Moisture Content Prediction Polynomial Regression and Regularization(LASSO and Ridge)

Abhinay Krishna Vellala

12/19/2020

We have the data of the result of study which investigated weather near infrared absorbance spectrum can be used to predict fat conent of a sample of meat. For each meat sample the data consists of a 100 channel spectrum of absorbance records and the levels of moisture (water), fat and protein. The absorbance is -log10 of the transmittance measured by the spectrometer. The moisture, fat and protein are determined by analytic chemistry.

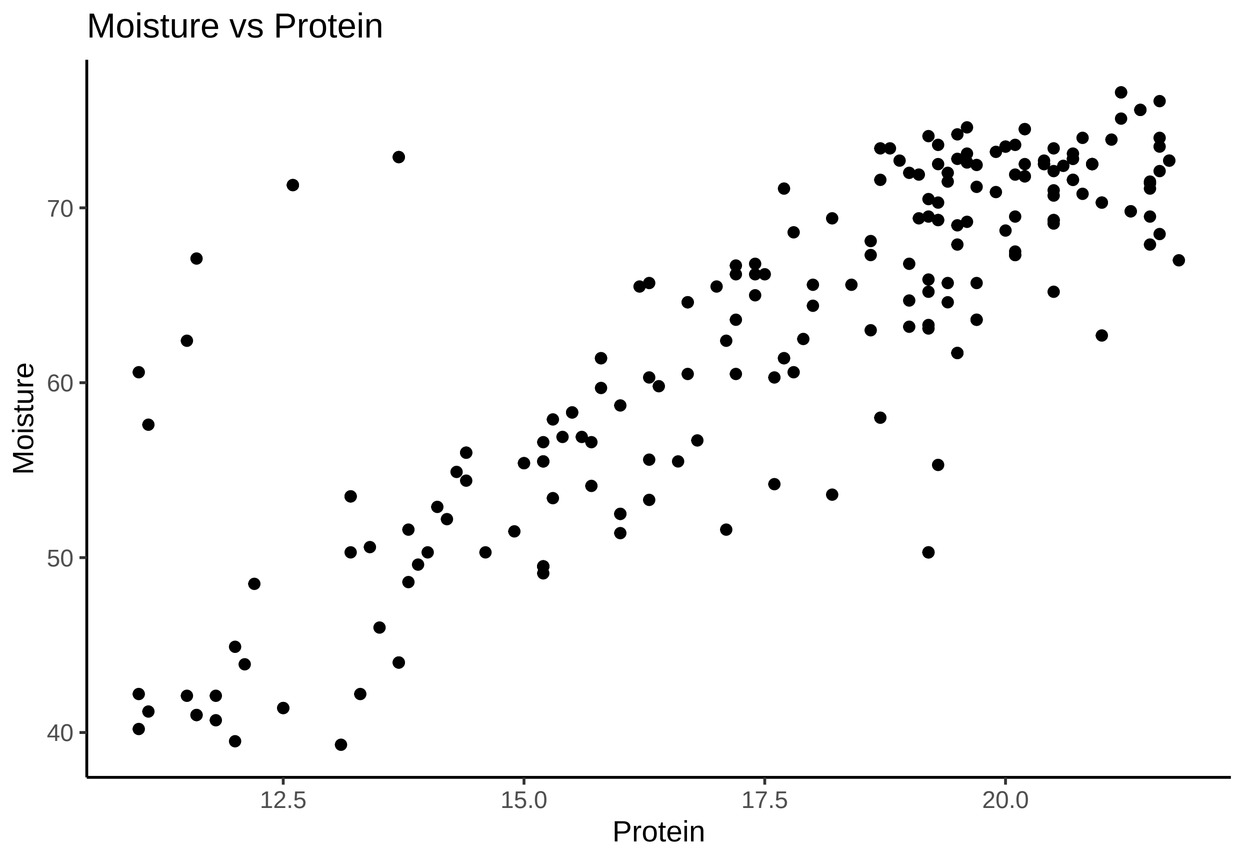
Data

Covariates 1. Channel 1-100 (100 variables) 2. Fat(1 variable) 3. Protein(1 Variable)

Target Moisture

4.1 Import and check

```
# load data
library(readxl)
library(ggplot2)
meat_data = read_excel("tecator.xlsx")
ggplot(meat_data,aes(x=Protein,y=Moisture)) + geom_point() + theme_classic() + ggtitle("Moisture vs Pro
```



The plot above shows there is a linear relation between Protein and Moisture as we can see with the increase in Protein value, Moisture is increasing. So, Linear Model can well explain this data.

4.2 Polynomial Model selection

Considering the model M_i in which Moisture is normally distributed, we can write the model as:

$$Moisture \sim N(\mu, \sigma^2)$$

And we know the Linear model in Matrix form as:

$$\hat{Y} = X(X^TX)^{-1}X^TY$$

Now, considering Moisture as Polynomial function of Protein, we can write it as:

$$Model(Moisture) = \beta_0 + \beta_1 * Protein + \beta_2 * (Protein)^2 ... + \beta_i * (Protein)^i$$

Let the polynomial function of protein be ϕ where,

$$\Phi = \begin{bmatrix} 1 & Protein & Protein^2 & ... & Protein^i \end{bmatrix}$$

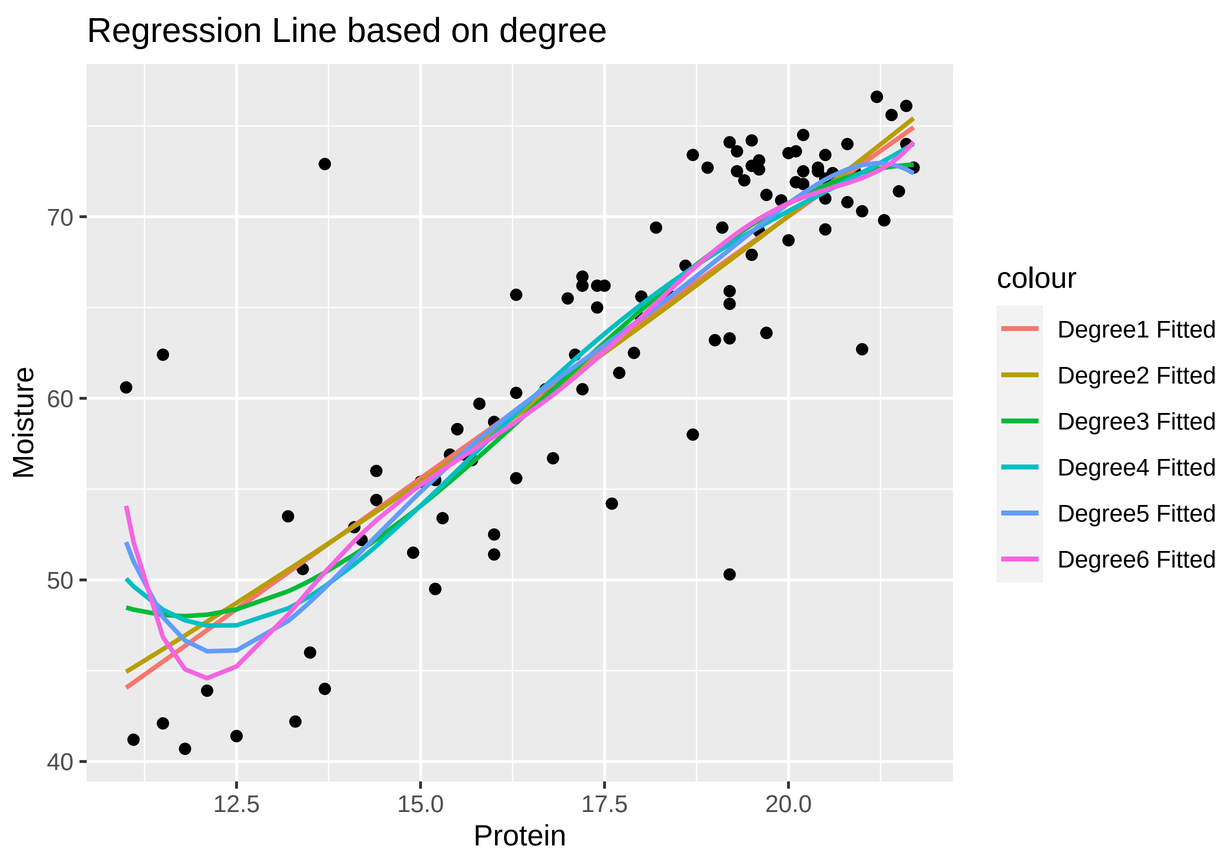
So, the probabilistic model becomes,

$$\hat{Y} = \Phi(\Phi^T\Phi)^{-1}\Phi^TY$$

Mean square error criterion estimates the values of Y that are close enough to the actual values because we try to minimize the Residual Square error. Hence, it is ideal to use MSE criterion in this case.

4.3 Polynomial model

```r
# divide data
n = dim(meat_data)[1]
set.seed(12345)
id = sample(1:n, floor(n*0.5))
train = meat_data[id,]
valid = meat_data[-id,]

fitted_values = list()
train_MSE = vector()
valid_MSE = vector()
for(i in 1:6){
  poly_model = lm(Moisture ~ poly(Protein,degree = i),data = train)
  fitted_values[[i]] = poly_model$fitted.values
  train_MSE[i] = mean((poly_model$residuals)^2)
  poly_predict = predict(poly_model, newdata = valid)
  valid_MSE[i] = mean((poly_predict - valid$Moisture)^2)
}

ggplot() + geom_point(aes(x = train$Protein, y = train$Moisture), color = "black") +
  geom_line(aes(x = train$Protein, y = fitted_values[[1]], color = "Degree1 Fitted"), size = 0.8) +
  geom_line(aes(x = train$Protein, y = fitted_values[[2]], color = "Degree2 Fitted"), size = 0.8) +
  geom_line(aes(x = train$Protein, y = fitted_values[[3]], color = "Degree3 Fitted"), size = 0.8) +
  geom_line(aes(x = train$Protein, y = fitted_values[[4]], color = "Degree4 Fitted"), size = 0.8) +
  geom_line(aes(x = train$Protein, y = fitted_values[[5]], color = "Degree5 Fitted"), size = 0.8) +
  geom_line(aes(x = train$Protein, y = fitted_values[[6]], color = "Degree6 Fitted"), size = 0.8) +
  ggtitle("Regression Line based on degree") + xlab("Protein") + ylab("Moisture")
```



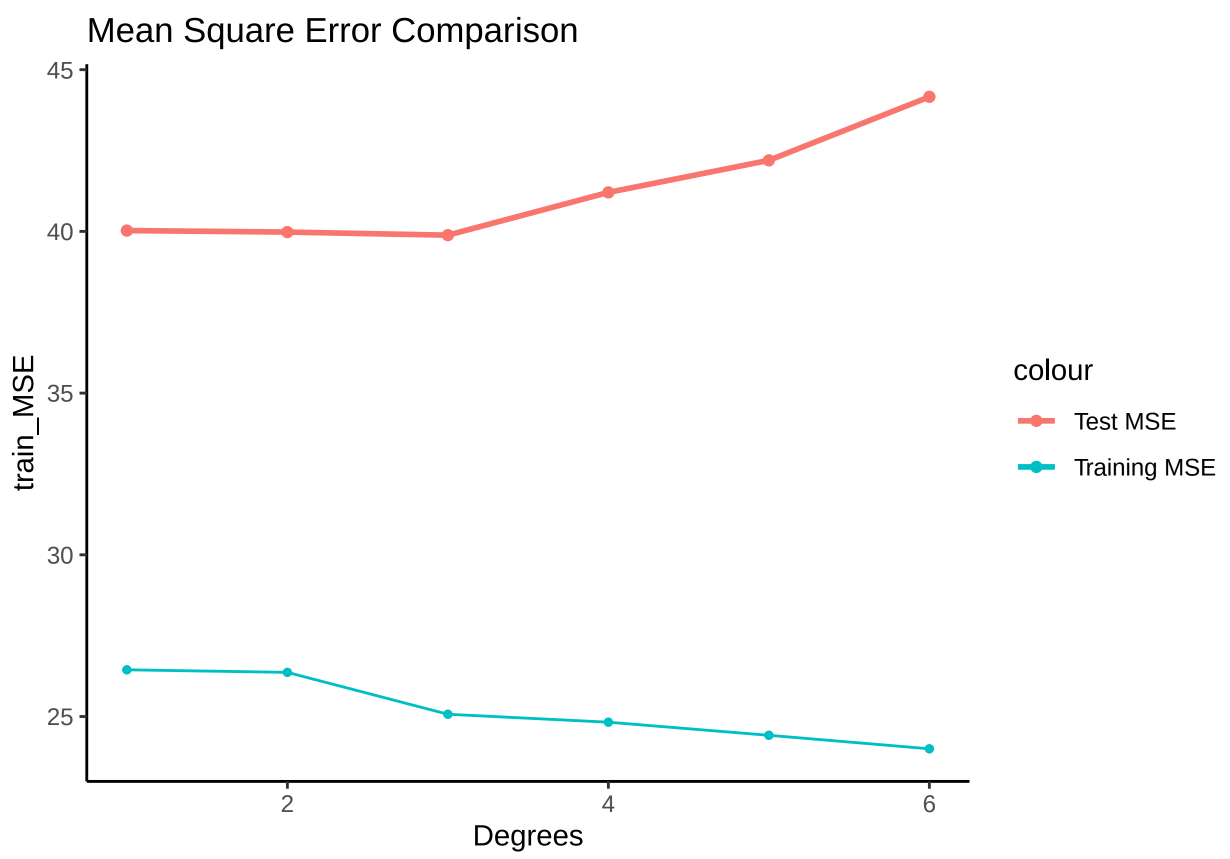
```r
# train and validation MSE
cat("The train errors are\n", train_MSE)
```

```
## The train errors are
##  26.44494 26.3659 25.07017 24.82475 24.41987 24.00218
```

```r
cat("\n\nThe test errors are\n", valid_MSE)
```

```
##
##
## The test errors are
##  40.02562 39.97895 39.88347 41.20548 42.19681 44.16041
```

```r
ggplot() + geom_line(aes(x = 1:6, y = train_MSE, color="Training MSE")) +
  geom_point(aes(x = 1:6, y = train_MSE, color="Training MSE"), size = 1) +
  geom_line(aes(x = 1:6, y = valid_MSE, color="Test MSE"), size = 1) +
  geom_point(aes(x = 1:6, y = valid_MSE, color="Test MSE")) +
  theme_classic() + ggtitle("Mean Square Error Comparison") + xlab("Degrees")
```



From the above graph it shows that the mean square error of training data is reduced with the increase in degree but it is increasing with validation data. This is the case of overfitting where the model might perform well with the training data but will do bad with the test data because of the high variance. Usually, if variance is high it will lead to overfitting and if bias is high it will lead to underfitting. The ideal model should have low bias and low variance which can perform well with the test data.

4.4 Variable selection using Step AIC

Step AIC is a method to choose most significant variables.

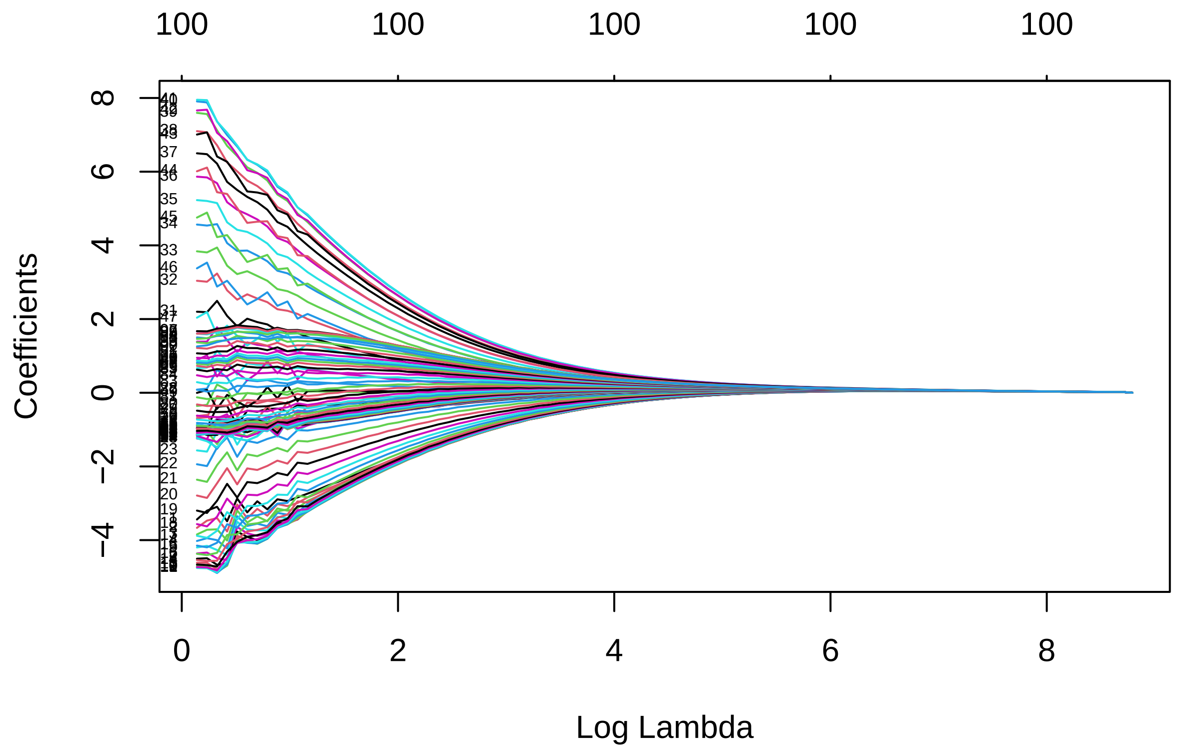
```r
# step AIC
library(MASS)
meat_data1 = meat_data[,2:102]
model_aic = lm(Fat ~ ., data = meat_data1)
select_step = stepAIC(model_aic, direction = "both", trace = FALSE)

cat("The number of variables selected using StepAIC are",length(select_step$coefficients))
```

```
## The number of variables selected using StepAIC are 64
```

```r
# Ridge Regression
library(glmnet)
```

```
## Loading required package: Matrix
## Loaded glmnet 4.0-2
```

```
covariates = meat_data1[,-101]
response = meat_data1$Fat
ridge_model = glmnet(as.matrix(covariates), response, alpha=0, family="gaussian")
plot(ridge_model, xvar = "lambda", label = TRUE)
```

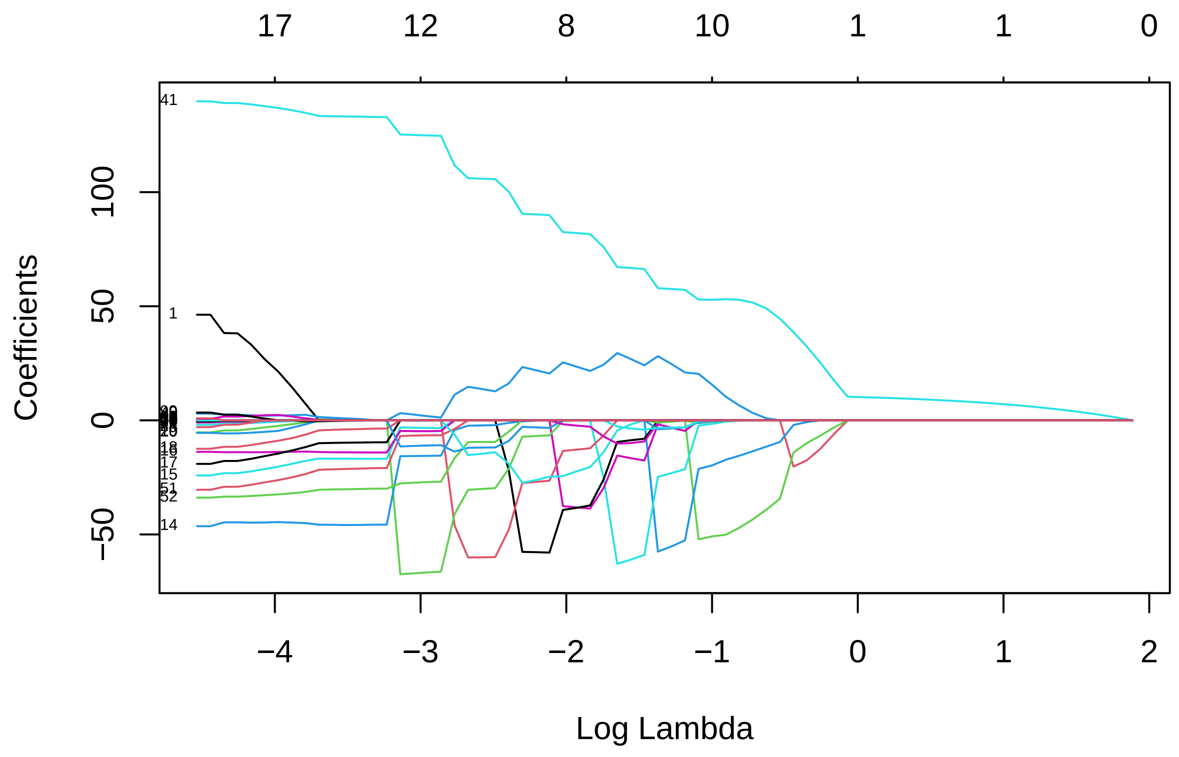


```
# choosing best value for lambda using cross validation
cv_ridge = cv.glmnet(as.matrix(covariates), response, alpha = 0, family="gaussian")
cat("The optimal lambda for Ridge regression is", (cv_ridge$lambda.min))
```

```
## The optimal lambda for Ridge regression is 1.150511
```

4.6 LASSO Regression

```
# LASSO Regression
lasso_model = glmnet(as.matrix(covariates), response, alpha = 1, family = "gaussian")
plot(lasso_model, xvar = "lambda", label = TRUE)
```



```r
cv_lasso = cv.glmnet(as.matrix(covariates), response, alpha = 1, family="gaussian")
cat("The optimal lambda for LASSO is", (cv_lasso$lambda.min))
```

```
## The optimal lambda for LASSO is 0.01072971
```

In ridge regression the penalty factor λ penalised all the coefficients. With increase in lambda, the shinkage is gradually increased upto 0 but it was never exactly 0. On the other hand, with LASSO regression, the penalisation happend quick as we take the absolute values of coefficients and coefficients of all the less significant features were shinked to 0 with the increase in lambda and only coefficients that are not equal to zero are considered in the model.

4.7 LASSO Cross validation

```r
# LASSO CV
cv_lasso = cv.glmnet(as.matrix(covariates), response, alpha = 1, family="gaussian")
cat("The optimal lambda for LASSO is", (cv_lasso$lambda.min))
```

```
## The optimal lambda for LASSO is 0.01072971
```

```r
lasso_coeff = coef.glmnet(cv_lasso, s = cv_lasso$lambda.min)
select_coeff = lasso_coeff[which(lasso_coeff[,1] != 0)]
cat("\n\nNumber of coefficients selected in LASSO are", length(select_coeff)-1)# -1 is to remove interc
```

```
##
##
## Number of coefficients selected in LASSO are 22
```