Machine Learning for Social Science - Lab 2

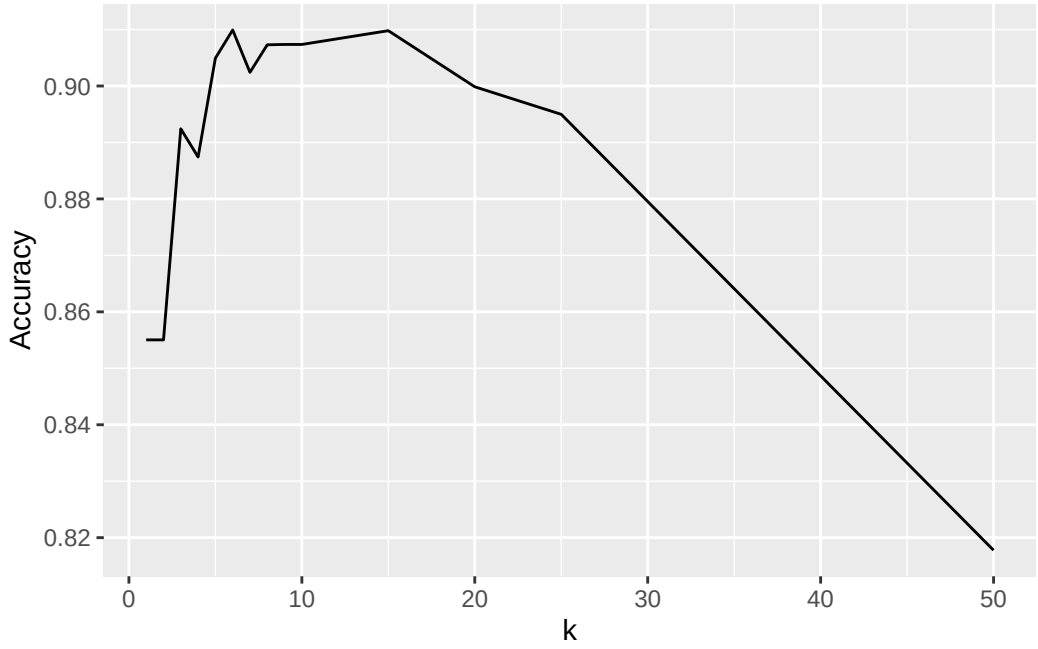
Marc Sparhuber

Table of contents

Part 1: Social Network Ad Purchase (cont.)

Task 2

The first non-parametric method you will use to predict ad purchase is the k-nearest neighbors algorithm. As we talked about in the lecture, it has one key parameter, k, that the researcher must set. Use the caret package to implement a 10-fold cross-validation procedure that examines the test accuracy of kNN under different choices of k (hint 1: set method="knn" to estimate a kNN with caret| hint 2: use the tuneGrid argument to specify your grid of k values). Consider the following values of k: {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20, 25, 50}. Because kNN is sensitive to the scale of variables, we must standardize our variables prior to estimation (hint: you can add the following argument to train(): preProcess=c("center","scale"), to do so). From the results, plot the test accuracy against k (hint: results can be extracted from the train object by using $results). Interpret this plot. How does the performance—for the best k—compare to that of GAM and standard linear regression (i.e., your results in lab 1)? Which do you prefer? Why?

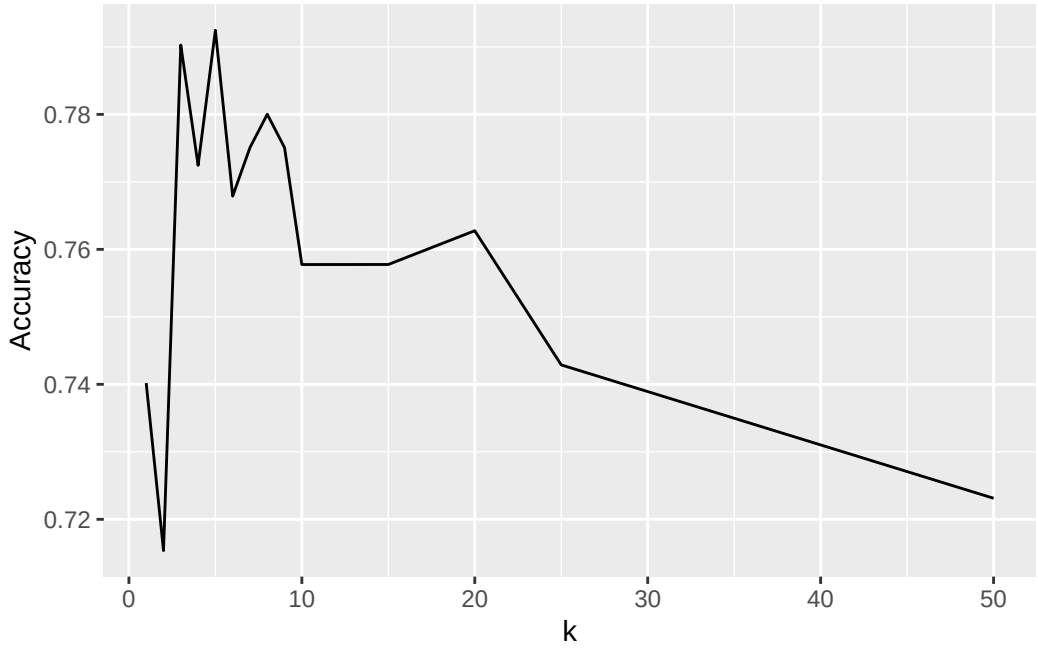


After a steep increase til about k = 6, accuracy stagnates and drops of after k = 15, gradually declining after that. This indicates that test accuracy falls off strongly after a certain point, using KNN. Inspecting the result with the highest accuracy, it appears that k = 6, at around ~0.91, results in the best test accuracy. With this result it performs better than the standard linear regression model and roughly as good as the best of the GAM models (df = 2). If I had to choose one of the three to

use, I would stick with the GAM, as it does not suffer from the same drawbacks as kNN in regards to a lack of interpretability, which is something GAMs are actually quite good for because the effect of each predictor on the outcome can be measured separately, if needed.

Task 4

Repeat task #2 for this new data set (from step #3). How does the prediction accuracy change? How do you explain this difference (or lack thereof)?



The test accuracy falls to ~0.79 at k = 5. As can be seen in the plot, the accuracy "line" is more erratic with just two "peaks" of accuracy. This is due to increase in (irrelevant) variables in the new data set. KNN's performance generally decreases with an increase in variables to consider, especially when compared to parametric approaches. For KNN specifically this problem comes from the curse of dimensionality, which is apparent here as the nearest neighbor to any nearby test observation may be very far away in the higher (p-)dimensional space.

Task 5

Motivated by the results obtained in #4, we want to explore an alternative method. Based on the properties—which we discussed in the lecture—of decision trees, do you think it has the

potential to perform better than kNN on the data set from #3? If yes, why?

> I expect slightly better results, as decision trees avoid some of the weaknesses of KNN. This is due to the fact how they partition the data, which is by means of recursive binary splitting. This procedure seeks to divide the data into regions with the goal of the greatest possible error reduction.

Task 6

Estimating decision trees is what you will do now. In R, we can use the package rpart for this. Decision trees have two main parameters that a researcher must choose prior to estimation: (i) the minimum number of observation in each leaf node (in rpart: minbucket), and (ii) the complexity parameter (in rpart: cp). In practice, the former is often set heuristically (using some rule-of-thumb) and the latter using cross-validation. Please do the following:

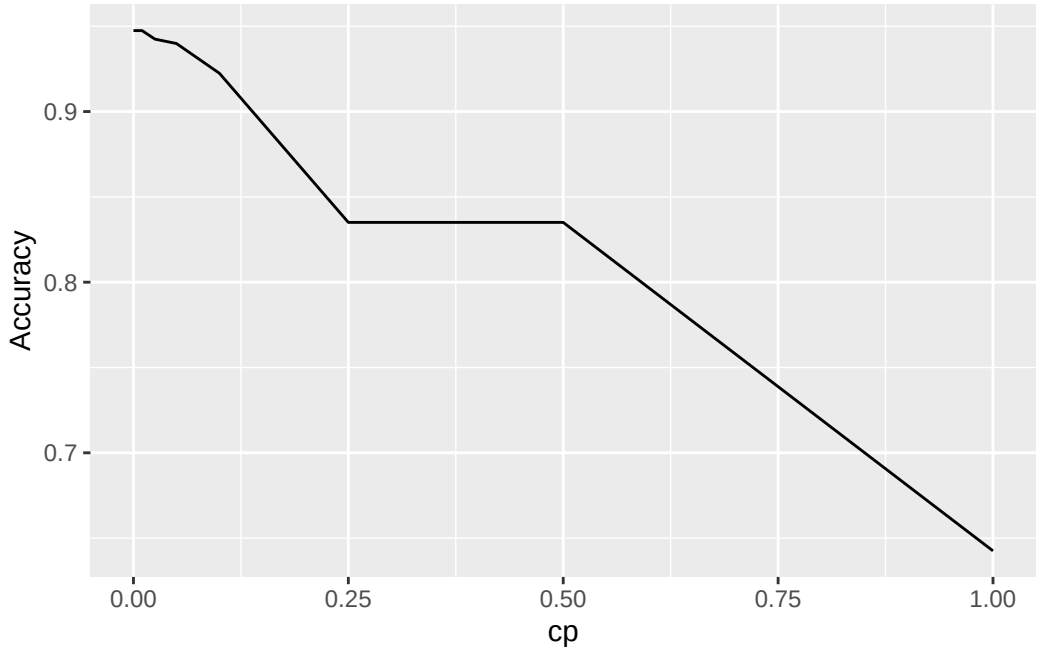
a.

Answer why it generally is not a good idea to set minbucket to either a very large number or a very small number (relative to the size of your data).

> Too large a minbucket parameter would lead to the potential oversight of further splits that could be made, whereas too small a minbucket might lead to trees that grow too large without any additional meaning being captured during the splits toward the leaf-ends of the tree.

b.

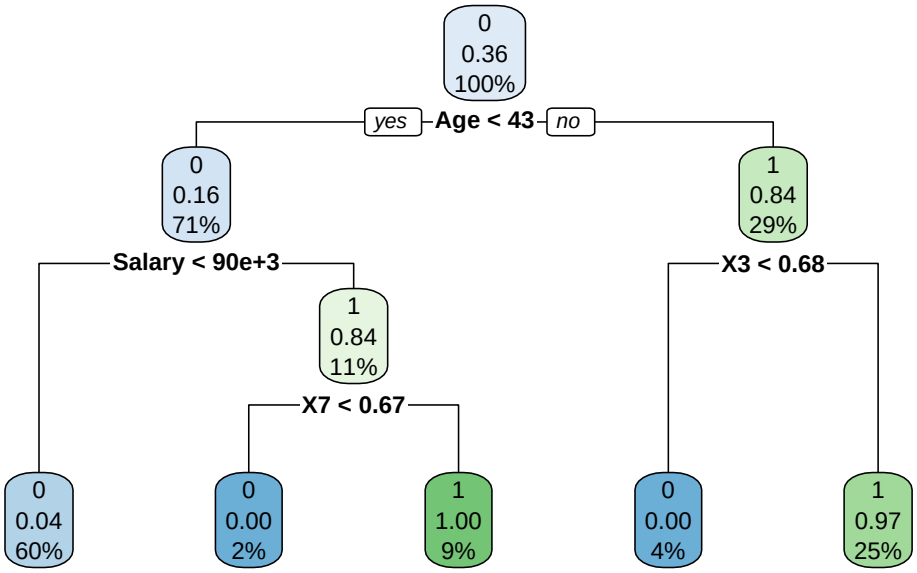
Use the caret package (method="rpart") to implement a 10-fold cross-validation procedure to assess how the test accuracy changes as a function of the complexity parameter (ISL: ; rpart: cp). Hint: use the tuneGrid argument to specify your grid of cp values. Consider the following values for cp: {0, 0.01, 0.025, 0.05, 0.10, 0.25, 0.5, 1.0}. (note: minbucket is here set to a default value=10). Plot the test accuracy against cp (hint: extract results as you did in #2 and #3). Interpret this plot. How does the accuracy for the best cp compare to kNN? What do you attribute this difference to?



Identical test accuracies of ~0.95 occur at cp = 0 and = 0.01. With an increasing complexity parameter they continually decrease. As such, the best test accuracies of this approach outperform those of the KNN model used earlier by about ~0.04. This is because the tuning parameter controls the pruning of the tree, which avoids overfitting by selecting the best subtrees, which is likely the reason why KNN is performing badly with the higher-dimensional dataset. The fact that the best test accuracy also occurs at cp = 0 shows, however, that pruning wasn't actually necessary to achieve the best possible performance.

c.

Use the rpart.plot package to plot the decision tree you found to be the best in b (hint: you can extract the model with the highest accuracy from a caret model using $finalModel). First report how many internal and terminal nodes this tree has. Then, provide a interpretation. Does any of the "new" variables (X1, . . . , X20) show up in the tree?



The generates tree has 4 internal nodes (if you count the root node) and 5 terminal nodes. The root node splits the data according to wether an individual is of age 43 or younger. Obviously, all data go through this split and here the probability for a negative purchase decision is around ~0.36. The 71% for whom age is below 43 are further split by their salary level and the 11% that earn more than 90000 are then further split by the additional variable X7, most of whom make a positive purchasing decision. The individuals earning less predominantly decide not to purchase. For the 29% who are above 43 years old, there is a .84 probability of a purchase. If their X3 is >= .68, a positive purchase decision is even more probable.

d.

Use caret's varImp() function to calculate variable importance scores. Interpret.

X3 and X7 seem to be of relatively high importance, while Gender and many of the additional variables hold very little importance. This lines up with the tree plot.

Part 2: Bernie Sanders and Donald Trump tweets (cont.)

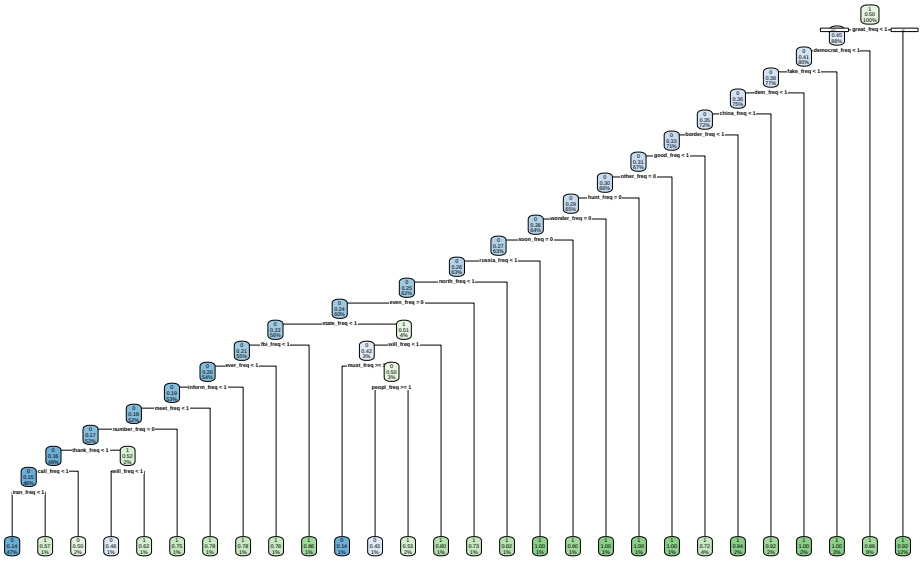
Task 2

Use caret to implement a 5-fold cross-validation procedure that estimates the test error for a decision tree with different cp (ISL:) values. Consider the following cp values: {0, 0.001, 0.01, 0.1, 0.25}. Report the accuracy of the best configuration. How does this compare to the performance of a standard logistic regression (LR) and a ridge regression (RR)? You do not need to estimate the LR and the RR yourself— I provide their performance here: Accuracy(LR)=55%. Accuracy(RR)=86%. Why do think this is the case? Doing the following two things might help you answer this question:

> The accuracy of the best configuration (0 and 0.001) is ~0.78. This is better than the LR but worse than the RR approach. This is, on one hand, likely due to the high-dimensionality of this data set, where p is approaching n (the data set in the previous lab even had n < p). I think that RR performed better as it is a complexity reducing approach which reduces irrelvant coefficients to near 0. This is essential to a getting a good predicition in this case, as can be seen in Task 2b: only 45 of the 780 variables actually carry non-zero values in a variable importance test.

a.

Plot the tree which achieved the highest test accuracy (hint: again, you can extract the best model using $finalModel, and plot it using rpart.plot). Is it a big or a small tree?



Big tree.

b. Calculate variable importance. How many variables have non-zero values?

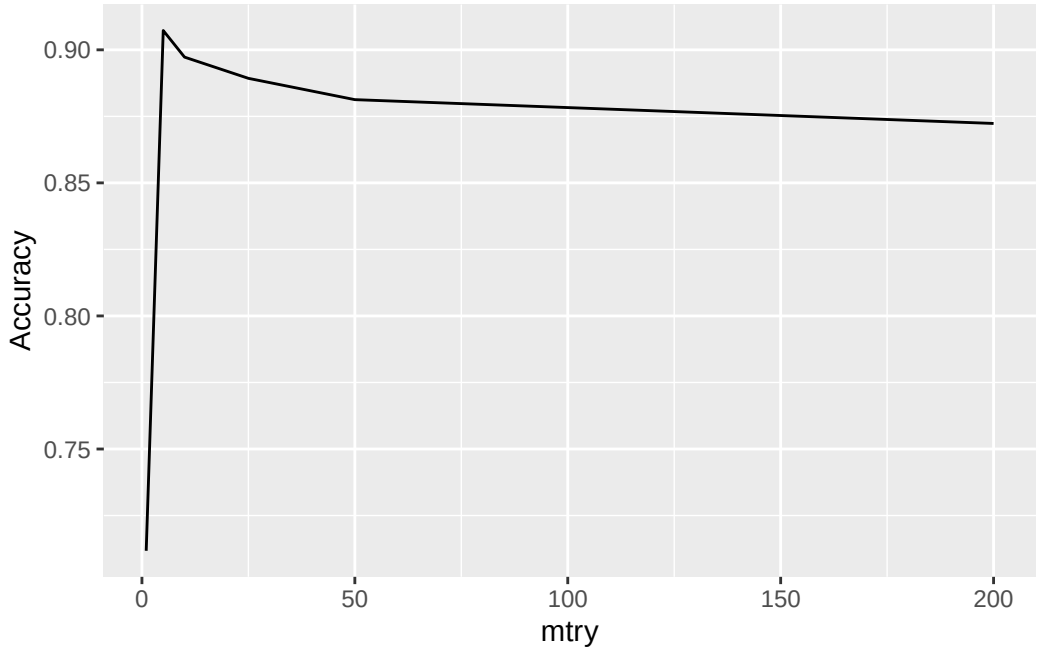
```
[1] 45
```

Task 3

Given the results in #2, we want to explore other alternative methods. Using once again the caret package, you shall now implement a 5-fold cross-validation procedure that fits a random forest model (method="rf") with different mtry values. mtry controls the number of variables that are considered at each split (corresponding to parameter m in ISL). Specifically, you shall consider the following values for mtry: {1, 5, 10, 25, 50, 200} (hint: again, you specify this grid using the argument tuneGrid). Another important parameter of random forest models is the number of trees to use. Here you shall use 200 (hint: you may enter ntree=200 as an argument in train() to do so). The estimation may take a couple of minutes to finish. Once it has finished, please do the following:

a.

Plot accuracy against mtry. Interpret this plot. Does decorrelating the trees seem to help performance?



Yes, the decorrelation of trees has helped, as the best accuracy achieved is higher than with a "normal" decision tree. The Accuracy first rises to above 0.90 and then gradually decreases to just below 0.875 at mtry = 200.

b.

Relate the performance of the best random forest model to the best decision tree model. Do you find a meaningful improvement using random forest? Why do you think that is (or isn't)? Examining the variable importance scores may provide some clues; how many non-zero variables do you find for the random forest?

```
[1] 780
```

Yes, random forest improves the test accuracy quite a bit from ~0.78 with the decision tree model to ~0.91. I think that this is due to the decorrelation process which I suppose is vital with this data set as the words occuring in tweets are likely very correlated amongst one another in their use. Instead of considering all available variables at each split, random forest only allows for a subset of them to be considered (5 at the best model here), which makes the trees not look like

another because they would otherwise all look very alike due to the strongest predictors always being chosen for a split. This reduces the variability of results when averaging the results of the different trees together at the end of the algorithm. There are no non-zero variables for the random forest. I have to admit I don't know 100% why but I assume it's because RF randomly chose all of them at some iteration and took into account their contribution, later adding them to the averaging process, making none of them zero.

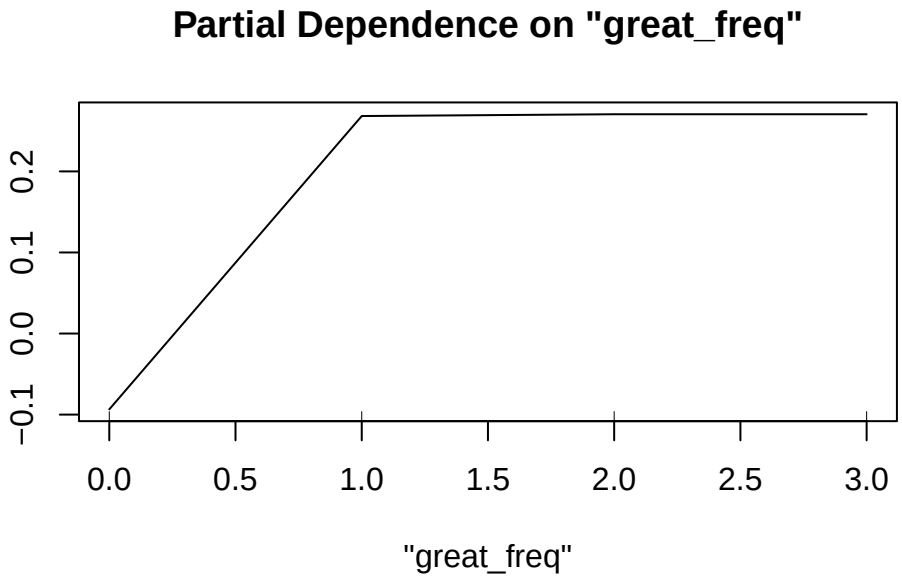
c.

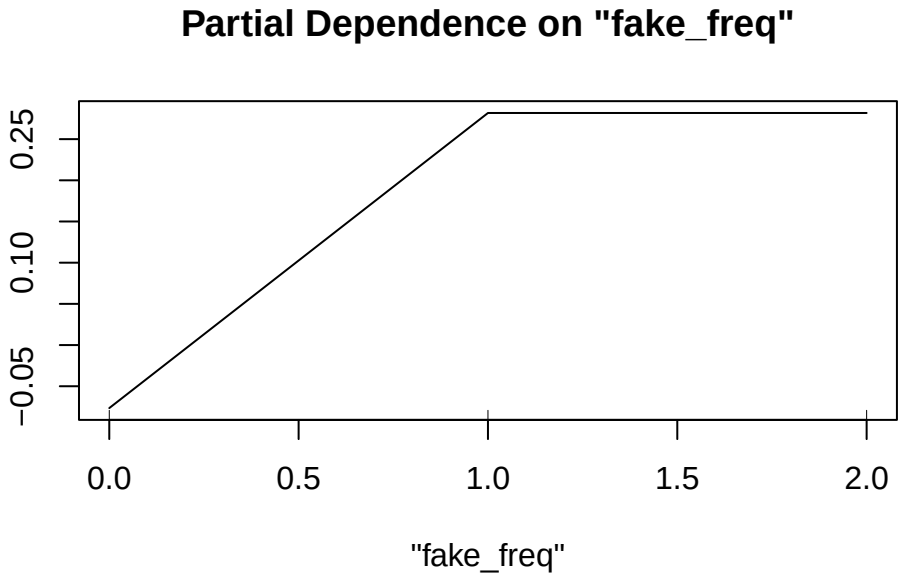
Relate the performance of the best random forest model and the ridge regression from #2. What do you think this difference indicates about the data, and more specifically the relationship between X and y?

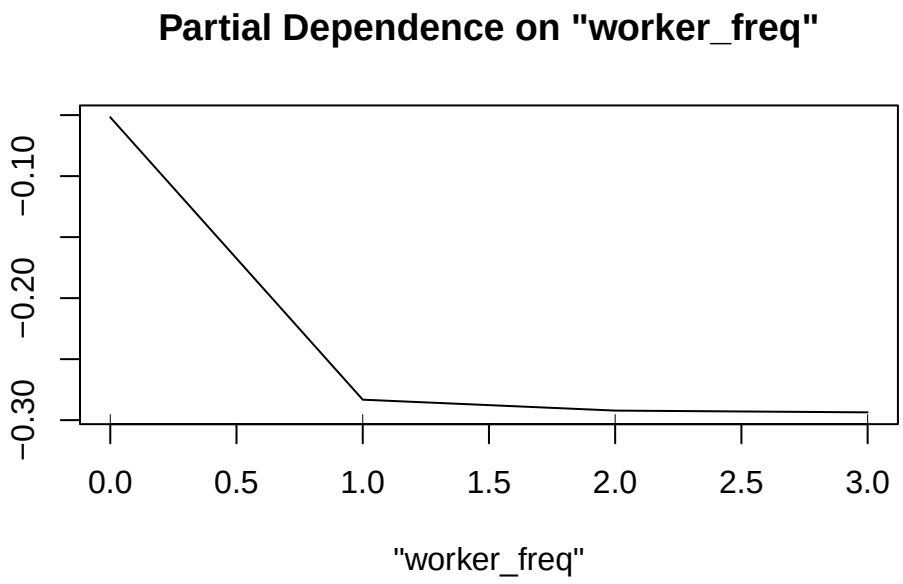
Of these two models, RF performs better. I think that on the one hand this shows the non-linearity and high-dimensionality, both of which ridge regression models can have difficulty dealing with. On the other hand, it seems that when predicting from the variables it is benefitial to include many of them in the model to arrive at the right prediction for y - something that ridge regression specifically avoids doing by reducing coefficients to near zero.

d.

Consider the variable importance scores in a bit more detail: does the variables that appear in the top 20 align with your expectations? Note that variable importance does not reveal the direction; e.g., whether a certain word is associated with Trump or with Bernie. For this, we can use the randomForest's function plotPartial() which produces a partial dependence plot for a variable of choice. Select three variables (words) which you find interesting, and create three separate partial dependency plots. In order to interpret the y-axis of the plot, use the argument which.class to specify which class ("0" or "1") should represent the positive class.


Partial Dependence on "great_freq"
−0.1
0.0
0.1
0.2
0.0
0.5
1.0
1.5
2.0
2.5
3.0
"great_freq"


Partial Dependence on "fake_freq"
−0.05
0.10
0.25
0.0
0.5
1.0
1.5
2.0
"fake_freq"



Looking at the undirected variable importance scores it shows words that are very typical for either Trump or Bernie to have used much more than the other. This aligns with my expectations. Picking three words from the top 20 it is apparent that words such as “great” or “fake” are associated much more with the language used in Trump’s tweets than Bernie’s and vice versa for a word like “worker”.