Introduction

The project mainly focused on training the dataset into various regression models. The dataset is describing the features and price of second-hand cars.
In this dataset, the sample size (n) of the dataset is 1000. My Response variable (Y) is the current price of the second-hand cars and there are 11 predictor variables (X), which includes:
X_1: v.id
X_2: on road old
X_3: on road now
X_4: years
X_5: km
X_6: rating
X_7: condition
X_8: economy
X_9: top speed
X_{10}: hp
X_{11}: torque

Since X_1 is the indicator (id) of different cars, I will not use this feature to train the following models. Besides, the whole dataset does not have any missing value.

Methodology

I mainly use RStudio to assist with the progress of mini project. I will try to train and fit three regression, like linear regression model, ridge regression model and lasso regression model by the information of the second-hand dataset. I will compare the performance of these three models by using MSE, R^2 and RMSE.

Discussion

Firstly, the csv data file will be input to RStudio. The First 6 rows of the dataset is shown in the below figure by head() function and we can see the data type of every variables by using str() function. It shows that the data type of all of the variables are integer expect the dependent variable, which is numeric. Then, I just drop the column of "v.id" which is only an indicator for various second-hand car. Also, I will set up the option of controlling the number of digits to show when printing numeric values.

```
> head(data)
  v.id on.road.old on.road.now years     km rating condition economy top.speed hp torque current.price
1    1      535651      798186     3  78945      1         2      14       177 73    123      351318.0
2    2      591911      861056     6 117220      5         9       9       148 74     95      285001.5
3    3      686990      770762     2 132538      2         8      15       181 53     97      215386.0
4    4      573999      722381     4 101065      4         3      11       197 54    116      244295.5
5    5      691388      811335     6  61559      3         9      12       160 53    105      531114.5
6    6      650007      844846     6 148846      2         9      13       138 61    109      177933.5
```

```
> str(data)
'data.frame':   1000 obs. of  12 variables:
 $ v.id         : int  1 2 3 4 5 6 7 8 9 10 ...
 $ on.road.old  : int  535651 591911 686990 573999 691388 650007 633344 662990 543184 573043 ...
 $ on.road.now  : int  798186 861056 770762 722381 811335 844846 756063 891569 841354 879481 ...
 $ years        : int  3 6 2 4 6 6 5 6 7 2 ...
 $ km           : int  78945 117220 132538 101065 61559 148846 78025 76546 57662 132347 ...
 $ rating       : int  1 5 2 4 3 2 1 1 4 2 ...
 $ condition    : int  2 9 8 3 9 9 9 2 7 3 ...
 $ economy      : int  14 9 15 11 12 13 15 12 14 12 ...
 $ top.speed    : int  177 148 181 197 160 138 171 146 151 200 ...
 $ hp           : int  73 74 53 54 53 61 94 109 50 115 ...
 $ torque       : int  123 95 97 116 105 109 132 96 132 82 ...
 $ current.price: num  351318 285002 215386 244296 531114 ...
```

Processing

In the preprocessing part, I will drop off the "v.id" column as this is only an indicator for various samples, which are the id of different second-hand cars. And then the dataset separated into a dataset of response variable and a dataset of predictor variables.

I will do scaling before using the dataset. By normalizing the dataset, it can ensure the parameters to keep on similar scales and it can improve the speed of computation.

In data partitioning part, I split the dataset into training dataset and testing dataset in order to test the performance of the fitted model. I will randomly split it by proportion. The training dataset comprised of 70 percent of the data which has 700 rows in the dataset and the testing dataset contained the remaining 30 percent. The dimension of training and testing dataset is shown in below figure. The number of rows and numbers of columns of the training dataset with attributes are 700 and 10. The numbers of columns of the testing dataset with attributes are 300 and 10.

```
> dim(x_train)
[1] 700  10
> dim(x_test)
[1] 300  10
```

Training model with ten predictor variables (Linear Regression)

We will train the training data of both features and dependent variables in a linear regression model and print a summary below. We can see that the features of "on.road.old", "on.road.now", "years", "km" and "condition" are statistically significant as the p-value of these five features are smaller than 0.001.

```
> summary(LReg)

Call:
lm(formula = y_train ~ ., data = train)

Residuals:
                  Min                   1Q               Median                   3Q                  Max
-0.100084546531188923 -0.057233340518444364 -0.015533648579776948  0.041928694363330708  0.168082912906809379

Coefficients:
                         Estimate              Std. Error               t value Pr(>|t|)
(Intercept) -0.00046893443171429323  0.00258078272663647690   -0.18170000000000000  0.85587
on.road.old  0.23602109675756821616  0.00260320587212961498   90.66554999999999609  < 2e-16 ***
on.road.now  0.22522628751701029293  0.00260120643561094966   86.58531999999999584  < 2e-16 ***
years       -0.02232716736124953916  0.00256374557636257884   -8.70880999999999972  < 2e-16 ***
km          -0.92494939503021944560  0.00258207881889491436 -358.21888000000001284  < 2e-16 ***
rating       0.00250345948192919141  0.00262572724832478956    0.95343000000000000  0.34070
condition    0.10131551416147525913  0.00259709693275873106   39.01106000000000051  < 2e-16 ***
economy      0.00285136505047608015  0.00262308843260174521    1.08702999999999994  0.27741
top.speed   -0.00122150455280546481  0.00259115636980894589   -0.47141000000000000  0.63750
hp           0.00348115292924665656  0.00261127733668564330    1.33312000000000008  0.18293
torque      -0.00244084539580943999  0.00262258238540511845   -0.93069999999999997  0.35233
---
Signif. codes:
0 '***' 0.00100000000000000000208 '**' 0.01000000000000000208 '*' 0.05000000000000002776 '.' 0.10000000000000000555 ' ' 1

Residual standard error: 0.068118104732573292459 on 689 degrees of freedom
Multiple R-squared:  0.99539327568531411,       Adjusted R-squared:  0.99532641466478167
F-statistic: 14887.497494929517 on 10 and 689 DF,  p-value: < 2.22044604925031e-16
```

After training the model, I use the testing data of features to do prediction by the fitted model. We can check the performance by MSE first. We can see the MSE of this fitted linear regression model is 0.0052248137161932202074. We can also check the model performance by calculating the R^2 and RMSE. We can check these two by inputting the true and predicted values by created function. The values of RMSE and R^2 are 0.072282872910484269324 and 0.99485548813426405435 in the figure below.

```
> mse_lr<-mean((y_test - predict)^2)
> mse_lr
[1] 0.0052248137161932202074
> eval_results(y_test, predict, x_test)
                        RMSE                 Rsquare
1 0.072282872910484269324 0.99485548813426405435
```

<u>Training model with ten predictor variables (Ridge Regression)</u>

The ridge regression model will add up with the penalized term lambda. The ridge regression model will implement the effect of L-2 term to reduce the overfitting problem.

I will train the same data in the ridge regression model this time. I will find the optimal cross-validated lambda, which is equal to 0.093226462414983016225 by using 5-fold cross-validation method. See the figure below. Then, we can predict the response variable by fitted model with the optimal lambda.

```
> optimal_lambda
[1] 0.00100000000000000000208
```

By comparing with the true values and the predicted values, the MSE of the fitted ridge regression model is 0.0052206500019575628746. Also, we can see that the RMSE and R^2 are equal to 0.072254065643101106353 and 0.99485958785503003643. The testing scores are shown in the below figure.

```
> mse_r
[1] 0.005220650001957562874 6
> eval_results(y_test, predictions_r, x_test)
                       RMSE                 Rsquare
1 0.07225406564310110635 3 0.9948595878550300364 3
```

Training model with ten predictor variables (Lasso Regression)

In lasso regression model, we also want to see the coefficient of the features when lambda equals the optimal lambda. Then, we will know about that the importance of various features to prevent the problem of overfitting too. However, lasso is usually used to train the model when the n is much smaller than p. In this case, the number of predictors is smaller than the number of samples, therefore using the ridge regression model may better than using lasso regression model.

We also try to train the data in the lasso regression. After using the 5-fold cross-validation method to find the tuning parameter, it is equal to 0.002913994302466957342 4 and shown in below figure. Then, I will use this lambda to train the model.

```
> best_lambda
[1] 0.002913994302466957342 4
```

I predict the testing data by the fitted lasso regression model with the best lambda. We can see that the performance with MSE is 0.005276938294916340595 2. Besides, I can check the values of RMSE and R^2, which are equal to 0.072642537778606963705 and 0.994804164675035851 32.

```
> mse_l
[1] 0.005276938294916340595 2
> eval_results(y_test, predictions_l, x_test)
                       RMSE                 Rsquare
1 0.072642537778606963705 0.994804164675035851 32
```

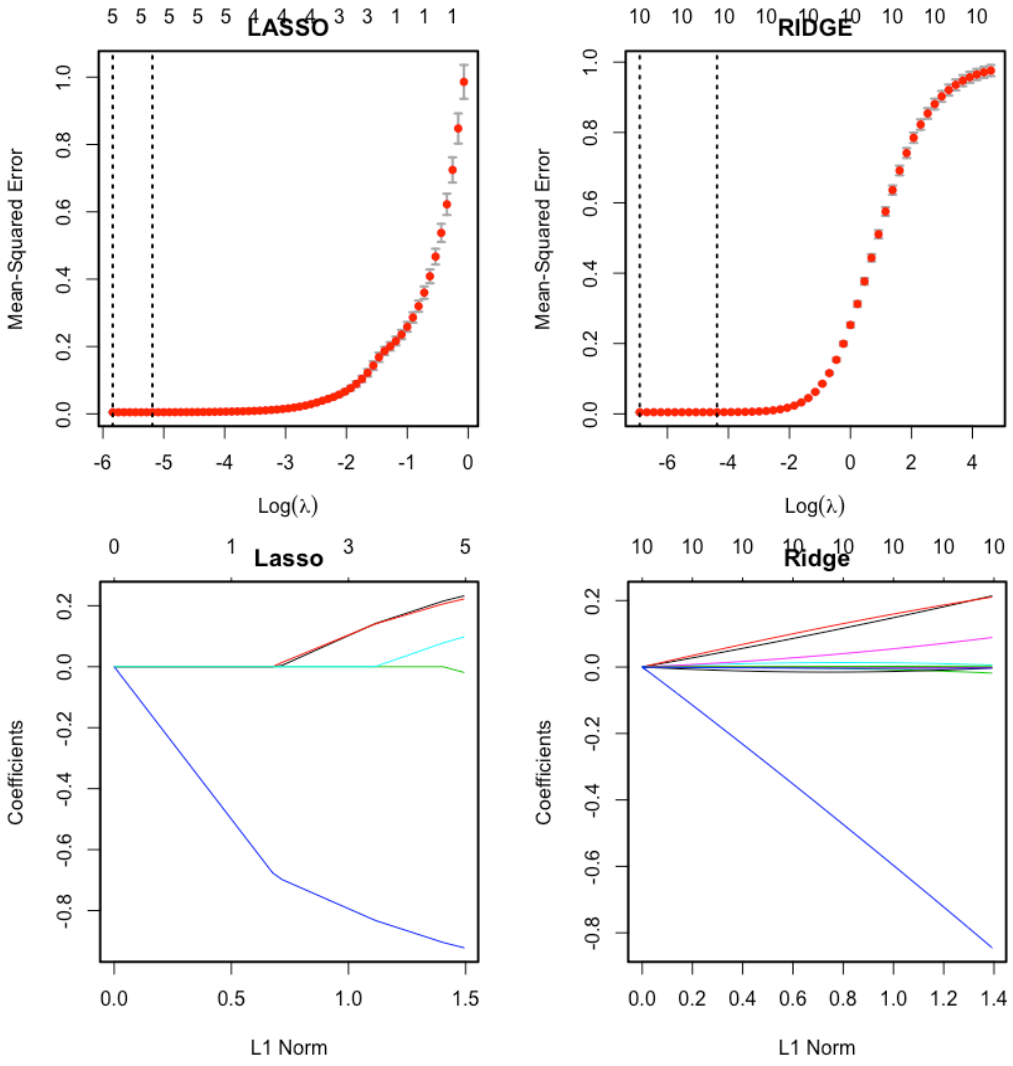
Comparison and Performance Evaluation

By comparing with three regression model, we can see that the results of three types of testing the performance and the ridge regression model has the best performance in these three models with the three score. The MSE and RMSE are the lowest in value and the R^2 is the highest one in these three regression models. However, the difference between the values of ridge and linear regression model are not significant. This may show that the data is already well fit in the linear regression model.

We also see that the when the coefficient become zero, the coefficients of lasso will become zero at different time in the plot when changing lambda. Besides, the coefficients of ridge regression model will equal zero at the same time.

Model	MSE	RMSE	R^2
Linear Regressi	0.005224813716193220 2074	0.07228287291048426 9324	0.994855488134264 05435

Ridge Regressi	0.005220650001957562 8746	0.07225406564310110 6353	0.9948595878550300 3643
Lasso Regressi	0.005276938294916340 5952	0.07264253777860696 3705	0.9948041646750358 5132



Conclusion

After dropping the X_1 predictor variable, we will train the three different models: linear regression model, ridge regression model and lasso regression. By comparing these three testing score, which are MSE, RMSE and R^2, it is known that the ridge regression model has the best performance and fitted regression model is

$$Y = \beta_0 + \beta_2 X_2 + \beta_3 X_3 + \beta_4 X_4 + \beta_5 X_5 + \beta_6 X_6 + \beta_7 X_7 + \beta_8 X_8 + \beta_9 X_9 + \beta_{10} X_{10} + \beta_{11} X_{11}$$

$$= -0.0004592008706536653689 4 + 0.2357632302935906420948 7X_2 + 0.2250601920841422964869 6X_3 - 0.0222751636683948366080 0X_4 - 0.9240124954496915510304 7X_5 + 0.0025471930332355206110 1X_6 + 0.1011664117635824727825 5X_7 + 0.0027824320272762338365 8X_8 - 0.0012220892084639975710 3X_9 + 0.0034753100849788205392 6X_{10} - 0.0024484651461513292497 5X_{11}$$

Appendix

```
options(digits = 20)
#Importing dataset
data = read.csv('train.csv')
names(data)
head(data)
str(data)
new_d = data%>%select(-v.id)
x = data.matrix(new_d[,1:10])
y = new_d[c(11)]

x = scale(x, center = TRUE, scale = TRUE)
y = scale(y, center = TRUE, scale = TRUE)


# Create the training data and test data
set.seed(100)
index = sample(1:nrow(x), 0.7*nrow(x))
x_train = x[index,]
x_test = x[-index,]

dim(x_train)
dim(x_test)

y_train = y[index,]
y_test = y[-index,]

#Training Linear Regression
train<-data.frame(y_train,x_train)
LReg<-lm(y_train~.,data =train)
summary(LReg)

#Prediction
xtest<-data.frame(x_test)
predict<-predict(LReg,xtest)

#MSE of Linear Regresssion Model
mse_lr<-mean((y_test - predict)^2)
mse_lr


#Create function for computing R squre and RMSE
Results <- function(true, predicted, df) {
  SSE <- sum((predicted - true)^2)
  SST <- sum((true - mean(true))^2)
  R_square <- 1 - SSE / SST
  RMSE = sqrt(SSE/nrow(df))

  data.frame(
    RMSE = RMSE,
    Rsquare = R_square
  )

}

eval_results(y_test, predict, x_test)


# RIDGE

library('glmnet')
ridge_reg <- glmnet(x_train, y_train, alpha = 0, lambda=10^seq(2, -3, by = -.1))
summary(ridge_reg)
cv.RIDGE = cv.glmnet(x_train,y_train,nfolds = 5, alpha=0,lambda=10^seq(2, -3, by = -.1))
optimal_lambda <- cv.RIDGE$lambda.min
optimal_lambda

predictions_r <- predict(ridge_reg, s = optimal_lambda, newx = x_test)
mse_r<-mean((y_test - predictions_r)^2)
mse_r
eval_results(y_test, predictions_r, x_test)

#LASSO
lasso_reg <- glmnet(x_train, y_train, alpha = 1, lambda=10^seq(2, -3, by = -.1))
cv.LASSO = cv.glmnet(x_train,y_train,nfolds = 5, alpha=1)
best_lambda <- cv.LASSO$lambda.min
best_lambda

predictions_l <- predict(lasso_reg, s = best_lambda, newx = x_test)
mse_l<-mean((y_test - predictions_l)^2)
mse_l
eval_results(y_test, predictions_l, x_test)
```

```r
#plot
par(mfrow=c(1,2))
plot(cv.LASSO)
title('LASSO')

plot(cv.RIDGE)
title('RIDGE')


outRIDGE = glmnet(x_train, y_train, alpha=0)
outLasso = glmnet(x_train, y_train)

out = glmnet(x_train, y_train,alpha = 0)
ridge.coef<-predict(ridge_reg, type = "coefficients", s = optimal_lambda)[1:11,]
ridge.coef

par(mfrow=c(1,2))
plot(outLasso)
title('Lasso')

plot(outRIDGE)
title('Ridge')
```