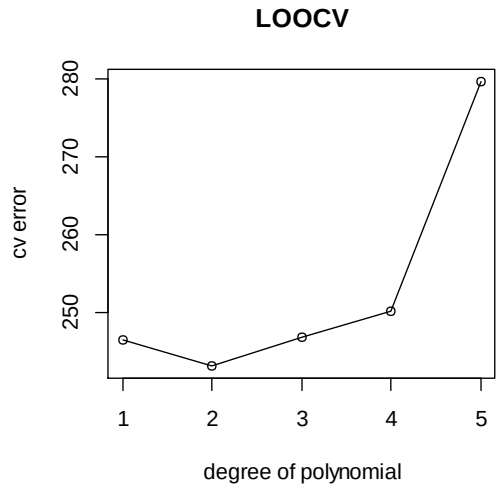
Problem 1: Speed and Stopping Distances of Cars

1. Using LOOCV, the plot of CV errors vs degree of polynomial is:

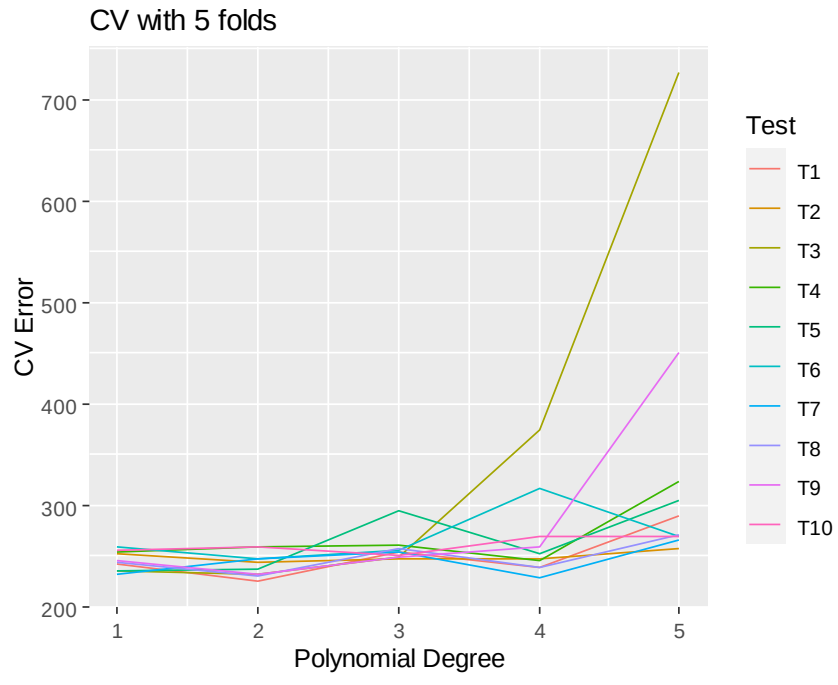


Code and result:

```
> cars=read.csv("cars.csv", header=TRUE,sep=",")
> library(boot)
> cv.error=rep(0,5)
> for (i in 1:5){
+   glm.fit=glm(dist~poly(speed,i),data=cars)
+   cv.error[i]=cv.glm(cars,glm.fit)$delta[1]}
> cv.error
[1] 246.4054 243.0292 246.8288 250.0914 279.6864
> plot(c(1:5),cv.error,type='o',xlab='degree of polynomial',ylab='cv error',main='LOOC
V')
```

We can see that the CV error of polynomial of degree 1 to 4 did not vary much but would increase when the degree reach 5. From the figures obtained, we can conclude the best model should be of degree 2.

2. Repeating 5-fold CV 10 times, the plot of CV errors vs degree of polynomial is:



Code and result:

```
> cv.error.mat=matrix(rep(0,50),5,10)
> for (i in 1:10){
+   for (j in 1:5){
+     glm.fit=glm(dist~poly(speed,j),data=cars)
+     cv.error.mat[j,i]=cv.glm(cars,glm.fit,K=5)$delta[1]
+   }
+ }
> df = data.frame(c(1:5),cv.error.mat)
> colnames(df) = c('Degree','T1','T2','T3','T4','T5','T6','T7','T8','T9','T10')
> df
  Degree       T1       T2       T3       T4       T5       T6       T7       T8       T9      T10
1      1 241.5463 252.5046 235.2885 254.2131 234.3958 259.3760 231.5003 243.5648 245.4029 256.2548
2      2 224.8081 243.2954 231.3836 259.1587 237.1943 247.6273 247.1858 229.7523 231.0094 259.4201
3      3 253.3828 246.4893 248.6640 261.0355 294.4739 256.1543 253.1166 257.2795 249.4252 251.0051
4      4 237.8517 247.5014 374.7347 244.5767 251.5076 316.4570 229.2805 238.7376 258.9115 269.2481
5      5 289.5313 256.7126 726.8828 323.5288 304.5082 269.7823 265.0906 270.3370 450.0374 269.2875
> rowMeans(df[,c(2:11)])
[1] 245.4047 241.0835 257.1026 266.8807 342.5698
> library(ggplot2)
> library(reshape2)
> df <- melt(df,id.vars = 'Degree', variable.name = 'Test')
> p = ggplot(df, aes(Degree,value)) + geom_line(aes(colour = Test))
> p + ggtitle("CV with 5 folds") + xlab("Polynomial Degree") + ylab("CV Error")
```

The CV errors for different testing are different as the elements in each folds are randomly drawn. The average CV errors for polynomial of degree 2 is the lowest and we can conclude that it is the best.

3. Both LOOCV and 5-fold CV yield the same conclusion in this dataset. In general, LOOCV is approximately unbiased and have a high variance while K-fold CV would have a higher bias and a lower variance. However, since the sample size is small in this problem, the variance of 5-fold CV is also high. LOOCV should be a better approach for this problem.

Problem 2: Titanic – Survival or Not

1. Result:

Predictor	Estimated coefficient	95% confidence interval
Sex/male	-2.701736	[-3.092228,-2.327693]
Pclass/3rd	-2.196367	[-2.768914,-1.631383]

Code:

```
titanic=read.csv("titanic.csv", header=TRUE,sep=",")
summary(titanic)
library(mice)
# perform mice imputation, based on predictive mean matching
t = mice(titanic,maxit=10,meth='pmm',)
titanic.data = complete(t,1)
titanic.data$Pclass = as.character(titanic.data$Pclass)
titanic.model = glm(Survived~Pclass+Sex+Age+SibSp+Fare,data=titanic.data,family=binomial)
summary(titanic.model)
confint(titanic.model,c('Sexmale','Pclass3'),level=0.95)
```

Output:

```
> titanic.model = glm(Survived~Pclass+Sex+Age+SibSp+Fare,data=titanic.data,family=binomial)
> summary(titanic.model)

Call:
glm(formula = Survived ~ Pclass + Sex + Age + SibSp + Fare, family = binomial,
    data = titanic.data)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.6945  -0.6112  -0.4051   0.6143   2.4960

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  3.689085   0.429120   8.597  < 2e-16 ***
Pclass2     -1.035632   0.291719  -3.550 0.000385 ***
Pclass3     -2.196367   0.289504  -7.587 3.28e-14 ***
Sexmale     -2.701736   0.194807 -13.869  < 2e-16 ***
Age         -0.034822   0.007042  -4.945 7.63e-07 ***
SibSp       -0.404496   0.105900  -3.820 0.000134 ***
Fare         0.002563   0.002340   1.095 0.273523
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1186.66  on 890  degrees of freedom
Residual deviance:  790.65  on 884  degrees of freedom
AIC: 804.65

Number of Fisher Scoring iterations: 5

> confint(titanic.model,c('Sexmale','Pclass3'),level=0.95)
Waiting for profiling to be done...
             2.5 %    97.5 %
Sexmale -3.092228 -2.327693
Pclass3 -2.768914 -1.631383
```

2. Result of applying Bootstrap with 1000 repetitions:

Predictor	95% confidence interval
Sex/male	[-3.131,-2.356]
Pclass/3rd	[-2.818,-1.628]

Code and Output:

```
> boot.fn=function(data,index){
+   return(coef(glm(Survived~Pclass+Sex+Age+SibSp+Fare,data=data,family=binomial,su
bset=index)))}
> library(boot)
> boot.obj = boot(titanic.data ,boot.fn ,1000)
> boot.ci(boot.obj,conf=0.95,type='perc',index=3)  #for Pclass3
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 1000 bootstrap replicates

CALL :
boot.ci(boot.out = boot.obj, conf = 0.95, type = "perc", index = 3)

Intervals :
Level     Percentile
95%   (-2.818, -1.628 )
Calculations and Intervals on Original Scale
> boot.ci(boot.obj,conf=0.95,type='perc',index=4)  #for Sexmale
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 1000 bootstrap replicates

CALL :
boot.ci(boot.out = boot.obj, conf = 0.95, type = "perc", index = 4)

Intervals :
Level     Percentile
95%   (-3.131, -2.356 )
Calculations and Intervals on Original Scale
```

The confidence interval obtained from Bootstrap for the coefficients of both Sex/male and Pclass/3rd are close to that obtained from part 1. It suggested that the coefficients distribution of both predictors are approximate normal.

3. It is observed from above that “Fare” is not statistically significant, the model is fitted again with logistic regression using all variables except “Fare”, “PassengerId”, “Name”, “Ticket” and “Cabin”. The result is as follows:

```
> titanic.model_1 = glm(Survived~.-Name-PassengerId-Ticket-Cabin-Fare,data=titanic.data,family=binomial)
> summary(titanic.model_1)

Call:
glm(formula = Survived ~ . - Name - PassengerId - Ticket - Cabin -
    Fare, family = binomial, data = titanic.data)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.6363  -0.6035  -0.3784   0.6264   2.4823

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  16.810228 608.267299   0.028 0.977952
Pclass2      -1.109570   0.273940  -4.050 5.11e-05 ***
Pclass3      -2.419372   0.261375  -9.256  < 2e-16 ***
Sexmale      -2.700902   0.202479 -13.339  < 2e-16 ***
Age          -0.043327   0.007395  -5.859 4.66e-09 ***
SibSp        -0.385564   0.111411  -3.461 0.000539 ***
Parch        -0.060166   0.116939  -0.515 0.606896
EmbarkedC   -12.297350 608.267235  -0.020 0.983870
EmbarkedQ   -12.486235 608.267293  -0.021 0.983623
EmbarkedS   -12.774171 608.267222  -0.021 0.983245
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1186.66  on 890  degrees of freedom
Residual deviance:  774.42  on 881  degrees of freedom
AIC: 794.42

Number of Fisher Scoring iterations: 13
```

The result shows that only the ticket class, sex, age and number of siblings/spouses aboard are statistically significant. From the estimate of the coefficient, we can conclude that:

i. Passengers with ticket class 2 and 3 have negative impact to survival in which the extent of ticket class 3 is larger;
ii. Male passengers has higher chance to die;
iii. The older the passengers, the chance of survival is less;
iv. The number of siblings/spouse aboard has negative impact to survival.

```
> titanic.prob = predict(titanic.model_1,titanic.data,type='response')
> titanic.pred = rep(1,nrow(titanic.data))
> titanic.pred = rep(0,nrow(titanic.data))
> titanic.pred[titanic.prob>.5]=1
> table(titanic.pred,titanic.data$Survived)

titanic.pred   0   1
           0 476  96
           1  73 246
```

From the confusion matrix above, the model correctly predict (476+246)/(476+96+73+246)=81% of passenger whether they were died or survived.

Problem 3: Predicting First-Year College Students' GPA

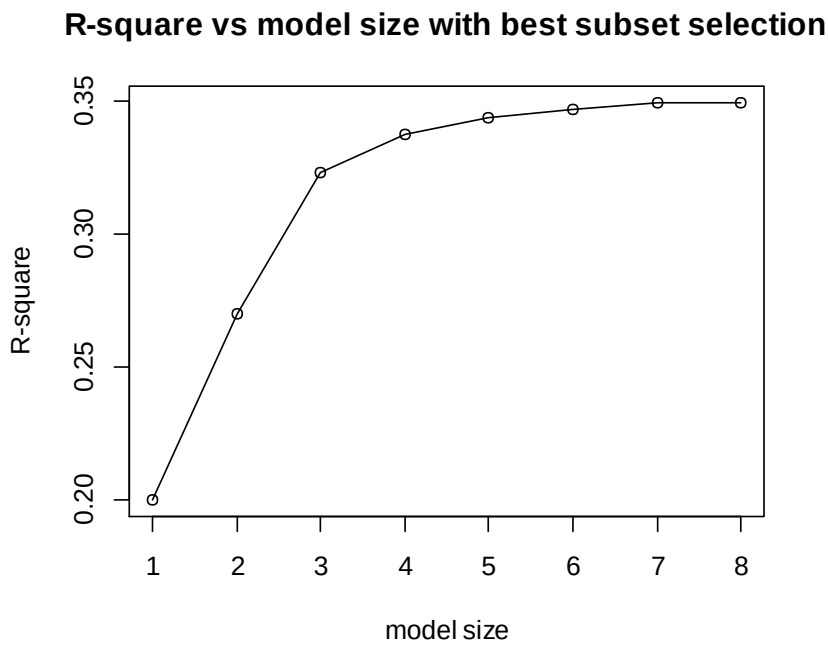
1. The 8 best linear models by best subset selection are illustrated below:

```
Selection Algorithm: exhaustive
         HSGPA SATV SATM Male HU  SS  FirstGen White CollegeBound
1  ( 1 ) "*"   " "  " "  " "  " " " " " "      " "   " "
2  ( 1 ) "*"   " "  " "  " "  "*" " " " "      " "   " "
3  ( 1 ) "*"   " "  " "  " "  "*" " " " "      "*"   " "
4  ( 1 ) "*"   "*"  " "  " "  "*" " " " "      "*"   " "
5  ( 1 ) "*"   "*"  " "  " "  "*" "*" " "      "*"   " "
6  ( 1 ) "*"   "*"  " "  "*"  "*" "*" " "      "*"   " "
7  ( 1 ) "*"   "*"  " "  "*"  "*" "*" "*"      "*"   " "
8  ( 1 ) "*"   "*"  " "  "*"  "*" "*" "*"      "*"   "*"
```

The best model using adjusted R-square is of size 6, in which the predictors include: HSGPA, SATV, Male, HU, SS and White.

Code:

```
> gpa = read.csv("FirstYearGPA.csv", header=TRUE,sep=",")
> library(leaps)
> regfit.full=regsubsets(GPA~.,data=gpa)
> reg.summary=summary(regfit.full)
> which.max(reg.summary$adjr2)
[1] 6
```



2. Repeating 5-fold CV 10 times, the best model is of size 4, in which the predictors include: HSGPA, SATV, HU, and White.

Code and output:

```
> library(magrittr)
> library(purrr)
> get_model_formula = function(id, object, response){
+   models = summary(object)$which[id,-1]
+   predictors = names(which(models == TRUE))
+   predictors = paste(predictors, collapse = "+")
+   as.formula(paste0(response, "~", predictors))
+ }
> get_cv_error = function(model.formula, data){
+   glm.fit = glm(model.formula,data=data)
+   cv.glm(data,glm.fit,K=5)$delta[1]
+ }
> model.ids = c(1:8)
> cv.errors = matrix(nrow=10,ncol=8,dimnames = list(paste("Test",1:10),paste("Size",1:8)))
> for (i in 1:10){
+   cv.errors[i,] =  map(model.ids, get_model_formula, regfit.full, "GPA") %>%
+     map(get_cv_error, data = gpa) %>%
+     unlist()
+ }
> cv.errors.avg = colMeans(cv.errors)
> cv.errors.avg
   Size 1    Size 2    Size 3    Size 4    Size 5    Size 6    Size 7    Size 8
0.1763319 0.1627130 0.1520591 0.1502044 0.1520018 0.1502901 0.1538532 0.1561210
> names(which.min(cv.errors.avg))
[1] "Size 4"
```

CV errors table:

```
> cv.errors
           Size 1    Size 2    Size 3    Size 4    Size 5    Size 6    Size 7    Size 8
Test 1  0.1767299 0.1643802 0.1511110 0.1503885 0.1535756 0.1500860 0.1555502 0.1559147
Test 2  0.1760187 0.1622581 0.1506310 0.1504829 0.1529727 0.1519817 0.1526908 0.1548226
Test 3  0.1739297 0.1660655 0.1522235 0.1534815 0.1499882 0.1470181 0.1559913 0.1611396
Test 4  0.1800583 0.1628768 0.1511246 0.1497746 0.1501075 0.1512890 0.1562442 0.1565792
Test 5  0.1755917 0.1623437 0.1537934 0.1493142 0.1530524 0.1494561 0.1523725 0.1508612
Test 6  0.1749986 0.1606842 0.1504741 0.1492894 0.1552596 0.1535237 0.1559711 0.1561115
Test 7  0.1792965 0.1613535 0.1518487 0.1498829 0.1524050 0.1489386 0.1491997 0.1585619
Test 8  0.1757185 0.1607180 0.1496797 0.1477635 0.1499776 0.1498727 0.1476628 0.1620407
Test 9  0.1761327 0.1647102 0.1541602 0.1517347 0.1483765 0.1486540 0.1544185 0.1490388
Test 10 0.1748443 0.1617399 0.1555448 0.1499320 0.1543028 0.1520815 0.1584313 0.1561392
```

Summary of the model:

```
> summary(glm(get_model_formula(4,regfit.full,'GPA'),data=gpa))

Call:
glm(formula = get_model_formula(4, regfit.full, "GPA"), data = gpa)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-1.06370  -0.26286   0.02436   0.27338   0.87190

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 0.6409767  0.2787933   2.299  0.02246 *
HSGPA       0.4761952  0.0710947   6.698 1.83e-10 ***
SATV        0.0007372  0.0003417   2.157  0.03209 *
HU          0.0150566  0.0036383   4.138 5.03e-05 ***
White       0.2121164  0.0686196   3.091  0.00226 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for gaussian family taken to be 0.1462258)

    Null deviance: 47.234  on 218  degrees of freedom
Residual deviance: 31.292  on 214  degrees of freedom
AIC: 207.39

Number of Fisher Scoring iterations: 2
```

3. The 8 best linear models by forward stepwise selection are illustrated below:

```
Selection Algorithm: forward
          HSGPA SATV SATM Male HU  SS  FirstGen White CollegeBound
1  ( 1 ) "*"   " "  " "  " "  " " " " " "       " "   " "
2  ( 1 ) "*"   " "  " "  " "  "*" " " " "       " "   " "
3  ( 1 ) "*"   " "  " "  " "  "*" " " " "       "*"   " "
4  ( 1 ) "*"   "*"  " "  " "  "*" " " " "       "*"   " "
5  ( 1 ) "*"   "*"  " "  " "  "*" "*" " "       "*"   " "
6  ( 1 ) "*"   "*"  " "  "*"  "*" "*" " "       "*"   " "
7  ( 1 ) "*"   "*"  " "  "*"  "*" "*" "*"       "*"   " "
8  ( 1 ) "*"   "*"  " "  "*"  "*" "*" "*"       "*"   "*"
```

The best model using BIC is of size 3, in which the predictors include: HSGPA, HU and White.

Code:

```
> regfit.fwd=regsubsets(GPA~.,data=gpa,nvmax=8, method ="forward")
> reg.fwd.summary = summary(regfit.fwd)
> plot(c(1:8),summary(regfit.fwd)$adjr2,
+      main='Adjusted R-square vs model size with forward stepwise selection',xlab='model size',
+      ylab='R-square',type='o')
> which.min(reg.fwd.summary$bic)
[1] 3
```

4. Repeating 5-fold CV 10 times, the best model is of size 4, in which the predictors include: HSGPA, SATV, HU, and White.

   Code and output:

```
> model.ids = c(1:8)
> cv.fwd.errors = matrix(nrow=10,ncol=8,dimnames = list(paste("Test",1:10),paste("Size",1:8)))
> for (i in 1:10){
+   cv.fwd.errors[i,] =  map(model.ids, get_model_formula, regfit.fwd, "GPA") %>%
+     map(get_cv_error, data = gpa) %>%
+     unlist()
+ }
> cv.fwd.errors.avg = colMeans(cv.fwd.errors)
> cv.fwd.errors.avg
   Size 1    Size 2    Size 3    Size 4    Size 5    Size 6    Size 7    Size 8
0.1759450 0.1628610 0.1524762 0.1505745 0.1509413 0.1510919 0.1533589 0.1564127
> names(which.min(cv.fwd.errors.avg))
[1] "Size 4"
```

   CV errors table:

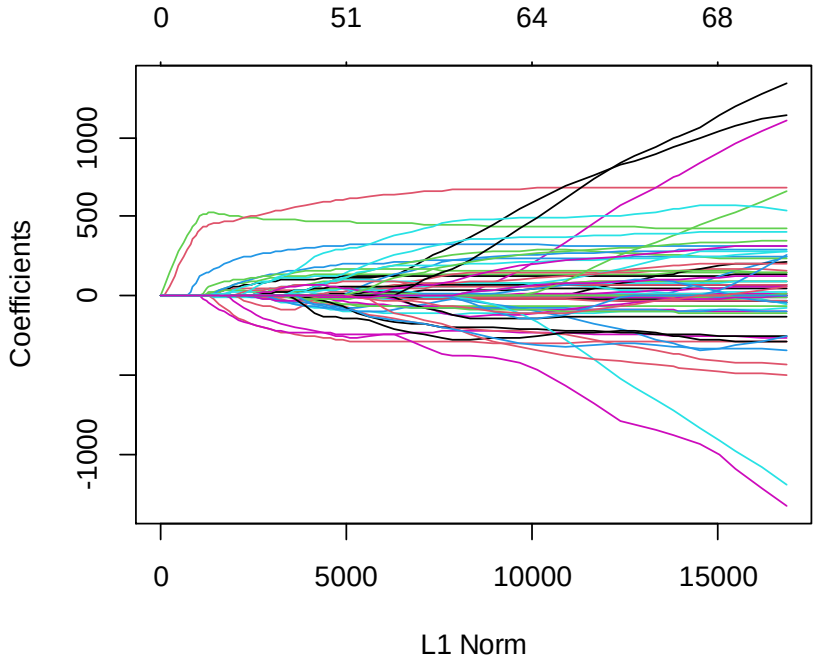
```
> cv.fwd.errors
           Size 1    Size 2    Size 3    Size 4    Size 5    Size 6    Size 7    Size 8
Test 1  0.1749228 0.1674991 0.1529714 0.1563430 0.1472763 0.1498790 0.1541489 0.1563532
Test 2  0.1779181 0.1612231 0.1491836 0.1518615 0.1544380 0.1502752 0.1567923 0.1572314
Test 3  0.1761455 0.1607983 0.1566875 0.1535964 0.1522228 0.1495392 0.1585732 0.1573119
Test 4  0.1781330 0.1613865 0.1538115 0.1466898 0.1536137 0.1499726 0.1466636 0.1559744
Test 5  0.1763464 0.1654573 0.1498047 0.1496789 0.1505917 0.1500794 0.1564524 0.1615178
Test 6  0.1769970 0.1620781 0.1544800 0.1490404 0.1528696 0.1548070 0.1563885 0.1502225
Test 7  0.1747304 0.1634060 0.1514460 0.1511307 0.1459786 0.1548054 0.1530988 0.1591494
Test 8  0.1760507 0.1604518 0.1510767 0.1485297 0.1567989 0.1479553 0.1479979 0.1537253
Test 9  0.1743112 0.1616967 0.1534987 0.1500308 0.1464720 0.1506619 0.1513111 0.1586291
Test 10 0.1738945 0.1646130 0.1518019 0.1488436 0.1491510 0.1529435 0.1521620 0.1540124
```

   Summary of the model:

```
> summary(glm(get_model_formula(4,regfit.fwd,'GPA'),data=gpa))

Call:
glm(formula = get_model_formula(4, regfit.fwd, "GPA"), data = gpa)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-1.06370  -0.26286   0.02436   0.27338   0.87190

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 0.6409767  0.2787933   2.299  0.02246 *
HSGPA       0.4761952  0.0710947   6.698 1.83e-10 ***
SATV        0.0007372  0.0003417   2.157  0.03209 *
HU          0.0150566  0.0036383   4.138 5.03e-05 ***
White       0.2121164  0.0686196   3.091  0.00226 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for gaussian family taken to be 0.1462258)

    Null deviance: 47.234  on 218  degrees of freedom
Residual deviance: 31.292  on 214  degrees of freedom
AIC: 207.39

Number of Fisher Scoring iterations: 2
```

Problem 4: Prediction of the Progression of Diabetes

1. Coefficients vs L1 norm:



Code:

```
> q4train=read.csv("diabetes_train.csv", header=TRUE,sep=",")
> q4test=read.csv("diabetes_test.csv", header=TRUE,sep=",")
> xtrain=model.matrix(Y~.,q4train)[,-1]
> ytrain=q4train$Y
> xtest=model.matrix(Y~.,q4test)[,-1]
> ytest=q4test$Y
> library(glmnet)
> grid=10^seq(4,-2,length=100)
> lasso.mod=glmnet(xtrain,ytrain,alpha=1,lambda=grid)
> plot(lasso.mod)
```

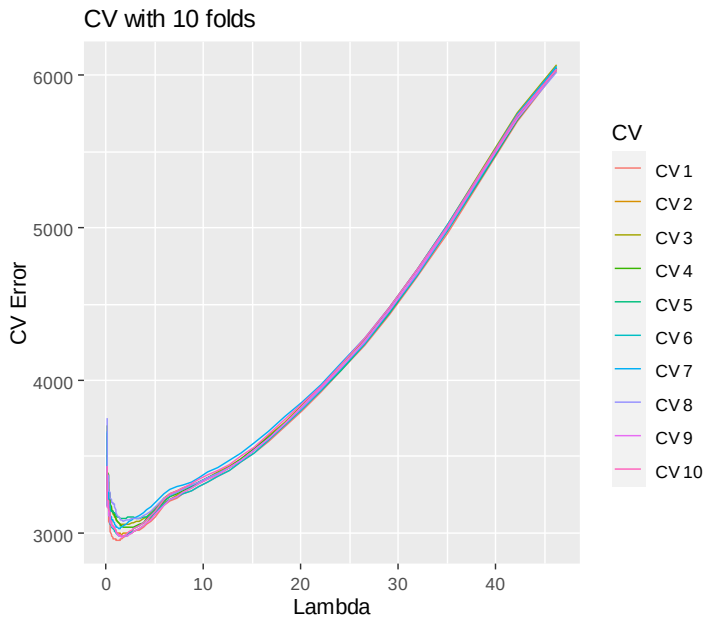
From the above plot of coefficient vs L1 norm, we can see that when L1 norm is 0, all the coefficients are 0 also which corresponds to an empty model. This is due to a large lambda is set such that the minimum of the loss function is achieved when all coefficients are 0. As the L1 norm getting larger and larger, more and more betas become non-zero which is equivalent to saying that more and more variables enter the model. One interesting thing is that once the variable has entered the model, it will not leave when L1 norm is getting larger. We can also see that most of the coefficient will get either larger (for positive ones) or smaller (for negative ones) until a threshold is reached. However, there is one obvious exception (the first variable entering the model showing in the green line) that the coefficient first increase with L1 norm but later decrease until a threshold is reached. This phenomenon may be due to its correlation with other variables.

2. Repeating 10-fold CV 10 times, the best lambda value is 1.624036 which yield the lowest average CV error. There are 33 variables included in the model with the best lambda, including:

```
> names(lasso.best$beta[,1][lasso.best$beta[,1]!=0])
 [1] "sex"     "bmi"     "map"     "tc"      "hdl"     "ltg"     "glu"     "sex.1"   "bmi.1"
[10] "map.1"   "tc.1"    "hdl.1"   "ltg.1"   "glu.1"   "age.2"   "bmi.2"   "ltg.2"   "glu.2"
[19] "age.sex" "age.ldl" "age.hdl" "age.ltg" "sex.bmi" "sex.ldl" "bmi.map" "bmi.tc"  "bmi.hdl"
[28] "bmi.ltg" "map.hdl" "tc.hdl"  "tc.tch"  "ldl.ltg" "tch.glu"
```

Code and plot:

```
> cv.out = cv.glmnet(xtrain,ytrain,alpha=1,K=10)
> lambdas = cv.out$lambda
> cv.lasso.errors = matrix(nrow=10,ncol=100,dimnames = list(paste("CV",1:10),lambdas))
> lambda_min = matrix(nrow=10,ncol=1)
> for (i in 1:10){
+    cv.out = cv.glmnet(xtrain,ytrain,alpha=1,K=10)
+     cv.lasso.errors[i,] = cv.out$cvm
+     lambda_min[i] = cv.out$lambda.min
+ }
> cv.lasso.errors.avg = colMeans(cv.lasso.errors)
> lambda_min
          [,1]
 [1,] 1.348303
 [2,] 1.479761
 [3,] 1.956157
 [4,] 1.348303
 [5,] 1.782377
 [6,] 2.146880
 [7,] 1.479761
 [8,] 1.624036
 [9,] 1.782377
[10,] 3.114755
> bestlam=as.numeric(names(which.min(cv.lasso.errors.avg)))
> bestlam
[1] 1.624036
> cv.lasso.errors.t = t(cv.lasso.errors)
> df = data.frame(as.numeric(rownames(cv.lasso.errors.t)),cv.lasso.errors.t)
> colnames(df) = c('Lambda',colnames(cv.lasso.errors.t))
> library(ggplot2)
> library(reshape2)
> df = melt(df,id.vars = 'Lambda', variable.name = 'CV')
> p = ggplot(df, aes(Lambda,value)) + geom_line(aes(colour = CV))
> p + ggtitle("CV with 10 folds") + xlab("Lambda") + ylab("CV Error")
```



3. Using the best mode to predict the progression of diabetes on the test dataset, the mean test error is 3033.535.

```
> lasso.pred=predict(lasso.mod,s=bestlam ,newx=xtest)
> mean((lasso.pred-ytest)^2)
[1] 3033.535
```