Homework 2 - Data Science Methods

Hans de Ferrante, 910957

1 Question 1: Lasso & Ridge

a) Explain what Lasso and Ridge tend to do to the coefficients of two highly correlated variables in a linear model.

First, let us start by stating the obvious: Lasso and Ridge are both shrinkage methods. This will mean that both methods will tend to decrease the average norm of the coefficient estimates as λ increases. One important difference between the two is that Lasso will set some coefficients explicitly to zero if we increase the tuning parameter λ ("Lasso kicks out"), whereas Ridge will simply keep shrinking both coefficients proportionally. This effectively means that Lasso will tend to select only one of the highly correlated variables for large λ, whereas Ridge will keep both coefficients.

b) Does Lasso select the (true non-zero) regressors with probability one as the sample grows large? Explain.

I don't think we have discussed this at length in the lectures (although we did see that for the adaptive lasso this result holds since adaptive lasso satisfies the oracle properties and consistent variable selection is one of these properties). On the slides, however, it is briefly mentioned that indeed Lasso is able to select the true model with probability one as the sample grows large provided that the true model is sparse and has p not too large compared to n. I did not recall having discussed this in the lecture so I decided to consult the literature on Lasso. I found it quite technical and difficult to understand but hope I understood it correctly.

In this literature, authors make the distinction between (i) estimation consistency and (ii) model consistency of Lasso (e.g. Zhao2006). With regards to (i) it has been shown that under mild regularity conditions Lasso is indeed estimation consistent. Hence, we can expect the zero regressor coefficient estimates to converge in probability to 0 as $n \to \infty$. (ii) is more tricky and requires the Irrepresentable Condition, variations of which have been derived independently by different authors. These include Zhao2006, Zou2006 and Meinshausen2006. Zhao2006 is a highly cited paper that provides a simple example of a case where the condition does not hold, and consequently Lasso is unable to select the correct model [Zhao and Yu, 2006]. The example works with $n = 1000$ and $p = 3$ so I do think it presents a situation where n grows large compared to p as we assumed in the question.

I find it therefore difficult to come up with a non-ambiguous answer to this question as the answer depends on how we define Lasso selecting the true non-zero regressors with probability one as the sample grows large. As $n \to \infty$ the zero regressors will converge to 0 so for sure we will know which regressors are truly non-zero and which are not in the limit. However, we would need to check the Irrepresentable Condition to verify whether Lasso would truly kick out non-zero regressors (arbitrarily close to 0 is not 0). Downside is that this is practically impossible as we would need the true covariance matrix for checking the condition.

Long story short: If we only need estimation consistency, yes: zero coefficients converging to 0 is established result. If we want model consistency in the sense that we want Lasso to kick all zero regressors out, not generally; we would need to verify IC.

References

[Zhao and Yu, 2006] Zhao, P. and Yu, B. (2006). On model selection consistency of lasso. *J. Mach. Learn. Res.*, 7:2541–2563.

2 Question 2

Prove that the l2-norm $||\hat{\beta}_{RIDGE}||$ of the ridge estimates $\hat{\beta}_{RIDGE}$ from a ridge regression of y on X ($n \times p$ full rank matrix, with $p < n$) is always decreasing in λ.

Recall from the LN that $\hat{\beta}_{RIDGE} = (X'X + \lambda I)^{-1}X'y$. Note that under the assumptions $p < n$ and X $(n \times p)$ full rank, we can use the SVD for X, i.e. $X = UDV'$ with U and V unitary matrices, D a diagonal matrix with diagonal entries $d_{ii} > 0 \quad \forall \quad i, 0$ otherwise. This implies in particular $D' = D$, $U'U = I_p$, and $V' = V^{-1}$. Hence, we can rewrite

$$\begin{aligned}
\hat{\beta}_{RIDGE} &= [(UDV')'UDV' + \lambda I]^{-1}(UDV')'y = (VD'U'UDV' + \lambda I)^{-1}VD'U'y \\
&= (VD'U'UDV' + \lambda I)^{-1}VD'U'y = (VD^2V' + \lambda I)^{-1}VDU'y = \\
&= (VD^2V' + \lambda VV')^{-1}VDU'y = (V(D^2 + \lambda I)V')^{-1}VDU'y = \\
&= (V')^{-1}(D^2 + \lambda I)^{-1}V^{-1}VDU'y = V(D^2 + \lambda I)^{-1}V^{-1}VDU'y = \\
&= V(D^2 + \lambda I)^{-1}DU'y.
\end{aligned}$$

where we use in addition to the mentioned implications the following (trivial) results from linear algebra:

$$(ABC)^{-1} = C^{-1}B^{-1}A^{-1} \qquad \& \qquad (ABC)' = C'B'A'.$$

Now we know that by definition for the l_2-norm of vectors, we can write $||\hat{\beta}_{RIDGE}||^2 = \hat{\beta}'_{RIDGE}\hat{\beta}_{RIDGE}$. If we plug in the expression above, we find therefore:

$$\begin{aligned}
||\hat{\beta}_{RIDGE}||^2 &= (V(D^2 + \lambda I)^{-1}DU'y)'V(D^2 + \lambda I)^{-1}DU'y \\
&= (U'y)'D((D^2 + \lambda I)^{-1})'V'V(D^2 + \lambda I)^{-1}DU'y \\
&= (U'y)'D(D^2 + \lambda I)^{-1}(D^2 + \lambda I)^{-1}D(U'y) \\
&= x'Ax
\end{aligned}$$

where the last equality shows that this is a quadratic form with $A = D^2(D^2+\lambda I)^{-2}$ and $x = U'y$. Now getting rid of the matrix notation, we see that

$$||\hat{\beta}_{RIDGE}||^2 = \begin{bmatrix} x_1 & x_2 & \dots & x_p \end{bmatrix} \begin{bmatrix} \frac{d_{11}^2}{(d_{11}^2+\lambda)^2} & 0 & \dots & 0 \\ 0 & \frac{d_{22}^2}{(d_{22}^2+\lambda)^2} & \dots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & \frac{d_{pp}^2}{(d_{pp}^2+\lambda)^2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_p \end{bmatrix} = \frac{x_1^2 d_{11}^2}{(d_{11}^2 + \lambda)^2} + \dots + \frac{x_p^2 d_{pp}^2}{(d_{pp}^2 + \lambda)^2}$$

It is trivial to see from this final equation that this final equation is decreasing in λ as all numerators increase in λ and denominators remain the same. This implies that $||\hat{\beta}_{RIDGE}|| = \sqrt{||\hat{\beta}_{RIDGE}||^2}$ is too decreasing in λ as the square root transformation is a monotonic one.

Question 3

March 16, 2018

Data pre-processing

The data has problems. Let us first alleviate some of these.

- The number of employees has “5-Jan” and “25-Jun”. I don’t quite believe the description that this means e.g. 25 employees in June. I think it is an artefact from a computer translating 1-5 and 6-25 to dates.
- Gender has all different types of entries. Some of these are misspellings of male/female. There are about 30 some other entries. These include non-binary, neuter, etc. Let’s collapse these into “LBGTQ”.
- Work-interference has NAs. Initially I thought these were missing values. However, the question is “If you have a mental health condition, do you feel that it interferes with your work?” The other factors are “Never”, “Rarely”, “Sometimes”, “Often”. Hence, having NA reported on this probably reflects persons not having a mental illness. Plotting the distribution of treatment over these factors shows that this is indeed the case (see Appendix).
- Some people report ages below 0 and over 100. Let us remove any observations that fall not within the reasonable age limit of 28-75.
- I think originating countries are not very informative as is since it is extremely unbalanced. We will deal with this later.

```
# Read in data and fix no_employees. Also filter based on non-sensical ages.
set.seed(418910)
df <- read_csv("mentalhealth.csv") %>%
  mutate(no_employees = ifelse(no_employees == "25-Jun","6-25",no_employees)) %>%
  mutate(no_employees = ifelse(no_employees == "5-Jan","1-5",no_employees)) %>%
  mutate(care_options = ifelse(care_options == "Not sure", "Don't know", care_options)) %>%
  mutate(work_interfere = ifelse(is.na(work_interfere), "NA", work_interfere)) %>%
  mutate(Age = as.integer(Age)) %>%
  filter(Age > 17) %>% filter(Age < 76)
```

```
## Parsed with column specification:
## cols(
##   .default = col_character(),
##   Age = col_double()
## )
```

```
## See spec(...) for full column specifications.
```

```
# Remap genders to male, female or LBTGQ.
male <- c("male", "cis male", "m","make","mal","maile","mail","malr", "cis man", "M", "male (cis)", "ci
female <- c("female", "f", "cis female", "female","femake","woman","cis_female/femme","female (cis)","f
df <- df %>% mutate(Gender = ifelse(tolower(Gender) %in% male, "M", Gender)) %>%
  mutate(Gender = ifelse(tolower(Gender) %in% female, "F", ifelse(Gender == "M", "M", "LBGTQ")))

# Remove uninformative features.
df$Timestamp <- NULL
```

Pairwise correlations

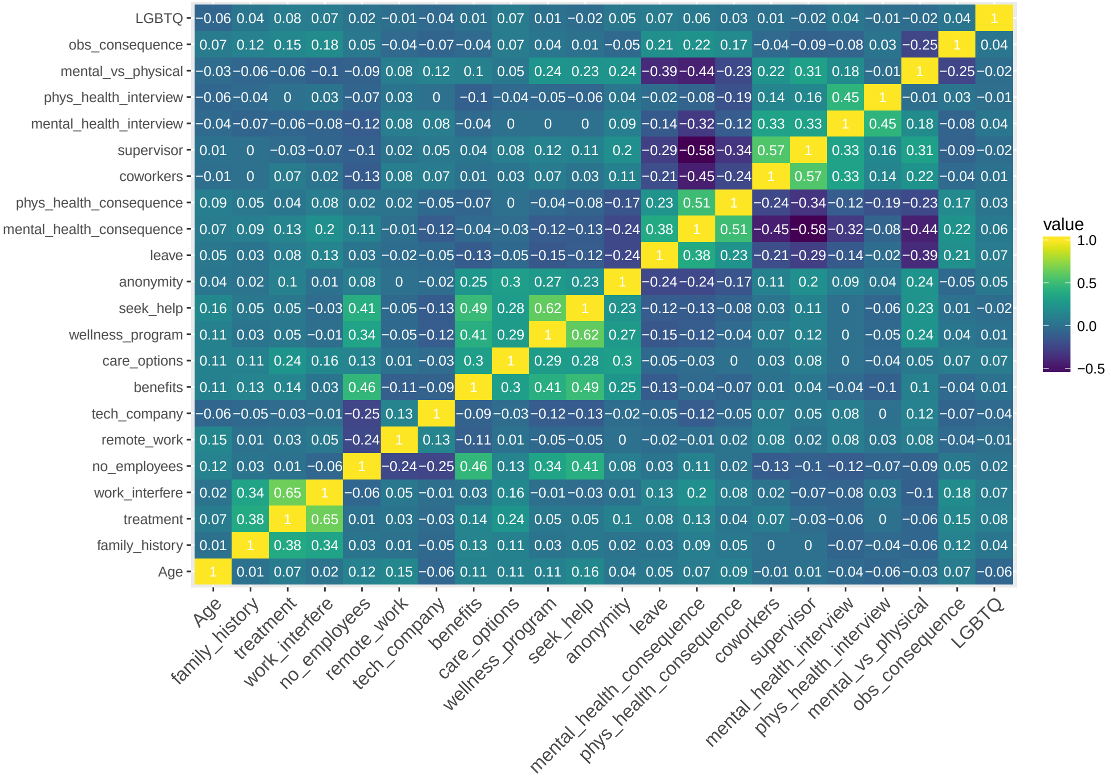
Translate some of the factors to a numeric format. This will allow us to display a correlation matrix.

```r
# Appropriate ordering for levels.
yn <- c("No","Yes")
ynm <- c("No","Maybe","Yes")
yndn <- c("No", "Don't know", "Yes")
yns <- c("No","Some of them","Yes")
wi <- c("NA","Never","Rarely","Sometimes","Often")
cs <- c("1-5", "6-25", "26-100", "100-500", "500-1000", "More than 1000")
lv <- c("Very easy", "Somewhat easy", "Don't know", "Somewhat difficult", "Very difficult")

df2 <- df %>% mutate(obs_consequence = as.numeric(factor(obs_consequence, levels = yn)) - 1) %>%
  mutate(tech_company = as.numeric(factor(tech_company, levels = yn)) - 1) %>%
  mutate(remote_work = as.numeric(factor(remote_work, levels = yn)) - 1) %>%
  mutate(treatment = as.numeric(factor(treatment, levels = yn)) - 1) %>%
  mutate(family_history = as.numeric(factor(family_history, levels = yn)) - 1) %>%
  mutate(phys_health_interview = as.numeric(factor(phys_health_interview, levels = ynm)) - 1) %>%
  mutate(mental_health_interview = as.numeric(factor(mental_health_interview, levels = ynm)) - 1) %>%
  mutate(phys_health_consequence = as.numeric(factor(phys_health_consequence, levels = ynm)) - 1) %>%
  mutate(mental_health_consequence = as.numeric(factor(mental_health_consequence, levels = ynm)) - 1) %
  mutate(benefits = as.numeric(factor(benefits, levels = yndn)) - 1) %>%
  mutate(care_options = as.numeric(factor(care_options, levels = yndn)) - 1) %>%
  mutate(wellness_program = as.numeric(factor(wellness_program, levels = yndn)) - 1) %>%
  mutate(seek_help = as.numeric(factor(seek_help, levels = yndn)) - 1) %>%
  mutate(anonymity = as.numeric(factor(anonymity, levels = yndn)) - 1) %>%
  mutate(mental_vs_physical = as.numeric(factor(mental_vs_physical, levels = yndn)) - 1) %>%
  mutate(supervisor = as.numeric(factor(supervisor, levels = yns)) - 1) %>%
  mutate(coworkers = as.numeric(factor(coworkers, levels = yns)) - 1) %>%
  mutate(leave = as.numeric(factor(leave, levels = lv)) - 1) %>%
  mutate(no_employees = as.numeric(factor(no_employees, levels = cs)) - 1) %>%
  mutate(work_interfere = as.numeric(factor(work_interfere, levels = wi)) - 1)

df2$LGBTQ <- as.numeric(df$Gender=="LBGTQ")

df2[,sapply(df2, is.numeric)] %>% cor() %>% round(2) %>% melt() %>%
  ggplot(aes(x=Var1, y=Var2, fill=value)) +
  geom_tile() +
  geom_text(aes(label = round(value, 2)),size=3,colour='white') +
  scale_fill_viridis() +
  theme(axis.text.x = element_text(angle = 45, vjust = 1, size = 12, hjust = 1),
        axis.title = element_blank())
```



From the heatmap of correlations, it becomes immediately clear that work interference and a family history of mental illness are variables that correlate most with having sought treatment. We conclude that it was a good idea we've kept work interference in there. Other variables, except for maybe knowing about care options, tend to correlate little with having sought treatment for mental illness. Despite lack of correlations for most variables, let us not discard any of the variables. We will probably not run into trouble as e.g. random forests are quite robust against overfitting.

Preprocessing of countries/states

We obviously do not want to discard all information on the region a person is from; whether a person seeks treatment may be influenced by both culture as well as where this person is from. Mapping countries to world regions is unsatisfactory for the Americas as the samples from the Americas are overrepresented. An approach to find reasonably balanced groups is to (i) map all non-Americas countries to their respective regions if are present at least 10 times, otherwise "Other", (ii) map states in the US to regions in the US. Let us do this and show what balance we end up with.

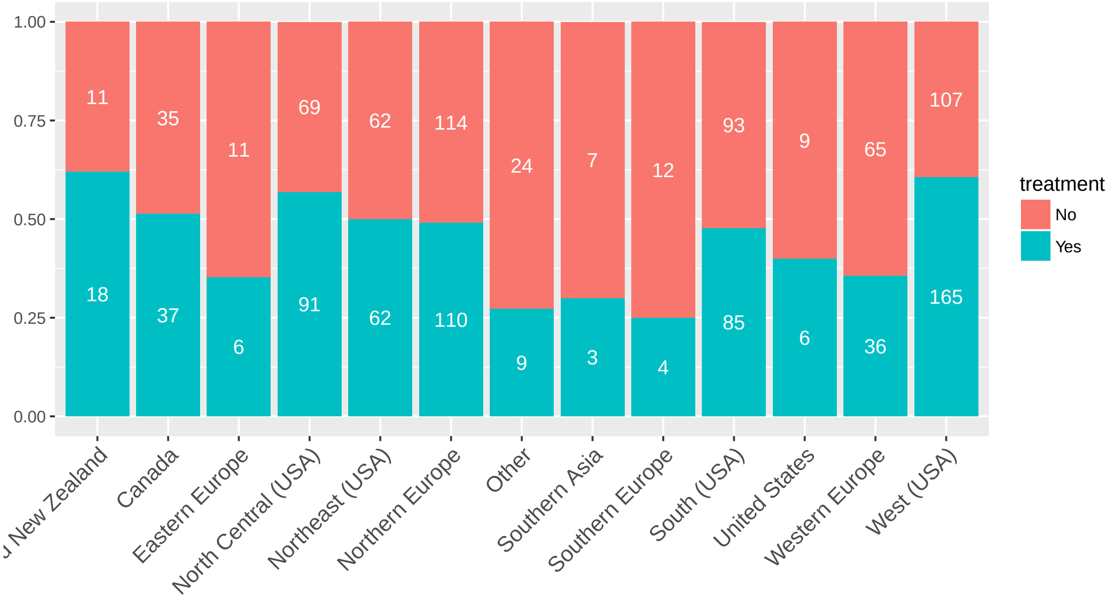
```
# Find world regions for countries with at least 10 counts.
df$worldregion <- countrycode(df$Country, "country.name", "region", warn = TRUE, custom_dict = NULL, cu
df <- df %>% group_by(worldregion) %>% mutate(n=n()) %>% ungroup() %>% mutate(worldregion = ifelse(n<10

# Find US regions for states.
USstates <- cbind(as.character(datasets::state.region), datasets::state.abb) %>% as.data.frame()
names(USstates) <- c("US_region","state")
df <- left_join(df,USstates)
```

```
## Joining, by = "state"
## Warning: Column `state` joining character vector and factor, coercing into
## character vector
```

```
# Define region as U.S. region, country if it is in Northern America or otherwise world region. Remove
df <- df %>% mutate(region = ifelse(is.na(US_region),ifelse(worldregion=="Northern America", Country, w
df$Country <- NULL
df$US_region <- NULL
df$worldregion <- NULL
df$state <- NULL

# Make a plot
df %>% mutate(treatment=factor(treatment), region=factor(region)) %>%
  group_by(region,treatment) %>%
  summarize(n=n()) %>% mutate(percent=n/sum(n)) %>%
  ggplot(aes(x=region, y=percent, fill=treatment)) +
  geom_bar(stat="identity") +
  geom_text(aes(label=n), position=position_stack(vjust=0.5), colour="white") +
  theme(axis.text.x = element_text(angle = 45, vjust = 1, size = 12, hjust = 1),
      axis.title = element_blank())
```



The figure shows cumulative distributions of having sought treatment for mental illness over different geographic regions. We see that they are reasonably balanced in terms of numbers of people per bin (in any case more balanced than the original data) and also that there are quite some differences between the different different geographic groups. This motivates why we should keep geographic data.

Transforming some of the original data and splitting.

```
# Where ordering is important.
df <- df %>% mutate(supervisor = as.numeric(factor(supervisor, levels = yns)) - 1) %>%
  mutate(coworkers = factor(coworkers, levels = yns)) %>%
  mutate(leave = factor(leave, levels = lv)) %>%
  mutate(no_employees = factor(no_employees, levels = cs)) %>%
  mutate(work_interfere = factor(work_interfere, levels = wi))

# Also cast other characters into factor.
df$n <- NULL
factorcols <- names(df[-1])
df[factorcols] <- lapply(df[factorcols], factor)

# Split into training and testing set. We will work with a 70:30 balance.
set.seed(1414)
is.train <- sample(c(TRUE,FALSE), nrow(df), replace = T, prob = c(0.7, 0.3))
train <- df[is.train,]
test <- df[is.train==FALSE,]
```

Learning from the data

Here, we will train classifiers on the training observations and evaluate test performance with the test set. For reproducibility, we will set the seed before training. We will evaluate the performance of bagging, logistic regression, random forests and constructing a simple classification tree.

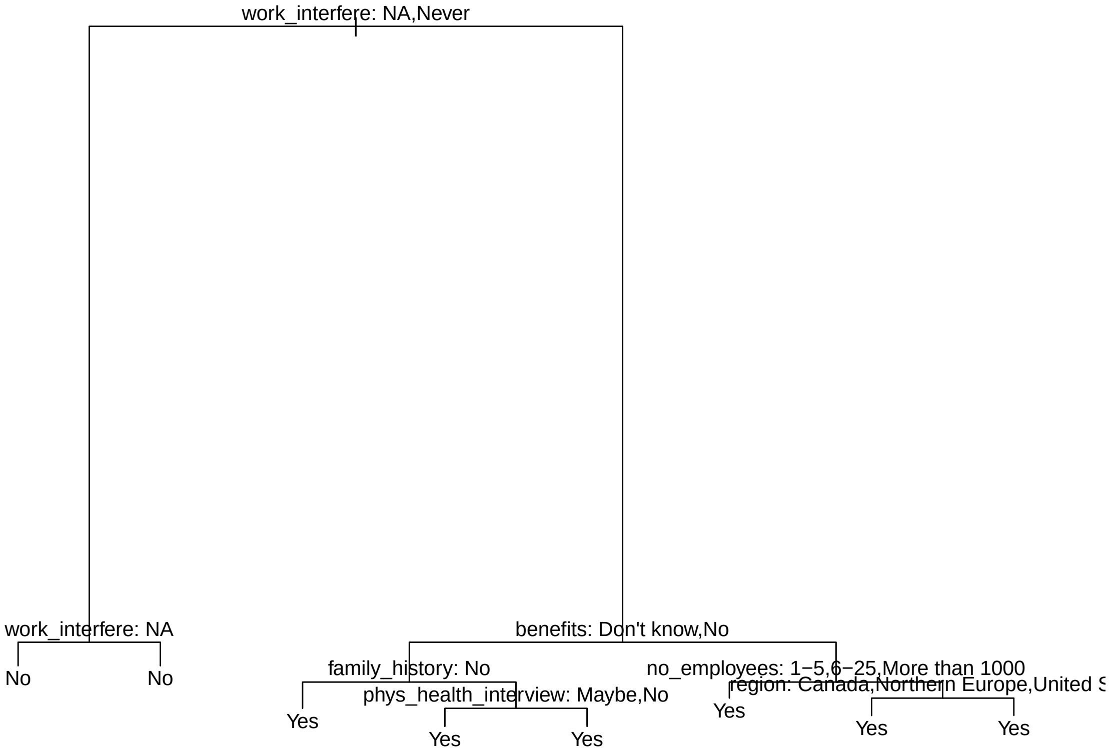
Classification tree

```
set.seed(149113)
tr.tree = tree(treatment ~ ., data = train)
tr.pred = predict(tr.tree, test, type = "class")
accuracy.tree = mean(tr.pred == test$treatment)
print(paste0("Accuracy of the classification tree: ",accuracy.tree))
```

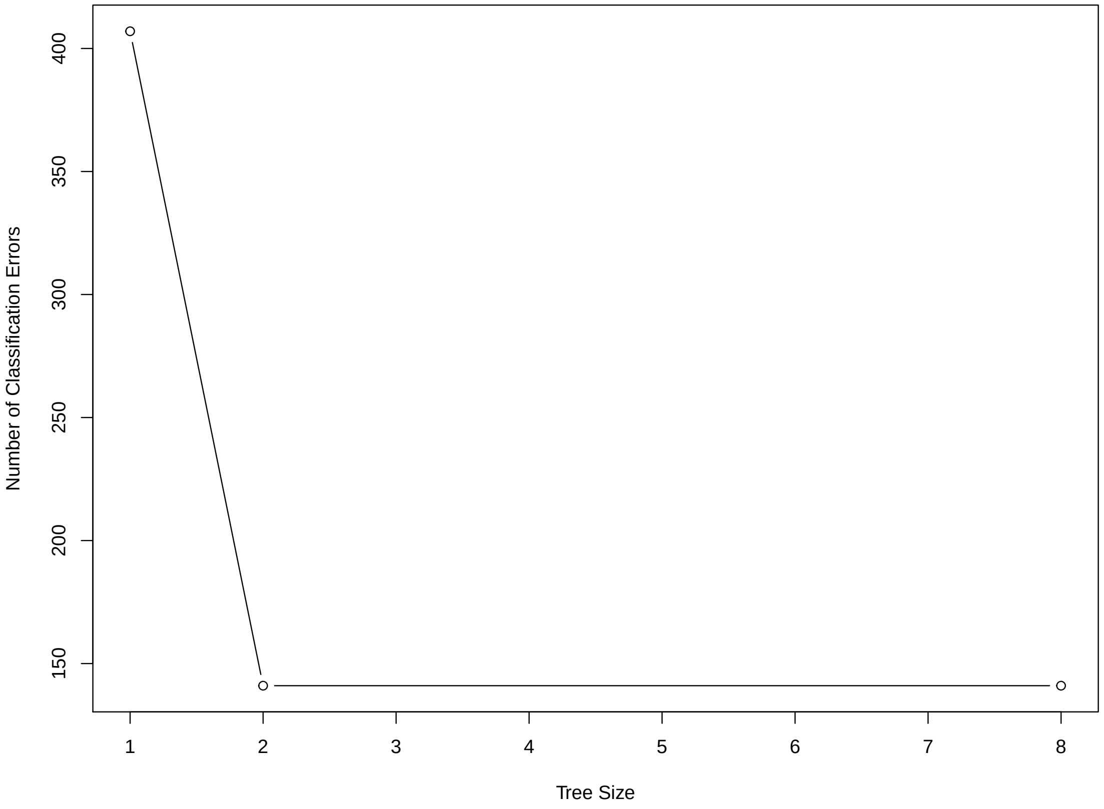
```
## [1] "Accuracy of the classification tree: 0.801955990220049"
```

We get an accuracy of 80.2% for the classification tree. If we plot the tree and plot the number of splits against the classification performance, we see that we reach this classification performance after already 2 splits on the basis of `work_interfere` and `benefits`.

```
plot(tr.tree)
text(tr.tree, pretty = 0)
```


work_interfere: NA,Never
work_interfere: NA
No
No
benefits: Don't know,No
family_history: No
Yes
phys_health_interview: Maybe,No
Yes
Yes
no_employees: 1−5,6−25,More than 1000
region: Canada,Northern Europe,United S
Yes
Yes
Yes

```
cv.tr = cv.tree(tr.tree, FUN = prune.misclass, K=20)
plot(cv.tr$size, cv.tr$dev, type = "b", xlab = "Tree Size", ylab = "Number of Classification Errors")
```



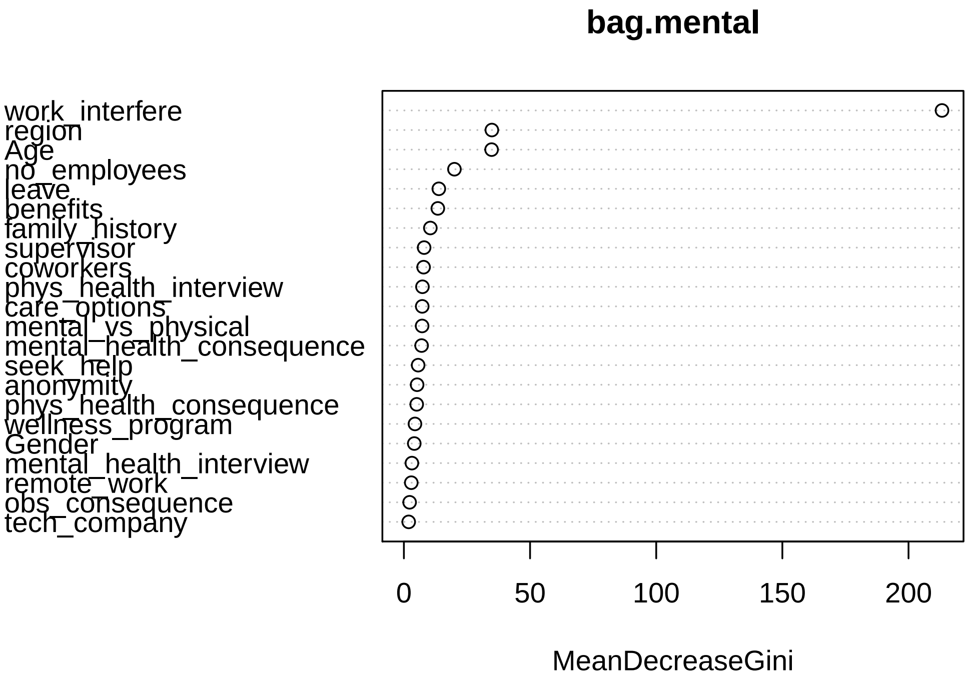
Bagging

Bagging can be done by training a random forest with mtry set to the number of parameters in the training set. Note that train in our case too contains the treatment variable, so we should set it to the number of columns minus 1. Evaluating the accuracy of the trained classifier on the test set shows that we do marginally better than the classification tree with an accuracy of 80.9%. Inspecting the variable importance, we see that again `work_interference` is the most important variable. `benefits` is also important but a few other variables also come into play. These include `region`, which indicates that it was a good decision not to discard geographic information altogether.

```
set.seed (12421)
bag.mental = randomForest(treatment ~ .,data=train, mtry = ncol(train-1))
yhat.bag = predict(bag.mental, newdata=test)
accuracy.bag = mean(yhat.bag==test$treatment)
print(paste0("Accuracy of the trained bagging classifier: ",accuracy.bag))
```

```
## [1] "Accuracy of the trained bagging classifier: 0.809290953545232"
```

```
varImpPlot(bag.mental)
```



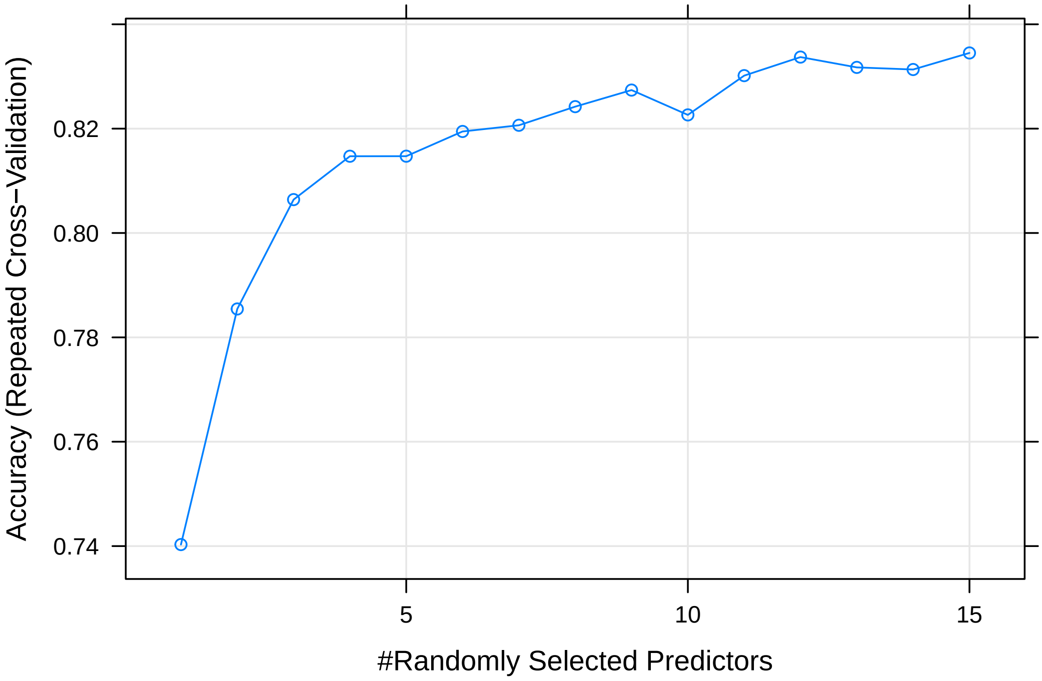
Random forest

The caret package includes methods to select the hyperparameter mtry through cross-validation. Note that this procedure is quite intense. Still, let us see if we can get better performance in this way.

```
library(caret)
set.seed (12421)
control <- trainControl(method="repeatedcv", number=5, repeats=3, search="grid")
set.seed(137182)
tunegrid <- expand.grid(.mtry=c(1:15))
rf_gridsearch <- train(treatment~., data=train, method="rf", metric="Accuracy", tuneGrid=tunegrid, trCo
print(rf_gridsearch)
```

```
## Random Forest
##
## 842 samples
##  22 predictors
##   2 classes: 'No', 'Yes'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 3 times)
## Summary of sample sizes: 674, 674, 673, 674, 673, 673, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    1    0.7402907  0.4795389
##    2    0.7854419  0.5694580
##    3    0.8064009  0.6111215
##    4    0.8147107  0.6276280
##    5    0.8147201  0.6276281
##    6    0.8194609  0.6369637
##    7    0.8206443  0.6393133
##    8    0.8242157  0.6465766
##    9    0.8273903  0.6528918
##   10    0.8226355  0.6433160
##   11    0.8301611  0.6584270
##   12    0.8337114  0.6655543
##   13    0.8317390  0.6615649
##   14    0.8313328  0.6607722
##   15    0.8345050  0.6672345
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 15.
```

```
plot(rf_gridsearch)
```



The optimal final value found by 10-fold cross-validation is `mtry` = 15. Working with this parameter value gives an MSE of 81.2% on the test set. If we do not specify `mtry` we get a slightly higher accuracy at 81.9%. Hence, let us work with the default parameters even though tuning with caret points towards optimality of `mtry` 15. Plotting the OOB error against the number of trees shows that we have reached the plateau on which performance can no longer be increased. The variable importance plot points towards importance of mostly the same variables as in bagging.

```
set.seed(12121)
rf.model <- randomForest(treatment ~ ., data=train, mtry=15, ntrees = 1000)
set.seed(12121)
rf.model.default <- randomForest(treatment ~ ., data=train, ntrees=1000)
rf.predict <- predict(rf.model, test)
rf.predict.default <- predict(rf.model.default, test)
```

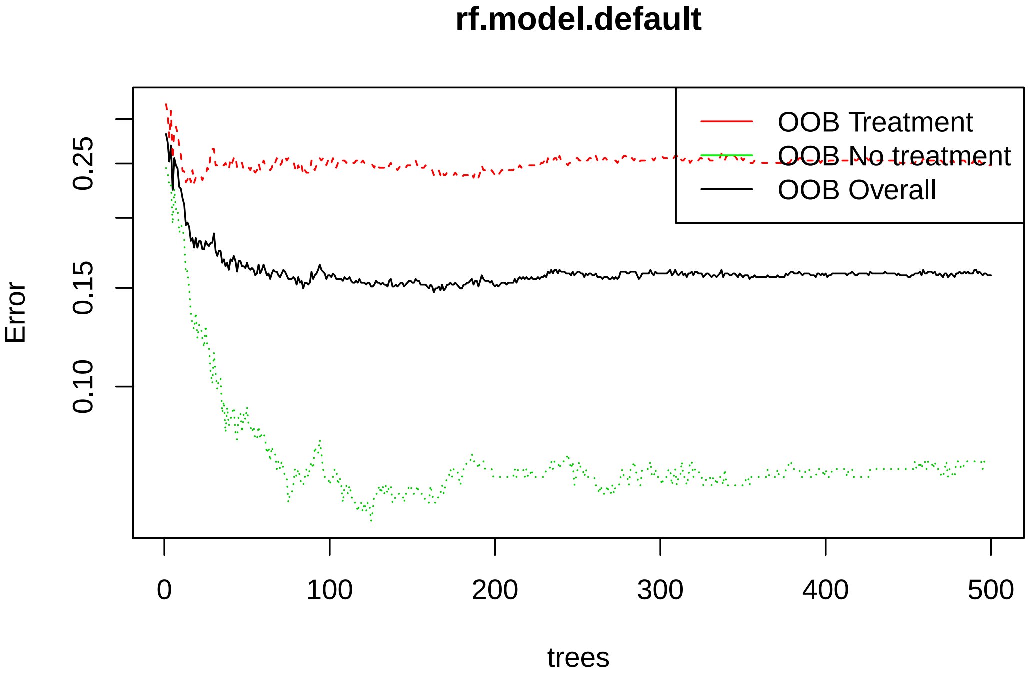
```
accuracy.randomforest <- mean(rf.predict==test$treatment)
accuracy.randomforest.default <- mean(rf.predict.default == test$treatment)
print(paste0("Accuracy random forest mtry = 15: ",accuracy.randomforest))
```

```
## [1] "Accuracy random forest mtry = 15: 0.814180929095355"
```

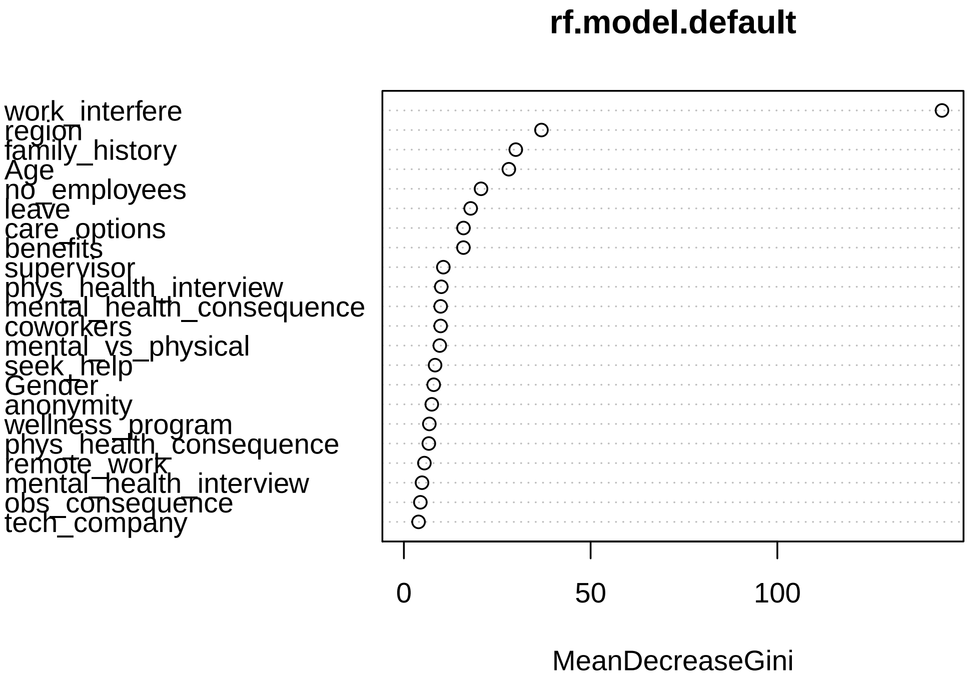
```
print(paste0("Accuracy random forest default mtry: ",accuracy.randomforest.default))
```

```
## [1] "Accuracy random forest default mtry: 0.819070904645477"
```

```
plot(rf.model.default, log="y")
legend('topright', c("OOB Treatment","OOB No treatment","OOB Overall"),lty=1,col=c('red','green','black
```



```
varImpPlot(rf.model.default)
```



Logistic regression

Logistic regression reaches an accuracy of 80.9% and thereby the same accuracy as the tree. The very significant coefficients include family history, work interference and benefits. Our previous results also have pointed towards importance of these variables.

```
glm.fit = glm(treatment ~ ., data = train, family = binomial)
summary(glm.fit)
```

```
##
## Call:
## glm(formula = treatment ~ ., family = binomial, data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.7166  -0.2952   0.1328   0.5357   3.1523
##
## Coefficients:
##                                Estimate Std. Error z value Pr(>|z|)
## (Intercept)                    -6.79004    1.41615  -4.795 1.63e-06 ***
## Age                             0.04071    0.01730   2.353  0.01862 *
## GenderLBGTQ                    -0.93019    0.82018  -1.134  0.25674
## GenderM                        -0.69874    0.31953  -2.187  0.02876 *
## family_historyYes               1.27827    0.24108   5.302 1.14e-07 ***
## work_interfereNever             2.06965    0.66417   3.116  0.00183 **
## work_interfereRarely            4.96090    0.66990   7.405 1.31e-13 ***
## work_interfereSometimes         5.43511    0.63786   8.521  < 2e-16 ***
## work_interfereOften             6.54833    0.74082   8.839  < 2e-16 ***
## no_employees6-25                0.38933    0.40755   0.955  0.33943
## no_employees26-100            0.59096    0.43164   1.369  0.17097
## no_employees100-500           0.64703    0.49604   1.304  0.19210
## no_employees500-1000          0.45081    0.68444   0.659  0.51011
## no_employeesMore than 1000    0.25158    0.48002   0.524  0.60021
## remote_workYes                0.13249    0.27461   0.482  0.62948
## tech_companyYes               0.32685    0.31889   1.025  0.30538
## benefitsNo                    0.04475    0.35553   0.126  0.89985
## benefitsYes                   1.08232    0.36844   2.938  0.00331 **
## care_optionsNo               -0.01550    0.31650  -0.049  0.96095
## care_optionsYes               0.57616    0.37067   1.554  0.12009
## wellness_programNo            0.14506    0.40286   0.360  0.71879
## wellness_programYes          -0.29735    0.48067  -0.619  0.53617
## seek_helpNo                  -0.66806    0.34230  -1.952  0.05098 .
## seek_helpYes                 -0.52395    0.42409  -1.235  0.21666
## anonymityNo                   0.38575    0.52402   0.736  0.46165
## anonymityYes                  0.56839    0.30791   1.846  0.06490 .
## leaveSomewhat easy           -0.62389    0.40527  -1.539  0.12369
## leaveDon't know              -0.27477    0.38328  -0.717  0.47344
## leaveSomewhat difficult      -0.06151    0.49476  -0.124  0.90105
## leaveVery difficult           0.22956    0.56973   0.403  0.68701
## mental_health_consequenceNo   0.34862    0.32798   1.063  0.28782
## mental_health_consequenceYes  0.10436    0.33487   0.312  0.75530
## phys_health_consequenceNo    -0.08680    0.29815  -0.291  0.77096
## phys_health_consequenceYes   -0.41255    0.60923  -0.677  0.49830
## coworkersSome of them         0.30316    0.32540   0.932  0.35151
## coworkersYes                  1.12794    0.46573   2.422  0.01544 *
## supervisor1                   0.15571    0.32982   0.472  0.63686
## supervisor2                  -0.45939    0.37225  -1.234  0.21717
## mental_health_interviewNo     0.32551    0.38841   0.838  0.40200
## mental_health_interviewYes    0.33908    0.76556   0.443  0.65783
## phys_health_interviewNo       0.09256    0.27828   0.333  0.73942
## phys_health_interviewYes      0.93165    0.40095   2.324  0.02015 *
## mental_vs_physicalNo         -0.02640    0.30028  -0.088  0.92993
## mental_vs_physicalYes        -0.22953    0.32210  -0.713  0.47609
## obs_consequenceYes            0.13447    0.35715   0.376  0.70655
## regionCanada                 -0.82039    0.90316  -0.908  0.36369
## regionEastern Europe         -0.78861    1.28253  -0.615  0.53863
## regionNorth Central (USA)    -0.31721    0.83996  -0.378  0.70569
## regionNortheast (USA)        -0.60930    0.87875  -0.693  0.48808
## regionNorthern Europe         0.53000    0.82201   0.645  0.51908
## regionOther                  -0.99545    1.02670  -0.970  0.33227
## regionSouthern Asia           2.48105    1.67762   1.479  0.13916
## regionSouthern Europe        -0.50284    1.39292  -0.361  0.71810
## regionSouth (USA)            -0.31881    0.85249  -0.374  0.70842
## regionUnited States          -0.74959    1.49057  -0.503  0.61504
## regionWestern Europe          0.01879    0.89423   0.021  0.98324
## regionWest (USA)              0.22171    0.84260   0.263  0.79245
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1166.33  on 841  degrees of freedom
## Residual deviance:  545.79  on 785  degrees of freedom
## AIC: 659.79
##
## Number of Fisher Scoring iterations: 6
```

```
glm.predict <- ifelse(predict.glm(glm.fit, newdata=test,type="response")>0.5, "Yes", "No")
accuracy.logisticregression = mean(glm.predict==test$treatment)
print(paste0("Accuracy of logistic regression with glm: ",accuracy.logisticregression))
```

```
## [1] "Accuracy of logistic regression with glm: 0.809290953545232"
```

Discussion of results

Ranking the methods based on accuracy, we obtain (i) random forests, (ii) bagging, (iii) logistic regression and the classification tree. Note, however, that all methods are close to each other in terms of the accuracy. We conclude therefore that random forests for this data set has little added value over a simple logistic regression tree. One reason why we don't see large differences in classification performance might be that `work_interference` by itself correlates already quite strongly with treatment. This is also reflected by the other methods: the classification tree splits directly based on `work_interference`, random forests and bagging both have the highest decrease in the mean gini coefficient for this variable, and logistic regression's coefficient estimates have lowest p-values for the dummy variables based on `work_interference`.

We can finally check whether ensemble prediction performs better. The accuracy of the most simple ensemble formed by joining the different classifiers (not tree) is calculated below. Note that the ensemble performs a bit worse than bagging and has the same accuracy as bagging. Model averaging here thus does not improve classification performance very significantly. This may indicate that the methods classify correctly the same subjects.

With regards to variable importance, we have seen that `work_interference` is by and large the most important variable. We expand on this a little in the appendix. Other variables reported to be important by many methods include `region`, `age`, `care_options` and `benefits`. Interestingly, the correlation matrix showed for most of these that they correlated to some extent with `treatment`.

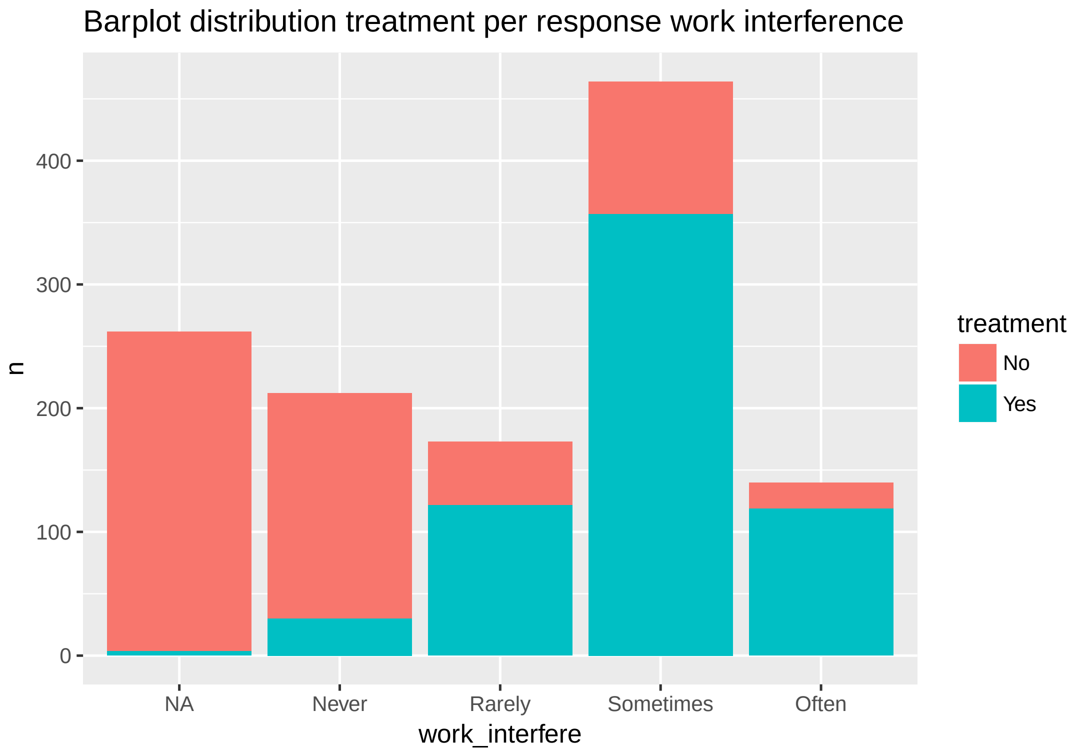
```
glm.predictnum = ifelse(glm.predict == "Yes", 1, 0)
majorityvote <- ifelse(as.numeric(rf.predict)-1 + as.numeric(rf.predict)-1 + glm.predictnum > 1.5, 1, 0
accuracy.ensemble <- mean(majorityvote == ifelse(test$treatment == "Yes", 1, 0))
```

Appendix - Short discussion of variable `work_interference`.

I initially thought it was odd for the coefficients on `work_interference`s dummies all positive, indicating that ceteris paribus the log-odds of having treatment increase with having responded to this question per se, rather than having a mental illness interfere with one's work. Looking at how the corresponding question is phrased, it turns out respondents are asked to answer this question only if they have a mental illness. By not responding one thus signals that one does not have a mental illness. This also becomes clear if we plot it (see below) and explains why the variable is so important; we cannot expect people to seek mental illness who do not have a mental illness in the first place.

The plot also quite clearly shows why work_interference is so important; if a respondent responds "Never" or does not fill in the question, we can be quite certain that this person has not had treatment for mental illness.

```
df %>% subset(select=c('treatment','work_interfere')) %>% group_by(work_interfere, treatment) %>% tally
  ggplot(aes(x=work_interfere, y=n, fill = treatment)) +
  geom_bar(stat="identity") +
  ggtitle("Barplot distribution treatment per response work interference")
```


Barplot distribution treatment per response work interference
n
400
300
200
100
0
NA
Never
Rarely
Sometimes
Often
work_interfere
treatment
No
Yes