Stat 435 Project 1

Andrews, Clancy (11674964) Yu-Tung, Cheng (11678647)

General guidelines

Please show your work in order to get points. Providing correct answers without supporting details does not receive full credits. This project requires you to synthesize your knowledge on linear models and model selection, apply such knowledge to a dataset, make scientific discoveries, and concisely present your findings. *Please follow the rubrics on projects given in the syllabus.* This project can be done by up to 2 students as a group assignment. Please write each team member's name and student ID on your report, and only one copy of your report needs to be submitted.

You DO NOT have to submit your project report using typesetting software. However, your answers must be legible for grading. Please upload your answers to the course space. Specifically, if you are not able to knit a .Rmd/.rmd file into an output file such as a .pdf, .doc, .docx or .html file that contains your codes, outputs from your codes, your interpretations on the outputs, and your answers in text (possibly with math expressions), you can organize your codes, their outputs and your answers in a document in the format given below:

```
Problem or task or question ...
Codes ...
Outputs ...
Your interpretations ...
```

It is absolutely not OK to just submit your codes only. This will result in a considerable loss of points on your assignments or projects.

Dataset and its description

Please use the data set `Boston` (which is contained in the library `MASS`). This data set contains 506 observations on 14 variables. The response variable is `medv`, and the rest are potential predictors. Namely, we are interested in predicting `medv` using a linear model. Please make sure you fully understand the meaning and type of each variable in the data set.

```
library(MASS)
library(leaps)
library(caTools)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following object is masked from 'package:MASS':
##
##     select

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(glmnet)
```

```
## Loading required package: Matrix

## Loaded glmnet 4.1-6
```

```
library(hdi)
```

```
## Loading required package: scalreg

## Loading required package: lars

## Loaded lars 1.3
```

```
str(Boston)
```

```
## 'data.frame':    506 obs. of  14 variables:
##  $ crim   : num  0.00632 0.02731 0.02729 0.03237 0.06905 ...
##  $ zn     : num  18 0 0 0 0 0 12.5 12.5 12.5 12.5 ...
##  $ indus  : num  2.31 7.07 7.07 2.18 2.18 2.18 7.87 7.87 7.87 7.87 ...
##  $ chas   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ nox    : num  0.538 0.469 0.469 0.458 0.458 0.458 0.524 0.524 0.524 0.524 ...
##  $ rm     : num  6.58 6.42 7.18 7 7.15 ...
##  $ age    : num  65.2 78.9 61.1 45.8 54.2 58.7 66.6 96.1 100 85.9 ...
##  $ dis    : num  4.09 4.97 4.97 6.06 6.06 ...
##  $ rad    : int  1 2 2 3 3 3 5 5 5 5 ...
##  $ tax    : num  296 242 242 222 222 222 311 311 311 311 ...
##  $ ptratio: num  15.3 17.8 17.8 18.7 18.7 18.7 15.2 15.2 15.2 15.2 ...
##  $ black  : num  397 397 393 395 397 ...
##  $ lstat  : num  4.98 9.14 4.03 2.94 5.33 ...
##  $ medv   : num  24 21.6 34.7 33.4 36.2 28.7 22.9 27.1 16.5 18.9 ...
```

```
names(Boston)
```

```
##  [1] "crim"    "zn"      "indus"   "chas"    "nox"     "rm"      "age"
##  [8] "dis"     "rad"     "tax"     "ptratio" "black"   "lstat"   "medv"
```

Tasks

(0) Please use `set.seed(1)` for all operations that involve user-induced randomness (such as the command `sample` for resampling from a data set, `cv.glmnet` for cross-validation on the LASSO method or ridge regression, etc). Otherwise, your results will *not be reproducible* when they are checked and graded.

(1) Please *randomly split (using the `sample` command)* the observations into a training set and a validation set, so that the training set can be used to fit a linear model, and the validation set can be used to evaluate the prediction accuracy of the fitted model. Here you have the freedom on splitting. But please be careful with the number of observations for each set, since a training set with a few observations cannot produce a relatively good fitted model. Caution: these 2 sets should be non-intersecting; sample from the row indices but do not sample with replacement when creating the two sets.

```
# Splitting the data into training and testing sets.  Split
# ratio is 75% training data, 25% testing data
set.seed(1)
split = sample.split(Boston, SplitRatio = 0.75)
train = subset(Boston, split == T)
test = subset(Boston, split == F)
```

(2) Apply best subset selection on all potential predictors without interactions between them, report the best model and its fitted model, perform model diagnostics on the model, conduct hypothesis tests on some coefficients of the model and report your findings, and assess the prediction accuracy of the fitted model and report your findings. Note: you can use the mean squared error to measure prediction accuracy.

```
# Applying best subset selection on all predictors with
# response variable being medv
regfit.full = regsubsets(medv ~ ., data = train, nvmax = 13)
# Summary of Best subset selection output
reg.summary = summary(regfit.full)
reg.summary
```

```
## Subset selection object
## Call: regsubsets.formula(medv ~ ., data = train, nvmax = 13)
## 13 Variables  (and intercept)
```

```
##         Forced in Forced out
## crim        FALSE      FALSE
## zn          FALSE      FALSE
## indus       FALSE      FALSE
## chas        FALSE      FALSE
## nox         FALSE      FALSE
## rm          FALSE      FALSE
## age         FALSE      FALSE
## dis         FALSE      FALSE
## rad         FALSE      FALSE
## tax         FALSE      FALSE
## ptratio     FALSE      FALSE
## black       FALSE      FALSE
## lstat       FALSE      FALSE
## 1 subsets of each size up to 13
## Selection Algorithm: exhaustive
##           crim zn  indus chas nox rm  age dis rad tax ptratio black lstat
## 1  ( 1 )  " "  " " " "   " "  " " " " " " " " " " " "     " "   "*"
## 2  ( 1 )  " "  " " " "   " "  " " "*" " " " " " " " "     " "   "*"
## 3  ( 1 )  " "  " " " "   " "  " " "*" " " " " " " " "*"     " "   "*"
## 4  ( 1 )  " "  " " " "   " "  " " "*" " " " " " " " "*"     "*"   "*"
## 5  ( 1 )  " "  " " " "   " "  "*" "*" " " "*" " " " " " "*"     " "   "*"
## 6  ( 1 )  " "  " " " "   " "  "*" "*" " " "*" " " " " " "*"     "*"   "*"
## 7  ( 1 )  " "  " " " "   "*"  "*" "*" " " "*" " " " " " "*"     "*"   "*"
## 8  ( 1 )  "*"  " " " "   " "  "*" "*" " " "*" "*" " " " "*"     "*"   "*"
## 9  ( 1 )  "*"  " " " "   " "  "*" "*" " " "*" "*" "*" "*"     "*"   "*"
## 10  ( 1 ) "*"  "*" " "   " "  "*" "*" " " "*" "*" "*" "*"     "*"   "*"
## 11  ( 1 ) "*"  "*" " "   "*"  "*" "*" " " "*" "*" "*" "*"     "*"   "*"
## 12  ( 1 ) "*"  "*" " "   "*"  "*" "*" "*" "*" "*" "*" "*"     "*"   "*"
## 13  ( 1 ) "*"  "*" "*"   "*"  "*" "*" "*" "*" "*" "*" "*"     "*"   "*"
```

```
# Best Models according to RSS, adjusted R^2, CP, and BIC
best.models = data.frame(adj.rsq = which.max(reg.summary$adjr2),
    cp = which.min(reg.summary$cp), bic = which.min(reg.summary$bic))
best.models
```

```
##   adj.rsq cp bic
## 1      11 11   9
```

```
# Best Models
model.11 = lm(medv ~ . - indus - age, data = train)
model.9 = lm(medv ~ . - indus - age - chas - zn, data = train)
```

Based on the output above, there are several different models that are considered the best according to different criteria. The model with 11 predictors was considered the best by Adjusted R^2 and Mallow's CP. The fitted model with 11 predictors is $y = 36.38 - 0.13crim + 0.04zn + 2.12chas - 17.08nox + 3.79rm - 1.43dis + 0.33rad - 0.01tax - ptratio + 0.01black - 0.52lstat$. The model with 11

predictors was considered by Bayesian Information Criterion. The fitted model with 9 predictors is $y = 37.02 - 0.12crim - 17.38nox + 4.01rm - 1.20dis + 0.33rad - 0.01tax - 1.15ptratio + 0.01black - 0.51lstat$.

```
summary(model.11)
```

```
##
## Call:
## lm(formula = medv ~ . - indus - age, data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -15.0512  -2.7301  -0.5551   1.6957  26.1629
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  36.379987   6.194958   5.873 9.98e-09 ***
## crim         -0.125500   0.039492  -3.178 0.001616 **
## zn            0.039300   0.016668   2.358 0.018933 *
## chas          2.115983   1.060515   1.995 0.046791 *
## nox         -17.082397   4.438620  -3.849 0.000141 ***
## rm            3.785004   0.482988   7.837 5.61e-14 ***
## dis          -1.431436   0.224609  -6.373 5.85e-10 ***
## rad           0.325287   0.076288   4.264 2.59e-05 ***
## tax          -0.012763   0.004091  -3.119 0.001962 **
## ptratio      -0.998475   0.157696  -6.332 7.44e-10 ***
## black         0.011952   0.003268   3.658 0.000294 ***
## lstat        -0.515091   0.056792  -9.070  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.876 on 350 degrees of freedom
## Multiple R-squared:  0.7394, Adjusted R-squared:  0.7312
## F-statistic: 90.29 on 11 and 350 DF,  p-value: < 2.2e-16
```

Based on the outputs for the model with 11 predictors, we have a p-value of $4.0581165 \times 10^{-95}$ and an Adjusted R^2 of 0.7312323.

```
summary(model.9)
```

```
##
## Call:
## lm(formula = medv ~ . - indus - age - chas - zn, data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
```

```
## -13.5624  -2.8771  -0.7848   1.8097  26.9025
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  37.023008   6.241050   5.932 7.15e-09 ***
## crim         -0.122283   0.039705  -3.080 0.002234 **
## nox         -17.375893   4.432555  -3.920 0.000106 ***
## rm            4.013386   0.481960   8.327 1.86e-15 ***
## dis          -1.203836   0.201501  -5.974 5.66e-09 ***
## rad           0.325992   0.076896   4.239 2.87e-05 ***
## tax          -0.011802   0.004063  -2.905 0.003903 **
## ptratio      -1.150877   0.148187  -7.766 8.91e-14 ***
## black         0.012381   0.003297   3.755 0.000202 ***
## lstat        -0.512335   0.057372  -8.930  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.926 on 352 degrees of freedom
## Multiple R-squared:  0.7325, Adjusted R-squared:  0.7256
## F-statistic: 107.1 on 9 and 352 DF,  p-value: < 2.2e-16
```

Based on the outputs for the model with 9 predictors, we have a p-value of $3.6135918 \times 10^{-95}$ and an Adjusted R^2 of 0.7256386. To test the best model, we will go with the Adjusted R^2 criteria, which means we will use the model with 11 predictors. The p-values for the coefficients in the model with 11 predictors are shown below:

```
summary(model.11)$coefficients[, 4]
```

```
##  (Intercept)         crim           zn         chas          nox           rm
## 9.981834e-09 1.615616e-03 1.893319e-02 4.679139e-02 1.411961e-04 5.611841e-14
##          dis          rad          tax      ptratio        black        lstat
## 5.846462e-10 2.586650e-05 1.962329e-03 7.440368e-10 2.935428e-04 8.727749e-18
```

```
# Validating fitted model on test set
pred = predict(model.11, newdata = test)
# Calculating Accuracy
MSE = mean((test$medv - pred)^2)
# Printing MSE
print(MSE)
```

```
## [1] 19.63469
```

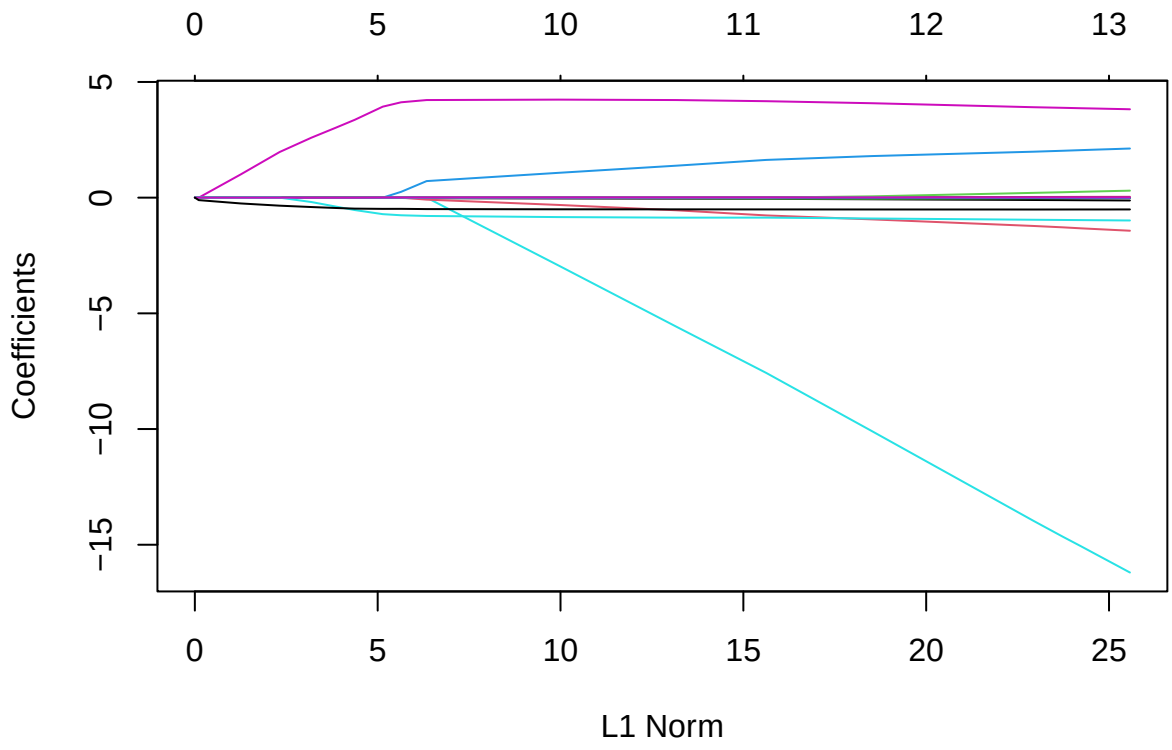
As observed in the output, we get a mean squared error of 19.6346946.

(3) Implement LASSO (with cross-validation to select the optimal tuning parameter) on all potential predictors without interactions between them, report the best model (that is based on the optimal tuning parameter) and its fitted model, conduct hypothesis tests on some coefficients of the model and report your findings, and assess the prediction accuracy of the fitted model and report your findings.

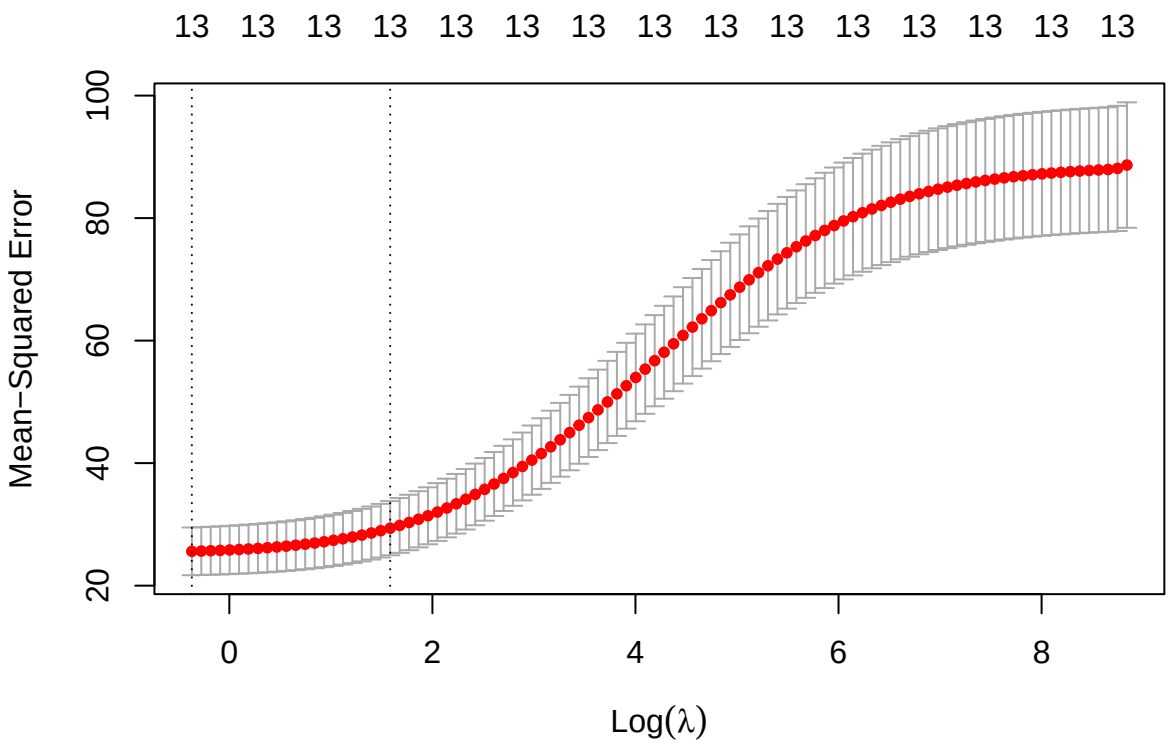
```
set.seed(1)
grid = 10^seq(10, -2, length = 100)

# Splitting off predictor and response data
x_train = select(train, -medv)
y_train = select(train, medv)
x_test = select(test, -medv)
y_test = select(test, medv)

lasso.mod = glmnet(x_train, y_train$medv, alpha = 1, lambda = grid)
plot(lasso.mod)
```




```
# Cross Validation on training data
set.seed(1)
cv.out = cv.glmnet(as.matrix(x_train), y_train$medv, alpha = 0)
plot(cv.out)
```




```
# Lambda that minimizes testing MSE
bestlam = cv.out$lambda.min

best.model = glmnet(x_train, y_train$medv, alpha = 1, lambda = bestlam)
coef(best.model)
```

```
## 14 x 1 sparse Matrix of class "dgCMatrix"
##                       s0
## (Intercept) 14.27107185
## crim        -0.01269111
## zn           .
## indus        .
## chas         0.13923967
## nox          .
## rm           4.09386829
## age          .
## dis          .
## rad          .
## tax          .
## ptratio     -0.75454577
## black        0.00754422
## lstat       -0.48389493
```

The best lambda value that minimizes the testing MSE is 0.6911461. When using the best lambda in our Lasso model, we get the best model (coefficients above):

$$y = 14.27 - 0.01crim + 0.14chas + 4.09rm - 0.75ptratio + 0.01black - 0.48lstat.$$

```
lasso_pred = predict(lasso.mod, s = bestlam, newx = as.matrix(x_test))

out = glmnet(x_test, y_test$medv, alpha = 1, lambda = grid)

# Fit lasso model on full dataset
lasso_coef = predict(out, type = "coefficients", s = bestlam)[1:14,
    ]

# Display coefficients using lambda chosen by CV
lasso_coef
```

```
## (Intercept)        crim          zn       indus        chas         nox
##  10.0510915   0.0000000   0.0000000   0.0000000   3.0232982   0.0000000
##          rm         age         dis         rad         tax     ptratio
##   4.4129545   0.0000000   0.0000000   0.0000000   0.0000000  -0.4957849
##       black       lstat
##   0.0000000  -0.5169286
```

Above we can see the Lasso coefficients.

```
# Display only non-zero coefficients
lasso_coef[lasso_coef != 0]
```

```
## (Intercept)        chas          rm     ptratio       lstat
##  10.0510915   3.0232982   4.4129545  -0.4957849  -0.5169286
```

```
# interpretation: Know that we can see the relationship
# between log lambda and the MSE, we are going to find the
# best lambda that reduces the error the most for the lasso
# model and test it on the hold-out data.
```

Above we can see the Lasso coefficients that are non-zero. The p-values for the coefficients in the Lasso model are shown below:

```
hypo = lasso.proj(x_train, y_train$medv, family = "gaussian",
    standardize = TRUE, multiplecorr.method = "none")
```

```
## Nodewise regressions will be computed as no argument Z was provided.

## You can store Z to avoid the majority of the computation next time around.

## Z only depends on the design matrix x.
```

```r
hypo$pval[which(hypo$pval < 0.1)]
```

```
##         crim           zn         chas          nox           rm          dis
## 7.587716e-03 6.012321e-02 4.455740e-02 1.814982e-03 9.658552e-15 9.634643e-09
##          rad          tax      ptratio        black        lstat
## 1.911008e-04 2.928560e-03 3.798754e-09 3.284483e-04 3.077902e-16
```

```r
# Calculate test MSE
Lasso.MSE = mean((lasso_pred - y_test$medv)^2)
```

Based on the testing data set used in Lasso, we have a following mean squared error of 23.8854301.

(5) Among the best/optimal models you would find in (2) and (3) respectively, which one has the best prediction accuracy? If you consider a trade-off between the number of predictors in a model and its prediction accuracy, which among the best models you found in (2) and (3) would you prefer?

According to the analysis conducted, best subset selection had a smaller mean squared error of 19.6346946 while Lasso had a mean squared error of 23.8854301. However, given the quantity of predictors in Lasso versus best subset, and the small difference in mean squared error between those models, we would prefer to use the Lasso model for future data prediction.

```r
# License Information
sessionInfo()
```

```
## R version 4.2.1 (2022-06-23 ucrt)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 22621)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=English_United States.utf8
## [2] LC_CTYPE=English_United States.utf8
## [3] LC_MONETARY=English_United States.utf8
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United States.utf8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] hdi_0.1-9      scalreg_1.0.1  lars_1.3       glmnet_4.1-6   Matrix_1.5-3
##  [6] dplyr_1.0.9    caTools_1.18.2 leaps_3.1      MASS_7.3-58.1  knitr_1.40
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_1.0.9       highr_0.9        pillar_1.8.1     compiler_4.2.1
##  [5] formatR_1.12     bitops_1.0-7     iterators_1.0.14 tools_4.2.1
##  [9] digest_0.6.29    evaluate_0.16    lifecycle_1.0.3  tibble_3.1.8
## [13] lattice_0.20-45  pkgconfig_2.0.3  rlang_1.0.6      foreach_1.5.2
## [17] cli_3.3.0        DBI_1.1.3        rstudioapi_0.14  parallel_4.2.1
## [21] yaml_2.3.5       xfun_0.32        fastmap_1.1.0    withr_2.5.0
## [25] stringr_1.4.1    generics_0.1.3   vctrs_0.5.0      grid_4.2.1
## [29] tidyselect_1.2.0 glue_1.6.2       R6_2.5.1         fansi_1.0.3
## [33] survival_3.3-1   rmarkdown_2.16   magrittr_2.0.3   splines_4.2.1
## [37] codetools_0.2-18 htmltools_0.5.3  assertthat_0.2.1 lpSolve_5.6.17
## [41] shape_1.4.6      linprog_0.9-4    utf8_1.2.2       stringi_1.7.8
```