables using ten fold cross validation as follows:

1. Start with no predictors. Choose the one for which cross validated prediction error is lowest (1 of 30 models)
2. Given i variables in the model, compare the test error of adding one of the remaining variables not in the model. Add the variable for which the $i+1$ predictor model has the lowest cross validation test error (1 of 30-i models).
3. At the end of the procedure, there will be 30 total random forest models, with precisely one having i variables, $i \in \{1, \ldots, 30\}$.

4. Our final selection is the model with the lowest test error.

This setup is similar to forward selection using the *step* function in linear or logistic regression. The problem with it is that it entails fitting $\sum_{i=1}^{30} 1 = (30)(31)/2$ sets of variables, and cross validating on each one. This implies fitting and prediction 4650 times on random forest models, which is too computationally expensive in our current case (it takes two and a half hours to run on my personal computer - there are undoubtedly more efficient ways to use this exact algorithm) - the optimal model produced via this method includes the variables: *crack*, *methadone*, *benzodiaz*, *mushrooms*, *amphetamine*, *cocaine*, *chocolate* and *legalhighs* with estimated test error in cross validation of 0.08592217 (or accuracy of 0.914). My code which implements this algorithm is presented below:

```
#eval = FALSE due to knitting time issues
system.time({

candidates = predictors #these are candidates we consider adding to the model
set.seed(1)
permutation_inds = sample(nrow(full))
optimalmodslist = list()
errorforoptimalmodels = NULL

for (i in 1:ncol(predictors)){
  print(ncol(candidates))

  CVerr = 2 #guarantees the first model will be "accepted"
  for (j in 1:ncol(candidates)){
    if (ncol(candidates) == ncol(predictors)){
      temppreds = data.frame(candidates[,j])
      colnames(temppreds) = colnames(candidates)[j]
    } else {
      if(ncol(candidates)==1){
        temppreds = predictors
      } else{
        temppreds = data.frame(currentpreds, candidates[,j])
        colnames(temppreds)[i] = colnames(candidates)[j]
      }
    } #extract variable to potentially add to model

    i_varmodel_temp = names(temppreds) #stores names of potential model

    temporaryfullset = data.frame(temppreds, heroin_binary = full$heroin_binary)
    errvec = NULL

    for (k in 1:10){
      inds = (188*(k-1)+1):(188*k)
      if (k ==10){
        inds = (188*(k-1)+1):1885
      }

      trainset = temporaryfullset[permutation_inds[-inds],]
      testset = temporaryfullset[permutation_inds[inds],]

      rfmod = randomForest(heroin_binary~., data = trainset)
      preds = predict(rfmod, newdata = testset)

      error = mean(preds != testset$heroin_binary)
      errvec = c(errvec, error)
    } #ten fold cross validation on potential model

    overall_err = mean(errvec)

    if (overall_err < CVerr){
      CVerr = overall_err
      optmod = i_varmodel_temp
    } #compares CV error on temporary model to lowest so far
```

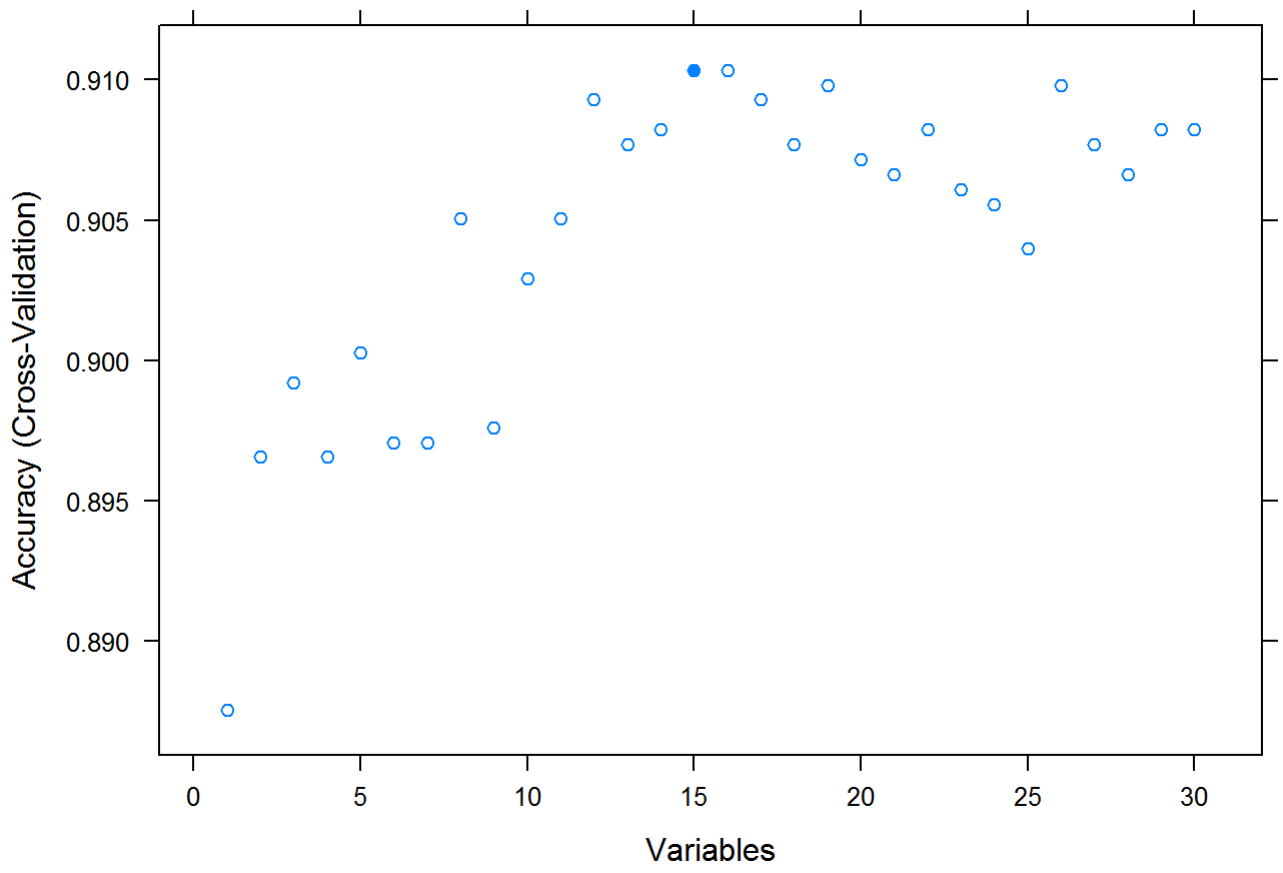
```
    #SUMMARY: CROSS VALIDATE ON EACH CANDIDATE AND INCLUDE THE ONE WITH THE SMALLEST CV ERROR
  }
  optimalmodslist[[i]] = optmod #store optimal model with i variables
  errorforoptimalmodels = c(errorforoptimalmodels, CVerr) #store the CV error for the optimal mo
del
  candidates = data.frame(candidates[,-(which(names(candidates) %in% optimalmodslist[[i]]))])
  #remove the added predictor from candidates
  currentpreds = data.frame(predictors[, optimalmodslist[[i]]]) #construct the current model's d
ata frame
  colnames(currentpreds) = optimalmodslist[[i]]
  print(colnames(currentpreds))
}
})
```

Additionally, we consider selecting a classifier using the (more efficient) caret package as follows:

```
set.seed(1)
controlset = rfeControl(functions = rfFuncs, method = "cv", number = 5) #5 fold CV with random f
orest method
res = rfe(predictors, response, size = 1:30, rfeControl = controlset) #sets up to 30 variables i
n fit
res$optVariables
```

```
##  [1] "crack"           "methadone"       "cocaine"
##  [4] "LSD"             "benzodiaz"       "amphetamine"
##  [7] "country"         "ketamine"        "ecstasy"
## [10] "mushrooms"       "agegroup"        "sensation"
## [13] "volatiles"       "amylnitrite"     "opentoexperience"
```

```
plot(res)
```




```
#8 variable model from RFE, to contrast its predictors with mine:
varmtx = res$variables[res$variables[,6]=="Fold1",4:5]
varmtx$var[varmtx$Variables==i]
```

```
##  [1] "crack"           "methadone"       "LSD"
##  [4] "benzodiaz"       "cocaine"         "country"
##  [7] "amphetamine"     "ketamine"        "sensation"
## [10] "opentoexperience"
```

We see that a 15 variable method is chosen, and the 8 variable model selected by the RFE function is different from the one selected in my algorithm (see above). It would appear that any of the models from 12 variables to 20 produce similar expected test errors. While we have estimated test error rates for these models from separate cross validations, we desire to compare them directly. We do this using using ten fold cross validation and we choose the model with the smallest estimated error rate:

```
set.seed(12)
permutation_inds = sample(nrow(full))
RFE_error = NULL
my_method_error = NULL
RFE_df_full = full[, c(res$optVariables, "heroin_binary")]
my_method_df_full = full[, c("crack", "methadone", "benzodiaz", "mushrooms", "amphetamine", "coc
aine", "chocolate", "legalhighs", "heroin_binary")]

for (i in 1:10){
  inds = (188*(i-1)+1):(188*i)
  if (i ==10){
    inds = (188*(i-1)+1):1885
  }

  train_mm = my_method_df_full[permutation_inds[-inds],]
  test_mm = my_method_df_full[permutation_inds[inds],]

  train_RFE = RFE_df_full[permutation_inds[-inds],]
  test_RFE = RFE_df_full[permutation_inds[inds],]

  mymod = randomForest(heroin_binary ~., data=train_mm)
  mypreds = predict(mymod, newdata = test_mm)
  myerr = sum(mypreds != test_mm$heroin_binary)/nrow(test_mm)
  my_method_error = c(my_method_error, myerr)

  RFEmod = randomForest(heroin_binary ~., data=train_RFE)
  RFEpreds = predict(RFEmod, newdata = test_RFE)
  RFEerr = sum(RFEpreds != test_RFE$heroin_binary)/nrow(test_RFE)
  RFE_error = c(RFE_error, RFEerr)

}

#Error rate from RFE function:
mean(RFE_error)
```

```
## [1] 0.09343788
```

```
#Error rate from my model selection algorithm:
mean(my_method_error)
```

```
## [1] 0.09182835
```

Note that the 8 variable model has a slightly lower estimate of the expected test error. We have an estimated error of 9.183% and thus estimated accuracy of 90.817%. Additionally, having fewer predictors, this model is more parsimonious and less time consuming to fit. Thus we use it to form our final model:

```
finalmodQ4.1 = randomForest(heroin_binary ~ ., data = my_method_df_full)
finalmodQ4.1
```

```
##
## Call:
##  randomForest(formula = heroin_binary ~ ., data = my_method_df_full)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 8.81%
## Confusion matrix:
##       no yes class.error
## no  1555  50  0.03115265
## yes  116 164  0.41428571
```

4.2

It is often conjectured that marijuana is a “gateway drug” in the sense that it is used as a first illegal substance for many individuals who end up using other substances. With this background interest in mind, we attempt to build a classifier for whether an individual has ever used any drugs *other than* cannabis. The set of potential predictors are the 16 personal traits and non illegal substances used in questions 2 and 3, along with *cannabis* We first define our variable of interest:

```
druguse$allelse = apply(druguse[,c(17:19, 21:31)],1,sum)
druguse$allelse = factor(ifelse(druguse$allelse>0, "yes", "no"))
full = druguse[,c(1:16,20,ncol(druguse))]
```

We are interested in determining whether *cannabis* is a significant predictor when adjusting for all other potential predictors, so we attempt to find the “best” model INCLUDING *cannabis* as a predictor, and the “best” model EXCLUDING it as a predictor, and compare the accuracy of these two models using cross validation. We are approaching the problem similarly to one of ANOVA, except the models are not necessarily nested and we are relying on prediction accuracy as our main criterion and not maximum likelihood theory.

The definition of “best” will be the one provided by stepwise regression, as was detailed in question 1. We select two models as follows based on the Akaike Information Criterion (I also performed the following using BIC, but prediction errors were higher - we allow the reader to change k=log(nrow(full)) if they desire to see this):

```
k = 2
null_excl = glm(allelse~1, family=binomial(link="logit"), data=full[,-17])
full_excl = glm(allelse~., family=binomial(link="logit"), data=full[,-17])
AICmod_excl = step(null_excl, scope = list(lower=null_excl, upper=full_excl), direction="both",
k=k, trace=FALSE)
#extract variable names for model excluding cannabis
AICmodvars_excl = all.vars(AICmod_excl$formula)
AICmod_excl$formula
```

```
## allelse ~ nicotine + country + sensation + conscientiousness +
##     opentoexperience + extraversion + gender + ethnicity + agegroup +
##     impulsiveness + agreeableness
```

```
null_incl = glm(allelse~cannabis, family=binomial(link="logit"), data=full)
full_incl = glm(allelse~., family=binomial(link="logit"), data=full)
AICmod_incl = step(null_incl, scope = list(lower=null_incl, upper=full_incl), direction="both",
k=k, trace=FALSE)
#extract variable names for model excluding cannabis
AICmodvars_incl = all.vars(AICmod_incl$formula)
AICmod_incl$formula
```

```
## allelse ~ cannabis + conscientiousness + nicotine + agegroup +
##     country + impulsiveness + opentoexperience + extraversion
```

Note the formulas indicating which variables are included in each. We also note that excluding the feature selection from the cross validation procedure will optimistically bias the estimate of the error found below, it is reasonable to assume that our conclusion regarding *cannabis* will be unaffected (and this may also produce a good final model regardless). We compare the two models below:

```
set.seed(123)
permutation_inds = sample(nrow(full))
model_inc_err = NULL
model_excl_err = NULL
model_inc_df = full[,AICmodvars_incl]
model_excl_df = full[,AICmodvars_excl]

for (i in 1:10){
  inds = (188*(i-1)+1):(188*i)
  if (i ==10){
    inds = (188*(i-1)+1):1885
  }

  train_incl = model_inc_df[permutation_inds[-inds],]
  test_incl = model_inc_df[permutation_inds[inds],]

  train_excl = model_excl_df[permutation_inds[-inds],]
  test_excl = model_excl_df[permutation_inds[inds],]

  modincl = rpart(allelse ~., data=train_incl)
      #modincl = glm(allelse ~., data=train_incl, family=binomial(link="logit"))
  predsincl = predict(modincl, newdata = test_incl, type="class")
      #predsincl = ifelse(predict(modincl, test_incl, type = "response")>.5, "yes", "no")
  errincl = sum(predsincl != test_incl$allelse)/nrow(test_incl)
  model_inc_err = c(model_inc_err, errincl)

  modexcl = rpart(allelse ~., data=train_excl)
      #modexcl = glm(allelse ~., data=train_excl, family=binomial(link="logit"))
  predsexcl = predict(modexcl, newdata = test_excl, type="class")
      #predsexcl = ifelse(predict(modexcl, test_excl, type = "response")>.5, "yes", "no")
  errexcl = sum(predsexcl != test_excl$allelse)/nrow(test_excl)
  model_excl_err = c(model_excl_err, errexcl)

}

#Error rate from excluding cannabis:
mean(model_excl_err)
```

```
## [1] 0.1867655
```

```
#Error rate from including cannabis:
mean(model_inc_err)
```

```
## [1] 0.1304652
```

The error rates for other models can be found simply by replacing *rpart* with *glm* or *randomForest* and I list them below:

1. AIC, logistic regression : 0.1624 excluding, 0.1326 including
2. AIC, random forest: 0.167 excluding, 0.1405 including

3. BIC, decision tree: 0.184 excluding, 0.1310 including
4. BIC, logistic regression: 0.162 excluding, 0.144 including
5. BIC, random forest: 0.178 excluding, 0.149 including

For the reason of optimizing estimated test error, we choose a DECISION TREE including *cannabis* with the following call:

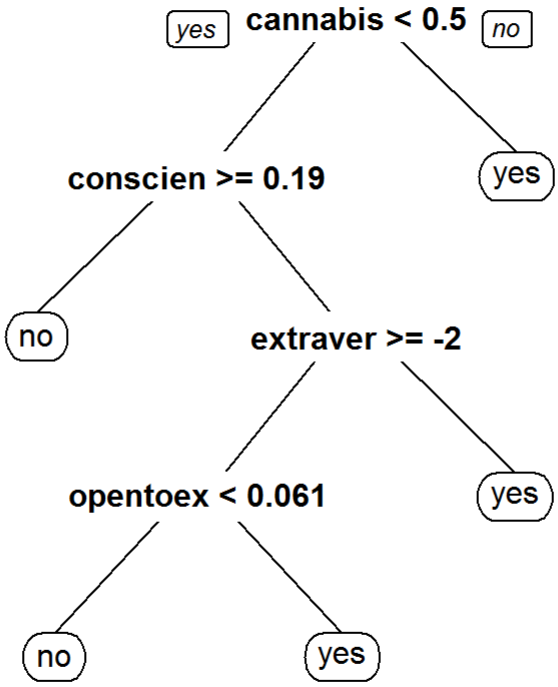
```
AICmod_incl$formula
```

```
## allelse ~ cannabis + conscientiousness + nicotine + agegroup +
##     country + impulsiveness + opentoexperience + extraversion
```

```
finalmod = rpart(AICmod_incl$formula, data = model_inc_df)
```

The above output indicates that this model performs better than all of the others listed, with an estimated accuracy of 86.95%. Below is its plot to indicate the decisions graphically:

```
rpart.plot.version1(finalmod)
```



Finally, we address the estimate of the loss. We have used the typical loss function $\frac{1}{n^*} I(y_{pred} \neq y_{obs})$, where $n^* =$ number of predictions. Our estimate of the loss is given by the estimated cross validation error of 13.04%.

Mathematical Notes

(1)

I address our implementation of naive Bayes in question 3 because we did not discuss the following in class. We proceeded as follows: Let $Y =$ *UseLevel* and $X = (x_1, \ldots, x_p)^T$ be our vector of covariates. For each test observation, we classify Y to $y_k \in \{$*high*,*low*$\}$ if the following is maximized:

$$\prod_{i=1}^{p} \hat{P}(x_i|y_{obs})\hat{P}(Y = y_{obs}) \propto \hat{P}(Y = y_k|X)$$

For categorical data, this procedure is the same as in class with $\hat{P}$ equal to the population proprotions in all cases. However, with the continuous variables in our dataset, we have set $\hat{P}(x_i|y_k) = \phi(\mu_{y_k}, \sigma_{y_k})$, where ϕ is a Gaussian density and μ, σ are the conditional sample means for x_i in the y_k category.

(2)

In computing our distances between points in k-nearest-neighbours in question 3, an alternative method could have been to add a discrete metric (for x, y categorical observations $d(x, y) = \delta > 0$ if $x \neq y$, $d(x, y) = 0$ if $x = y$), instead of encoding to numeric and using the overall Euclidean metric on the data. This may make more sense for categorical variables that are not ordinal.

Informal Citations

https://machinelearningmastery.com/feature-selection-with-the-caret-r-package/ (https://machinelearningmastery.com/feature-selection-with-the-caret-r-package/)

Our course webpage and practicals on Canvas