IEOR165 PROJECT 1

Zeyu Xing(xzy@berkeley.edu); Jenny(Qiuyi) Chen(qchen9797@berkeley.edu);
ElinaYu(elinayuyue@berkeley.edu); Yi Xu(eddi@berkeley.edu)

April 19, 2018

Introduction

The project is intended to model wine preferences by data mining from physicochemical properties. Following is the data frame and initial statistics for each property from the source of *winquanlity* provided.

```
##
## Please cite as:

##  Hlavac, Marek (2018). stargazer: Well-Formatted Regression and Summary Statistics Tables.

##  R package version 5.2.1. https://CRAN.R-project.org/package=stargazer

##
## ===========================================================
## Statistic               N    Mean  St. Dev.  Min    Max
## -----------------------------------------------------------
## fixed_acidity        1,599 8.320   1.741   4.600 15.900
## volatile_acidity     1,599 0.528   0.179   0.120  1.580
## citric_acid          1,599 0.271   0.195   0.000  1.000
## residual_sugar       1,599 2.539   1.410   0.900 15.500
## chlorides            1,599 0.087   0.047   0.012  0.611
## free_sulfur_dioxide  1,599 15.875  10.460  1.000 72.000
## total_sulfur_dioxide 1,599 46.468  32.895  6.000 289.000
## density              1,599 0.997   0.002   0.990  1.004
## pH                   1,599 3.311   0.154   2.740  4.010
## sulphates            1,599 0.658   0.170   0.330  2.000
## alcohol              1,599 10.423  1.066   8.400 14.900
## quality              1,599 5.636   0.808     3      8
## -----------------------------------------------------------
```

Regression

We've performed ordinary least squares regression, ridge regression, lasso regression as well as regression with elastic net.The default of the function is 10-fold, which is a reasonable value based on the project.

Gaussian is the default family option in the function *glmnet*. Suppose we have observations $x_i \in \mathbb{R}^p$ and the responses $y_i \in \mathbb{R}, i = 1, \ldots, N$. The objective function for the Gaussian family is

$$\min_{(\beta_0,\beta)\in\mathbb{R}^{p+1}} \frac{1}{2N}\sum_{i=1}^{N}(y_i - \beta_0 - x_i^T\beta)^2 + \lambda\left[(1-\alpha)||\beta||_2^2/2 + \alpha||\beta||_1\right]$$

where $\lambda \geq 0$ is a complexity parameter and $0 \leq \alpha \leq 1$ is a compromise between ridge ($0 \leq \alpha \leq 0$) and lasso ($0 \leq \alpha \leq 1$).

However, in terms of λ, there are two outcomes from the model *lambda.min* and *lambda.lse*. The main difference between these two value is:

1.The best model that may be too complex of slightly overfitted: *lambda.min* 2.The simplest model that has comparable error to the best model given the uncertainty: *lambda.*1*se*.

Given this case is not too complex, we choose *lambda.min* to further enchance the accuracy.

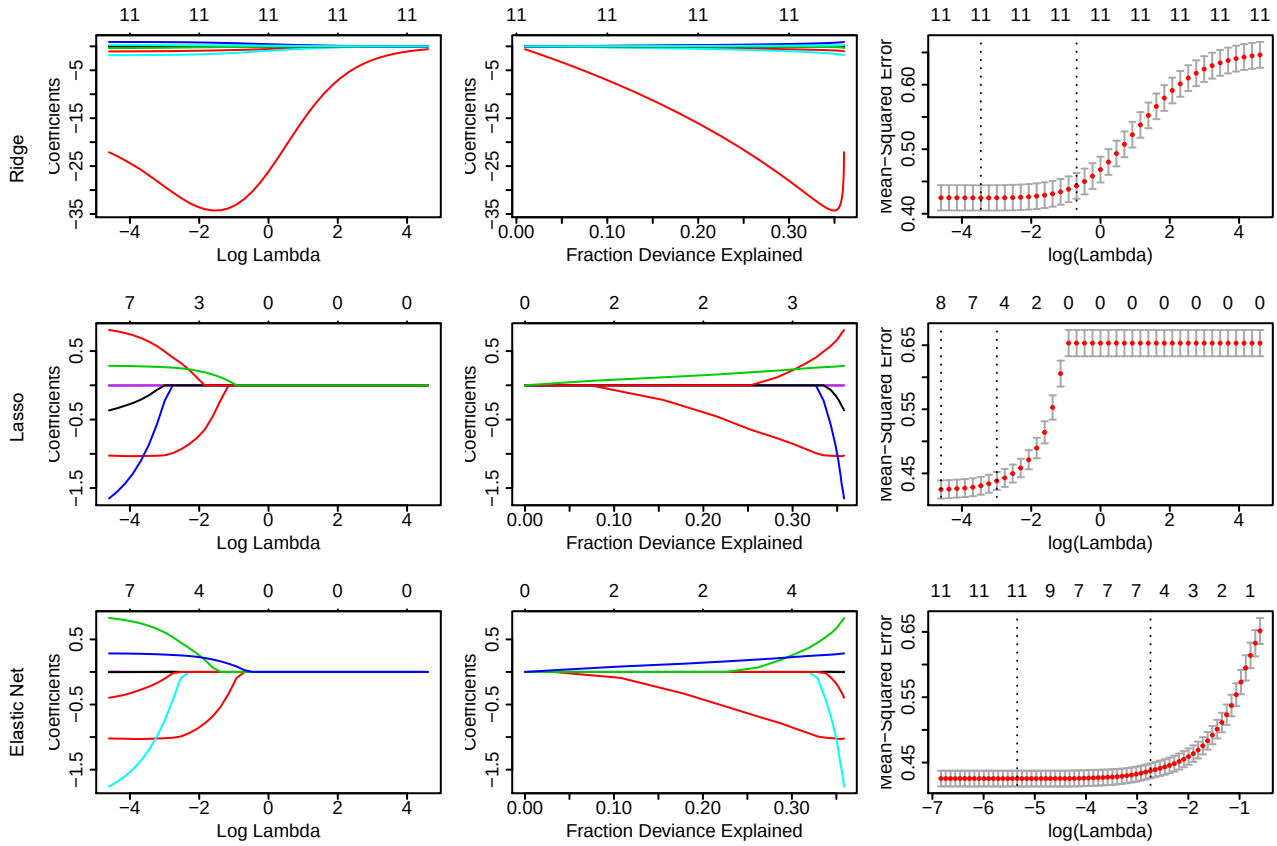
In terms of Elastic Net Regression, however, the *glmnet* or *cv.glumnet* don't help us find the optimal α, which ranges from 0 to 1. Therefore, we run 9 Elastic Regression with α from 0.1 to 0.9 with step width of 0.1. Based on the objective function, we may find optimal values for both α and λ.

So far, we've finished the regularized regression and cross validation by applying the R function *cv.glmnet* with different *alpha* value. The following diagrams show that the relationship between crossv alidation error and the value of lambdas by taking a log transformation to lambdas.

```
library(tidyverse)
library(broom)
library(glmnet)
library(stargazer)
library(dplyr)
library(xtable)
#define the response and regressors
y<-winequality$quality
x<-winequality
x$quality<-NULL
x<-as.matrix(x)
#ols regression
ols<-lm(quality~.,data=winequality)
#define the lambdas
lambdas <- 10^seq(-2, 2, by = .1)
#regression with Ridge, Lasso and elastic net
rr<-glmnet(x, y, alpha = 0, lambda = lambdas,family="gaussian")
lasso<-glmnet(x, y, alpha = 1, lambda = lambdas,family="gaussian")
for (i in 1:9){assign(paste("en", i, sep=""),glmnet(x,y,alpha=i/10,lambda=lambdas,family="gaussian"))}
#crossvalidation
rr.cv<- cv.glmnet(x, y, alpha = 0, lambda = lambdas,family="gaussian",type.measure="mse")
lasso.cv<- cv.glmnet(x, y, alpha = 1, lambda = lambdas,family="gaussian",type.measure="mse")
en.cvm<-data.frame(rep(1:9))
for (i in 1:9){
  assign(paste("en.cv", i, sep=""), cv.glmnet(x, y, type.measure="mse",alpha=i/10,family="gaussian"))
  en.cvm[i,]<-min(cv.glmnet(x, y, type.measure="mse",alpha=i/10,family="gaussian")$cvm)
}
en.cv<-get(paste("en.cv", which.min(as.matrix(en.cvm)), sep=""))
en<-get(paste("en", which.min(as.matrix(en.cvm)), sep=""))
#plot the regression outcomes
par(mfrow=c(3,3),cex=0.7,pch=19,cex=0.3,cex.axis=2,cex.lab=2)
{ plot(rr,xvar="lambda")
  plot(rr,xvar="dev")
  plot(rr.cv)

  plot(lasso,xvar="lambda")
  plot(lasso,xvar="dev")
  plot(lasso.cv)

  plot(en,xvar="lambda")
  plot(en,xvar="dev")
  plot(en.cv)
}
```



The first columes are the variation of coeficients with different lambda value. The second ones are the variation of coefficients with explained deviance. The last group show the relationship between MSE and different lambdas. The first, secon, third rows represent Ridge Regression, Lasso Regression and Elastic Net Regression repectively.

Summary

Followings are the optimal tuning parameters and optimal coefficients for each models repectively.

```
#obtain the optimal lambda and alpha for each regression
rr.coe<-coef(rr.cv,s=rr.cv$lambda.min)
lasso.coe<-coef(lasso.cv,s=lasso.cv$lambda.min)
zero<-c(0)
lasso.coe.value<-lasso.coe
#lasso.coe.value<-t(cbind(lasso.coe@x[1],0,lasso.coe@x[2],0,lasso.coe@x[3],
                         #lasso.coe@x[4],lasso.coe@x[5],lasso.coe@x[6],0,
                         #lasso.coe@x[7],lasso.coe@x[8],lasso.coe@x[9]))
en.coe<-coef(en.cv,s=en.cv$lambda.min)
rr<-glmnet(x, y, alpha = 0, lambda = rr.cv$lambda.min,family="gaussian")
lasso<-glmnet(x, y, alpha = 1, lambda = lasso.cv$lambda.min,family="gaussian")
en<-glmnet(x, y, alpha = which.min(as.matrix(en.cvm))/10, lambda = en.cv$lambda.min,family="gaussian")
rr.pre<-predict(rr,x)
lasso.pre<-predict(lasso,x)
en.pre<-predict(en,x)
tuning.optimal<-data.frame(
  matrix(c(0,0,sum(ols$residuals^2)/(nrow(x)-10),1-sum((y-ols$fitted.values)^2)/sum((y-mean(y))^2),1-(l
```

```
        0,rr.cv$lambda.min,sum((rr.pre-y)^2)/(nrow(x)-14),1-sum((y-rr.pre)^2)/sum((y-mean(y))^2),1-(leng
        lasso.cv$lambda.min,0,sum((lasso.pre-y)^2)/(nrow(x)-14),1-sum((y-lasso.pre)^2)/sum((y-mean(y))^2
        en.cv$lambda.min,which.min(as.matrix(en.cvm))/(nrow(x)-14),sum((en.pre-y)^2)/length(y),1-sum((y-
        nrow=5,ncol=4))
colnames(tuning.optimal)<-c("OLS Regression","Ridge Regression","Lasso Regression","Elastic Net Regressi
rownames(tuning.optimal)<-c("Lambda_optimal","Alpha_optimal","MSE","R^2","R^2_adj")
coe<-data.frame(matrix(c(ols$coefficients,rr.coe@x,lasso.coe[1:14],en.coe@x),nrow=nrow(rr.coe),ncol=4))
colnames(coe)<-c("ols","Ridge Regression","Lasso Regression","Elastic Net Regression")
rownames(coe)<-c("intercept",colnames(x)[1:11])
stargazer(coe,type="text",summary=F)
```

```
##
## ============================================================================================
##                          ols   Ridge Regression Lasso Regression Elastic Net Regression
## --------------------------------------------------------------------------------------------
## intercept             21.965       30.770           4.120
## fixed_acidity          0.025       0.030              0
## volatile_acidity      -1.084       -1.029           -1.026              9.543
## citric_acid           -0.183       -0.104             0                 0.008
## residual_sugar         0.016       0.019            0.0002             -1.055
## chlorides             -1.874       -1.823           -1.652             -0.092
## free_sulfur_dioxide    0.004       0.004            0.002              0.008
## total_sulfur_dioxide -0.003       -0.003           -0.003             -1.832
## density              -17.881     -26.905             0                 0.004
## pH                    -0.414       -0.325           -0.366             -0.003
## sulphates              0.916       0.895            0.808              -5.292
## alcohol                0.276       0.258            0.285              -0.431
## --------------------------------------------------------------------------------------------
```

```
stargazer(tuning.optimal,type="text",summary=F)
```

```
##
## ============================================================================================
##                 OLS Regression Ridge Regression Lasso Regression Elastic Net Regression
## --------------------------------------------------------------------------------------------
## Lambda_optimal       0               0              0.010               0.005
## Alpha_optimal        0             0.032              0                 0.004
## MSE                0.419           0.421            0.422               0.417
## R2                 0.361           0.360            0.358               0.360
## R2_adj             0.355           0.356            0.354               0.356
## --------------------------------------------------------------------------------------------
```

Though it's not fair to compare MSE of OLS with cross validation error of RR/LASSO/ELASTIC NET regression, we could still tell that Elastic Net is the optimal solution for this circumstance.

Part Two

For the given dataset, our goal is to construct the model using the SVM method. There are built-in functions in MATLAB we can direct use to generate the model. From the given dataset, we need to predict thechannel by the Region, Fresh, Milk, Grocery, Frozen, Detergents_Paper, Delicassen. We input the column of channel as the response and the other seven columns as the predictor. Run the SVM the function with the input above, set the k-out parameter, and try different kernels from linear, poly and gaussian, then the result data with the mean error and coefficients (for linear only) shows up.

Input Data:

X = (400x7 double) Region, Fresh, Milk, Grocery, Frozen, Detergents_Paper, Delicassen

Y = (400x1 double) Channel

Linear SVM (code)

```
SVMModel = fitcsvm(X,Y,'Standardize',true,'ClassNames',{'1','2'});
CVSVMModel = crossval(SVMModel,'Holdout',k)
```

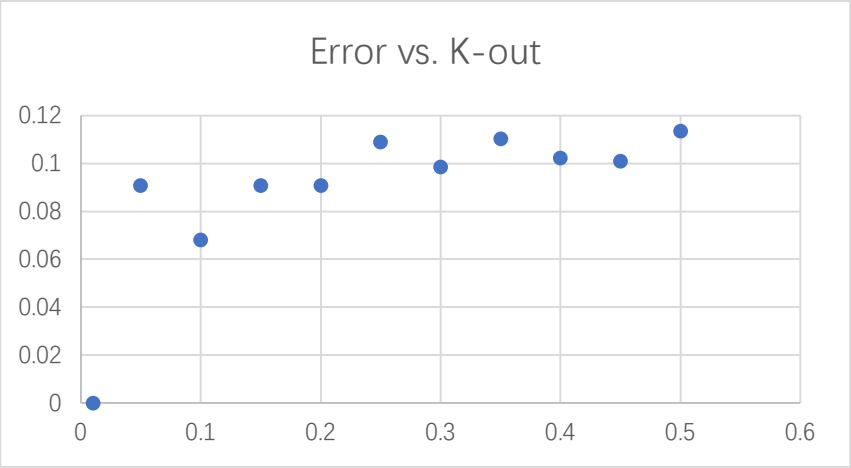
Alternatively,

```
CVSVMModel
= fitcsvm(X,Y,'Holdout',k,'ClassNames',{'1','2'},'Standardize',true);
```

k is the percent of holdout sample; for our model, we use k-Out Cross-Validation.

The output for the linear:

Linear								
k-out	error	coefficients						
0.05	0.0909	0.2741125	0.0485763	0.3768671	0.5131733	-0.362459	2.411575739	-0.0683206
0.1	0.0682	0.1227161	0.0291598	0.1972095	0.5076389	-0.2949826	2.360035407	-0.1652915
0.15	0.0909	0.3028351	-0.021299	0.4142067	0.4112489	-0.4704724	2.328857795	-0.0943383
0.2	0.0909	0.2548166	0.0409709	0.3666046	0.5295594	-0.3413221	2.519673373	-0.1270899
0.25	0.1091	0.2405753	0.0089857	0.357081	0.4816712	-0.3324532	1.976414785	0.0156295
0.3	0.0985	0.3598015	-0.0748424	0.4290036	0.4869561	-0.2619787	2.379610904	-0.2395512
0.35	0.1104	0.3910542	-0.172112	0.5074589	0.356207	-0.3697664	2.680523364	-0.2052772
0.4	0.1023	0.2212322	0.0556372	0.1917677	0.591487	-0.3627516	2.221806932	-0.4333383
0.45	0.101	0.2117887	-0.2084394	0.0743014	1.0956784	-0.5853845	2.105767645	0.1335061
0.5	0.1136	0.4592502	-0.1866631	0.422428	0.3656101	0.0346673	2.365724253	-0.3281133
0.01	0	0.2372294	0.0353294	0.3484889	0.5121178	-0.2829269	2.555969	-0.2219295
		Region	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen



For the linear model, the minimum cross-validation error is 0.0682 (when k-out = 0.1). Note for k-out very small, the size of the test data set is very small. Although the corresponding mean error is 0, it doesn't perfectly indicate the model is much better than others.

SVM with Polynomial Kernel (code)

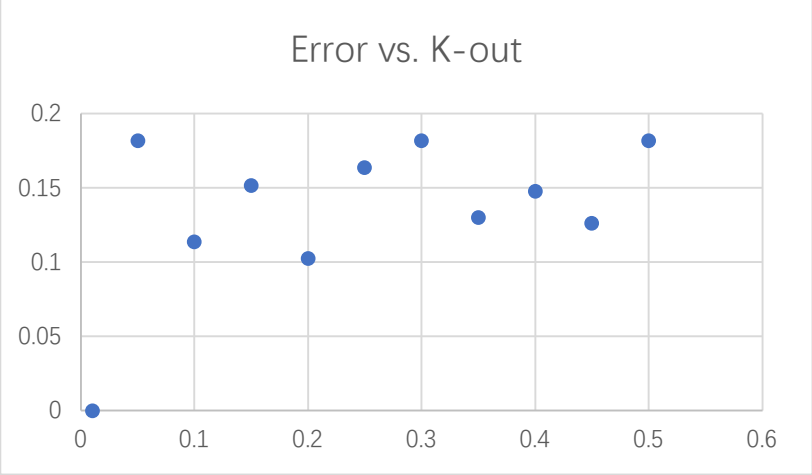
```
CVSVMModelPolynomial
= fitcsvm(X,Y,'Holdout',k,'KernelFunction','polynomial',...
'PolynomialOrder',i,'ClassNames',{'1','2'},'Standardize',true);
```

i is the order of polynomial; for our model, we choose polynomial order 4 to minimize error.

Output for polynomial:

Polynomial, d=4	
k-out	error
0.01	0
0.05	0.1818
0.1	0.1136
0.15	0.1515
0.2	0.1023
0.25	0.1636
0.3	0.1818
0.35	0.1299
0.4	0.1477
0.45	0.1263
0.5	0.1818

For the polynomial model, the minimum cross-validation error is 0.1023 (when k-out = 0.2).



SVM with Gaussian Kernel (code)

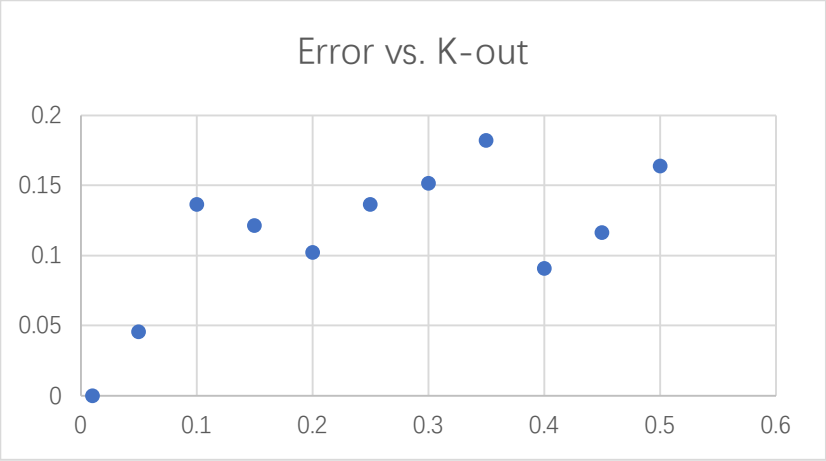
```
CVSVMModelPolynomial
= fitcsvm(X,Y,'Holdout',k,'KernelFunction','rbf',...
'ClassNames',{'1','2'},'Standardize',true);
```

'rbf' represents Gaussian Kernel.

Output for Gaussian:

Gaussian	
k-out	error
0.01	0
0.05	0.0455
0.1	0.1364
0.15	0.1212
0.2	0.1023
0.25	0.1364
0.3	0.1515
0.35	0.1818
0.4	0.0909
0.45	0.1162
0.5	0.1636

For the Gaussian model, the minimum cross-validation error is 0.0455 (when k-out = 0.05).



Cross-Validation Error (code)

```
kfoldLoss(CVSVMModel)
```

kfoldLoss returns loss obtained by cross-validated classification model.

Predict Channel

```
CVSVMModel = fitcsvm(X,Y,'Holdout',k,'ClassNames',{'1','2'},...
    'Standardize',true);
CompactSVMModel = CVSVMModel.Trained{1}; % Extract trained, compact
classifier
testInds = test(CVSVMModel.Partition);   % Extract the test indices
XTest = X(testInds,:);
YTest = Y(testInds,:);
[label,score] = predict(CompactSVMModel,XTest);
table(YTest(1:10),label(1:10),score(1:10,2),'VariableNames',...
    {'TrueLabel','PredictedLabel','Score'})
```

for k=0.50, we obtained the following results:

TrueLabel	PredictedLabel	Score
2	2	0.46725
2	2	0.018562
2	2	0.3486
2	2	3.8813
2	2	2.0233
2	2	1.928
2	2	3.2617
2	2	2.3933
1	1	-1.9556
1	1	-0.85003