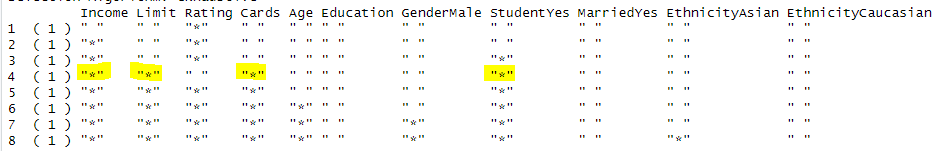
To simplify reading answers, all source code is at the end of the file with commented letter associations

**(a)-----------------------------------------------------------------------------------------------**

The best four variables where Income, Limit, Cards, and Student with coefficients

This is justified by looking at the columns with asterisks (\*) and numbered by 4 variables.



**(b)-----------------------------------------------------------------------------------------------**

**Cp**: Income Limit Rating Cards Age Student

(Intercept) Income Limit Rating Cards StudentYes

-526.1555233 -7.8749239 0.1944093 1.0879014 17.8517307 426.8501456

**BIC**: Income Limit Cards Student

(Intercept) Income Limit Cards StudentYes

-499.7272117 -7.8392288 0.2666445 23.1753794 429.6064203

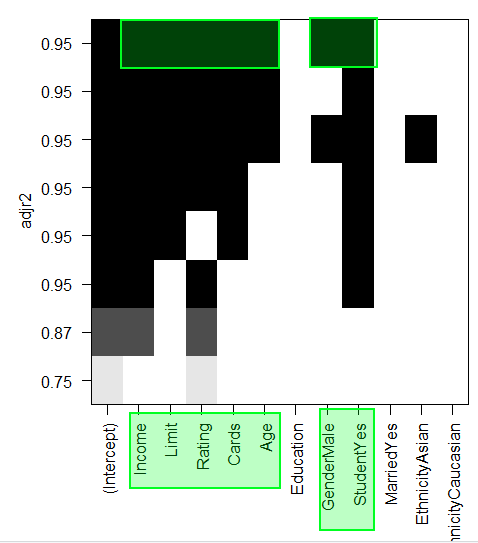
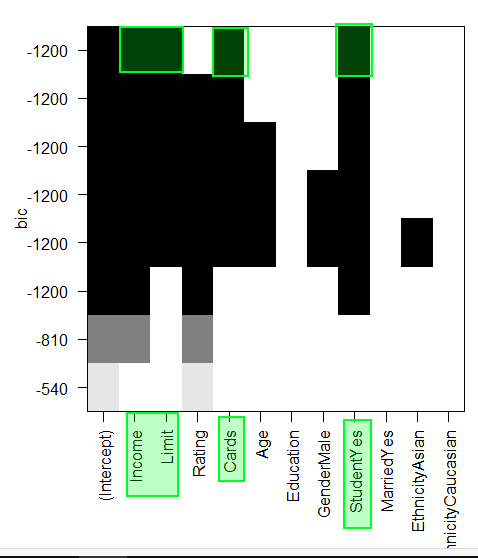
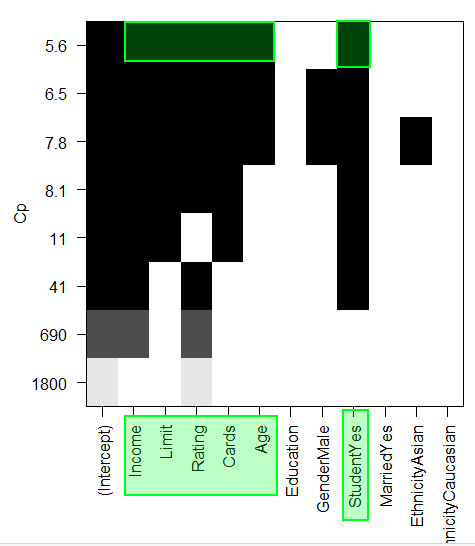
**Adjusted R2** : Income Limit Rating Cards Age Gender Student

(Intercept) Income Limit Rating

-499.0690216 -7.8036338 0.1936237 1.0940490

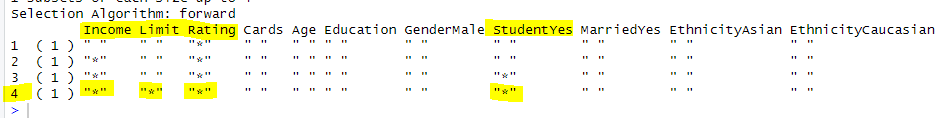
Cards Age GenderMale StudentYes

18.1091708 -0.6206538 10.4531521 426.5812620



**(c)-----------------------------------------------------------------------------------------------**

Best 4 variables (via forward stepping) are Income, Limit, Rating, and Student



**(d)-----------------------------------------------------------------------------------------------**

It is different because "Rating" was forced to stay included due to the stepping process. Whereas with all the possible combinations of size 4, rating had less importance and was not used in the model. The other 3 parameters ended up being the same.

**(e)-----------------------------------------------------------------------------------------------**

**Cp:** Income Limit Rating Cards Age Student

(Intercept) Income Limit Rating Cards Age

-493.7341870 -7.7950824 0.1936914 1.0911874 18.2118976 -0.6240560

StudentYes

425.6099369

**BIC:** Income Limit Rating Cards Student

(Intercept) Income Limit Rating Cards StudentYes

-526.1555233 -7.8749239 0.1944093 1.0879014 17.8517307 426.8501456

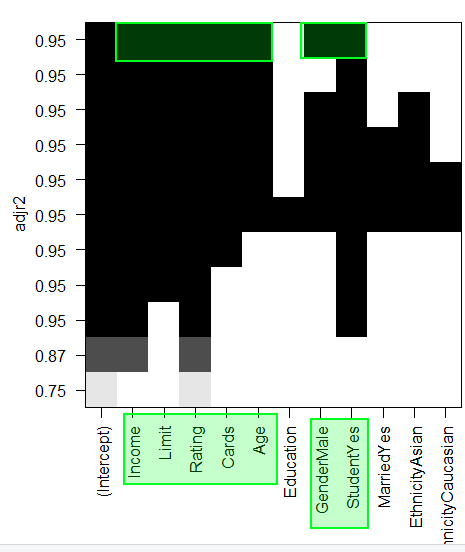
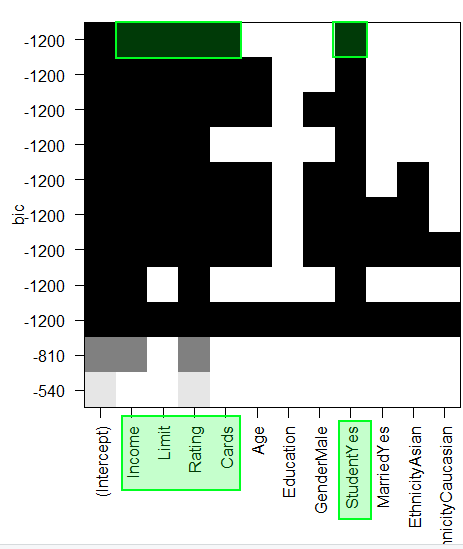
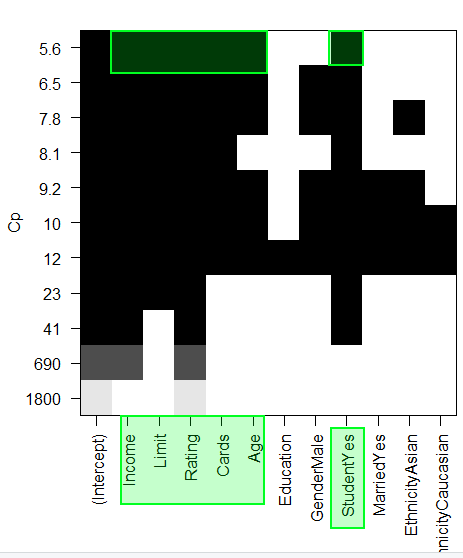
**Adjusted R2:** Income Limit Rating Cards Age Gender Student

(Intercept) Income Limit Rating Cards Age

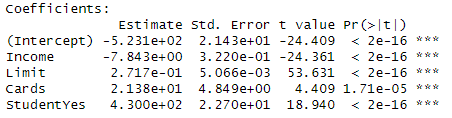
-499.0690216 -7.8036338 0.1936237 1.0940490 18.1091708 -0.6206538

GenderMale StudentYes

10.4531521 426.5812620



**(f)-----------------------------------------------------------------------------------------------**



**(g)-----------------------------------------------------------------------------------------------**

Test MSE = 10,485

**(h)-----------------------------------------------------------------------------------------------**

LOOCV MSE = 10046.43

10-Fold CV MSE: 10056.94

Thus LOOCV had the smallest test MSE for my seed (1). In either case both LOOCV and 10-Fold CV performed better than a 50-50 train test split.

**(i)-----------------------------------------------------------------------------------------------**

**Size MSE**

1 54504.00

2 24333.69

3 12885.25

4 11838.71

5 12595.75

6 12699.41

7 12774.41

8 12654.38

9 12458.07

10 12400.06

11 12380.05

Model size with smallest MSE: 4

**(j)-----------------------------------------------------------------------------------------------**

**Size MSE**

1 54965.475

2 27460.099

3 10489.300

4 9611.533

5 9718.811

6 9466.766

7 9631.280

8 9684.054

9 9740.664

10 9726.611

11 9673.000

Model size with smallest MSE: 6

**(k)-----------------------------------------------------------------------------------------------**

The results in (b) were that the models with only 4 to 7 parameters performed the best on the different error measures. This is similar to the results in (i) and (j) where the best sizes were 4 and 6.

**CODE**

library(readr)

library(leaps)

library(MASS)

library(boot)

# Load Credit dataset

creditData = read.csv("O:/Arr Matey/Credit.csv", header=T)

attach(creditData)

MAX\_PARAMS = 11

**# (a) -----------------------------------------------**

linearRegressionModel = regsubsets(Balance~., data=creditData, nvmax=MAX\_PARAMS)

summary(linearRegressionModel) # Row 4 asterisks: Income, Limit, Cards, Student

coef(linearRegressionModel, 4) # Get model equation's coefficients

**# (b) -----------------------------------------------**

plot(linearRegressionModel, scale="Cp")

plot(linearRegressionModel, scale ="bic")

plot(linearRegressionModel, scale="adjr2")

coef(linearRegressionModel, 5) # Cp [5 predictors]

coef(linearRegressionModel, 4) # BIC [4 predictors]

coef(linearRegressionModel, 7) # adj rsquare [7 predictors]

**# (c) -----------------------------------------------**

forwardSelectionLR = regsubsets(Balance~., data=creditData, nvmax=MAX\_PARAMS, method="forward")

summary(forwardSelectionLR)

**# (e) -----------------------------------------------**

plot(forwardSelectionLR, scale="Cp")

plot(forwardSelectionLR, scale ="bic")

plot(forwardSelectionLR, scale="adjr2")

coef(forwardSelectionLR, 6) # Cp [6 predictors]

coef(forwardSelectionLR, 5) # BIC [5 predictors]

coef(forwardSelectionLR, 7) # adj rsquare [7 predictors]

**# (f) -----------------------------------------------**

set.seed(1)

x\_train = sample(400, 200)

x\_test = -x\_train

split\_lm = lm(Balance~Income+Limit+Cards+Student, data=creditData, subset=x\_train)

summary(split\_lm)

**# (g) -----------------------------------------------**

y\_predictions = predict(split\_lm, newdata=creditData)

all\_residuals = (Balance - y\_predictions)

test\_residuals = all\_residuals[x\_test]

split\_lm\_mse = mean(test\_residuals^2)

split\_lm\_mse

**# (h) -----------------------------------------------**

loocv\_model = glm(Balance~Income+Limit+Cards+Student, data=creditData)

loocvMSE <- cv.glm(creditData, loocv\_model)$delta[2]

loocvMSE

kfold\_model = glm(Balance~Income+Limit+Cards+Student, data=creditData)

kfoldMSE = cv.glm(creditData, kfold\_model, K=10)$delta[2]

kfoldMSE

**# (i) -----------------------------------------------**

x\_train = sample(c(TRUE, FALSE), nrow(creditData), rep=TRUE)

x\_test = !x\_train

bestSubsets = regsubsets(Balance~., data=creditData[x\_train,], nvmax=MAX\_PARAMS)

testMatrix = model.matrix(Balance~., data=creditData[x\_test,])

test\_mses = rep(NA, MAX\_PARAMS)

for (modelSize in 1:MAX\_PARAMS){

coefficients = coef(bestSubsets, modelSize)

y\_predictions = testMatrix[, names(coefficients)]%\*%coefficients

y\_test = Balance[x\_test]

test\_mses[modelSize]= mean((y\_test - y\_predictions)^2)

}

test\_mses # Best MSEs for each size

which.min(test\_mses) # Best MSE's Model Size

**# (j) -----------------------------------------------**

k=10

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

}

folds = sample(1:k, nrow(creditData), replace=TRUE)

test\_mses = matrix(NA, k, MAX\_PARAMS, dimnames = list(NULL, paste(1:MAX\_PARAMS)))

for (currentFold in 1:k) {

bestSubsets = regsubsets(Balance~., data=creditData[folds != currentFold, ], nvmax=MAX\_PARAMS)

for (modelSize in 1:MAX\_PARAMS) {

y\_predictions = predict(bestSubsets, creditData[folds == currentFold, ], id=modelSize)

y\_test = Balance[folds == currentFold]

test\_mse = mean((y\_test-y\_predictions)^2)

test\_mses[currentFold, modelSize] = test\_mse

}

}

mean\_errors = apply(test\_mses, 2, mean)

mean\_errors

which.min(mean\_errors)