Practical Machine Learning Course Project

Josh Weisberg

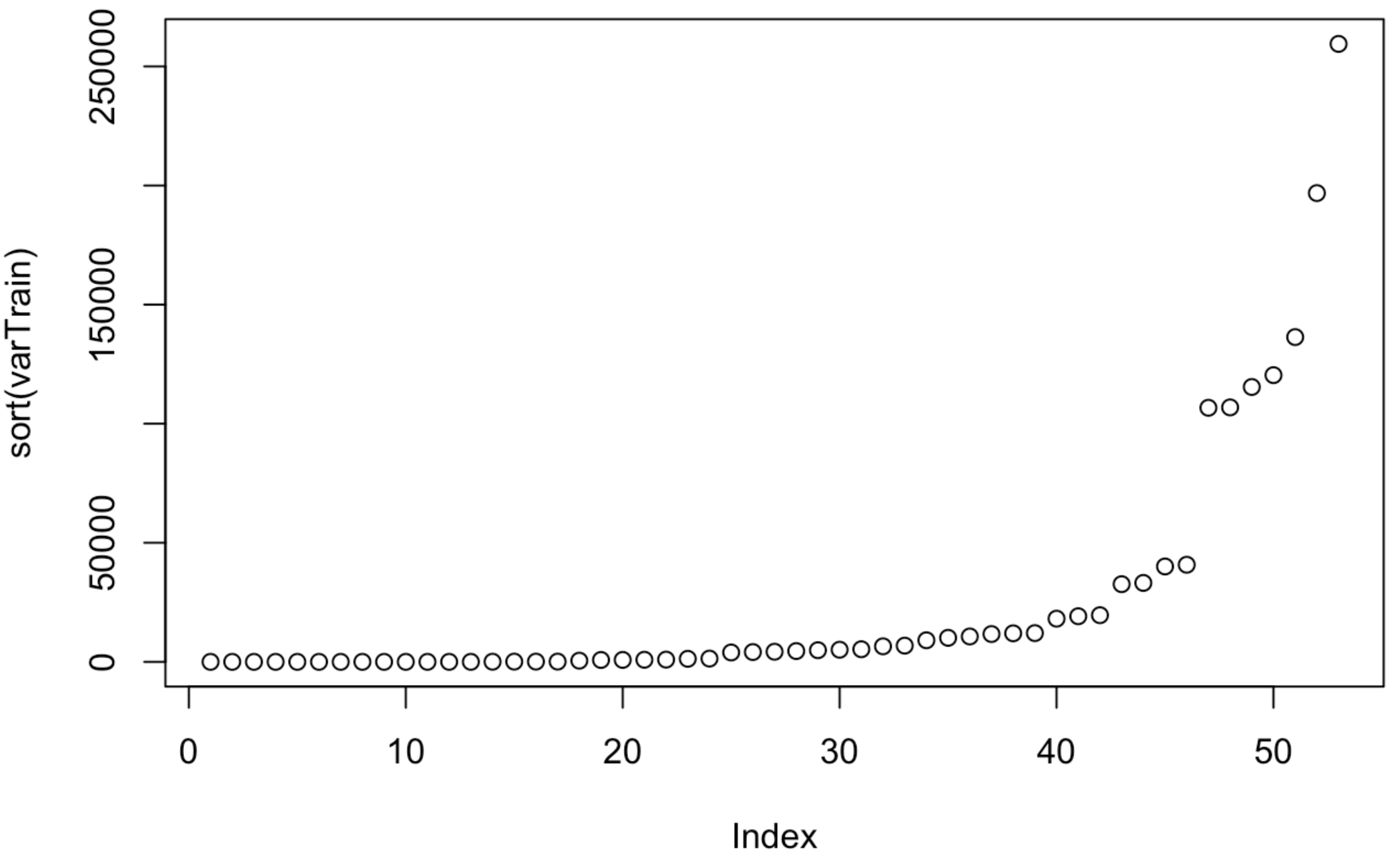
2/7/2021

##Get & Clean data First, I had to get and then clean the data. The dataset had many variables, but several of them were null. Thus, the first major cleaning step I took was to remove any variables with any NAs. There were some that had mostly NAs, but not all, but I removed them too because the vast majority of observations were NA. I made sure to remove any variables with NAs from both the test & training set, so the same predictors would be available for each.

Next, I removed the non-numeric or administrative columns which shouldn't be considered as predictors. For example, I noticed the data was sorted, and thus including the index (X) made the algorithm unrealistically accurate.

The final step of cleaning was to remove columns with low variance. I noticed, using the plot below, that the variance for the majority of the columns was about the same, but then there were ~25 columns with much higher variance, so I decided to only use those.

```
##Read in the data from these urls
urlTrain<- 'https://d396qusza40orc.cloudfront.net/predmachlearn/pml-
training.csv'
urlTest <- 'https://d396qusza40orc.cloudfront.net/predmachlearn/pml-
testing.csv'
Train<- read.csv(urlTrain)
Test<-read.csv(urlTest)

#Turn the dependent/class variable into a factor so that it can be c
lassified
Train$classe<-as.factor(Train$classe)


#Determine which columns have Null values in them, to remove the who
le column (since most that have any nulls, are nearly all null)
nullTrain<-apply(is.na(Train),2,sum)
nullTest<-apply(is.na(Test),2,sum)
nulls <- nullTrain + nullTest
Train<-subset(Train, select = (nulls == 0 ))
Test<-subset(Test, select = (nulls == 0 ))

#Remove non-numeric & administrative columns
Train<-subset(Train, select =  -c(X, user_name, raw_timestamp_part_1
, raw_timestamp_part_2, cvtd_timestamp, new_window, num_window ))
Test<-subset(Test, select =  -c(X, user_name, raw_timestamp_part_1,
raw_timestamp_part_2, cvtd_timestamp, new_window, num_window ))


Train2<-as.data.frame(lapply(Train, as.numeric))
Test2<-as.data.frame(lapply(Test, as.numeric))

Train2[is.na(Train2)]=0
Test2[is.na(Test2)]=0

#now remove columns with variance under 5000
varTrain<-apply(Train2,2, var )
Train2<-subset(Train2, select = varTrain>3000)
Test2<-subset(Test2, select = names(Train2))
```



Next, I got into predictions. I knew I would want to tune my model, so I split the train set into a training & cross-validation set, with 70% & 30% of the observations, respectively. As I ran through many iterations of different approaches, this came in handy so I could watch my accuracy improve.

The dataset is very large, including many variables. I tried a few different methods using the Caret package, but they all had extremely large runtimes, likely due to the dimensionality of the data. Thus, I opted to choose High-Dimensional Discriminate Analysis, which ran very quickly despite the large number of variables. I saw some success with this method, and then went back to preProcess the data, hoping to improve my results. Knowing that discriminate analysis assumes a gaussian distribution of each variable, with all variables having similar variance, I decided to center & scale the data before fitting the model.

```
#set the stage for the caret package
set.seed(233)
library(caret)
```

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

```
## Loading required package: ggplot2
```

```
#Remove 20% of the training set to be used as cross validation data
inTrain<-createDataPartition(Train$classe, p = .7, list = FALSE)

Train2$classe <-Train$classe

Training<-Train2[inTrain,]
CrossVal<-Train2[-inTrain,]



#center & scale it for discriminant analysis
preproc<- preProcess(Training, method = c("center", "scale"))

Training<-predict(preproc, Training)
Testing<-predict(preproc, Test2)
CrossVal<-predict(preproc, CrossVal)

#Use a high dimensional discriminant analysis
modFit<-train(classe ~ . , method = "hdda",  data = Training)
#hdda

#Run predictions on crossVal
preds<-predict(modFit, CrossVal)
cm<-confusionMatrix(data = preds, reference = CrossVal$classe)
cm$overall["Accuracy"]
```

```
##  Accuracy
## 0.7563297
```

With an accuracy on my cross validation set of .76, I was satisfied with my model & made my predictions. I did not use the cross-validation set to train my model, so it is an out-of-sample score, however, I did use it to refine my approach, so it's not a perfect experiment, and the out-of-sample accuracy rate could be a little lower.

```
#Now predict the test data
# predict(modFit, Testing) )
```