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.

```
         Income Limit Rating Cards Age Education GenderMale StudentYes MarriedYes EthnicityAsian EthnicityCaucasian
1  ( 1 ) " "    " "   "*"    " "   " " " "       " "        " "        " "        " "            " "
2  ( 1 ) "*"    " "   "*"    " "   " " " "       " "        " "        " "        " "            " "
3  ( 1 ) "*"    " "   "*"    " "   " " " "       " "        "*"        " "        " "            " "
4  ( 1 ) "*"    "*"   " "    "*"   " " " "       " "        "*"        " "        " "            " "
5  ( 1 ) "*"    "*"   "*"    "*"   " " " "       " "        "*"        " "        " "            " "
6  ( 1 ) "*"    "*"   "*"    "*"   "*" " "       " "        "*"        " "        " "            " "
7  ( 1 ) "*"    "*"   "*"    "*"   "*" " "       "*"        "*"        " "        " "            " "
8  ( 1 ) "*"    "*"   "*"    "*"   "*" " "       "*"        "*"        " "        "*"            " "
```

(b)---

C_p: Income Limit Rating Cards Age Student

```
(Intercept)       Income        Limit       Rating        Cards   StudentYes
-526.1555233   -7.8749239    0.1944093    1.0879014   17.8517307  426.8501456
```

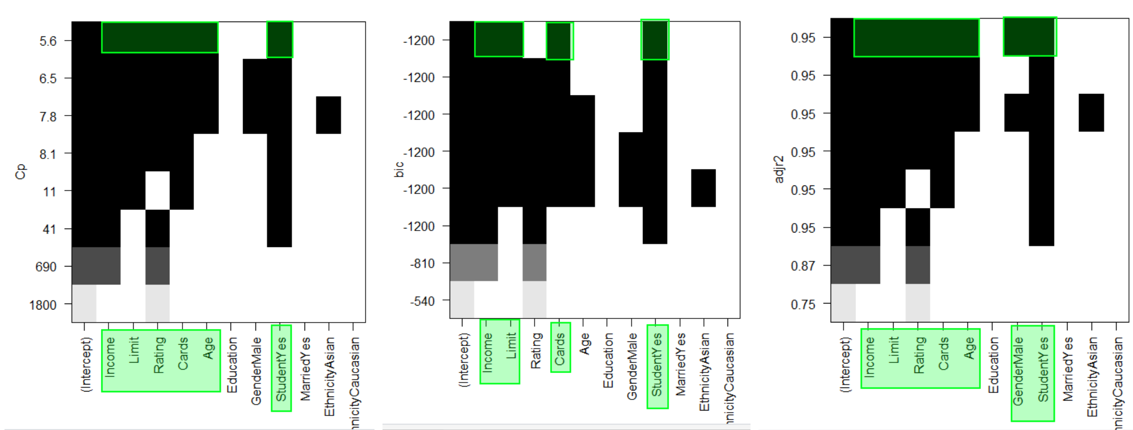
BIC: Income Limit Cards Student

```
(Intercept)       Income        Limit        Cards   StudentYes
-499.7272117   -7.8392288    0.2666445   23.1753794  429.6064203
```

Adjusted R^2 : 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

```
Selection Algorithm: forward
         Income Limit Rating Cards Age Education GenderMale StudentYes MarriedYes EthnicityAsian EthnicityCaucasian
1  ( 1 ) " "    " "   "*"    " "   " " " "       " "        " "        " "        " "            " "
2  ( 1 ) "*"    " "   "*"    " "   " " " "       " "        " "        " "        " "            " "
3  ( 1 ) "*"    " "   "*"    " "   " " " "       " "        "*"        " "        " "            " "
4  ( 1 ) "*"    "*"   "*"    " "   " " " "       " "        "*"        " "        " "            " "
>
```

(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)---

C_p: 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

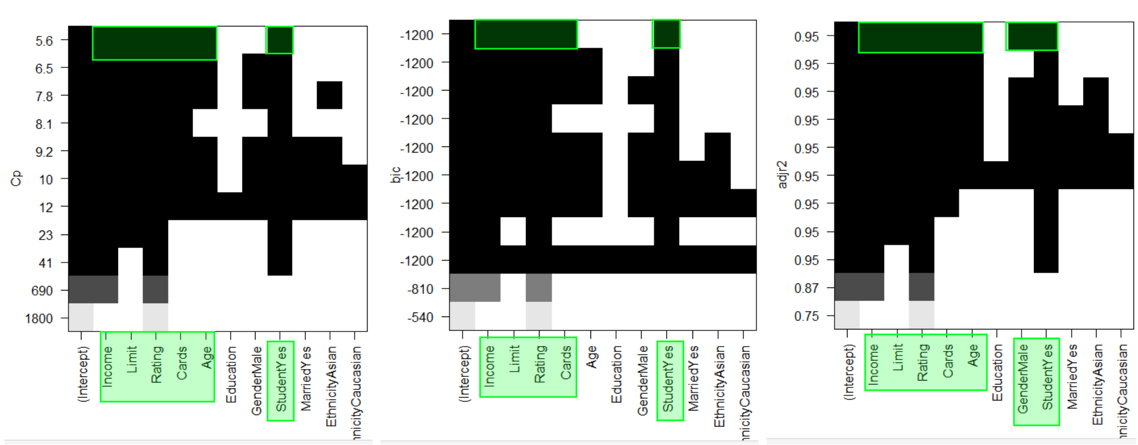
BIC: Income Limit Rating Cards Student

(Intercept)	Income	Limit	Rating	Cards	StudentYes
-526.1555233	-7.8749239	0.1944093	1.0879014	17.8517307	426.8501456

Adjusted R^2: 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)---

```
Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept) -5.231e+02  2.143e+01 -24.409  < 2e-16 ***
Income      -7.843e+00  3.220e-01 -24.361  < 2e-16 ***
Limit        2.717e-01  5.066e-03  53.631  < 2e-16 ***
Cards        2.138e+01  4.849e+00   4.409 1.71e-05 ***
StudentYes   4.300e+02  2.270e+01  18.940  < 2e-16 ***
```

(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)
```