Supervised Learning - Assignment 1

Dean Hope Robertson (RBRDEA003)

4/8/2018

Abstract

In this assignment students are tasked with building a regression model for predicting an individual's credit balance using a variety of variables. The students are required to download the **Assignment_1_Data.txt.** dataset which contains approximately **400** observations, each with **10** predictor variables. The students are tasked with using their new found skills in the art of statistical learning, in order to train model that best predict the individuals credit balance using a various methods covered in class.

Introduction

In the given dataset there is one response variable, Balance, and 10 predictor variables (See Table 1). The Variables are a mixture of **quantitative** (e.g. Income) and **qualitative** (e.g. Gender) variables. Quantitative variables are numerical values, and in contrast qualitative variables take categorical values. **Income** is the only **continuous** quantitative variables, while the other remaining quantitative variables are **discrete**. The table below shows the first row of the credit balance dataset as well as variable characterization.

Table1 : Summary of predictor variables.

Income	**Limit**	**Rating**	**Cards**	**Age**	**Education**	**Gender**	**Student**	**Married**	**Ethnicity**
14.89	3606	283	2	34	11	Male	No	Yes	Caucasian
Quant	Qaunt	Qaunt	Qaunt	Qaunt	Qaunt	Qual	Qual	Qual	Qual
Cont.	Disc	Disc	Disc	Disc	Disc	Nominal	Nominal	Nominal	Nominal

Using the 10 predictive variables above, the objective is to build a regression model that can predict **Balance**. In order to do this, it is required that the models be trained on a training dataset, and then used to predict from a smaller test dataset. Regression analysis is used determine the relationship between the response variable and the predictor variables, and thus can be seen as a regression problem as defined below:

$$Y = f(X)) + \epsilon$$

where Y is the response variable and the X is the predictor variables

Seeing as there are a number of predictor variables, the formula can be expanded to:

$$Y_i = \beta_0 + \beta_1 X1_i + \beta_2 1X2_i + \beta_3 X3_i + \beta_4 X4_i + \beta_5 X5_i + \beta_6 X6_i + \beta_7 X7_i + \beta_8 X8_i + \beta_9 X9_i + \beta_1 0X10_i + e$$

where the Y_i is the predicted respsonse and the predictor variables are $X_i = [X1_i ... X10_i]$

The regression model attempts to estimate the *coeffiecents* $= \beta_0, \beta_1, \beta_2 ...$ in an effort to determine the response varaible Y.

In **Appendix A**, under the **'Balance Plots'** subheading it is possible to see the relationship which Income, Limit, Age and Education have with the response variable, Balance. It can be derived that the four predictor variables seem to have a **linear relationship** with the response variable. This forms the basis of the linear argument when determining what type of regression to explore in this assignment.

Measuring the fit

There are many metrics one can compare regression models across. In terms of regression modeling, the most commonly used measure for 'best fit' is the *mean squared error* (**MSE**).Alternatively one could use, the R^2 statistic, which is the proportionality of variance explained. This value can be misleading as it will always increases with the addition of more predictors. For this reason the R^2 statistic is not a good metric to compare models as it favors models with greater number of variables. For this assignment, *mean squared error* will be used as the benchmark for which a model's accuracy is evaluated. The **MSE** is calculate by the following equation:

$$MSE = \frac{1}{n}\sum(y_i - \hat{f}(x_i))^2$$

Null Model

The initial step is to compute the NULL model, that is the model in which no predictive variables are taken into account and thus have zero regression *coefficients*. This is done by taking the average of the response variables $\bar{Y}$ to predict the continuous response variables. The Null model had a training MSE and test MSE of **30.21**, and **37.172** respectively. This is a very large *mean squared error* and confirms that the Null Model is not accurate for this specific dataset.

1. Multiple Linear Regression

After computing the Null model, the second step is to regress all the all the predictor variables to the response variable via basic linear regression. This is the first step in **backwards selection**. This can be conducted by using the built-in **Linear Model (lm)** function in R which utilizes *least squares* linear regression in order to determine the coefficients for best fit. The *least squares* approach selects coefficients in such a way as to minimize the *residual sum of squares* (**RSS**).

$$RSS = \sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2$$

Its is important to realize that the dataset contains categorical data, which are not measurable on a continuous scale. Categorical variables are also know as *factors*. These variables are included into the regression model using **dummy variables** that take only two possible numerical values. The built-in *lm* function encodes these categorical variables by replacing the categories with 1' and 0's. For example the 'Yes' and 'No' outcome for the *Student* category is replaced with 1 and 0 respectively. In cases where there is more than two possible categories (i.e. Ethnicity) two or *dummy variables* are required.

$$X_i = \begin{cases} 1 - \text{if the ith person is a student} \\ 0 - \text{if the ith person is not a student} \end{cases}$$

It is possible to fit the basic linear model to the training data using the following code :

```
#Fitting the training data
fit1_train = lm(Balance~.,data=train_data)
```

For this assignment the training data represents 75% of the entire dataset and the remainder is allocated to a test dataset. Once the model is trained on the training dataset, it is then tested on the test dataset and the MSE is then calculated by running the following code:

```r
#predict of test data
pred_test = predict(fit1_train, test_data)
pred_train = predict(fit1_train, train_data)

#compute mse
mse_test = mean((test_data$Balance-pred_test)^2)
mse_train = mean((train_data$Balance-pred_train)^2)
```

The **mean squared error** for the basic linear model containing all 10 variables is:

$$MSE(train) = 1.964$$
$$MSE(test) = 1.816$$

From this point, it is possible to perform backward selection by reviewing the predictor variables coefficients, t-value's and p-value's as can be seen in the table below:

Table 1: Parameter estimates for Mulitvariant Linear Model.

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	4.338	0.611	7.096	0.000
Income	0.153	0.004	39.080	0.000
Limit	0.000	0.001	0.624	0.533
Rating	-0.004	0.008	-0.528	0.598
Cards	0.076	0.074	1.037	0.301
Age	0.022	0.005	4.541	0.000
Education	0.057	0.027	2.081	0.038
GenderFemale	0.027	0.168	0.159	0.873
StudentYes	0.959	0.272	3.522	0.000
MarriedYes	-0.448	0.174	-2.576	0.010
EthnicityAsian	0.057	0.230	0.248	0.804
EthnicityCaucasian	0.074	0.203	0.367	0.714

Note the above table contains an extra *dummy variable* for the Ethnicity categorical variable.

It can be seen from the table above that that **Gender** and **Ethnicity** are the least significant variables and have a relatively weaken relationship with the response variable, **Balance**. This is determined by looking at the **t-statistic** and **p-values**. A large **t-statistic** and corresponding small **p-value** indicates that there is a relationship between that predictor and the response, therefore it is inferred that Gender and Ethnicity affect on the response variable is of less of a significance than other predictors. Variables such as **Income** and **Student** have very small p-values and thus have a strong relationship with the response variable.

The variables such as **Rating** and **Limit** also have a weaker relationship witn the response variable. In fact, these two variables have a correlation of **0.996** and can be treated as a case of collinearity. Using backwards selection, both variables are removed in the second iteration leaving a model with only 6 predictor variables, with the final model returning a MSE of **1.80**.

Table2 : Summary of backwards selection Linear Regression Iterations.

	R^2	**F-statistic**	**MSE**
Fit1	0.935	415	1.816
Fit2	0.9349	522.8	1.822
Fit3	0.9347	699	1.800

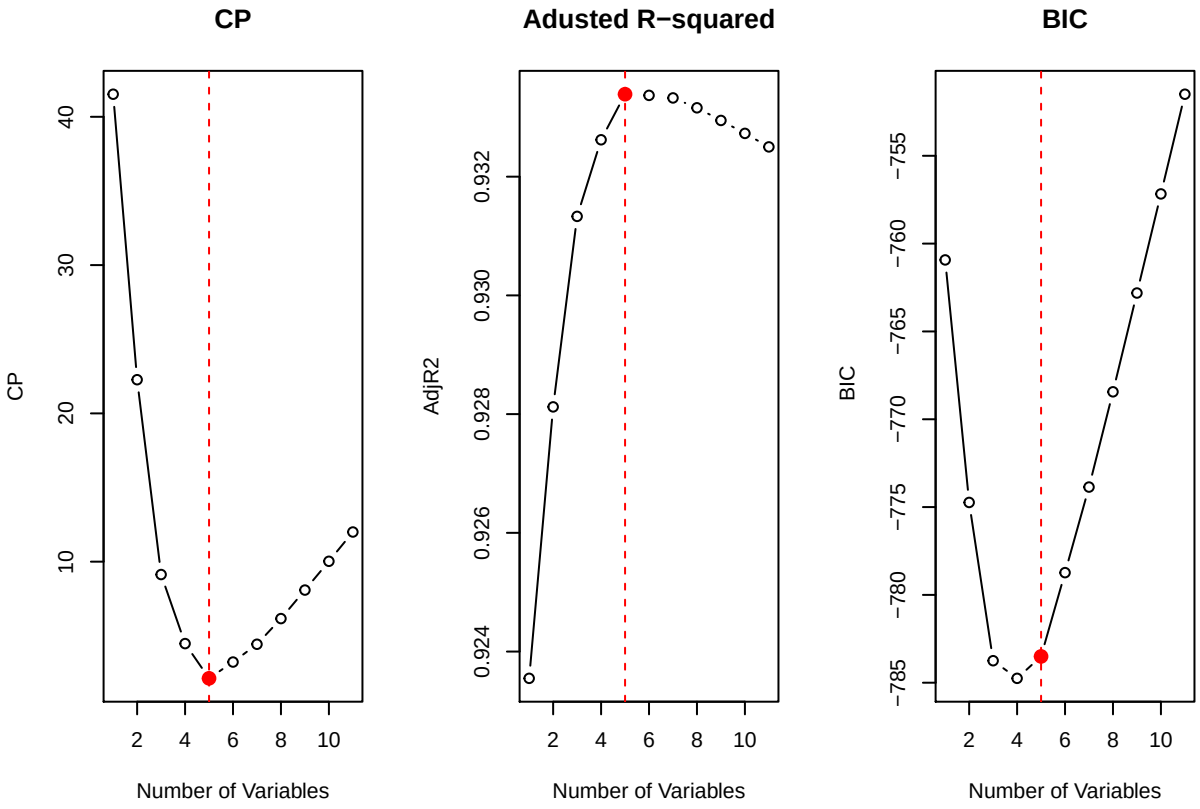
2. Subsect Selection

The subset selection is a variable selection method that only selects the best combination variable predictors that are the most significantly related to the response variable. This can be through both *forward and backward stepwise* methods, as well as *best subset selection*.

The subset algorithm essentially fits the *least mean squares* to find the best combination of variables for every possible subset size from 1 to p predictors (in this case our dataset includes 11 predictors including the extra dummy variable). The algorithm is incredibly powerful seeing as there are 55 different combinations of models that contain only two predictors, and it compute this for very subset size.

Subset selection is conducted on the Credit Balance dataset in order to find the best combination of predictors for each subset. To Assist in the evaluation of resulting models, Mallows Cp value, Bayesian Information Criterion and Adjusted R^2 values are all computed and can be seen below:

Figure 1 : Mallows's Cp, Adjusted R^2, and BIC for Subset Selection .



It is also possible to compute the **MSE** using cross validation. From the **MSE** and get the indicators above, it is apparent that the subset size with lowest MSE is **five**.

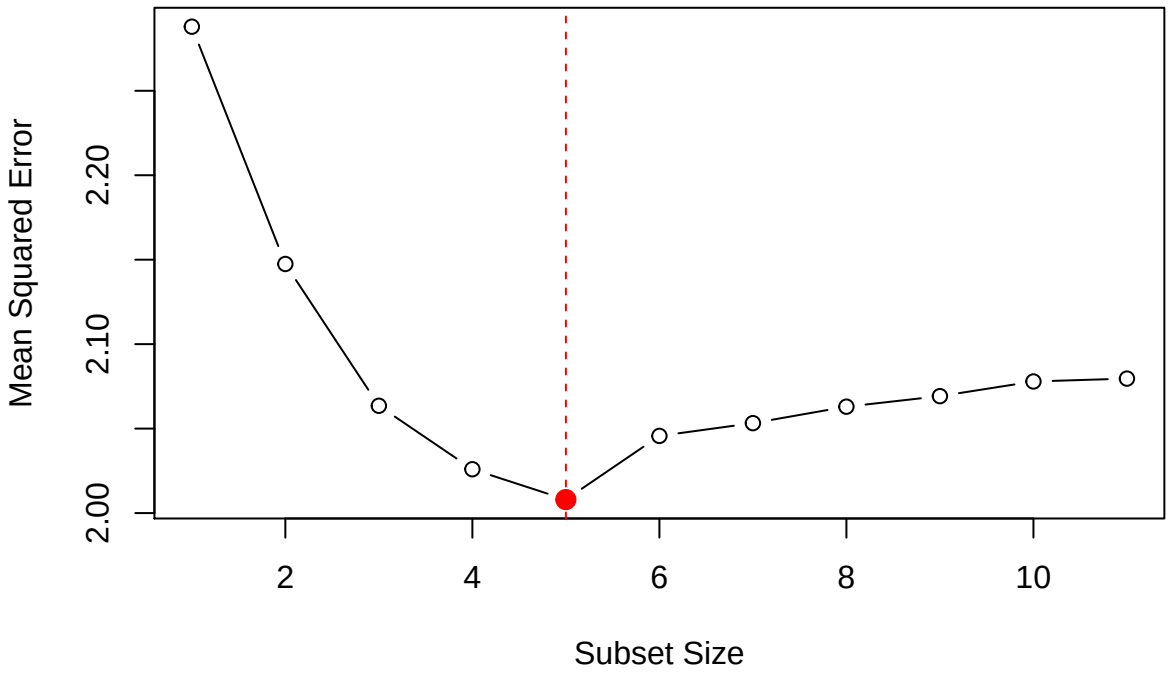


Figure 2 : MSE using Cross Validation for Subset Selection .

The subset selection algorithm has a subset with five coefficients, namely **Income**, **Age**, **Education**, **Student**, and **Married** predictor variables. The five coefficients of the subset selection model are as follows:

Table3 : Subset Selection coefficients.

Income	Age	Education	Student	Married
0.1537	0.0209	0.0758	0.6433	-0.4353

Run the following code to fit the model:

```
#Fitting subset selection variables via linear regression
sub_fit = lm(Balance~ Income + Age + Education + Student + Married,data=train_data)
```

The Subset fitted model returns the following results:

$$MSE(train) = 1.979$$
$$MSE(test) = 1.799$$

3. Shrinkage and Regularization

Shrinkage methods differ from subset selection due to the fact that subset selection tries to find the optimal amount predictors in a subset, whilst shrinkage methods try to fit all the predictor variables by **constraining** thier coefficients. Shrinkage methods do this by constraining ("shrinking") the coefficients towards zero.

As we recalled earlier in the assignment, the previous two models focus on minimizing RSS, which has been adapted below to include all the estimated coefficients. Shrinkage methods work on a similar principle, but include a *shrinkage penalty*

$$\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2 + penalty$$

The **shrinkage penalty**, which is added to the RSS, consists of a tuning parameter multiplied by the total sum of the coefficients. This creates a trade off between optimal fit and the size of coefficients, thus penalizing the coefficients for being large. When minimizing the equation below, the model is now essentially paying a price for the size of its coefficients, and thus pushing predictor variables' coefficients towards zero.

The tuning parameter λ is critical for the performance of the model. The tuning parameter determines the size of the penalty paid for coefficient size. If λ is too small, the penalty will be insignificant resulting in a fit closer to ordinary least squares estimate. λ is determined via cross validation. The two methods of regularization are **LASSO** and **Ridge Regression**.

3.1 Ridge Regression

$$RSS + \lambda \sum_{j=1}^{p} \beta_j^2$$

It is important to note in previous models using standard least squares, the coefficients are *equivariaint*, meaning that if you were to scale the coefficients there would be no effect on the RSS. In contrast, when performing ridge regression is necessary to standardize the variables. Standardizing is necessary seeing as the coefficients are all used in the penalty function. Should one variable use a different scale/unit, that same variable might the penalized unfairly, and thus standardizing the variables makes the penalization process a level play field.

The **glmnet** package R automatically standardizes the variables, and thus there is no reason to scale the predictor variables. Fitting the ridge regression model with a is done below:

```
#Ridge: alpha=0 abd LASSO: alpha=1
ridge.mod =glmnet(x,y,alpha=0)
cv.ridge = cv.glmnet(x,y,alpha =0)
bestlam=cv.ridge$lambda.min
```

The final Ridge Regression model has a λ of 0.58 and constrains the **limit** variable to the smallest coefficient of **0.002** as shown below. The Ridge regression test results are as follows:

$$MSE(train) = 2.37$$
$$MSE(test) = 2.20$$

3.2 LASSO

$$RSS + \lambda \sum_{j=1}^{p} |\beta_j|$$

The primary difference separating **Ridge Regression** and **LASSO** is the fact that coefficients can be constrained completely to zero, if the tuning parameter is sufficiently large enough. This action produces a simpler and more interpretable model. The resulting model will include a subset of the predictors.

Cross Validation is utilized to determine the tuning parameter λ. The results are as follows:

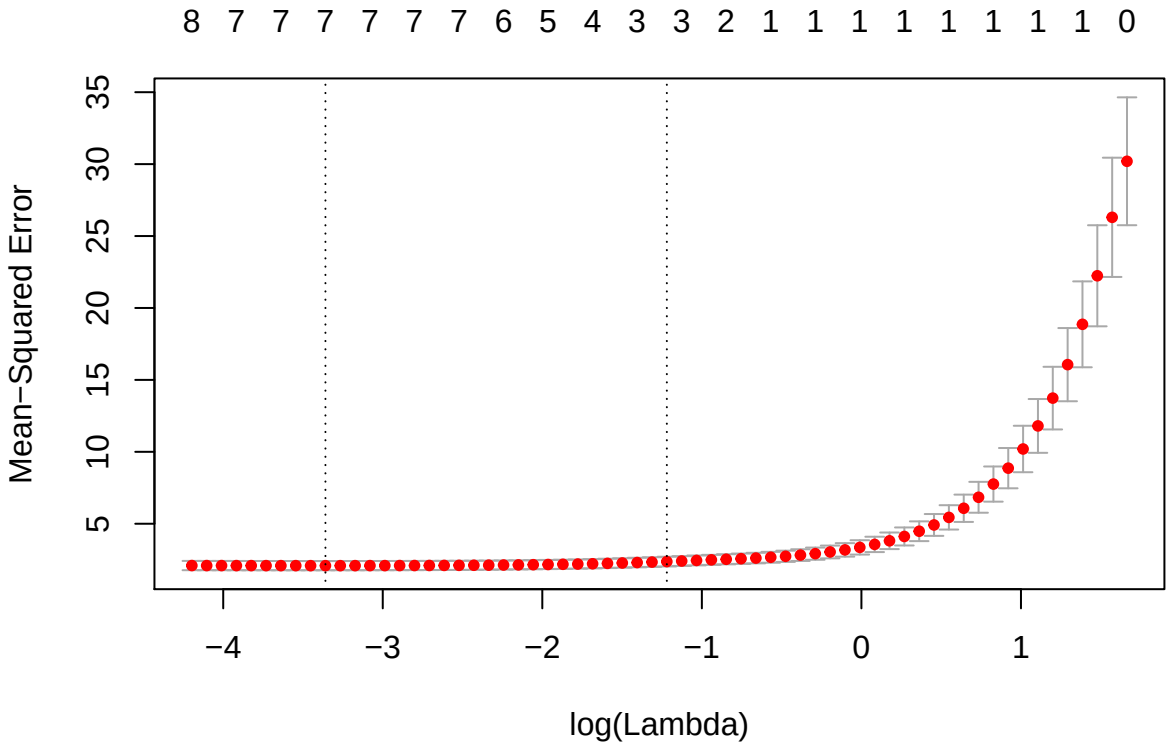


Figure 3 : Determing Tuning Paramter λ via Cross Validation for LASSO.

The above figure shows lambda's relationship with the MSE for this particular dataset. The left most horizontal dotted line shows for what λ values the minimum MSE occurs. The second dotted line on the right is situated one standard error of the minimum λ value. In this case the optimal λ is **0.0347** which corresponds to **-3.36** on the graph above. The numbers located on the top horizontal axis represent the number of predictor variables, which still have non-zero coefficients.

Once obtaining the optimal value for λ, it is now possible to train the LASSO model and determine the 'best fit' coefficients. glment cross validation algorithm selects the larger lambda of **0.2953895** resulting in the below coefficients.

Table4 : LASSO coefficients .

Income	**Age**	**Student**
0.1470	0.007	0.1023

The results from the LASSO model are shown below:

$$MSE(train) = 1.97$$
$$MSE(test) = 1.77$$

Conclusion

As stated in the beginning of the assignment, *mean squared error* MSE would be used as the benchmark for evaluating a model's performance. Throughout this assignment, 4 models were trained, tested using a random 75/25 training to test dataset split. The results are as shown below:

Model	**Train**	**Test**	**Number of Variables**
Multiple Linear Regresion	1.964	1.816	10
Back Selection Linear Regresion	1.973	1.800	6
Subset Selection	1.979	1.799	5
Ridge Regression	2.377	2.205	10
Lasso Regression	1.974	1.776	3

Table5 : Final models and corresponding MSE results.

Based on the information above, the model with the lowest *mean squared error* is the lasso regression model. The final lasso model reduced the predictor variables to a subset of 3, namely Income, Age and Student. The model with the most predictor variables, Ridge Regression, as also the least accurate model, returning the largest *mean squared error*.

Final Formula

$$F(x) = 6.363 + 0.147X1_i + 0.007X2_i + 0.102X3_i$$

PLAGIARISM DECLARATION

1.I know that plagiarism is wrong. Plagiarism is to use another's work and pretend that it is one's own.

2.I have used a generally accepted citation and referencing style. Each contribution to, and quotation in, this tutorial/report/project from the work(s) of other people has been attributed, and has been cited and referenced.

3. This tutorial/report/project is my own work.

4.I have not allowed, and will not allow, anyone to copy my work with the intention of passing it off as his or her own work.

5.I acknowledge that copying someone else's assignment or essay, or part of it, is wrong, and declare that this is my own work.

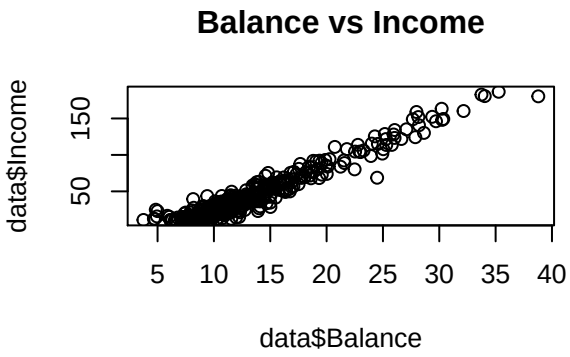
Date: 08/04/2018

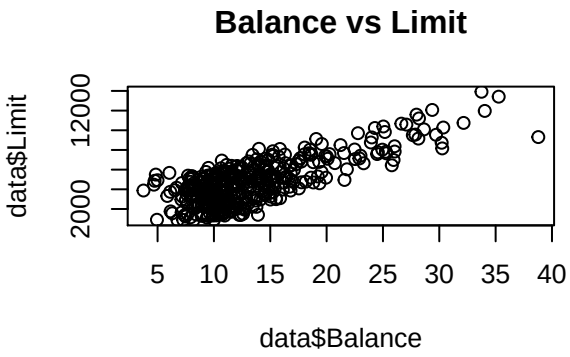
Appendix A

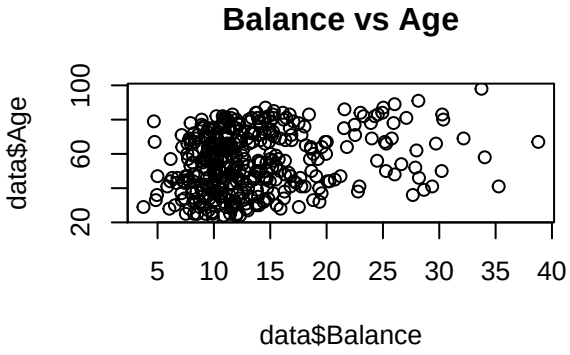
Correlation Matrix

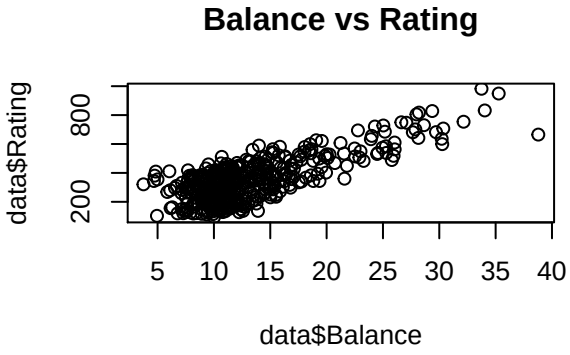
```
## Loading required package: lattice

## Loading required package: survival

## Loading required package: Formula

## Loading required package: ggplot2

##
## Attaching package: 'Hmisc'

## The following objects are masked from 'package:base':
##
##     format.pval, units

##           Balance Income Limit Rating Cards  Age Education
## Balance      1.00   0.97  0.76   0.76 -0.01 0.23      0.01
## Income       0.97   1.00  0.79   0.79 -0.02 0.18     -0.03
## Limit        0.76   0.79  1.00   1.00  0.01 0.10     -0.02
## Rating       0.76   0.79  1.00   1.00  0.05 0.10     -0.03
## Cards       -0.01  -0.02  0.01   0.05  1.00 0.04     -0.05
## Age          0.23   0.18  0.10   0.10  0.04 1.00      0.00
## Education    0.01  -0.03 -0.02  -0.03 -0.05 0.00      1.00
##
## n= 400
##
##
## P
##           Balance Income Limit  Rating Cards  Age    Education
## Balance           0.0000 0.0000 0.0000 0.9034 0.0000 0.9000
## Income    0.0000         0.0000 0.0000 0.7156 0.0004 0.5808
## Limit     0.0000  0.0000        0.0000 0.8384 0.0437 0.6387
## Rating    0.0000  0.0000 0.0000        0.2881 0.0392 0.5479
## Cards     0.9034  0.7156 0.8384 0.2881        0.3916 0.3081
## Age       0.0000  0.0004 0.0437 0.0392 0.3916        0.9425
## Education 0.9000  0.5808 0.6387 0.5479 0.3081 0.9425
```

Balance Plots (linear)









Appendix B

Loading Data

```
rm(list = ls())

library(GGally)
library(ISLR)
library(glmnet)
library(leaps)

#load data
data<-read.table('Assignment_1_Data.txt', h = T)

#summary of the data
summary(data)

#Split up training and test data
train_data = data[1:300,] #75%
test_data = data[-(1:300),] #25%
```

Null Model

```
#NUll model
Null_mean = function(data){
  mse = NULL
  n = length(data$Balance)
  mean_balance = mean(data$Balance)
  for (i in 1:n){
  mse[i] = (data$Balance[i] - mean_balance)^2
  }
  temp = sum(mse)/n
  return(temp)
}
#Null model has a MSE of 32.06
Null_mean(train_data)
Null_mean(test_data)
```

Linear Regression All varaibles

```
#First model with all variables
fit1_train = lm(Balance~.,data=train_data)

#predict of test data
pred_test = predict(fit1_train, test_data)
pred_train = predict(fit1_train, train_data)

#compute mse
mse_test = mean((test_data$Balance-pred_test)^2)
mse_train = mean((train_data$Balance-pred_train)^2)
```

Backwards Selection

```
#removing Gender and Ethnicity
summary(fit1_train)
fit2_train = lm(Balance~.-Gender -Ethnicity,data=train_data)
summary(fit2_train)

#calculat the new MSE
pred_test2 = predict(fit2_train, test_data)
mse_test2 = mean((test_data$Balance-pred_test2)^2)

#correlation matrix show the high correlation between Rating and Limit - can assume Collinearity
cor(data$Limit,data$Rating)
fit3_train = lm(Balance~.-Gender -Ethnicity - Rating - Limit,data=train_data)

#calculat the new MSE
pred_test3 = predict(fit3_train, test_data)
pred_train3 = predict(fit3_train, train_data)

mse_test3 = mean((test_data$Balance-pred_test3)^2)
mse_train3 = mean((train_data$Balance-pred_train3)^2)
```

Subset Selection

```
regfit.full=regsubsets (Balance ~.,data=train_data,nvmax =11)
reg.summary = summary(regfit.full)

#optimal number of variables (5 varibales)
which.min(reg.summary$cp)
which.max(reg.summary$adjr2)
which.min(reg.summary$bic)

#can plot Mallow's CP/BIC/AdjR2 across the all models
par(mfrow =c(1,3))
plot(reg.summary$cp ,xlab=" Number of Variables ",ylab="CP",type = 'b')
points (5, reg.summary$cp[5], col ="red",cex =2, pch =20)
abline(v=5,col='red', lty=2)
plot(reg.summary$adjr2 ,xlab=" Number of Variables ",ylab="AdjR2",type = 'b')
points (5, reg.summary$adjr2[5], col ="red",cex =2, pch =20)
abline(v=5,col='red', lty=2)
plot(reg.summary$bic ,xlab =" Number of Variables ",ylab=" BIC",type="b")
points (5, reg.summary$bic[5], col ="red",cex =2, pch =20)
abline(v=5,col='red', lty=2)

#another plot
plot(regfit.full ,scale ="Cp")
plot(regfit.full ,scale ="adjr2")
plot(regfit.full ,scale ="bic")

coef(regfit.full ,5)

#CROSS VALIDATION
#define prediction function for subsect
predict.regsubsets =function (object ,newdata ,id ,...){
  form=as.formula(object$call [[2]])
  mat=model.matrix (form ,newdata )
  coefi =coef(object ,id=id)
  xvars =names (coefi )
  mat[,xvars ]%*% coefi
}

#Can Also evaluate using Cross Validation
k=10
set.seed (1)
folds=sample(1:k,nrow(train_data),replace =TRUE)

table(folds)
cv.errors =matrix(NA ,k,11)

for(j in 1:k){
  best.fit=regsubsets(Balance~.,data=train_data[folds!=j,],nvmax=11)
  for(i in 1:11){
    pred=predict.regsubsets(best.fit ,train_data[folds ==j,],id=i)
    cv.errors[j,i]= mean((train_data$Balance[folds==j]-pred)^2)
  }
```

```
}

mean.cv.errors =apply(cv.errors ,2, mean)
which.min(mean.cv.errors)

plot(mean.cv.errors ,type='b')
points(5,mean.cv.errors[5], col ='red', pch=20, cex =2)
abline(v=5,col='red', lty=2)

#extract the coefficents
reg.best=regsubsets(Balance~.,data=data , nvmax =11)
coef(reg.best ,5)

# only use teh five variables recommended by the subset selction
sub_fit = lm(Balance~ Income + Age + Education + Student + Married,data=train_data)

#test and train
pred_test_sub = predict(sub_fit, test_data)
pred_train_sub = predict(sub_fit, train_data)

#compute mse
mse_test_sub = mean((test_data$Balance-pred_test_sub)^2)
mse_train_sub = mean((train_data$Balance-pred_train_sub)^2)
```

Shrinkage Methods

Ridge Regression

```
x=model.matrix(Balance~.-1,train_data)
x_test =model.matrix(Balance~.-1,test_data)
y=train_data$Balance
y_test = test_data$Balance

#Ridge: alpha=0 abd LASSO: alpha=1
ridge.mod =glmnet(x,y,alpha=0)

#plot
plot(ridge.mod,xvar = "lambda" ,label = TRUE)

#cross validation
cv.ridge = cv.glmnet(x,y,alpha =0)
plot(cv.ridge)

bestlam=cv.ridge$lambda.min

ridge.test=predict(ridge.mod,s=bestlam,x_test)
ridge.train=predict(ridge.mod,s=bestlam,x)

mse_reg_test = mean((y_test-ridge.test)^2)
mse_reg_train = mean((y-ridge.train)^2)
mse_reg_test
mse_reg_train
```

Lasso

```
x=model.matrix(Balance~.-1,train_data)
x_test =model.matrix(Balance~.-1,test_data)
y=train_data$Balance
y_test = test_data$Balance

lasso.mod =glmnet(x,y,alpha =1)
plot(lasso.mod,xvar = "lambda" ,label = TRUE)

set.seed (1)
cv.out =cv.glmnet(x,y,alpha =1)
plot(cv.out)
xbestlam =cv.out$lambda.min

lasso_train =predict(lasso.mod,s=xbestlam,x)
lasso_test =predict(lasso.mod,s=xbestlam,x_test)

mse_lass_test = mean((y_test-lasso_test)^2)
mse_lass_train = mean((y-lasso_train)^2)
mse_lass_train
mse_lass_test

coef(cv.out)
```

Principle Components

```
library (pls)
set.seed(1)
pcr.fit=pcr(Balance~., data=train_data,scale=TRUE ,validation ="CV")
summary(pcr.fit)

validationplot(pcr.fit ,val.type="MSEP")

pcr.pred_test=predict (pcr.fit ,test_data, ncomp =10)
pcr.pred_train=predict (pcr.fit ,train_data, ncomp =10)

mse_prc_test = mean((test_data$Balance -pcr.pred_test)^2)
mse_prc_train = mean((train_data$Balance -pcr.pred_train)^2)

#fit on to full data
pcr.fit=pcr(y~x,scale =TRUE ,ncomp =10)
pcr.pred=predict (pcr.fit ,data, ncomp =10)
mean((pcr.pred -Balance)^2)

coef(pcr.fit)
```