1. **Study and describe the predictor variables. Do you see any issues that are relevant for making predictions?**

The dataset comprises 97 clinical measurements and 8 variables. Table 1 provides a summary of the variables, including a brief statistical description. Notably, *svi* (seminal vesicle invasion) is represented as a dummy variable (no = 0 / yes = 1).

Attributes	**Description**	**Mean±SD**
Cscore	progression of the cancer score	36.152±52.72
lcavol	log of cancer volume (cc)	1.350±1.179
lweight	log of prostate weight (g)	3.629±0.428
age	age of a patient (years)	68.866±7.445
lbph	log of the amount of benign prostatic hyperplasia (BPH) (cm^2). A noncancerous enlargement of the prostate gland, as an area in a digitized image	0.100±1.451
svi	Seminal vesicle invasion; 1=Yes, 0=No. Indicator of whether prostate cancer cells have invaded the seminal vesicle.	{'0'=76, '1'=21}
lcp	log of capsular penetration (cm). Represents the level of extension of cancer into the capsule (the fibrous tissue which acts as an outer lining of the prostate gland)	-0.179±1.398
lpsa	log Prostate specific antigen (PSA) (ng/mL)	2.478±1.154

Table 1. Description and statistics summary of all eight attributes for prostate cancer dataset.

The distribution of the response variable, Cscore, is shown in Figure 1, ranging from -19.473 to 373. The Shapiro test was used to assess the normality of the distribution, and the results showed that the Cscore was not normally distributed (p-value = 4.139e-13).

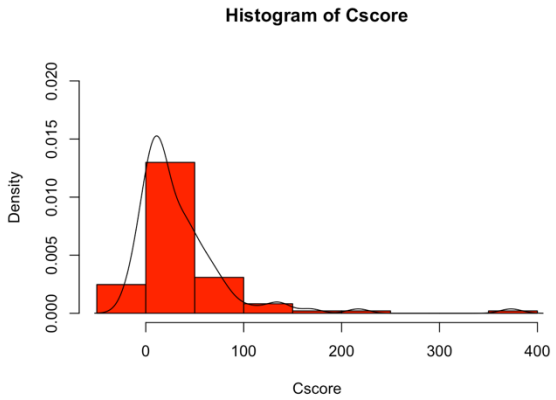


Figure 1. The distribution of Cscore

```
## histograms: shape of Cscore distribution
par(mfrow=c(1,1))
hist(data$Cscore, prob = TRUE,col = 'red',xlab='Cscore',
     main='Histogram of Cscore',ylim=c(0,0.02))
lines(density(data$Cscore))
shapiro.test(data$Cscore) # p<0.05 => not normal distribution
```

Figure 2 and Table 2 compare the Cscore between group 0 and group 1. As the Cscore distribution is non-normal, we performed the Mann-Whitney U test to analyze whether the scores differed between the two groups. The results showed that the Cscore in group 0 was significantly lower than in group 1 ($p < 0.001$). However, the number of patients in group 1 ($n = 21$) was much smaller than in group 0 ($n = 76$), leading to unequal sample sizes that could cause heterogeneity of variance (F-test to compare two variances: $p < 0.05$). This heterogeneity could dramatically affect the type I error rate (p-value) and result in a loss of statistical power (type II error), which would decrease the accuracy of prediction. Therefore, caution is needed when interpreting the results.

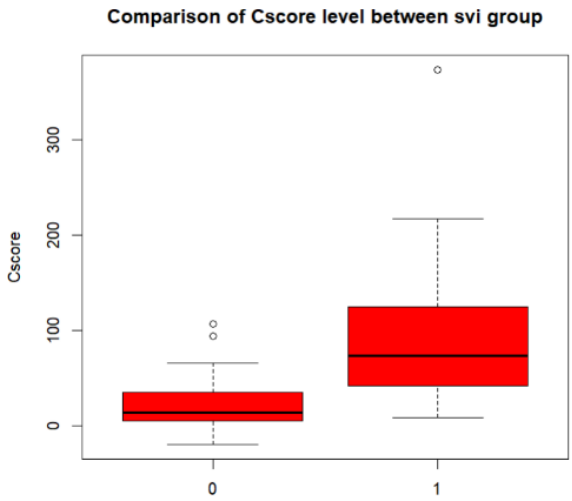


Figure 2. Comparison of Cscore in 2 svi group

	Cscore			
group	**N**	**mean**	**std. error**	**Variance**
0	76	20.5	24.1	579
1	21	92.8	82.9	6870

Table 2. The summary of Cscore in two svi groups.

```
## boxplot
# by_svi: Cscore are significantly different between group 0 and 1.
by_svi <- group_by(data, svi)
summarise(by_svi, n=n(), mean(Cscore), sd(Cscore), var(Cscore))
boxplot(Cscore ~ svi, data = data,col = 'red',
        main='Comparison of Cscore level between svi group', ylab = "Cscore")

data.wilcox <- wilcox.test(data$Cscore ~ data$svi)
data.wilcox # p<0.05 =>  two populations have different continuous distribution

res <- var.test(Cscore ~ svi, data = data)
res # p<0.05 => there is a significant difference between the two variances
```

In terms of issues relevant for making predictions, it's important to note that the dataset has a relatively small sample size, with only 97 observations, which may limit the generalizability of any predictive model and may lead to overfitting and hinder accurate predictions. Additionally, there may be issues with multicollinearity, as some of the predictor variables may be correlated with each other. Multicollinearity refers to high correlation among predictor variables in multiple regression, which can increase the standard errors of the regression coefficients and result in noisy estimates. To validate the effect of correlations between variables, we utilized a scatterplot, which is shown in Figure 3. The results indicate that the correlation between the variables *lcavol* and *lpsa* is higher than 0.7, which can be considered a strong correlation.

To quantify the effect of collinearity on the variance of our regression estimates, we calculated the variance inflation factor (VIF) using the formula VIF = 1/(1 - R-squared). A VIF greater than 5 is typically considered indicative of multicollinearity. Our results indicate that all VIF values are lower than 3, suggesting that there is no multicollinearity issue that need to be addressed in our dataset.

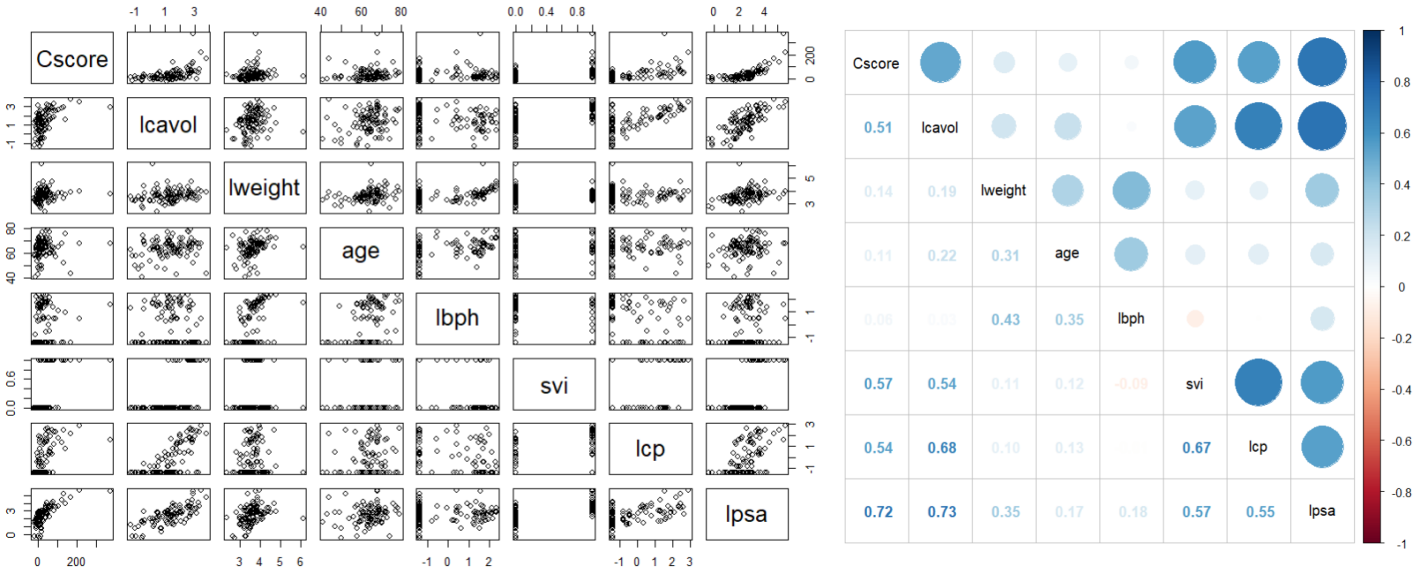


Figure 3. Result of correlation between variables.

```
## correlation
par(mfrow=c(1,1))
data$svi<-as.numeric(data$svi)
data.cor <- cor(data)
data.cor # Pearson's linear correlations
corrplot.mixed(data.cor, tl.col = "black",tl.cex=1)
data$svi<-as.factor(data$svi)
plot(data)

## collinearity check
data.lm.all <- lm(Cscore~.,data=data)
vif(data.lm.all)
# Rule of thumb:
# => all vif lower than 5
# => no collinearity
```

2. **Generate your best linear regression model using only linear effects. Are there any indications that assumptions underlying inferences with the model are violated? Evaluate the effect of any influential point, or outlier.**

Outliers can have a significant impact on prediction, especially in models that rely on minimizing errors or distances between the predicted values and the actual values. In regression models, residuals should fit a dataset well and fall randomly around zero, following a normal distribution for an unbiased fit. However, the influence plot (Figure 4) revealed that the patient at index 96 was an outlier. The residual plots and a quantile-quantile (Q-Q) plot (Figure 5) also confirmed that the patient at index 96 was an influential point with a Cook's Distance larger than 0.5. The presence of an outliers can pull the regression line or decision boundary away from most of the data, which will increase the variance of estimated coefficients and lead to less accurate predictions. Therefore, the decision was made to remove the entire row at index 96 from the dataset. After removing the outlier, both the residual standard error and variance decreased (residual error: 35 -> 24, variance: 2779 -> 1601), indicating a more accurate fit. It is important to note that high leverage points with high residuals can

have a significant impact on the slope of the regression line. On the other hand, the point at index 32 is a high leverage point with a low residual. Therefore, it is unlikely to have a significant impact on the coefficient or accuracy of the regression model, and it was decided to keep this point in the analysis.

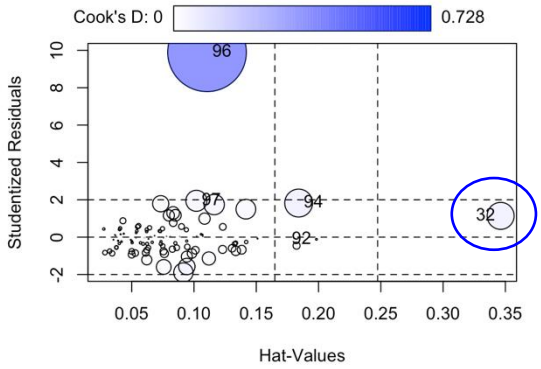


Figure 4 .Influence Plot

```
par(mfrow=c(2,3))
influencePlot(data.lm.all)
plot(data.lm.all)
```

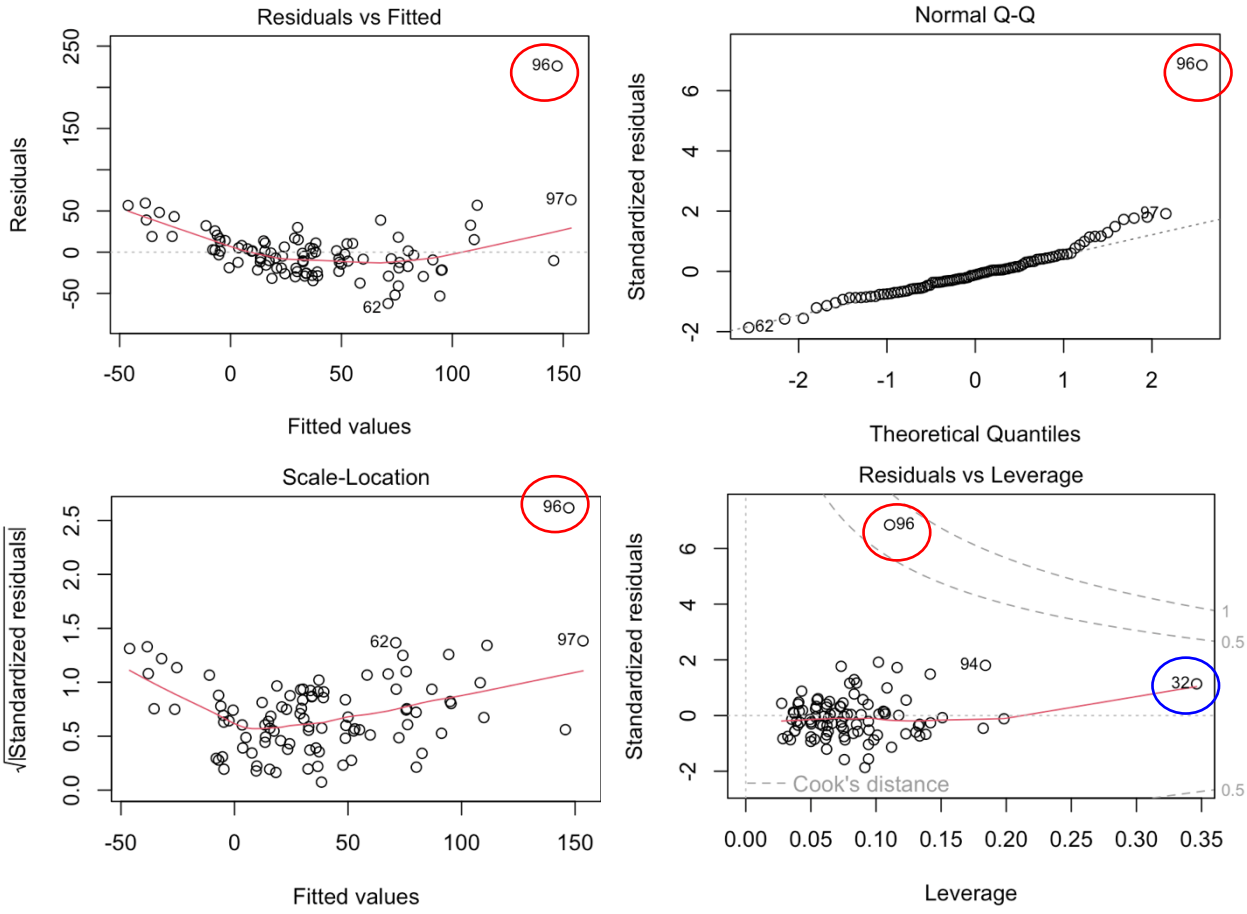


Figure 5. Diagnostic Plot for evaluating the assumptions of linear regression model

The Best Subset Selection, Forward Selection, and Backward Selection techniques are employed to identify the optimal model based on the Bayesian information criterion (BIC) score. The BIC score penalizes the model for its complexity, aligning with the principle of Occam's razor, which states that the simplest explanation is often the best one. Additionally, within the Bayesian probability framework, the BIC provides a higher probability of selecting the true model based on the prior probability of the candidate models that include the true model. As shown in Figure 6, the results of all three techniques indicate that the two variables *lcp* and *lpsa* are the best predictors of Cscore.

Best Subset Selection, Forward Selection, and Backward Selection are all examples of deterministic search algorithms that are relatively fast and straightforward to implement. These methods can provide a good balance between model complexity and predictive accuracy and can help identify a small set of predictors that are highly related to the outcome variable. While cross-validation is a more data-driven approach that involves splitting the data into training and testing sets to evaluate model performance. Cross-validation is a more robust method to estimate the predictive performance of a model because it evaluates the model's performance on a hold-out set of data that was not used to fit the model. This provides a more realistic estimate of the model's performance on new, unseen data, which help estimate the generalizability of a model and identify overfitting.

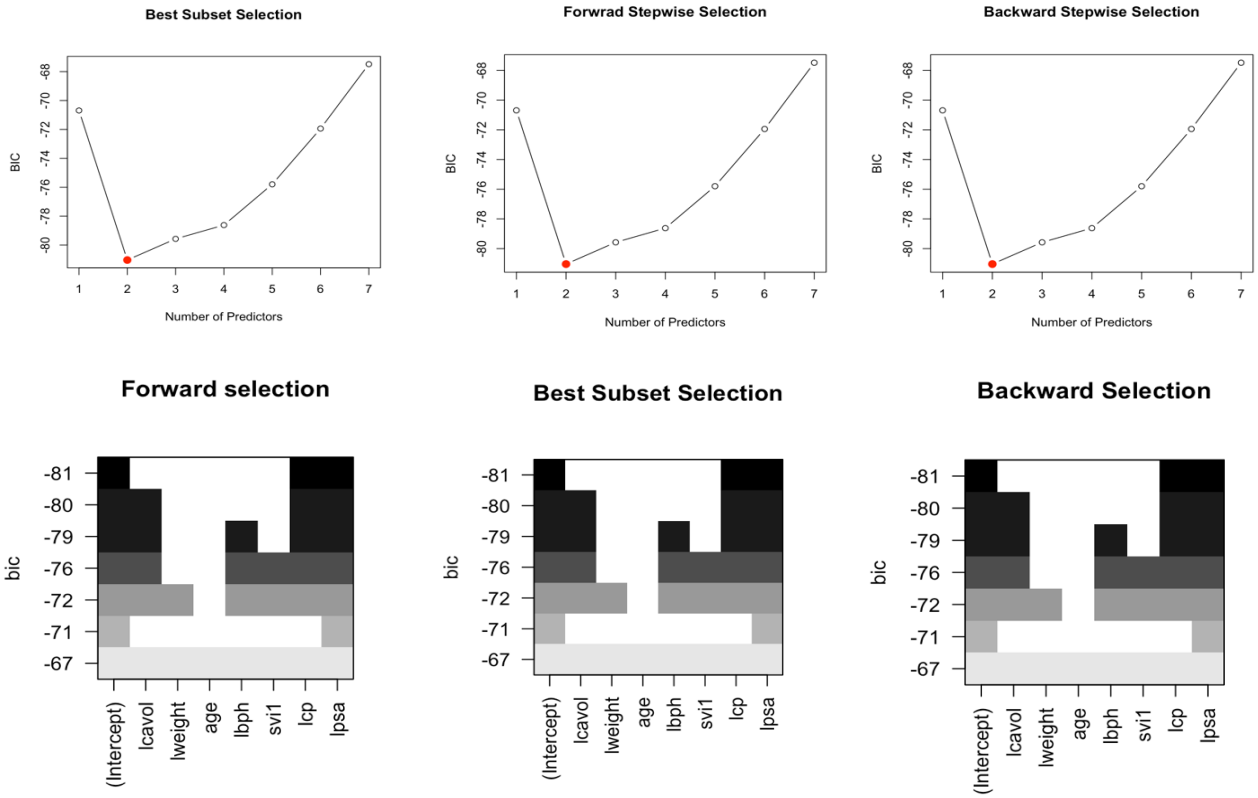


Figure 6. Best Subset Selection, Forward Selection and Backward Selection

```
## best Subset Selection
best <- regsubsets(Cscore~.,data=omit_data)
bestSUM <- summary(best)

par(mfrow=c(1,2))
plot(bestSUM$bic,type="b",ylab="BIC",
     main ="Best Subset Selection")
points(2,bestSUM$bic[2], col ="red",cex =2, pch =20)
plot(best, scale="bic",main ="Best Subset Selection")
which.min(bestSUM$bic)
co=coef(best,2)
```

```
## forward selection
fwd <- regsubsets(Cscore~.,data=omit_data,method="forward")
fwdSUM <- summary(fwd)
plot(fwdSUM$bic,type="b",ylab="BIC",main ="Forward selection")
points(2,fwdSUM$bic[2], col ="red",cex =2, pch =20)
plot(fwd, scale="bic",main ="Forward selection")
which.min(fwdSUM$bic)
co=coef(fwd,2)
```

```
## backward selection
bwd <- regsubsets(Cscore~.,data=omit_data,method="backward")
bwdSUM <- summary(bwd)
plot(bwdSUM$bic,type="b",ylab="BIC",main ="Backward selection")
points(2,bwdSUM$bic[2], col ="red",cex =2, pch =20)
plot(bwd, scale="bic",main ="Backward selection")
which.min(bwdSUM$bic)
co=coef(bwd,2)
```

As shown in Figure 7, the validation set approach and cross-validation have resulted in different models. The validation set approach selected the model with only one variable, lpsa (Figure 7. Left)., while cross-validation selected the model with two variables, lcp and lpsa (Figure 7. Right). The two functions perform model selection using different approaches. Validation set approach split the data into training and validation sets. The training set is used to fit models of increasing complexity and the validation set is used to evaluate the performance of each model and choose the best one based on the lowest validation error. This approach is prone to overfitting since the model selection is based on a single validation set, and the results may not be reliable. K-fold cross-validation, on the other hand, uses all the data for training and validation and therefore provides a more robust estimate of the test error. In summary, the validation set approach is simpler and faster to implement but can lead to overfitting, while k-fold cross-validation is more reliable but requires more computation time.

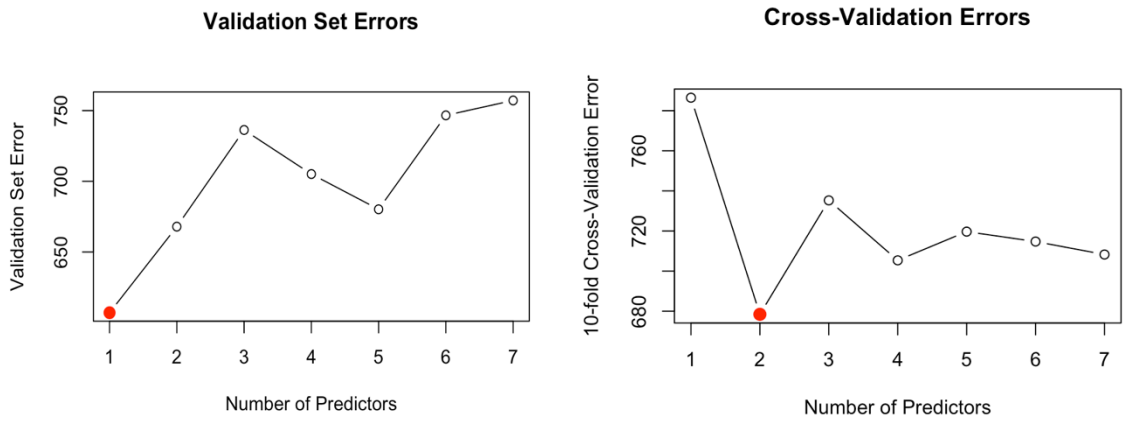


Figure 7. Validation Set vs 10-fold Cross Validation

```r
## validation set approach
set.seed(1)
train <- sample(c(TRUE, FALSE), nrow(omit_data ), rep=TRUE)
test <- (!train)

#apply regsubsets() to the training set to perform best subset selection
regfit.best <- regsubsets(Cscore~., data = omit_data [train,], nvmax=7)

#compare the validation set error
test.mat=model.matrix (Cscore~., data=omit_data [test ,])
#compute the test MSE
val.errors <- rep(NA,7)
for(i in 1:7){
  coefi = coef(regfit.best, id=i)
  pred=test.mat[,names(coefi)]%*%coefi
  val.errors[i]=mean((omit_data $Cscore[test]-pred)^2)
}
val.errors
par(mfrow =c(1,1))
plot(val.errors, type="b", xlab="Number of Predictors",
     ylab="Validation Set Error",main="Validation Set Errors")
points(which.min(val.errors), val.errors[which.min(val.errors)],
       col = "red", cex = 2, pch = 20)
which.min(val.errors)
coef(regfit.best, 1)
```

```r
## 10-fold cross validation
set.seed(1)
k=10
folds=sample(1:k, nrow(omit_data), replace=TRUE)
cv.errors=matrix(NA, k, 7, dimnames=list(NULL, paste(1:7)))
#perform cross-validation
for(j in 1:k){
  best.fit=regsubsets(Cscore~., data=omit_data[folds!=j,], nvmax=7)
  for(i in 1:7){
    pred=predict(best.fit, omit_data[folds==j,], id=i)
    cv.errors[j,i]=mean((omit_data$Cscore[folds==j]-pred)^2)
  }
}
#obtain cross-validation error
mean.cv.errors =apply(cv.errors ,2, mean)
mean.cv.errors
which.min(mean.cv.errors)
#2

par(mfrow =c(1,1))
plot(mean.cv.errors,xlab="Number of Predictors",
     ylab="10-fold Cross-Validation Error",
     main="Cross-Validation Errors", type="b")
points(which.min(mean.cv.errors), mean.cv.errors[which.min(mean.cv.errors)],
       col = "red", cex = 2, pch = 20)
```

Base on the consistency we are able to obtain among different approach, the final linear model for the selection of variables includes the predictors lcp and $lpsa$, and the formula for this model is $Cscore = -17.46 + 8.5\ lcp + 21.2\ lpsa + irreduciable\ error$. This outcome aligns with our expectations since the severity of cancer is likely to increase with a higher level of invasion or migration of cancer cells into the capsule, which is represented by the lcp variable. The PSA protein, which is produced by both normal and malignant prostate cells, is a useful indicator for prostate cancer detection. Higher PSA levels are often associated with prostate cancer, and this is reflected in the positive coefficient of 21.2 for the lpsa variable in the final model. The coefficient values indicate

that a one-unit increase in lcp is associated with an increase of 8.5 in the mean value of the response variable, holding other variables constant, and a one-unit increase in lpsa is associated with an increase of 21.2 in the mean value of the response variable, holding other variables constant.

3. **Make an appropriate LASSO model, with the appropriate link and error function, and evaluate the prediction performance. Do you see evidence that over-learning is an issue?**

Lasso regression is a linear regression technique that incorporates a penalty term to the cost function, which is controlled by the tuning parameter λ. The penalty term imposes a constraint on the sum of the absolute values of the regression coefficients, resulting in the shrinking of coefficients towards zero. This leads to variable selection, whereby some coefficients are set exactly to zero, making lasso particularly useful in high-dimensional datasets where not all features may be important in predicting the outcome variable. By selecting the most important features, lasso regression can reduce overfitting and improve model interpretability. Unlike ridge regression, which shrinks all coefficients towards zero, lasso can eliminate some coefficients, resulting in a sparse model with only a subset of the original predictors.

Overfitting occurs when a model is too complex and captures noise instead of the underlying signal in the data, leading to poor generalization performance on new data. One way to assess overfitting in Lasso is to compare the model performance on the training and testing sets. If the model has learned too much from the training set and does not generalize well to the test set, this may indicate overfitting. The prediction performance of the Lasso model can be evaluated using the test set, and the mean squared error (MSE) can be calculated. If the MSE on the test set is significantly higher than the MSE on the training set, this may indicate overfitting.

In our study, we fitted a Lasso model using the glmnet package to predict *Cscore* based on all predictor variables. The model was trained on a randomly selected two-thirds of the data and tested on the remaining one-third. The optimal lambda parameter was chosen via cross-validation, as shown in the Figure 8 (Left). We observed that a shrinkage penalty λ of 0.74 resulted in the smallest cross-validation error of 643.20, with cross-validation standard deviation (CVSD) of 155.03. We calculated the MSE for both the training and test sets using the selected lambda. We found that the MSE on the test set (750.17) was higher than the MSE on the training set (468.57), suggesting that the model may be overfitting to the training data. This indicates that the model may have learned the noise in the training data and may not generalize well to new data. The formula is $Cscore = -20.13 - 3.09lcavol - 2.79lweight - 1.66lbph + 11.53svi + 7.03\,lcp + 22.38\,lpsa + irreduciable\ error$, with only one predictor *age* reduced to zero.

```
> #Generate standardized LASSO coefficients
> out=glmnet(omit_data[,-1],omit_data$Cscore, alpha =1, lambda =grid)
> lasso.coef.standardized <- coef(out, s = lambda_best,
+                                 x = x_train, y = y_train,
+                                 standardize = TRUE)[1:8,]
> lasso.coef.standardized
(Intercept)      lcavol     lweight         age        lbph         svi         lcp        lpsa
 -20.217552   -3.097977   -2.789784    0.000000   -1.655355   11.534354    7.027591   22.381557
```

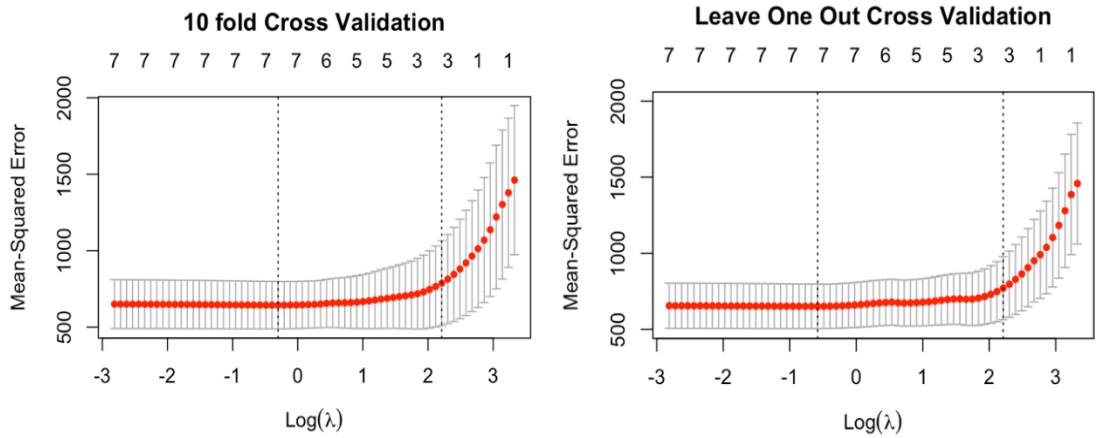


Figure 8. (Left) 10-fold cross-validation error (Right) Leave-One-Out-Cross-Validation error as a function

```
#LASSO
set.seed(1)
train_index <- sample(nrow(omit_data), nrow(omit_data) * 2/3)
train <- omit_data[train_index, ]
test <- omit_data[-train_index, ]

grid =10^seq(10,-2, length =100)
x_train <- model.matrix(Cscore ~ ., train)[,-1]
y_train <- train$Cscore
x_test <- model.matrix(Cscore ~ ., test)[,-1]
y_test <- test$Cscore

### Lasso regression with 10-fold Cross Validation ###
lasso_mod <- glmnet(x_train, y_train, alpha =1, lambda =grid)
lasso_fit <- cv.glmnet(x_train, y_train, alpha = 1, nfolds=10)
plot(lasso_fit,main="10 fold Cross Validation")

## lambda.min
lambda_best <- lasso_fit$lambda.min
lambda_best
# lambda_best = 0.7
```

```
## Get the index of the lambda with the minimum CVM
# (cross-validation error mean)
lambda_idx <- which.min(lasso_fit$cvm)
lambda_idx
# Get the CVM for the lambda with the minimum CVM
cvm <- lasso_fit$cvm[lambda_idx]
cvm  #smallest cross-validation error 643.2
# Get the CVSD (standard deviation) with the minimum CVM
cvsd <- lasso_fit$cvsd[lambda_idx]
cvsd #155.03

## Calculate the MSE on the training set
y_train_pred <- predict(lasso_mod,s=lambda_best, newx = x_train)
mse_train <- mean((y_train_pred - y_train)^2)
mse_train
#468

# Calculate the MSE on the test set
y_test_pred <- predict(lasso_mod,s=lambda_best, newx = x_test)
mse_test <- mean((y_test_pred - y_test)^2)
mse_test
#750
```

Nevertheless, the small size of the test set (only 32 observations) makes it difficult to draw definitive conclusions about the model's performance. By varying the size of the training and test groups, even by changing the set.seed() function, we observed that the test and training MSEs differed significantly. Therefore, additional evaluation may be necessary. We decided to perform LOOCV (leave-one-out-cross-validation) to further assess the Lasso model's performance.

LOOCV is a resampling technique that is used to estimate the performance of a statistical model on new data. It works by fitting the model on n-1 data points and testing it on the remaining one. This process is repeated n times, each time leaving out a different data point. By averaging the results over all the n iterations, we obtain an estimate of the model's prediction performance on new data. LOOCV is often used when the sample size is small, as it uses all the data for training and testing and can give a good estimate of the model's performance on new data. In our case, LOOCV was performed with nfolds = 96, which means that the model is trained and tested 96 times, leaving out one observation at a time. As shown in Figure 8 (Right), a shrinkage penalty of $\lambda = 0.55$ was selected to minimize the cross-validation error, resulting in a value of 651.01 with cross-validation standard deviation (CVSD) 146.05. However, it is worth noting that LOOCV did not result in better performance than using 10-fold cross-validation. The performance on the test set was not as good as expected, with a MSE of 757.24, suggesting that the model may still be overfitting.

One thing needs to be highlight is that we can also select the model with one standard error rule, which lambda value is indicated in the second vertical dash line in Figure 8. By selecting the smallest model for which the estimated test error is within one standard error of the lowest MSE. We can find the simpler model that appears to be more or less equally good, in the sense that model with the smallest number of predictors. Even though the MSE is a bit higher (844.18) than the lowest MSE, lambda is large enough (9.11) to introduce heavier penalty. In this case, applying the one-standard-error rule leads to selection of the three-variable model. The formula is $Cscore = -10.30 + 5.00svi + 3.48\ lcp + 15.36\ lpsa + irreducible\ error$.

```
> ## Generate 1se LASSO coefficients
> out=glmnet(omit_data[,-1],omit_data$Cscore, alpha =1, lambda =grid)
> lasso.coef.standardized <- predict(out, s = lambda_1se, type = "coefficients",
+                                    standardize = TRUE)[1:8,]
> lasso.coef.standardized
(Intercept)      lcavol     lweight         age        lbph         svi         lcp        lpsa
 -10.296668    0.000000    0.000000    0.000000    0.000000    5.003118    3.488188   15.357703
```

4. Look at the coefficient for "lcavol" in your LASSO model. Does this coefficient correspond to how well it can predict Cscore? Explain your observation.

The coefficient of *lcavol* in a LASSO model represents its correlation with the target variable, *Cscore*, while accounting for the impact of other predictor variables in the model. A negative coefficient indicates an inverse relationship between *lcavol* and *Cscore*, implying that as *lcavol* increases, the predicted value of *Cscore* decreases. However, the coefficient alone does not provide an accurate measure of *lcavol*'s predictive ability and must be considered in the context of other variables in the model, as well as the model's overall performance in predicting the target variable.

As shown in Figure 9, there are two different LASSO models with distinct coefficient values for *lcavol.* The 10-fold Cross-validation LASSO model's coefficient for *lcavol* is -3.097977, indicating that a one-unit increase in the natural log of *lcavol* corresponds to a -3.097977 unit decrease in *Cscore* while keeping other predictors constant. On the other hand, the LOOCV LASSO model's coefficient for *lcavol* is -3.969890. Therefore, it is not appropriate to draw direct comparisons between these two coefficients or to determine *lcavol*'s predictive capability based solely on the coefficient value.

We conducted a linear regression analysis with *lcavol* as the only predictor variable and *Cscore* as the response variable. The Adjusted R-squared value was 0.3138, indicating that only 31.38% of the variability in *Cscore* can be explained by the variability in *lcavol*, while controlling for other predictor variables in the model. As shown in Figure 10, we also generated a scatter plot with *lcavol* on the x-axis and *Cscore* on the y-axis, which included a linear regression line to visualize the overall trend. This plot allowed us to visually evaluate how well *lcavol* predicts *Cscore*, and we observed a moderate positive correlation of 0.5665924 between the two variables.

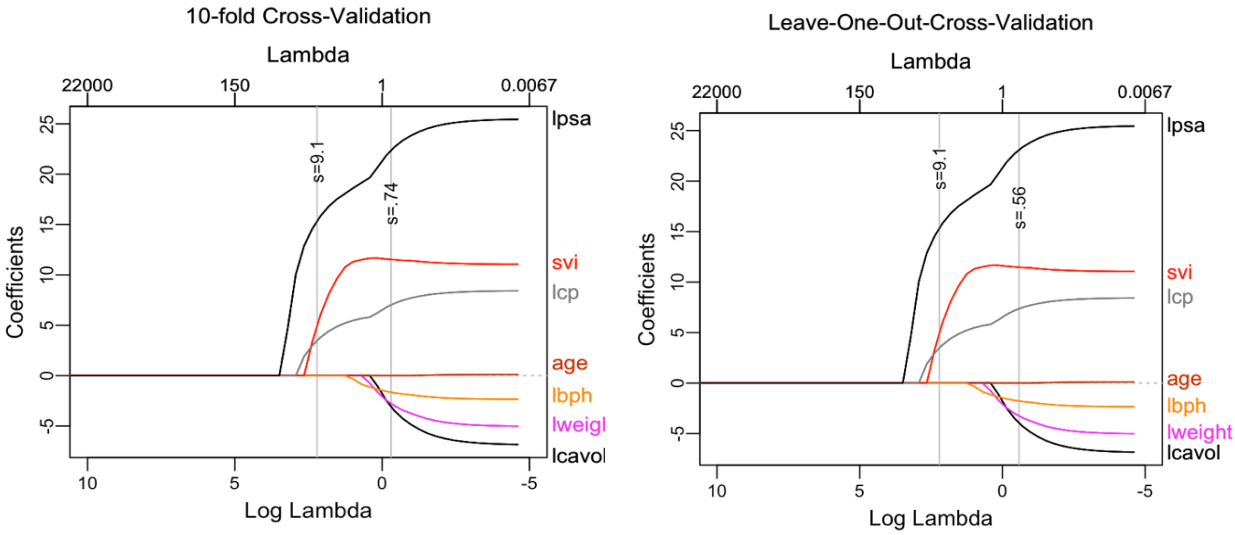
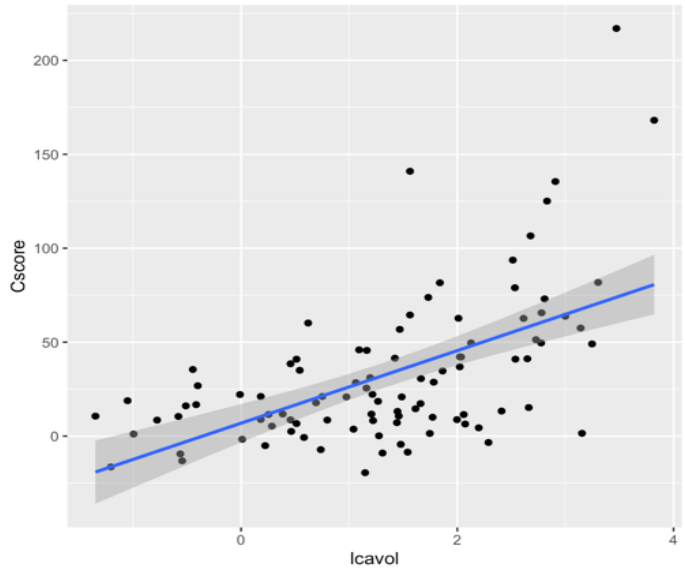


Figure 9. (Left) 10-fold Cross Validation (Right) LOOCV standardized Lasso coefficients as functions of log λ

```
## Lasso regression with 10-fold Cross Validation ##
out=glmnet(omit_data[,-1],omit_data$Cscore, alpha =1, lambda =grid)

# Generate standardized LASSO coefficients
##lambda_best
lasso.coef.standardized <- coef(out, s = lambda_best, x = x_train, y = y_train, standardize = TRUE)[1:8,]
lasso.coef.standardized #lcavol:-3.09

plot_glmnet(out, label = TRUE, s = lambda_best, xlim = c(10, -5), main="10-fold Cross-Validation")

##lambda_1se
lasso.se.coef.standardized <- predict(out, s = lambda_1se, type = "coefficients", standardize = TRUE)[1:8,]
lasso.se.coef.standardized #only 3 variables left (svi, lcp,lpsa)

plot_glmnet(out, label = TRUE, s = lambda_1se , xlim = c(10, -5), main="10-fold Cross-Validation",add = TRUE)
```

It is important to note that correlation does not necessarily imply causation, and that there may be other variables not included in the model that can influence the relationship between *lcavol* and *Cscore*. In the context of the LASSO regression model, it is possible for the coefficient for *lcavol* to be negative, even if there is a positive correlation between *lcavol* and *Cscore*. This can occur because other predictor variables in the model may have a stronger association with *Cscore* or because the relationship between *lcavol* and *Cscore* may be non-linear or conditional on the values of other variables. Therefore, it is essential to evaluate the overall performance of the model and consider the interpretation of coefficients in the context of the other predictor variables.




```
#scatterplot => how well "lcavol" predicts "Cscore"
ggplot(data = omit_data, aes(x = lcavol, y = Cscore)) +
  geom_point() +
  geom_smooth(method = "lm")


model <- lm(Cscore ~ lcavol, data = omit_data)
summary(model)
```

Figure 10. Relationship between lcavol and Cscore

As illustrated in Figure 3, *lcavol* and *lpsa* exhibit high correlation, which suggests that they can be used interchangeably to predict the response variable. Same can also be observed in LASSO with 1 standard error lambda, which reduced the *lcavol* into zero (Figure.9). When dealing with a group of variables that have high correlations, the LASSO may select one variable from the group while reducing the influence of the others. This means that the coefficient of *lcavol* in the LASSO model may not be representative of its true relationship with *Cscore*, and it may be necessary to consider other models or techniques to better understand the relationship between the variables.

5. Fit your best model with *appropriate* non-linear effects. Report a comparison of performance to LASSO and your model reported under question 2. Explain what you find and indicate relevant issues or limitations of your analysis.

To fit the best model with appropriate non-linear effects, we utilized the gam function from the gam package to fit generalized additive models that incorporate non-linear effects of variables using smooth functions. To evaluate the performance of this model, we compared it to the LASSO and the best linear model using the mean squared error (MSE) on the test set. It is important to note that adding more terms or interactions can increase the complexity of the model and risk overfitting, so careful evaluation and consideration of the trade-off between model complexity and performance is necessary. In other words, simpler models are preferred within the marginal worse performance.

To determine the best subset of predictors, we split the data into a training and test set using a 2/3 split ratio and performed both best subset and forward stepwise selection. We found that, in both conditions, the best model includes the predictors *lcavol*, *lbph*, *lcp*, and *lpsa*. We then fit a generalized additive model using smoothing spline functions with 4 degrees of freedom to identify the non-linear effects of each predictor. The performance of the GAM is evaluated by computing the MSE (379.97) on the test data. We identify that only *lpsa* had a significant non-linear effect and simplified the model by removing the predictor *lbph* and the smoothed term of predictor *lcp* and *lcavol*, respectively. For further simplification, we reduce the degrees of freedom and then the test MSE (332.66) was computed for the simplified models as shown in figure 11 below. An ANOVA test was performed to compare the performance of the models, simplification justified as expected.

The reason why we choose to use smoothing splines as a starting point of fitting GAM model is that it provides smooth curves that can capture non-linear relationships without introducing unwanted wiggles or abrupt changes. It is especially important when the true relationship between variables is expected to be smooth. Secondly, smoothing splines is a shrunken version of a natural cubic spline, while lambda controls the level of shrinkage. This means that it can capture complex and intricate non-linear relationships without introducing strong assumptions, which makes it more flexible than other non-linear models. Additionally, smoothing splines can be useful in situations where the degree of non-linearity is unknown or varies across the range of the predictor since it can be adjusted to fit different degrees of non-linearity. Finally, smoothing splines have been shown to have good performance characteristics in terms of minimizing mean squared error and avoiding overfitting, it can also be easily visualized and interpreted, as they produce a smooth curve that can be easily understood.

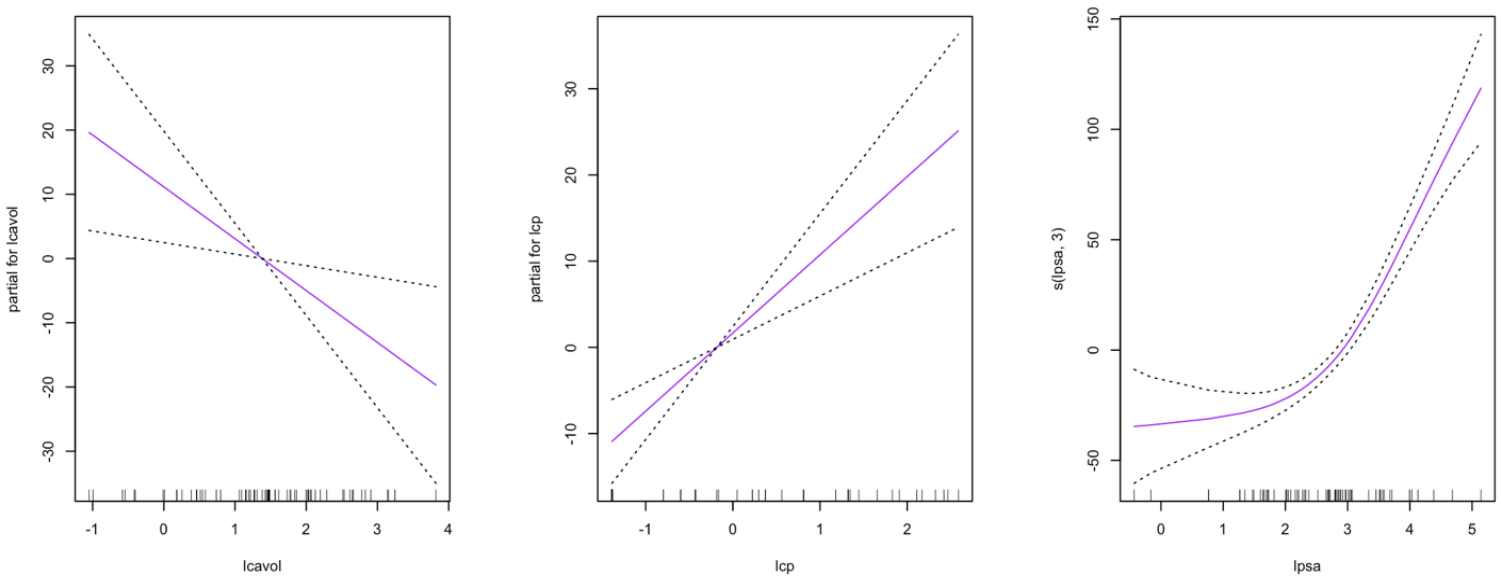


Figure 11. Generalized Additive Models for 3 variables, first 2 functions are linear in lcavol and lcp, the third function is a smoothing spline in lpsa with 3 degrees of freedom.

```
# Fit a GAM, plot the results, evaluate the model
gam1=gam(Cscore~ s(lcavol,4) +s(lbph,4)+ s(lcp,4)+s(lpsa,4),data=train)
par(mfrow=c(2,2))
plot(gam1,se=TRUE,col="purple")
summary(gam1)
#only lpsa seems to have non-linear effect, lbph is not significant
predgam = predict(gam1, newdata=test)
msegam1 = mean((predgam-test$Cscore)^2)
msegam1
#379.97

# Remove lbph and remove smooth spline function
gam2=gam(Cscore~ lcavol + lcp + s(lpsa,4),data=train)
plot(gam2,se=TRUE,col="purple")
summary(gam2)
predgam2 = predict(gam2, newdata=test)
msegam2 = mean((predgam2-test$Cscore)^2)
msegam2
#350

# Reduce df
gam3=gam(Cscore~ lcavol + lcp +s(lpsa,3),data=train)
plot(gam3,se=TRUE,col="purple")
summary(gam3)
predgam3 = predict(gam3, newdata=test)
msegam3 = mean((predgam3-test$Cscore)^2)
msegam3
#332
anova(gam1, gam2, gam3) #choose gam3
# simplification justified as expected.
```

```
#fit same model with ns
par(mfrow=c(2,2))
lm1 = lm(Cscore~lcavol+ lcp +ns(lpsa,3) ,data=train)
summary(lm1)
plot(lm1)
predlm1 = predict(lm1, newdata=test)
mselm1 = mean((predlm1-test$Cscore)^2)
mselm1
#310, better than smoothing

#remove lcavol
lm2 = lm(Cscore~ lcp + ns(lpsa,3) ,data=train)
summary(lm2)
predlm2 = predict(lm2, newdata=test)
mselm2 = mean((predlm2-test$Cscore)^2)
mselm2
#267

#reduce df
lm3 = lm(Cscore~ lcp +ns(lpsa,2) ,data=train)
summary(lm3)
predlm3 = predict(lm3, newdata=test)
mselm3 = mean((predlm3-test$Cscore)^2)
mselm3 #marginally worse than msegam2
#277

anova(lm1,lm2,lm3) #choose lm3
```

A natural spline function was then applied to *lpsa*, and we found that the natural spline function had a lower MSE (310.34) on the test set than the GAM model, indicating better performance. We also identified that removing *lcavol* resulted in a model with a lower MSE (267.09). Further simplification of the model by reducing the degrees of freedom resulted in marginally worse test MSE, which was justified by an ANOVA test. However, further simplification by removing *lcp* led to a significant increase in the MSE (324.33, p-value 0.0009). Finally, a polynomial regression model was fitted with linear predictors *lcp* and a quadratic term of *lpsa*. The test MSE for this model was significantly lower than all other models (257.29), and it maintained high simplicity. Cubic term of *lpsa* was also fitted and display even lower MSE (246.70). However, with p-value of 0.1838 on ANOVA, simpler model was favoured.

```r
## polynomial transformation with lpsa variable
poly.lm <- lm(Cscore~ lcp +poly(lpsa,2,raw=TRUE),data=omit_data)
summary(poly.lm)
predlm5 = predict(poly.lm, newdata=test)
mselm5 = mean((predlm5-test$Cscore)^2)
mselm5
#257
summary(poly.lm)
#R-squared 0.8056
summary(poly.lm)$coef
# Cscore = 17.94 +6.18lcp -20.00lpsa +8.98lpsa^2 +irreduciable error

par(mfrow=c(2,3))
plot(poly.lm)
res <- resid(poly.lm)
plot(density(res))
shapiro.test(res) # p>0.05
# => residual is normal distribution

## polynomial 3 degree
poly.lm2 <- lm(Cscore~ lcp +poly(lpsa,3,raw=TRUE),data=omit_data)
summary(poly.lm2)
predlm6 = predict(poly.lm2, newdata=test)
mselm6 = mean((predlm6-test$Cscore)^2)
mselm6
#246

anova(poly.lm,poly.lm2)
# poly.lm2 is not significant
# => polynomial 2 degree is better (simpler is better)
```

```r
# Draw regression curve
par(mfrow=c(2,2))
#lcp
my_mod_lcp <- lm(Cscore ~ lcp, data = omit_data)
summary(my_mod_lcp)
plot(Cscore ~ lcp, omit_data , main="Cscore vs lcp")
lines(sort(omit_data$lcp),
      fitted(my_mod_lcp)[order(omit_data$lcp)],
      col = "red",
      type = "l")

#lpsa
my_mod_lpsa <- lm(Cscore ~ lpsa, data = omit_data)
summary(my_mod_lpsa)
plot(Cscore ~ lpsa, omit_data, main="Cscore vs lpsa")
lines(sort(omit_data$lpsa),
      fitted(my_mod_lpsa)[order(omit_data$lpsa)],
      col = "red",
      type = "l")

# Draw polynomial regression curve
#lpsa
my_mod_lpsa <- lm(Cscore ~ poly(lpsa, 2, raw=TRUE), data = omit_data)
summary(my_mod_lpsa)
plot(Cscore ~ lpsa, omit_data, main="Cscore vs lpsa (quadratic)")
lines(sort(omit_data$lpsa),
      fitted(my_mod_lpsa)[order(omit_data$lpsa)],
      col = "red",
      type = "l")
```

Comparing the performance of the best non-linear model to LASSO (MSE:750.17) and the best linear model reported in question 2(MSE: 684.49), we found that the non-linear model had a significantly lower MSE. Also, in figure 12, we include the regression curve for the selected variables. We can see that $lpsa^2$ can provide a better explanation for the response variables compared to the linear model. Therefore, a polynomial regression model with linear predictors *lcp* and a quadratic term of *lpsa* is identified as the best model for predicting the prostate cancer *Cscore*, which explains 80.56% of the variability in *Cscore*. The formula for this model is $Cscore = 17.94 + 6.18\ lcp - 20.00\ lpsa + 8.98\ lpsa^2 + irreduciable\ error$.

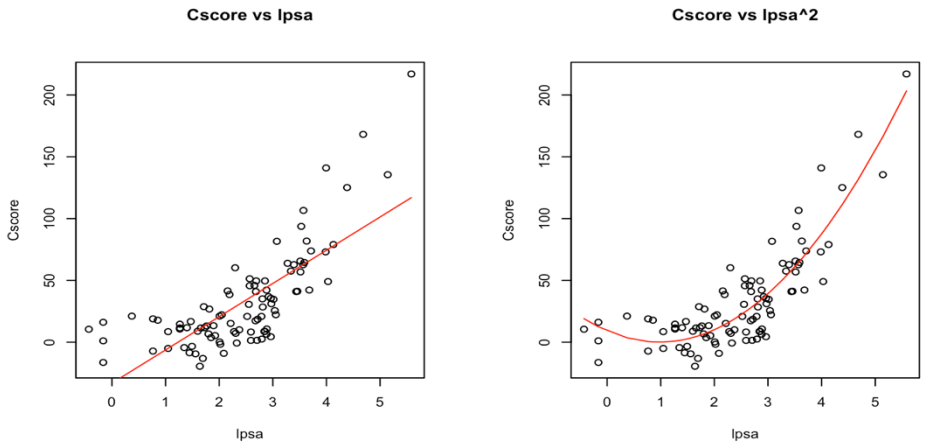


Figure 12. Visualization of the linear model and non-linear model (Polynomial)

One limitation of this analysis is that the sample size is relatively small, which may limit the generalizability of the findings. Additionally, the models are limited by the available predictors and potential unmeasured confounding variables that may affect the relationship between the predictors and the response variable. It is important to note that we only used the mean squared error as a measure of predictive performance, and non-linear models may have potential interpretability issues since the effects of the variables are not easily summarized by a single coefficient. Further research and data collection may be needed to improve the accuracy of the models.