401 Test 3

12/6/2021

Introduction

Healthcare is a contemporary problem in modern America that primarily focuses on the politics of implementation. Often overlooked but equally important, we will be considering the level of satisfaction of patients within the California healthcare system. Our dataset has 276 patient observations with the variable of interest being *quality*. Alongside quality, we have 10 measured covariate that includes data encompassing things such as physical/health status and socioeconomic positioning.

This paper has three aims:

1) To build a model that can significantly predict quality from our covariate

2) Infer the causal effect of physician group on patient satisfaction

3) Represent the relationship of pg, sex, and quality using graphical modeling

EDA (Exploratory Data Analysis)

Let us start the analysis by familiarizing ourselves with our 11 variables. Note: There is no missing data in our set.

```
##  pg           age        sex     educ    insu    drug    severity
##  0:104   Min.   :19.00   0: 83   1:  1   1:184   0:  5   1: 42
##  1:172   1st Qu.:34.00   1:193   2:  4   2: 78   1:271   2: 65
##          Median :40.00           3: 15   3:  1           3:126
##          Mean   :39.96           4: 80   5: 13           4: 43
##          3rd Qu.:47.00           5:100
##          Max.   :55.00           6: 76
##       com             pcsd            mcs.sd      quality
##  Min.   :0.000   Min.   : 9.176   Min.   :13.46   0: 90
##  1st Qu.:1.000   1st Qu.:39.905   1st Qu.:43.79   1:186
##  Median :2.000   Median :47.587   Median :50.67
##  Mean   :2.174   Mean   :45.650   Mean   :48.59
##  3rd Qu.:3.000   3rd Qu.:53.697   3rd Qu.:56.04
##  Max.   :9.000   Max.   :65.338   Max.   :65.50
```

pg = Physician Group, educ = Education, insu = Insurance Status, drug = Drug Coverage Status, com = Comorbidity, pcsd = Physical Comorbidity Scale, msc.sd = Mental Comorbidity Scale, Quality; 1 if satisfied with treatment, 0 if other

pg
age
sex
educ
insu
drug
severity
com
pcsd
mcs.sd
quality
Corr: −0.049
Corr: −0.119*
Corr: 0.217***
Corr: −0.050
Corr: −0.121*
Corr: −0.406***
Corr: 0.331***
Corr: −0.021
Corr: −0.174**
Corr: −0.030
Corr: 0.128*
Corr: −0.002
Corr: −0.016
Corr: 0.236***
Corr: −0.067
Corr: 0.018
Corr: 0.133*
Corr: −0.183**
Corr: −0.016
Corr: −0.069
Corr: 0.066
Corr: 0.041
Corr: 0.029
Corr: 0.155**
Corr: 0.111.
Corr: −0.128*
Corr: 0.113.
Corr: −0.002
Corr: −0.098
Corr: −0.186**
Corr: 0.059
Corr: −0.126*
Corr: −0.094
Corr: 0.151*
Corr: 0.036
Corr: −0.076
Corr: −0.168**
Corr: 0.120*
Corr: −0.049
Corr: −0.208***
Corr: 0.032
Corr: 0.079
Corr: −0.027
Corr: 0.054
Corr: 0.094
Corr: −0.043
Corr: −0.130*
Corr: 0.077
Corr: −0.256***
Corr: −0.211***
Corr: 0.184**
Corr: −0.080
Corr: −0.087
Corr: −0.102.

Upon initial inspection, Comorbidity has the highest absolute correlation coefficient. Admittedly, it is not very high, at just 0.184. The marjority of the other variables have an absolute correlation coefficient that range in [.075, .14], similarly low to Comorbidity. As quality is a binomial variable, it is difficult to visually judge what types of relationships exist between the covariate and quality.

Looking at their distributions, each continuous variable appears to have a near normal distribution that would more accurately be considered some variation of a beta distribution. Discrete variables such as pg, sex, educ, insu, drug, and severity have mild to very imbalanced distributions which reflect that the models we use might be skewed to favor the larger classes.

Considering all of this, we will take a look at some transformations that will help to properly normalize our distributions and result in linear relationships with age. Furthermore, we will be taking a look at nonparametric models as well.

Modeling

Feature Engineering

Our original dataset was not transformed and all variables are imported as numeric in order to take full advantage of the below transformations as well as to accomodate the diversity of models we tested over.

In addition to our original raw dataset, we will be considering 3 additional transformed datasets in order to address the above mentioned issues. The first engineered dataset contains the original 10 feature set as well as 5 common transformations applied to each of the original features. Those transformations are defined as:
log[*FeatureName*] = log(rawFeature+1)
sqrt[*FeatureName*] = sqrt(rawFeature)
recip[*FeatureName*] = 1/(rawFeature+1)
sqr[*FeatureName*] = rawFeature^2
cubed[*FeatureName*] = rawFeature^3
and will result in a total of 60 features within this dataset.

The second engineered dataset contains the prior mentioned 60 features as well as every possible interaction term between the original 10 features. This adds an additional 10 choose 2 (= 45) interaction variables for a total of 105 features within this dataset.

The third engineered dataset contains the original 10 features as well as every possible interaction term for a total of 55 features. This dataset will exclusively be used for our nonparametric models below.

Model Creating and Testing

We will be reserving 80% of our observations to a training set with the remaining 20% to be used for final model testing. This partition was randomly generated and applied to each of the above 4 datasets such that they share the same sample observations amongst their training and testing sets.

Fifteen models are considered within our framework; three logistic regressiono models, three standard linear regression models, three cross-validated LASSO regression models, two SVM models, two LDA models, and two random forests. The logistic, linear and LASSO models are tested on each of our first three datasets mentioned above. Our nonparametric and classifier models do not benefit from transformations and thus will only be tested on the original dataset and the fourth dataset mentioned above. This will allow us to best analyze which model has the strongest prediction performance given our problem.

While we do use train-test split for all of these models, we additionally perform 10-fold cross-validation on our LASSO regressions. The final LASSO models are determined by the lambda.1se resulting parameter.

Diagnostics and Model Selection

Even as we treat every feature to be numerical, we will be comparing model performance primarily via classification accuracy. Since quality is a binomial variable, and some of our models produced non-binomial predictions (i.e. a real number in [0, 1]), model predictions were rounded to the nearest integer. The below table summarizes these results calculated over the training and testing sets for our models:

```
##              Num. Predictors Accuracy (Train) Accuracy (Test)
## logModel1                 10        0.7000000       0.6785714
## logModel2                 60        0.6363636       0.5892857
## logModel3                105        0.7818182       0.4821429
## linearModel1              10        0.6863636       0.6964286
## linearModel2              60        0.7272727       0.6607143
## linearModel3             105        0.8227273       0.5714286
## lassoModel1               10        0.6727273       0.6785714
## lassoModel2               60        0.6727273       0.6785714
## lassoModel3              105        0.6727273       0.6785714
## svmModel1                 10        0.6772727       0.6785714
## svmModel2                 55        0.7000000       0.7142857
## ldaModel1                 10        0.7000000       0.6607143
## ldaModel2                 55        0.7500000       0.5892857
## forestModel1              10        0.9954545       0.6785714
## forestModel2              55        1.0000000       0.6785714
```

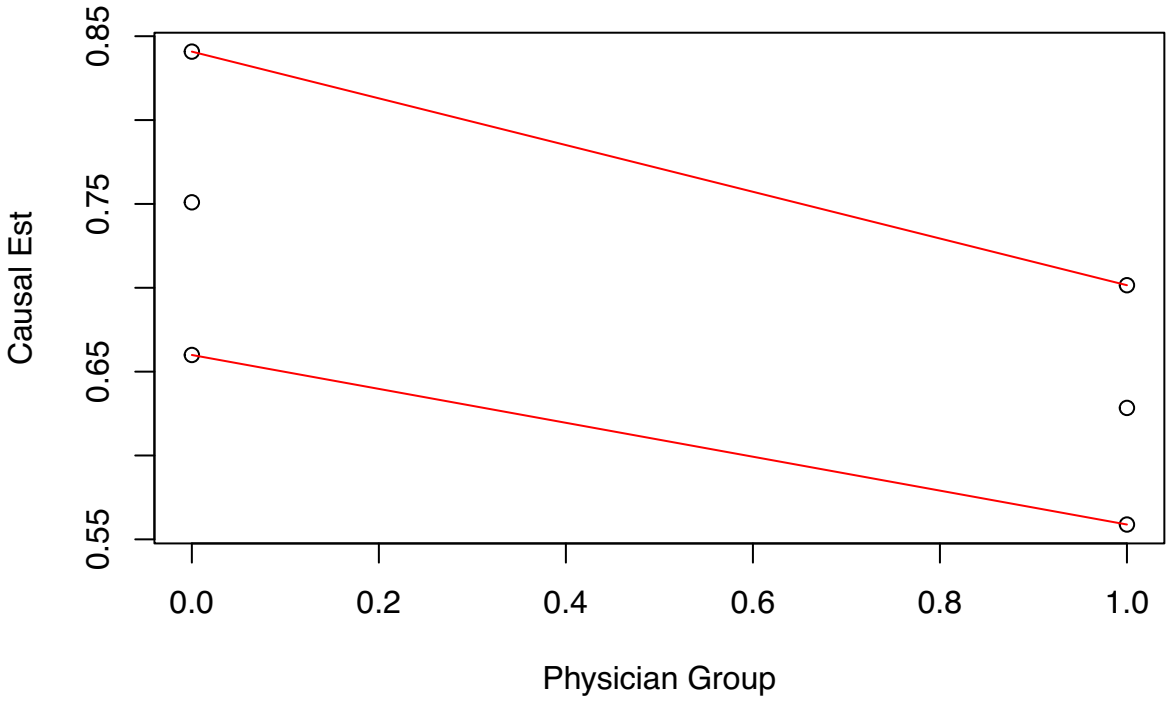
The above model performance is a mixed bag that tends towards the unimpressive. The logistic, linear, and lda models each saw decreasing test performance with larger datasets. The lasso, svm, and forest models tended to perform the best irrespective of dataset used, likely due to their lack of reliance on linearity. Interestingly, these three models achieve the exact same performance yet the lasso and svm models (excluding svmModel2) predict all 1 in the above test set while the RF models predicts a mixture of 1's and 0's. That is to say, the prediction accuracy of picking 1 in our testing dataset is 0.6785714. Only two of the above tested models surpassed this: linearModel1 and svmModel2, with the latter being the top performer at 0.7142857. This is a mere difference of ~.036 which is not significant and indicates the covariate within our dataset are not good predictors of quality.

I attempted removing some outliers and rebuilding and testing the models. Considering our original dataset, any observation that has at least one feature with a z-score greater than 4 was removed. 7 observations were removed via this method for a total of 269 observations. There was no noticeable improvement in the models.

Causal Inference:

Now let us consider our second question of the causal effect of physician group on patient satisfaction. Though there is not much predictive capability in this data as seen above, perhaps there is room for causal inference. To this end, we look to measure the causal effect of pg on quality through the use of a linear model as a plugin estimator. Specifically, the same techniques used to develop linearModel1 were used as it is a simple model and performs similar to the rest in the above results.

Two thousand iterations were performed to train the linear model on the entire dataset with the causal estimation being defined as the average model prediction on the entire dataset when pg is set to either 0 for the first thousand iterations or 1 for the second. The results can be seen plotted below alongside a 95% confidence band.



Though there is near no predictive power in our models, there appears to be room for causal inference. The causal estimate for pg=0 is .751 with a 95% confidence interval of [.662, .837]. The causal estimate for pg=1 is .627 with a 95% confidence interval of [.546, .704]. Thus it would appear that patients forced into the care of physician group 0 will have higher quality scores than if they were placed into physician group 1.

Graphical Modeling:

Now let us move onto our final question: understanding the graphical relationship of pg, sex, and quality.

```
## $call
## glm(formula = freq ~ pg * sex * quality, family = "poisson",
##     data = tbl)
##
## $coefficients
##                   Estimate Std. Error    z value     Pr(>|z|)
## (Intercept)     1.60943791  0.4472136  3.5988126 0.0003196734
## pg              1.64865863  0.4883252  3.3761487 0.0007350817
## sex             1.28093385  0.5055250  2.5338683 0.0112811117
## quality         1.33500107  0.5026247  2.6560595 0.0079059667
## pg:sex         -0.82545832  0.5642759 -1.4628629 0.1435049063
## pg:quality     -1.09659004  0.5669181 -1.9343006 0.0530761790
## sex:quality    -0.09823844  0.5694876 -0.1725032 0.8630419449
## pg:sex:quality  0.42292147  0.6567799  0.6439318 0.5196196679
```

This suggests the graph:

quality - pg - sex

In other words, quality is independent of sex conditional on physician group.

Final Conclusions

1) To build a model that can significantly predict quality from our covariate:

We were unsuccessful in finding a model that can predict quality based on the covariate in our dataset. Linear, logistic, and LASSO regression, LDA, SVM, and RF's all performed near equally poorly - achieving an accuracy similar to that of only predicting the dominant class of quality=1.

It is possible there is some predictive power within our features, but they are too nuanced for the binary decisions we are predicting. Furthermore, since quality is a subjective value based on patient satisfaction, I would advise attempting to collect more environment-oriented data such as lobby wait time, what actual physician treats the patient, or cost of treatment as these all may impact a person's mood more than education level.

2) Infer the causal effect of physician group on patient satisfaction:

We were successful in finding a difference in the causal effect of pg on quality. The causal estimate for pg=0 was higher than that of pg=1 and both estimates fell outside of their counterpart's 95% confidence interval.

3) Represent the relationship of pg, sex, and quality using graphical modeling:

We were mildly successful in finding a relationship between pg, sex, and quality. The graphical modeling results we found showed a tendency for a relationship: quality - pg - sex. However, the p-value for pg:sex and pg:quality are .144 and .053 respectively. This is larger than ideal and should thus serve as a starting point for determining a potential relationship. More advanced techniques should be applied to validate that this relationship truly exists.

Furthermore, it is not clear that all confounding variables are measured within the dataset - an essential assumption to the graphical modeling done here. In fact, being unsuccessful in (1) finding a model that could predict quality indicates that this is almost surely not the case. Thus, further data collection should be done in tandem with advanced graphical techniques.

Code Appendix

```
## ---- echo=FALSE, include=FALSE-----------------------------------------------
set.seed(42)
df = read.table("asthma.txt",head=TRUE)
tempDf = df
df[,1] = df[,1]-1

tempDf[,1] = factor(df[,1])
tempDf[,3] = factor(df[,3])
tempDf[,4] = factor(df[,4])
tempDf[,5] = factor(df[,5])
tempDf[,6] = factor(df[,6])
tempDf[,7] = factor(df[,7])
tempDf[,11] = factor(df[,11])

library("ggplot2")
library("GGally")
library("glmnet")
library("caret")
library("mgcv")
library("ranger")
library("e1071")
library("MASS")
library("plyr")


## ---- echo=FALSE-------------------------------------------------------------
set.seed(42)
## Exploratory Data Analysis
summary(tempDf)


## ---- echo=FALSE, message=FALSE, out.extra='angle=90', out.width="140%", out.height="150%"----
set.seed(42)
ggpairs(df, axisLabels = "none", lower=list(continuous=wrap("points",  size=.1)), upper=list(continuous=


## ---- echo=FALSE-------------------------------------------------------------
set.seed(42)
## Feature Engineering:

n = nrow(df)

createFeatures = function(df, colNum) {
  colName = names(df)[colNum]
  rawFeature = df[,colName]
  df[paste0("log", colName)] = log(rawFeature+1)
  df[paste0("sqrt", colName)] = sqrt(rawFeature)
  df[paste0("recip", colName)] = 1/(rawFeature+1)
  df[paste0("sqr", colName)] = rawFeature^2
  df[paste0("cubed", colName)] = rawFeature^3
  return(df)
```

```
}

addInteraction = function(colName1, colName2, df) {
  rawFeature = df[,colName1]
  if (!colName1==colName2) {
    if(!paste0(colName1, ":", colName2) %in% names(df)){
      df[paste0(colName2, ":", colName1)] = df[,colName2]*rawFeature}
  }
  return(df)
}

createFeatures2 = function(df, colNum) {
  colName = names(df)[colNum]
  rawFeature = df[,colName]
  df = addInteraction(colName, "pg", df)
  df = addInteraction(colName, "age", df)
  df = addInteraction(colName, "sex", df)
  df = addInteraction(colName, "educ", df)
  df = addInteraction(colName, "insu", df)
  df = addInteraction(colName, "drug", df)
  df = addInteraction(colName, "severity", df)
  df = addInteraction(colName, "com", df)
  df = addInteraction(colName, "pcsd", df)
  df = addInteraction(colName, "mcs.sd", df)
  return(df)
}

builtDf = df
for (i in 1:10) {
  builtDf = createFeatures(builtDf, i)
}
builtDf2 = builtDf
for (i in 1:10) {
  builtDf2 = createFeatures2(builtDf2, i)
}


## ---- echo=FALSE----------------------------------------------------------
set.seed(42)
## Train-test Split:
partition = sample(nrow(df), nrow(df)*.80)
testPartition = seq(nrow(df))[-partition]


## ---- echo=FALSE, warning=FALSE-------------------------------------------
set.seed(42)
# Build the models:
logModel1 = glm(quality ~ ., data=df[partition,], family="binomial")
logModel2 = glm(quality ~ ., data=builtDf[partition,], family="binomial")
logModel3 = glm(quality ~ ., data=builtDf2[partition,], family="binomial")

linearModel1 = lm(quality ~ ., data=df[partition,])
linearModel2 = lm(quality ~ ., data=builtDf[partition,])
```

```r
linearModel3 = lm(quality ~ ., data=builtDf2[partition,])

lassoModel1 = cv.glmnet(y=df[partition,11],x=as.matrix(df[partition,-11]),
                        alpha=1)
lassoModel2 = cv.glmnet(y=builtDf[partition,11],
                        x=as.matrix(builtDf[partition,-11]),
                        alpha=1)
lassoModel3 = cv.glmnet(y=builtDf2[partition,11],
                        x=as.matrix(builtDf2[partition,-11]), alpha=1)

svmModel1 = svm(quality ~.,data=df[partition,],kernel="linear")
svmModel2 = svm(quality ~.,data=builtDf2[partition,c(1:11, 62:106)],
                kernel="linear")

ldaModel1 = lda(quality ~., data=df[partition,])
ldaModel2 = lda(quality ~., data=builtDf2[partition, c(1:11, 62:106)])

forestModel1 = ranger(y=df[partition,11], x=df[partition,-11],
                      num.trees = 1000)
forestModel2 = ranger(y=builtDf2[partition, 11],
                      x=builtDf2[partition, c(1:10, 62:106)],
                      num.trees = 1000)



## ---- echo=FALSE----------------------------------------------------------------
set.seed(42)
# Evaluate the models:

generateAccuracy = function(model, y, X, parti, isForest, isLDA) {
  if(isLDA){preds = as.numeric(predict(model, X[parti,],
                                       type="response")[[1]])-1}
  if(isForest){preds = round(predict(model, X[parti,],
                                     type="response")$prediction)}
  if(!isForest & !isLDA){preds = round(predict(model, X[parti,],
                                         type="response"))}
  return(mean(preds==y[parti]))
}

modelDataOutput = function(model, y, X, isForest=F, isLDA=F) {
  temp = data.frame()
  name = deparse(substitute(model))
  temp[name, 1] = ncol(X)
  temp[name, 2] = generateAccuracy(model, y, X, partition, isForest, isLDA)
  temp[name, 3] = generateAccuracy(model, y, X, testPartition,
                                   isForest, isLDA)
  return(temp)
}

# Display model evaluation:
displayRMSEData = function() {
  temp = modelDataOutput(logModel1, df[,11], df[,-11])
  temp = rbind(temp, modelDataOutput(logModel2, builtDf[,11], builtDf[,-11]))
```

```
  temp = rbind(temp, modelDataOutput(logModel3, builtDf2[,11],
                                     builtDf2[,-11]))
  temp = rbind(temp, modelDataOutput(linearModel1, df[,11], df[,-11]))
  temp = rbind(temp, modelDataOutput(linearModel2, builtDf[,11],
                                     builtDf[,-11]))
  temp = rbind(temp, modelDataOutput(linearModel3, builtDf2[,11],
                                     builtDf2[,-11]))
  temp = rbind(temp, modelDataOutput(lassoModel1, df[,11],
                                     as.matrix(df[,-11])))
  temp = rbind(temp, modelDataOutput(lassoModel2, builtDf[,11],
                                     as.matrix(builtDf[,-11])))
  temp = rbind(temp, modelDataOutput(lassoModel3, builtDf2[,11],
                                     as.matrix(builtDf2[,-11])))
  temp = rbind(temp, modelDataOutput(svmModel1, df[,11], df[,-11]))
  temp = rbind(temp, modelDataOutput(svmModel2, builtDf2[,11],
                                     builtDf2[,c(1:10, 62:106)]))
  temp = rbind(temp, modelDataOutput(ldaModel1, df[,11], df[,-11], F, T))
  temp = rbind(temp, modelDataOutput(ldaModel2, builtDf2[,11],
                                     builtDf2[,c(1:10, 62:106)],
                                     F, T))
  temp = rbind(temp, modelDataOutput(forestModel1, df[,11], df[,-11], T))
  temp = rbind(temp, modelDataOutput(forestModel2, builtDf2[, 11],
                                     builtDf2[, c(1:10, 62:106)], T))
  temp = setNames(temp, c("Num. Predictors" ,"Accuracy (Train)", "Accuracy (Test)"))
  return(temp)
}


## ---- warning = FALSE, echo=FALSE-----------------------------------------------
set.seed(42)
## Model Perf
displayRMSEData()


## ---- echo=FALSE, eval=FALSE, warning=FALSE-----------------------------------
## 'set.seed(42)
## df = df[,-6]
## builtDf = builtDf[,-6]
## builtDf2 = builtDf2[,-6]
## z1 = as.data.frame(sapply(df, function(df) (abs(df-mean(df))/sd(df))))
##
## cutoff = 4
##
## # Only keep rows with all their abs value z-scores less than 4
## df = df[!rowSums(z1>cutoff),]
## builtDf = builtDf[!rowSums(z1>cutoff),]
## builtDf2 = builtDf2[!rowSums(z1>cutoff),]
##
## # Resample without outliers:
## partition = sample(nrow(df), nrow(df)*.80)
## testPartition = seq(nrow(df))[-partition]
##
## # Build the models:
```

```
## logModel1 = glm(quality ~ ., data=df[partition,], family="binomial")
## logModel2 = glm(quality ~ ., data=builtDf[partition,], family="binomial")
## logModel3 = glm(quality ~ ., data=builtDf2[partition,], family="binomial")
##
## linearModel1 = lm(quality ~ ., data=df[partition,])
## linearModel2 = lm(quality ~ ., data=builtDf[partition,])
## linearModel3 = lm(quality ~ ., data=builtDf2[partition,])
##
## lassoModel1 = cv.glmnet(y=df[partition,10],x=as.matrix(df[partition,-10]),
##                         alpha=1)
## lassoModel2 = cv.glmnet(y=builtDf[partition,10],
##                         x=as.matrix(builtDf[partition,-10]),
##                         alpha=1)
## lassoModel3 = cv.glmnet(y=builtDf2[partition,10],
##                         x=as.matrix(builtDf2[partition,-10]), alpha=1)
##
## svmModel1 = svm(quality ~.,data=df[partition,],kernel="linear")
## svmModel2 = svm(quality ~.,data=builtDf2[partition,c(1:10, 61:105)],
##                 kernel="linear")
##
## ldaModel1 = lda(quality ~., data=df[partition,])
## ldaModel2 = lda(quality ~., data=builtDf2[partition, c(1:10, 61:105)])
##
## forestModel1 = ranger(y=df[partition,10], x=df[partition,-10],
##                       num.trees = 1000)
## forestModel2 = ranger(y=builtDf2[partition, 10],
##                       x=builtDf2[partition, c(1:9, 61:105)],
##                       num.trees = 1000)
##
## displayRMSEData = function() {
##   temp = modelDataOutput(logModel1, df[,10], df[,-10])
##   temp = rbind(temp, modelDataOutput(logModel2, builtDf[,10], builtDf[,-10]))
##   temp = rbind(temp, modelDataOutput(logModel3, builtDf2[,10],
##                                      builtDf2[,-10]))
##   temp = rbind(temp, modelDataOutput(linearModel1, df[,10], df[,-10]))
##   temp = rbind(temp, modelDataOutput(linearModel2, builtDf[,10],
##                                      builtDf[,-10]))
##   temp = rbind(temp, modelDataOutput(linearModel3, builtDf2[,10],
##                                      builtDf2[,-10]))
##   temp = rbind(temp, modelDataOutput(lassoModel1, df[,10],
##                                      as.matrix(df[,-10])))
##   temp = rbind(temp, modelDataOutput(lassoModel2, builtDf[,10],
##                                      as.matrix(builtDf[,-10])))
##   temp = rbind(temp, modelDataOutput(lassoModel3, builtDf2[,10],
##                                      as.matrix(builtDf2[,-10])))
##   temp = rbind(temp, modelDataOutput(svmModel1, df[,10], df[,-10]))
##   temp = rbind(temp, modelDataOutput(svmModel2, builtDf2[,10],
##                                      builtDf2[,c(1:9, 61:105)]))
##   temp = rbind(temp, modelDataOutput(ldaModel1, df[,10], df[,-10], F, T))
##   temp = rbind(temp, modelDataOutput(ldaModel2, builtDf2[,10],
##                                      builtDf2[,c(1:9, 61:105)],
##                                      F, T))
##   temp = rbind(temp, modelDataOutput(forestModel1, df[,10], df[,-10], T))
```

```
##   temp = rbind(temp, modelDataOutput(forestModel2, builtDf2[, 10],
##                                      builtDf2[, c(1:9, 61:105)], T))
##   temp = setNames(temp, c("Num. Predictors" ,"Accuracy (Train)", "Accuracy (Test)"))
##   return(temp)
## }
##
## displayRMSEData()'


## ---- warning=FALSE, echo=FALSE-------------------------------------------------
Y = df$quality
A = df$pg
X = df[,-c(1, 11)]
pgGrid = 0:1
B = 1000
causalEstList = c()
for(a in pgGrid){
    causalEst = list(1:B)
    for(i in 1:B){
        n = length(Y)
        I = sample(1:n,size=n,replace=TRUE)
        XX = X[I,]
        YY = Y[I]
        AA = A[I]
        dfdf = data.frame(XX, pg=AA, quality=YY)
        out = lm(quality ~ ., data=dfdf)
        causalEst[[1]][i] = mean(predict(out, newdata=data.frame(XX, pg=a),
                                         type="response"))
    }
    causalEst = list(sort(causalEst[[1]]))
    causalEstList = append(causalEstList, causalEst)
}

lowerBounds = c()
upperBounds = c()
for(i in 1:length(pgGrid)){
    lowerBounds[i] = causalEstList[[i]][25]
    upperBounds[i] = causalEstList[[i]][975]
}
xGraph = append(append(pgGrid, pgGrid), pgGrid)
yGraph = append(append(unlist(lapply(causalEstList, FUN=mean)),
                lowerBounds), upperBounds)
plot(x=xGraph, y=yGraph, xlab="Physician Group", ylab="Causal Est")
lines(x=pgGrid, y=lowerBounds, col="red", cex=5)
lines(x=pgGrid, y=upperBounds, col="red", cex=5)


## ---- echo=FALSE---------------------------------------------------------------
X = df[,c(1,3,11)]
tbl = count(X)
out = glm(freq ~ pg*sex*quality, data=tbl, family="poisson")
summary(out)[c(1,12)]
```

```
## ----code = readLines(knitr::purl(knitr::current_input(), documentation = 1)), echo = TRUE, eval = FA
##
```