Assignment 1 Statistical Machine Learning

Darix SAMANI SIEWE

07/12/2024

```
library(lattice)
library(ggplot2)
library(caret)
require(reshape2)
```

```
## Loading required package: reshape2
```

Part 1: Dataset Exploration

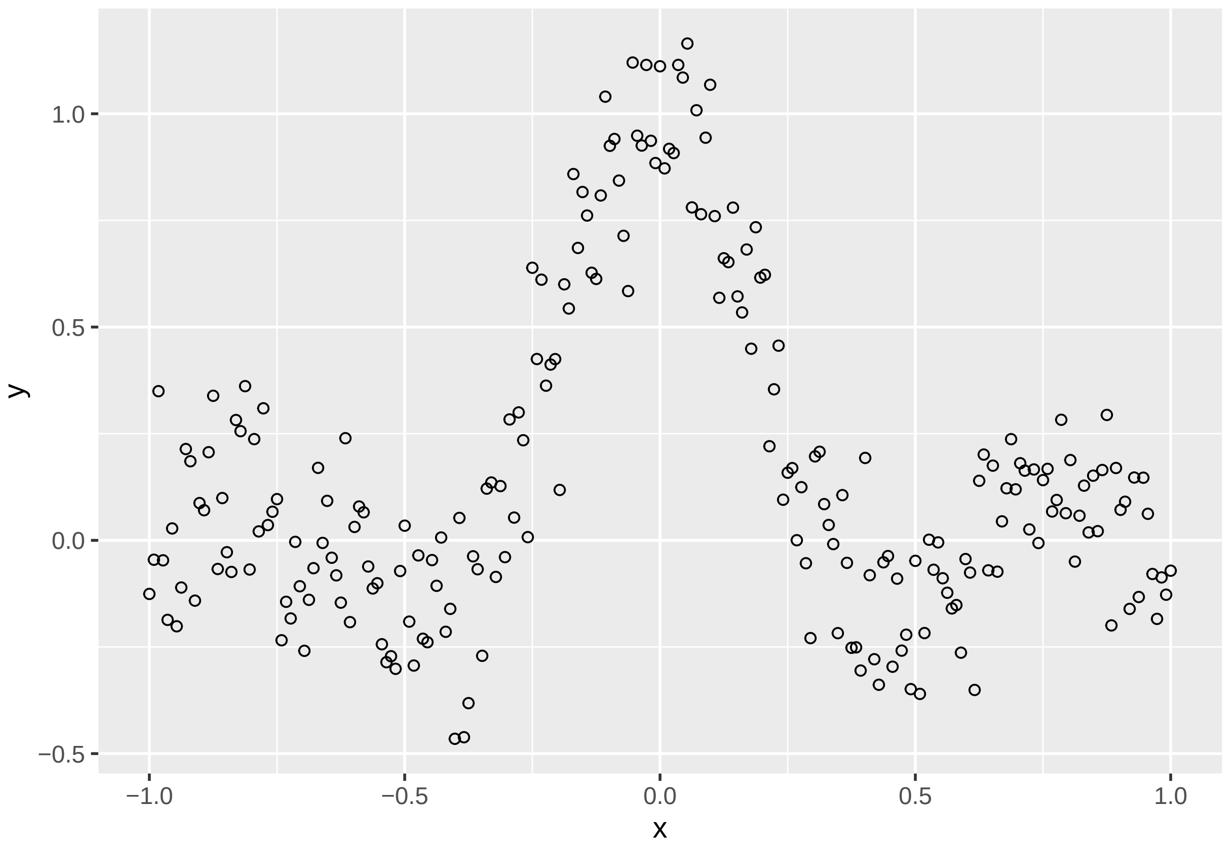
1. Download the file aims-sml-2024-2025-data.csv and load it into RMarkDown.
2. Determine the size of the dataset n.

```
data = read.csv("aims-sml-2024-2025-data.csv")
n = nrow(data)
print(paste("The size of our dataset is: ", n))
```

```
## [1] "The size of our dataset is:  225"
```

3. Create a scatter plot of y versus x.

```
ggplot(data, aes(x=x, y=y)) +
    geom_point(shape=1)
```



4. Determine whether this is a classification or regression task, and justify your answer.

This is the regression because the response variable are continuous random variable and based on the shape of our scatter plot, the shape tell us this a regression task, because the shape of scatter plot of our y versus x have a quadratic pattern. we can interpolate the shape of our dataset with polynomial newton or Lagrange.

Part 2: Theoretical Framework

1. Suggest a function space H for this task and write it in mathematical notation.

The function space in this case is the space of the polynomial function the expansion expressed can be written as:

$$\mathbb{H} := \{f \qquad \text{such as} \qquad f(x) = A_0 + A_1x + A_2x^2 + A_3x^3 + \ldots + A_px^p, \qquad x^i \in \mathbb{R} \qquad and \qquad A_i \in \mathbb{R}\}$$

and the general expression can be written as :

$$\mathbb{H} := \{f \qquad \text{such as} \qquad f(x) = \sum_{i=0}^{p} A_ix^i \qquad x \in \mathbb{X} \qquad \text{and} \qquad A_j \in \mathbb{R}\}$$

$$\mathbb{H} := \{f \qquad \text{such as} \qquad f(x) = AX \qquad X \in \mathbb{R}^{nxp} \qquad \text{and} \qquad A \in \mathbb{R}^p\}$$

where $A = [A_0, A_1, \ldots, A_p]^T$ and $X = [1, x, x^2, \ldots, x^p]$

2. Specify the loss function for this task and justify its use.

Since the Function Space is Polynomial function intuitively we can use $\mathcal{L}_2$ is suitable best for this task.

The expression of the loss function can be written as:

$$\mathcal{L}(y, f(x)) = (y - f(x))^2$$

3. Derive and write down the theoretical risk R(f) for a candidate function $f \in \mathcal{H}$.

The theoretical risk is the expectation of the loss function the general expression can be written as:

$$R(f) = \mathbb{E}(\mathcal{L}(y, f(x)))R(f) = \int_{\mathbb{X}X\mathbb{Y}} \mathcal{L}(y, f(x))P_{xy}dxdy$$

where P_{XY} is the joint probability density function of the population and $\mathcal{L}$ is the loss function that we express bellow.

4. Write down the expression for the Bayes learning machine $f^*(x)$ in this case.

In the function space $f^*(x)$ is a subset of $\mathcal{H}$ where we can locate the best. Hence $f^*(x)$ is the universal of the best function. Bayes learning machine is the universal of the best in function space, so his general expression can be express as :

$$f^*(x) := arinf(R(f))$$

5. Write down the empirical risk $\hat{R}(f)$ for a candidate function, including all components of the model space. Hint: Decide on a complexity prior to training.

$$\hat{R}(f) = \frac{1}{n}\sum_{i=0}^{n} \mathcal{L}(y_i, \hat{f}(x_i))\hat{R}(f) = \frac{1}{n}\sum_{i=0}^{n}(y_i - \sum_{i=0}^{p} A_i x^i)^2$$

where p is the degree of the polynomial and where $\hat{f}(x_i)$ is the predicted value of our model.

since the scatter plot of our model is like the square function, let's consider a candidature function $f = A_0 + A_1x + A_2x^2$ and our empirical risk become:

$$\hat{R}(f) = \frac{1}{n}\sum_{i=0}^{n}(y_i - (A_0 + A_1x + A_2x^2))^2$$

The complexity of our model is based on the degree of p, the high degree can interpolate almost all the points but but subject to overfitting and the lower degree is not able to capture all the points. Hence choose, the better function space in the function space is the function that able to interpolate almost all the data without extrapolate.

Part 3: Estimation and Model Complexity

1. Derive the expression for the OLS estimator $\hat{f}(x)$ for this problem.

$$f(x) = A_0 + A_1x + \ldots + A_px^p + \epsilon$$

we can write this expression as $f(x) = XA + \epsilon$ where X is the design matrix and A the parameter estimator of our machine or model and ϵ is the error who follow normal distribution.

The expression of X(design matrix) can be expressed as:

$$X = \begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^p \\ 1 & x_2 & x_2^2 & \cdots & x_2^p \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^p \end{bmatrix}$$

and general expression of A can be expressed as : $A = [A_0, A_1, A_2, \ldots, A_n]^T$, A is the parameter of our model in this case, it is regression task(polynomial regression with one variable).

where p is the degree of the polynomial and n is the size of the sample.

and $A = [a_0, a_1, \ldots, a_p]^T$

The Sum of Squared Errors (SSE) is defined as $SSE(A) = (Y - XB)^T(Y - XB)$

where Y is the vector of observed values.

To minimize $SSE(A)$, we take the derivative with respect to A and set it to zero:

$$\frac{\partial}{\partial A} SSE(A) = -2X^T(Y - XA) = -2X^TY + 2X^TXA = 0.$$

simplify expression we have:

$$X^TXA = X^TY.$$

If X^TX is invertible, the solution is:

$$A = (X^TX)^{-1}X^TY.$$

The estimated function $\hat{f}(x)$ is then:

$$\hat{f}(x) = X\hat{A} = X(X^TX)^{-1}X^TY.$$

2. Comment extensively on the properties of $\hat{f}(x)$.

since $\hat{f}(x) = XA + \epsilon$, $\hat{f}$ is a multiple linear model(polynomial regression with one variable),

- if $n > p$ the matrix X^TX which is design matrix is invertible.
- $\mathbb{E}(\hat{A}) = A$: $\hat{A}$ is a unbiased estimator since $\mathbb{E}(\epsilon) = 0$
- $\mathbb{E}(\hat{f}) = f(x)$
- $var(\hat{f}(x)) = \sigma^2 X(X^TX)^{-1}X^T$ where σ^2 is

3. Use V-fold cross-validation (e.g., V =, 5, 10) to determine the optimal complexity (degree

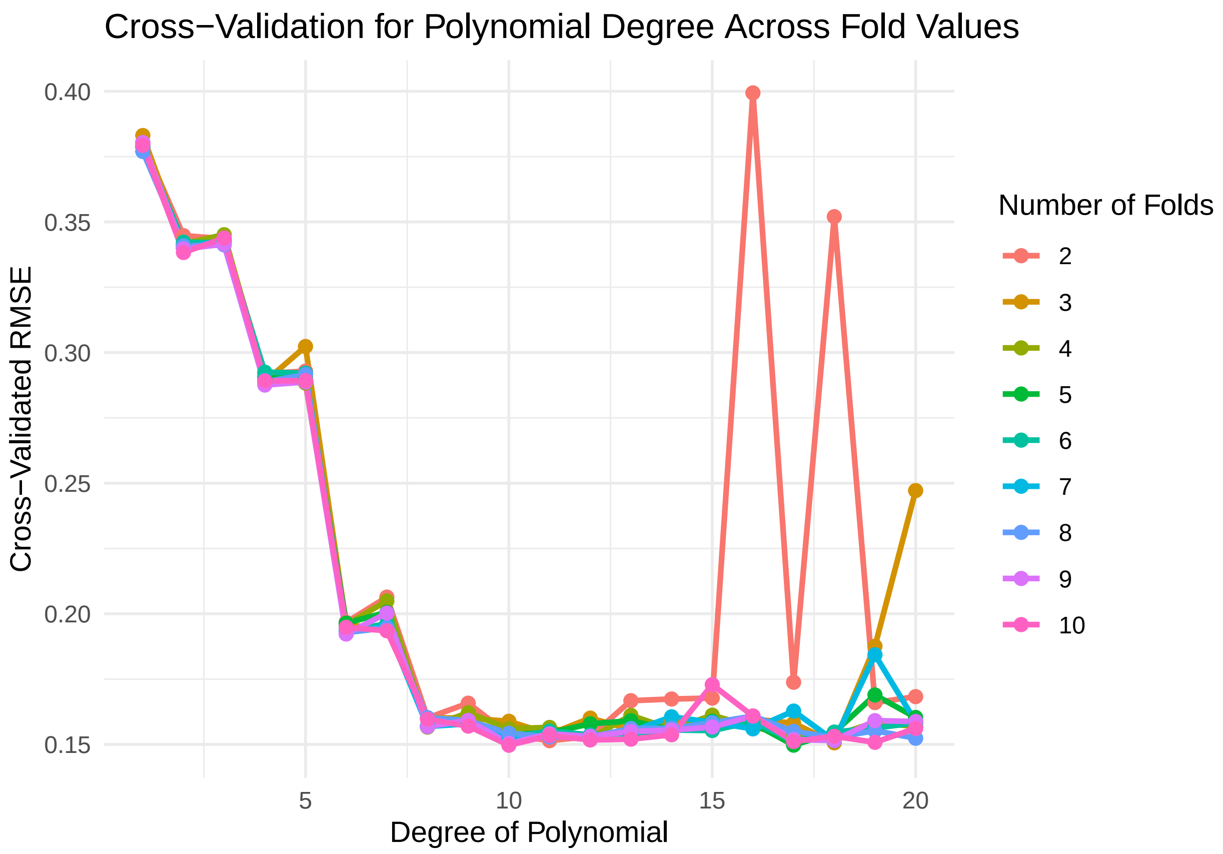
p) for the polynomial regression model. Explain what "optimal complexity" means.

```r
set.seed(20241208)


max_degree <- 20  # Maximum degree of the polynomial
fold_range <- 2:10  # Range of folds for cross-validation
set.seed(20241207)

# this function help us to calculate the cross_validation error given folds and degree
```

```r
cv_error <- function(degree, data, folds) {
  formula <- as.formula(paste("y ~ poly(x, ", degree, ", raw = TRUE)"))
  train_control <- trainControl(method = "cv", number = folds)
  model <- train(formula, data = data, method = "lm", trControl = train_control)
  return(mean(model$resample$RMSE))
}

# This function help us to calculate empirical risk given degree of the polynomial.

empirical_risk <- function(degree, data) {
  formula <- as.formula(paste("y ~ poly(x, ", degree, ", raw = TRUE)"))
  model <- lm(formula, data = data)
  predictions <- predict(model, data)
  return(mean((data$y - predictions)^2))  # Empirical risk (training error)
}

# create the dataframe to calculate the result.
results <- data.frame()

# This dataframe help us to store optimal degree
optimal_degrees <- data.frame(Folds = integer(), Optimal_Degree = integer())



for (folds in fold_range) {
  degrees <- 1:max_degree
  cv_errors <- sapply(degrees, cv_error, data = data, folds = folds)

  # Find the degree with the minimum error
  optimal_degree <- which.min(cv_errors)

  # Store the results
  optimal_degrees <- rbind(optimal_degrees, data.frame(Folds = folds, Optimal_Degree = optimal_degree))

  # Print results for this fold
  cat("Folds:", folds, "Optimal Degree:", optimal_degree, "\n")

  # Add results to data frame
  results <- rbind(
    results,
    data.frame(Folds = folds, Degree = degrees, CV_Error = cv_errors)
  )
}
```

```
## Folds: 2 Optimal Degree: 11
## Folds: 3 Optimal Degree: 18
## Folds: 4 Optimal Degree: 12
## Folds: 5 Optimal Degree: 17
## Folds: 6 Optimal Degree: 17
## Folds: 7 Optimal Degree: 18
## Folds: 8 Optimal Degree: 20
## Folds: 9 Optimal Degree: 10
## Folds: 10 Optimal Degree: 10
```

```r
# Plot using ggplot2
ggplot(results, aes(x = Degree, y = CV_Error, color = factor(Folds))) +
  geom_line(linewidth = 1) +
  geom_point(size = 2) +
  labs(
    title = "Cross-Validation for Polynomial Degree Across Fold Values",
    x = "Degree of Polynomial",
    y = "Cross-Validated RMSE",
    color = "Number of Folds"
  ) +
  theme_minimal()
```



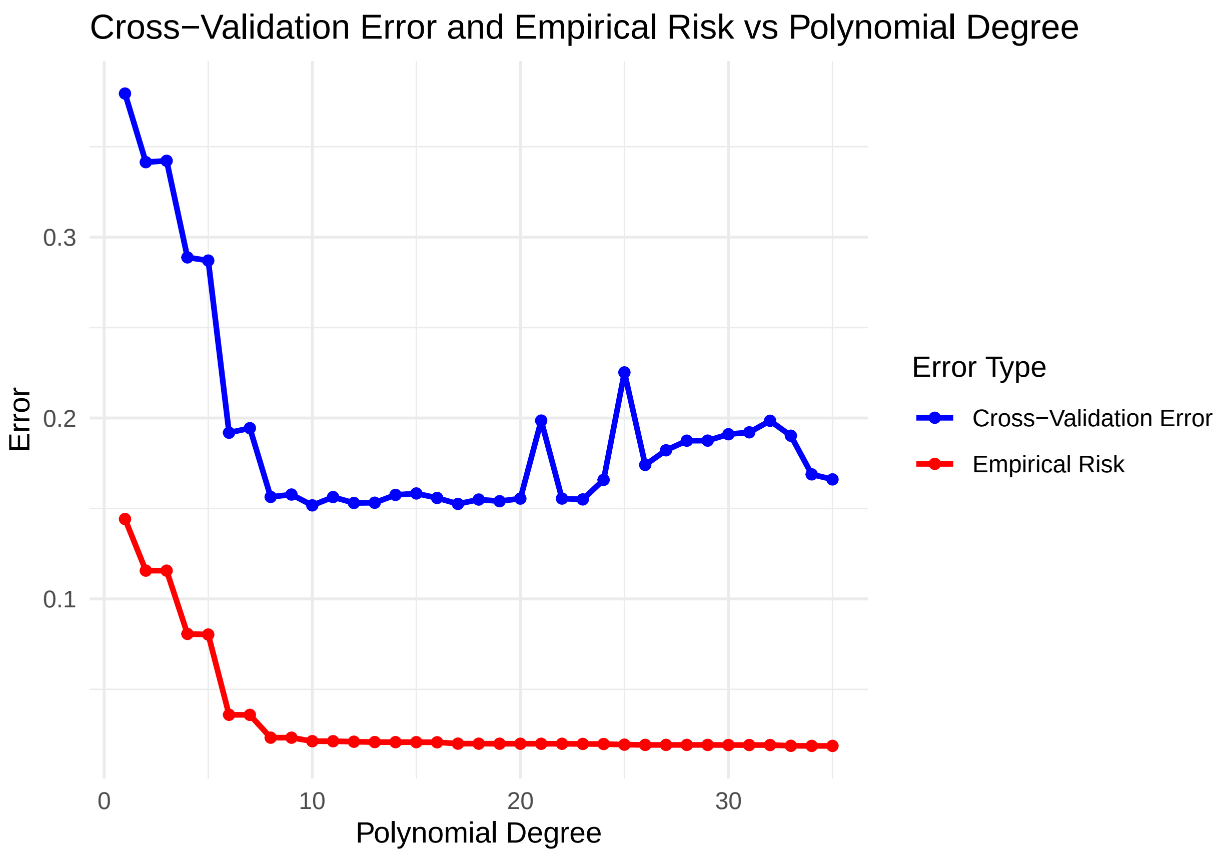
Explanation of the "optimal complexity"

In the context of polynomial regression, the optimal complexity of polynomial refers to the degree of the polynomial that balance between bias and variance. The hight degree of polynomial regression model, can capture more complex data but subject to over fitting in the model, and the lower degree of polynomial is subject to underfitting(the bias is hight), The optimal complexity of the model is the degree that can capture almost all data without add mode complexity. we can see that best value of v-fold is 10 because with this value the cross validation error have the batter stability over the degree p.

4. Plot the cross-validation error and empirical risk as functions of p. Comment on the plot.

```r
set.seed(20241208)


max_degree <- 35   # Maximum polynomial degree
fold_range <- 10   # Number of folds for cross-validation
```

```
# Data frame to store errors
results <- data.frame(Degree = integer(), CV_Error = numeric(), Empirical_Risk = numeric())

# Calculate errors for each degree
for (degree in 1:max_degree) {
  cv_err <- cv_error(degree, data, fold_range)
  emp_risk <- empirical_risk(degree, data)

  results <- rbind(
    results,
    data.frame(Degree = degree, CV_Error = cv_err, Empirical_Risk = emp_risk)
  )
}

# Plot cross-validation error and empirical risk
ggplot(results, aes(x = Degree)) +
  geom_line(aes(y = CV_Error, color = "Cross-Validation Error"), size = 1) +
  geom_point(aes(y = CV_Error, color = "Cross-Validation Error")) +
  geom_line(aes(y = Empirical_Risk, color = "Empirical Risk"), size = 1) +
  geom_point(aes(y = Empirical_Risk, color = "Empirical Risk")) +
  labs(
    title = "Cross-Validation Error and Empirical Risk vs Polynomial Degree",
    x = "Polynomial Degree",
    y = "Error"
  ) +
  scale_color_manual(name = "Error Type", values = c("Cross-Validation Error" = "blue", "Empirical Risk
  theme_minimal()
```



```r
# Find the degree with the minimum cross-validation error
optimal_degree <- results$Degree[which.min(results$CV_Error)]
cat("Optimal Polynomial Degree (Based on Cross-Validation Error):", optimal_degree, "\n")
```

```
## Optimal Polynomial Degree (Based on Cross-Validation Error): 10
```

Comment on the Plot

based on the plot above, we can see that the universal of the best is the degree of the polynomial from range 10 to 12, and after the the degree 13 the cross validation is most greater than empirical risk that means after the degree 13 the the model is subject to overfitting (the variance is very high).

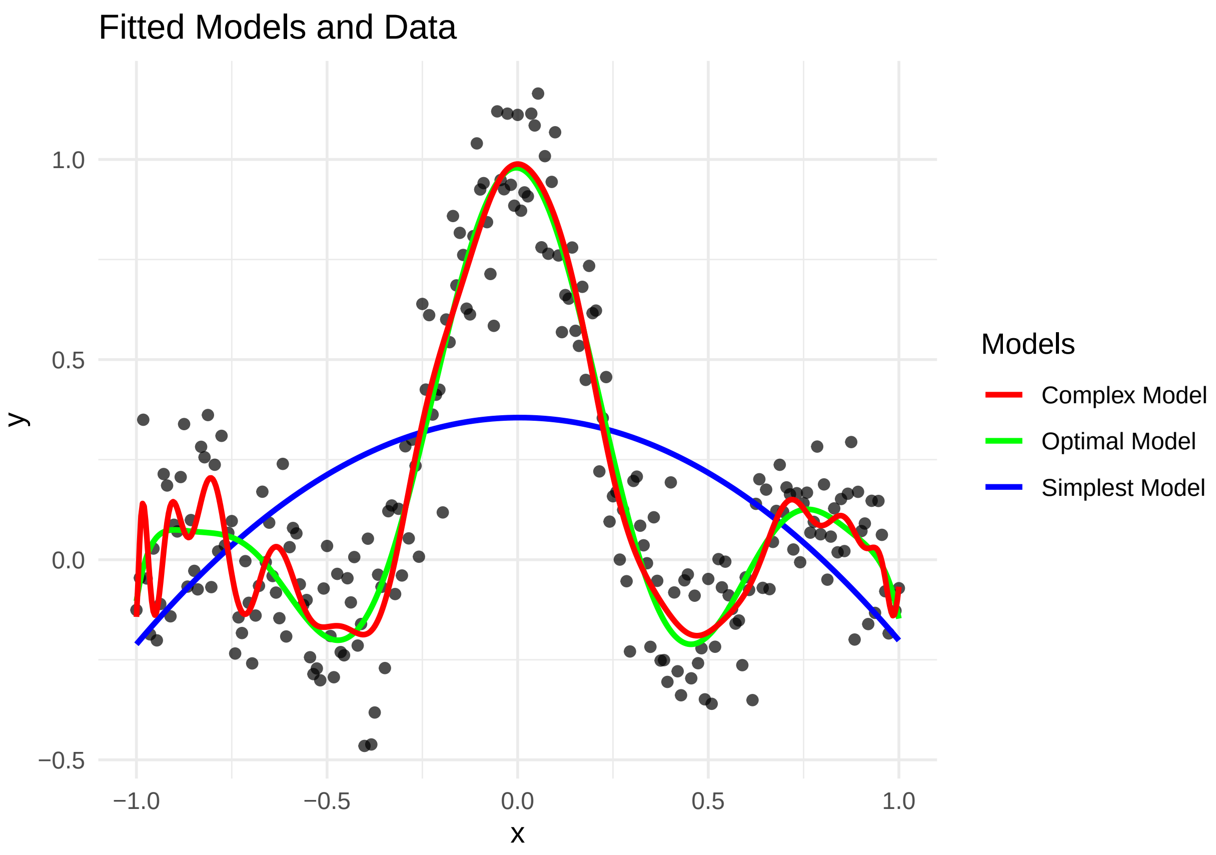
Part 4: Model Comparison and Evaluation

1. Fit and plot the following models on the same plot with the data:

- The simplest estimator $\hat{f}(x)$ that depends on x: in our case the simplest estimator that depends on x is a naive estimator which is model with degree 2(we can see it graphically)
- The optimal estimator determined by cross-validation: the optimal estimator is the model with optimal degree the plot of our cross validation show us that the optimal degree is p
- An overly complex model: in the case we can choice any degree of polynomial that is subject to overfitting Use a legend to distinguish the models and comment on their behaviors.

```r
set.seed(20241208)
optimal_degree = 10
max_degree = 35

## defined our three model
```

```r
simplest_model <- lm(y ~ poly(x, 2, raw = TRUE), data = data)  # Simplest model (linear)
optimal_model <- lm(y ~ poly(x, optimal_degree, raw = TRUE), data = data)  # Optimal model
complex_model <- lm(y ~ poly(x, max_degree, raw = TRUE), data = data)  # Overly complex model

# Predictions
x_pred <- seq(min(data$x), max(data$x), length.out = 500)
data_pred <- data.frame(x = x_pred)
data_pred$simplest <- predict(simplest_model, newdata = data_pred)
data_pred$optimal <- predict(optimal_model, newdata = data.frame(x = x_pred))
data_pred$complex <- predict(complex_model, newdata = data.frame(x = x_pred))

# Plot the data and models
ggplot(data, aes(x = x, y = y)) +
  geom_point(color = "black", size = 1.5, alpha = 0.7) +
  geom_line(data = data_pred, aes(x = x_pred, y = simplest, color = "Simplest Model"), size = 1) +
  geom_line(data = data_pred, aes(x = x_pred, y = optimal, color = "Optimal Model"), size = 1) +
  geom_line(data = data_pred, aes(x = x_pred, y = complex, color = "Complex Model"), size = 1) +
  labs(
    title = "Fitted Models and Data",
    x = "x",
    y = "y"
  ) +
  scale_color_manual(name = "Models", values = c(
    "Simplest Model" = "blue",
    "Optimal Model" = "green",
    "Complex Model" = "red"
  )) +
  theme_minimal()
```



2. Perform stochastic hold-out validation with S = 100 splits (70% training, 30% testing).

Compute and plot boxplots of the test errors for:

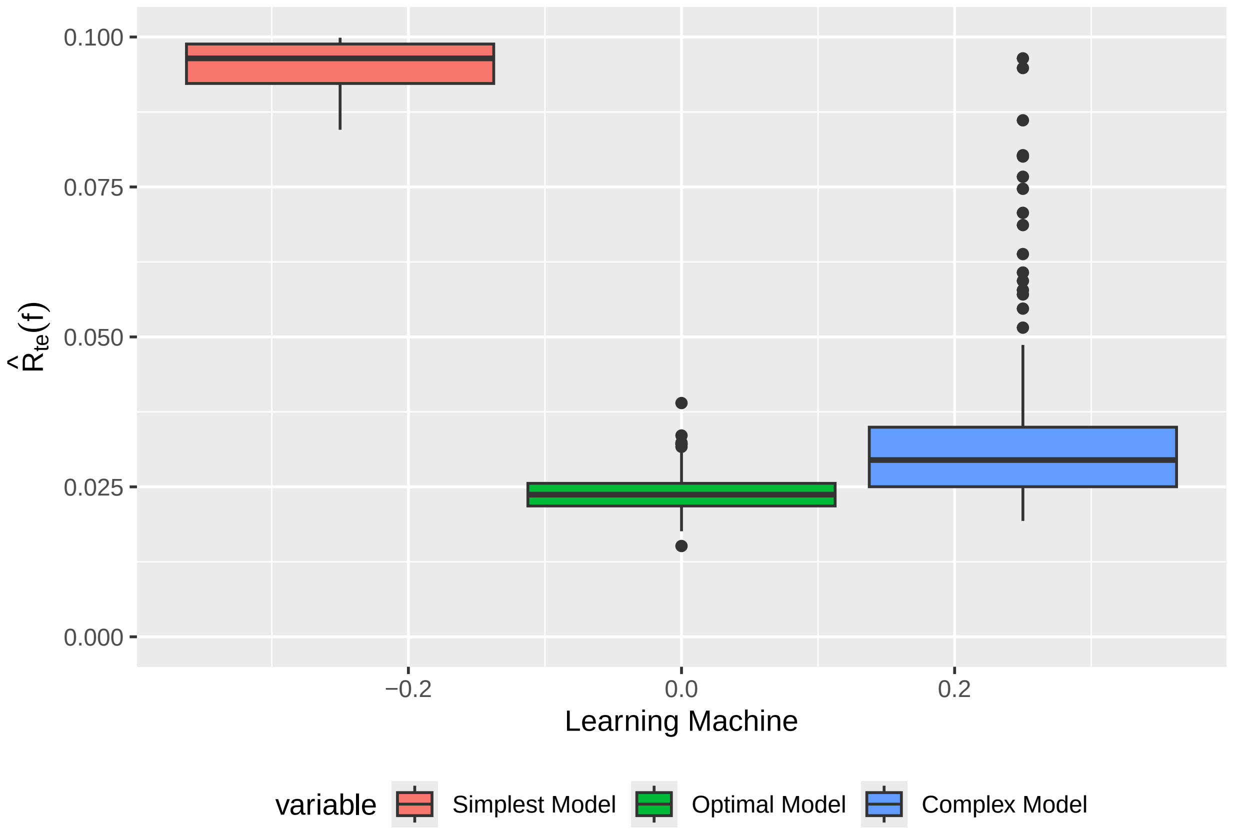
- The simplest model.
- The optimal model.
- The overly complex model.

```r
set.seed(20241208)
epsilon <- 3/10              # Proportion of observations in the test set
nte     <- round(n*epsilon)  # Number of observations in the test set
ntr     <- n - nte

S <- 100   # Number of replications
test.err <- matrix(0, nrow=S, ncol=3)

for(s in 1:S)
{
  # Split the data

  id.tr   <- sample(sample(sample(n)))[1:ntr]                   # For a sample of ntr indices from {
  id.te   <- setdiff(1:n, id.tr)


  y.te         <- data$y[id.te]                                   # True responses in test set

  # Simplest model
  simplest_model <- lm(y ~ poly(x, 2, raw = TRUE), data = data[id.tr,])
  y.te.hat       <- predict(simplest_model, newdata = data[id.te, ])      # Predicted responses in tes
```

```
    test.err[s,1]  <- mean(((y.te - y.te.hat))^2)

    # Optimal Model
    optimal_model <- lm(y ~ poly(x, optimal_degree, raw = TRUE), data = data[id.tr,])
    y.te.hat       <- predict(optimal_model, newdata =  data[id.te, ])        # Predicted responses in
    test.err[s,2]  <- mean(((y.te - y.te.hat))^2)

    # Complex Model
    complex_model <- lm(y ~ poly(x, max_degree, raw = TRUE), data = data[id.tr,])
    y.te.hat       <- predict(complex_model, newdata =  data[id.te, ])        # Predicted responses in
    test.err[s,3]  <- mean(((y.te - y.te.hat))^2)

  }
  test <- data.frame(test.err)
  Method<-c('Simplest Model', 'Optimal Model', 'Complex Model')
  colnames(test) <- Method
```

The Box plot of the test error

```
ggplot(data = melt(test, id.vars = NULL), xlim = c(0, 0.1),  aes(x=, y=value)) + geom_boxplot(aes(fil
  labs(x='Learning Machine', y=expression(hat(R)[te](f))) +
  scale_y_continuous(limits = c(0, 0.1)) +
  theme(legend.position="bottom")
```



The box plot above show us that the complex model have more outliers which means the complex model is subject to overfitting in unknown data and the optimal complex we can see that all the data is in LB(Lower Bound) and UP(Upper Bound) that means in this model there are no outliers.

Part 5: Further Analysis

1. Perform an analysis of variance (ANOVA) on the test errors. What does it reveal?

```
aov.method <- aov(value~variable, data=melt(test, id.vars = NULL))
anova(aov.method)
```

```
## Analysis of Variance Table
##
## Response: value
##            Df  Sum Sq Mean Sq F value  Pr(>F)
## variable    2   40.32 20.1580  3.5157 0.03097 *
## Residuals 297 1702.90  5.7337
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

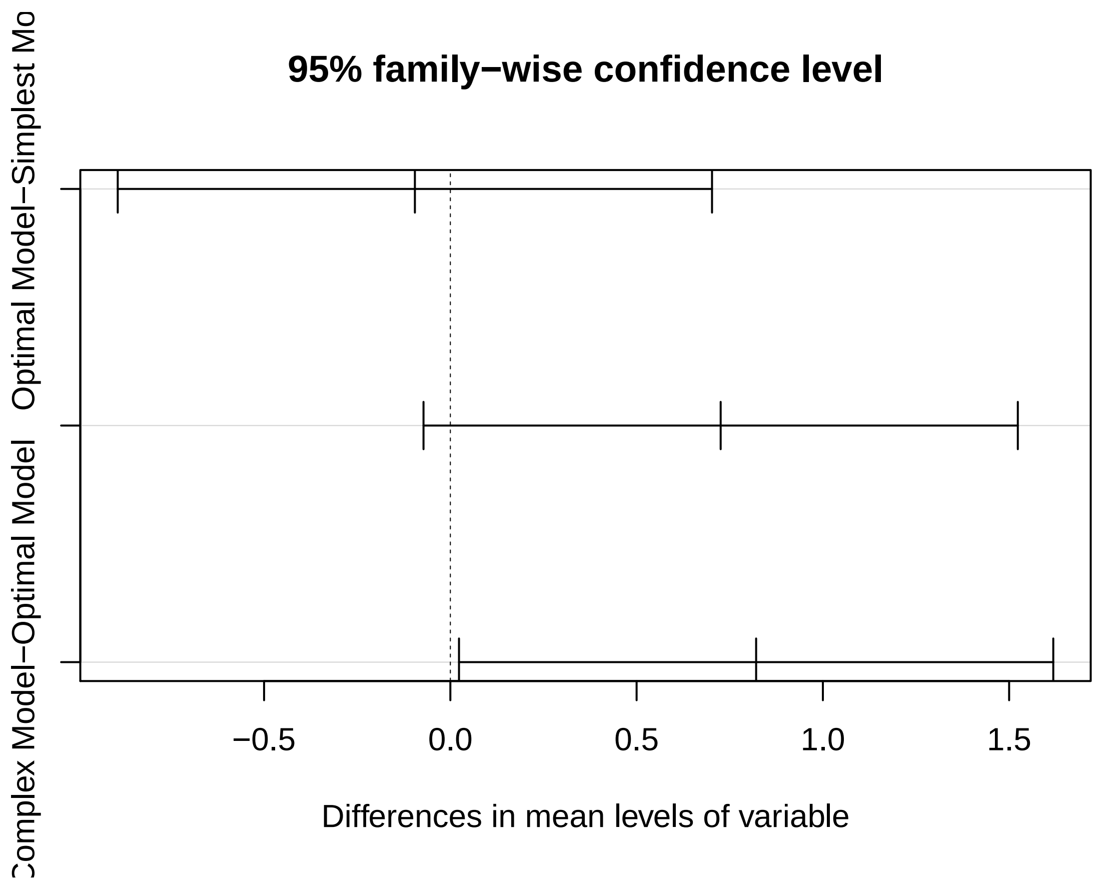
```
summary(aov.method)
```

```
##              Df Sum Sq Mean Sq F value Pr(>F)
## variable      2   40.3  20.158   3.516  0.031 *
## Residuals   297 1702.9   5.734
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
TukeyHSD(aov.method, ordered = TRUE)
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##     factor levels have been ordered
##
## Fit: aov(formula = value ~ variable, data = melt(test, id.vars = NULL))
##
## $variable
##                                    diff         lwr       upr     p adj
## Simplest Model-Optimal Model 0.09519332 -0.70246990 0.8928565 0.9573813
## Complex Model-Optimal Model  0.82086411  0.02320089 1.6185273 0.0420519
## Complex Model-Simplest Model 0.72567079 -0.07199243 1.5233340 0.0830827
```

```
#plot version
plot(TukeyHSD(aov.method))
```

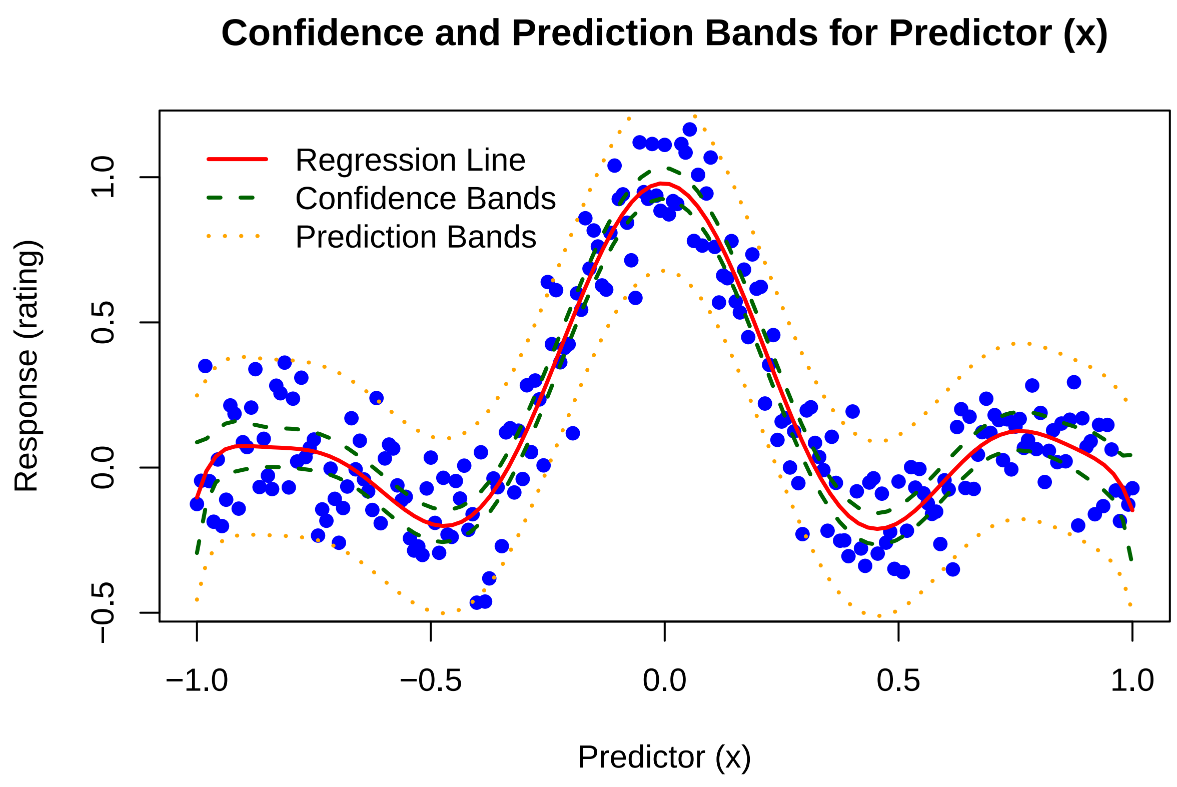


The Analysis of variance show us that us that Simplest Model-Optimal Model and complex Model-Simplest Model is not significant and the complex Model-Optimal Model is significant and the model, Simplest Model-Optimal Model and complex Model-Simplest Model contains the value 0 which means that the estimator parameter can be zero.

2. Obtain and plot the 95% confidence and prediction bands for the dataset D_n.

```
# Define the response and best predictor
response <- data$y
best_predictor_values <- data$x

best_model <- lm(y ~ poly(x, optimal_degree, raw = TRUE), data = data)

# Generate a sequence of predictor values for smoother bands
x_new <- seq(min(best_predictor_values), max(best_predictor_values), length.out = 100)

# Create a new data frame for predictions
new_data <- data.frame(x = x_new)

# Compute the confidence intervals and prediction intervals
predictions <- predict(
  best_model,
  newdata = new_data,
  interval = "confidence", # Confidence bands
  level = 0.95             # 95% confidence level
)
```

```r
predictions_pred <- predict(
  best_model,
  newdata = new_data,
  interval = "prediction", # Prediction bands
  level = 0.95             # 95% prediction level
)

# Plot the data and the regression line
plot(
  best_predictor_values, response,
  main = paste("Confidence and Prediction Bands for", "Predictor (x)"),
  xlab = "Predictor (x)",
  ylab = "Response (rating)",
  pch = 16, col = "blue"
)

# Add the regression line
lines(x_new, predictions[, "fit"], col = "red", lwd = 2)

# Add the confidence bands
lines(x_new, predictions[, "lwr"], col = "darkgreen", lwd = 2, lty = 2)
lines(x_new, predictions[, "upr"], col = "darkgreen", lwd = 2, lty = 2)

# Add the prediction bands
lines(x_new, predictions_pred[, "lwr"], col = "orange", lwd = 2, lty = 3)
lines(x_new, predictions_pred[, "upr"], col = "orange", lwd = 2, lty = 3)

# Add a legend
legend(
  "topleft",inset=0.02,
  legend = c("Regression Line", "Confidence Bands", "Prediction Bands"),
  col = c("red", "darkgreen", "orange"),
  lty = c(1, 2, 3),
  lwd = 2,
  bty = "n"
)
```



3. Write the mathematical expression for:

- The confidence band for a single observation (X_i, Y_i).

Confidence band represented the uncertainly to estimate based on limited data,

$$p(\hat{f}(x) - w(x_i) \leq y_i \leq \hat{f}(x) + w(x_i))$$

where wx_i is the confidence interval

- The prediction band for a single observation (X_i, Y_i).

  The prediction band is used to represent the uncertainty about the new value data in the curve, but with noise

$$p(\hat{f}(x) - w(x_i) \leq y^* \leq \hat{f}(x) + w(x_i))$$

  where y^* is an observation taken from data-generating process at given point x_i that independent of the data used to construct the point estimate.

4. Comment extensively on what the confidence and prediction bands reveal about the model.

The figure above show us that the confidence band is must closed to our model that means the uncertainly to predict a new data to very closed to the real value and in the others hand, the prediction band contains almost our D_n which means the uncertainly to predict the new data will be in the curve.

Exercise 2

```
#import requirement document for this part
library(kernlab)
```

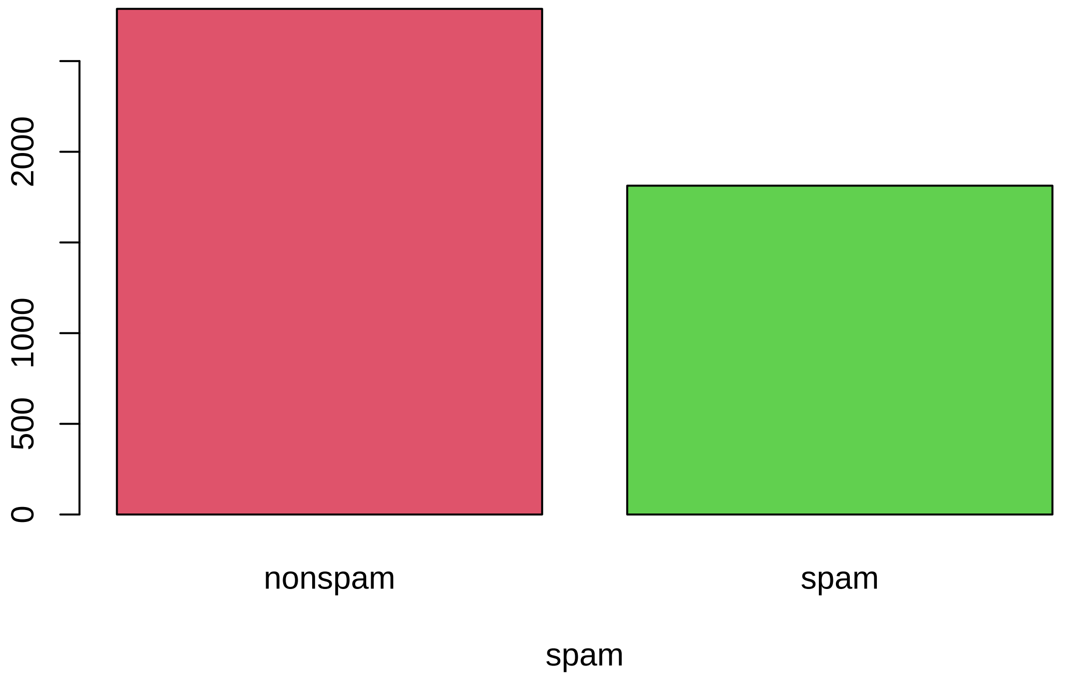
```
##
## Attaching package: 'kernlab'
## The following object is masked from 'package:ggplot2':
##
##     alpha
```

```
library(kernlab)   # For the spam dataset (optional if using your own dataset)
library(MASS)      # For LDA and QDA
library(e1071)     # For Naive Bayes
library(ROCR)
library(klaR) # we use this RDA(Regularized QDA) that solve error
```

Consider the spam dataset from library(kernlab). You are supposed to provide a thorough comparison of four learning machines namely LDA, QDA, Naive Bayes and FLD, and your comparison

will be solely based on the test error. 1. Plot the distribution of the response for this dataset and comment.

```
data(spam)
data_spam <- spam

barplot(table(data_spam$type), col=2:3, xlab='spam')
```



The distribution above show us that 50% for our dataset are spam

2. Comment on the shape of this dataset in terms of the sample size and the dimensionality of the input space

```
n = nrow(data_spam) #  number of sample
p   <- ncol(data_spam) - 1   # Dimensionality of the input space

# Kapper calculation
kapper <- n / p

cat("Kapper (size divided by input space) is:", kapper, "\n")
```

```
## Kapper (size divided by input space) is: 80.7193
```

Since the kapper is 80 that means that we have in the case of infra hight dimension, another intrepretation is the dataset have approximation, 80% samples per dimension so we can use it for many machine leaning task

3. Comment succinctly from the statistical perspective on the type of data in the input space

```
X <- data_spam[,1:57]
```

```
summary(data_spam[,1:57]) ## summary of Input space
```

```
##       make           address            all              num3d         
##  Min.   :0.0000   Min.   : 0.000   Min.   :0.0000   Min.   : 0.00000  
##  1st Qu.:0.0000   1st Qu.: 0.000   1st Qu.:0.0000   1st Qu.: 0.00000  
##  Median :0.0000   Median : 0.000   Median :0.0000   Median : 0.00000  
##  Mean   :0.1046   Mean   : 0.213   Mean   :0.2807   Mean   : 0.06542  
##  3rd Qu.:0.0000   3rd Qu.: 0.000   3rd Qu.:0.4200   3rd Qu.: 0.00000  
##  Max.   :4.5400   Max.   :14.280   Max.   :5.1000   Max.   :42.81000  
##       our               over            remove          internet      
##  Min.   : 0.0000   Min.   :0.0000   Min.   :0.0000   Min.   : 0.0000  
##  1st Qu.: 0.0000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.: 0.0000  
##  Median : 0.0000   Median :0.0000   Median :0.0000   Median : 0.0000  
##  Mean   : 0.3122   Mean   :0.0959   Mean   :0.1142   Mean   : 0.1053  
##  3rd Qu.: 0.3800   3rd Qu.:0.0000   3rd Qu.:0.0000   3rd Qu.: 0.0000  
##  Max.   :10.0000   Max.   :5.8800   Max.   :7.2700   Max.   :11.1100  
##      order              mail             receive              will       
##  Min.   :0.00000   Min.   : 0.0000   Min.   :0.00000   Min.   :0.0000  
##  1st Qu.:0.00000   1st Qu.: 0.0000   1st Qu.:0.00000   1st Qu.:0.0000  
##  Median :0.00000   Median : 0.0000   Median :0.00000   Median :0.1000  
##  Mean   :0.09007   Mean   : 0.2394   Mean   :0.05982   Mean   :0.5417  
##  3rd Qu.:0.00000   3rd Qu.: 0.1600   3rd Qu.:0.00000   3rd Qu.:0.8000  
##  Max.   :5.26000   Max.   :18.1800   Max.   :2.61000   Max.   :9.6700  
##     people             report            addresses           free        
##  Min.   :0.00000   Min.   : 0.00000   Min.   :0.0000   Min.   : 0.0000  
##  1st Qu.:0.00000   1st Qu.: 0.00000   1st Qu.:0.0000   1st Qu.: 0.0000  
##  Median :0.00000   Median : 0.00000   Median :0.0000   Median : 0.0000  
##  Mean   :0.09393   Mean   : 0.05863   Mean   :0.0492   Mean   : 0.2488  
##  3rd Qu.:0.00000   3rd Qu.: 0.00000   3rd Qu.:0.0000   3rd Qu.: 0.1000  
##  Max.   :5.55000   Max.   :10.00000   Max.   :4.4100   Max.   :20.0000  
##    business          email             you              credit        
##  Min.   :0.0000   Min.   :0.0000   Min.   : 0.000   Min.   : 0.00000  
##  1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.: 0.000   1st Qu.: 0.00000  
##  Median :0.0000   Median :0.0000   Median : 1.310   Median : 0.00000  
##  Mean   :0.1426   Mean   :0.1847   Mean   : 1.662   Mean   : 0.08558  
##  3rd Qu.:0.0000   3rd Qu.:0.0000   3rd Qu.: 2.640   3rd Qu.: 0.00000  
##  Max.   :7.1400   Max.   :9.0900   Max.   :18.750   Max.   :18.18000  
##       your              font             num000           money         
##  Min.   : 0.0000   Min.   : 0.0000   Min.   :0.0000   Min.   : 0.00000  
##  1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.:0.0000   1st Qu.: 0.00000  
##  Median : 0.2200   Median : 0.0000   Median :0.0000   Median : 0.00000  
##  Mean   : 0.8098   Mean   : 0.1212   Mean   :0.1016   Mean   : 0.09427  
##  3rd Qu.: 1.2700   3rd Qu.: 0.0000   3rd Qu.:0.0000   3rd Qu.: 0.00000  
##  Max.   :11.1100   Max.   :17.1000   Max.   :5.4500   Max.   :12.50000  
##        hp               hpl              george            num650      
##  Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.0000   Min.   :0.0000  
##  1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.:0.0000  
##  Median : 0.0000   Median : 0.0000   Median : 0.0000   Median :0.0000  
##  Mean   : 0.5495   Mean   : 0.2654   Mean   : 0.7673   Mean   :0.1248  
##  3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.:0.0000
##  Max.   :20.8300   Max.   :16.6600   Max.   :33.3300   Max.   :9.0900
##       lab              labs            telnet           num857
##  Min.   : 0.00000   Min.   :0.0000   Min.   : 0.00000   Min.   :0.00000
##  1st Qu.: 0.00000   1st Qu.:0.0000   1st Qu.: 0.00000   1st Qu.:0.00000
##  Median : 0.00000   Median :0.0000   Median : 0.00000   Median :0.00000
##  Mean   : 0.09892   Mean   :0.1029   Mean   : 0.06475   Mean   :0.04705
##  3rd Qu.: 0.00000   3rd Qu.:0.0000   3rd Qu.: 0.00000   3rd Qu.:0.00000
##  Max.   :14.28000   Max.   :5.8800   Max.   :12.50000   Max.   :4.76000
##       data             num415            num85          technology
##  Min.   : 0.00000   Min.   :0.00000   Min.   : 0.0000   Min.   :0.00000
##  1st Qu.: 0.00000   1st Qu.:0.00000   1st Qu.: 0.0000   1st Qu.:0.00000
##  Median : 0.00000   Median :0.00000   Median : 0.0000   Median :0.00000
##  Mean   : 0.09723   Mean   :0.04784   Mean   : 0.1054   Mean   :0.09748
##  3rd Qu.: 0.00000   3rd Qu.:0.00000   3rd Qu.: 0.0000   3rd Qu.:0.00000
##  Max.   :18.18000   Max.   :4.76000   Max.   :20.0000   Max.   :7.69000
##     num1999          parts              pm              direct
##  Min.   :0.000   Min.   :0.0000   Min.   : 0.00000   Min.   :0.00000
##  1st Qu.:0.000   1st Qu.:0.0000   1st Qu.: 0.00000   1st Qu.:0.00000
##  Median :0.000   Median :0.0000   Median : 0.00000   Median :0.00000
##  Mean   :0.137   Mean   :0.0132   Mean   : 0.07863   Mean   :0.06483
##  3rd Qu.:0.000   3rd Qu.:0.0000   3rd Qu.: 0.00000   3rd Qu.:0.00000
##  Max.   :6.890   Max.   :8.3300   Max.   :11.11000   Max.   :4.76000
##       cs             meeting          original         project
##  Min.   :0.00000   Min.   : 0.0000   Min.   :0.0000   Min.   : 0.0000
##  1st Qu.:0.00000   1st Qu.: 0.0000   1st Qu.:0.0000   1st Qu.: 0.0000
##  Median :0.00000   Median : 0.0000   Median :0.0000   Median : 0.0000
##  Mean   :0.04367   Mean   : 0.1323   Mean   :0.0461   Mean   : 0.0792
##  3rd Qu.:0.00000   3rd Qu.: 0.0000   3rd Qu.:0.0000   3rd Qu.: 0.0000
##  Max.   :7.14000   Max.   :14.2800   Max.   :3.5700   Max.   :20.0000
##       re               edu              table            conference
##  Min.   : 0.0000   Min.   : 0.0000   Min.   :0.000000   Min.   : 0.00000
##  1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.:0.000000   1st Qu.: 0.00000
##  Median : 0.0000   Median : 0.0000   Median :0.000000   Median : 0.00000
##  Mean   : 0.3012   Mean   : 0.1798   Mean   :0.005444   Mean   : 0.03187
##  3rd Qu.: 0.1100   3rd Qu.: 0.0000   3rd Qu.:0.000000   3rd Qu.: 0.00000
##  Max.   :21.4200   Max.   :22.0500   Max.   :2.170000   Max.   :10.00000
##  charSemicolon     charRoundbracket charSquarebracket charExclamation
##  Min.   :0.00000   Min.   :0.000    Min.   :0.00000   Min.   : 0.0000
##  1st Qu.:0.00000   1st Qu.:0.000    1st Qu.:0.00000   1st Qu.: 0.0000
##  Median :0.00000   Median :0.065    Median :0.00000   Median : 0.0000
##  Mean   :0.03857   Mean   :0.139    Mean   :0.01698   Mean   : 0.2691
##  3rd Qu.:0.00000   3rd Qu.:0.188    3rd Qu.:0.00000   3rd Qu.: 0.3150
##  Max.   :4.38500   Max.   :9.752    Max.   :4.08100   Max.   :32.4780
##    charDollar         charHash          capitalAve        capitalLong
##  Min.   :0.00000   Min.   : 0.00000   Min.   :   1.000   Min.   :   1.00
##  1st Qu.:0.00000   1st Qu.: 0.00000   1st Qu.:   1.588   1st Qu.:   6.00
##  Median :0.00000   Median : 0.00000   Median :   2.276   Median :  15.00
##  Mean   :0.07581   Mean   : 0.04424   Mean   :   5.191   Mean   :  52.17
##  3rd Qu.:0.05200   3rd Qu.: 0.00000   3rd Qu.:   3.706   3rd Qu.:  43.00
##  Max.   :6.00300   Max.   :19.82900   Max.   :1102.500   Max.   :9989.00
##   capitalTotal
##  Min.   :    1.0
##  1st Qu.:   35.0
##  Median :   95.0
##  Mean   :  283.3
##  3rd Qu.:  266.0
##  Max.   :15841.0
```

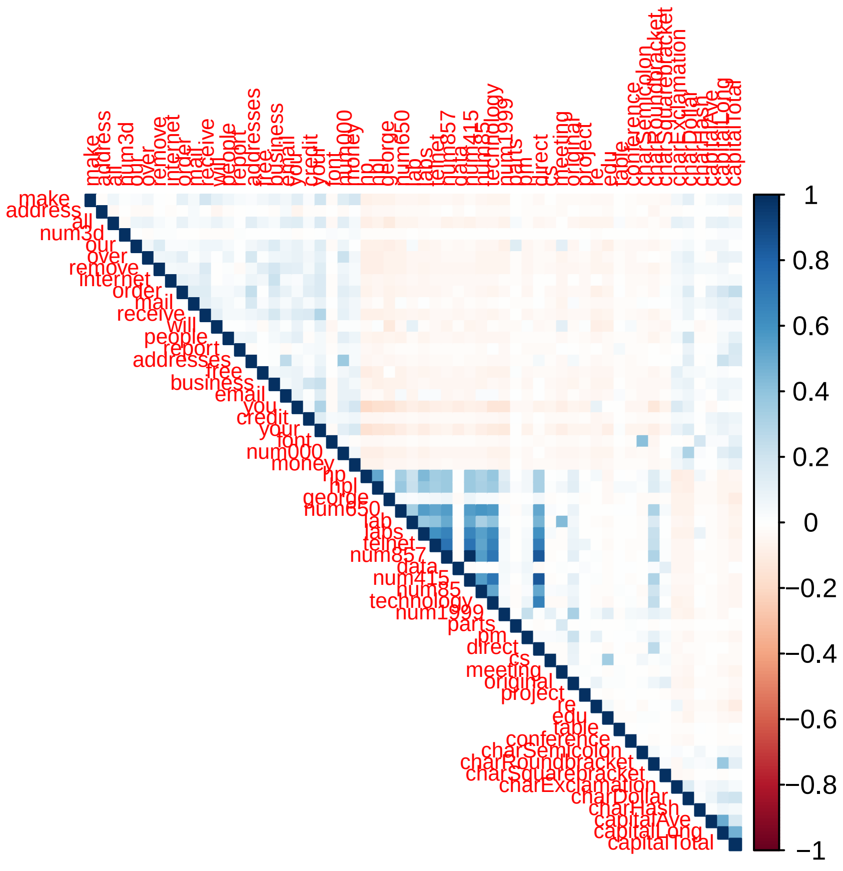
Comment of the Summary of Numerical Input Space

The summary above show us that our input space are numerical, that the range value of all numerical variable is well normalized. So it not necessary to scaling the Input space. Hence with this actual value, we can directly train some machine without normalized or scaling.

```
library(corrplot)
```

```
## corrplot 0.94 loaded
```

```
numeric_data <- data_spam[, sapply(data_spam, is.numeric)]
cor_matrix <- cor(numeric_data)
corrplot(cor_matrix, method = "color", type = "upper", tl.cex = 0.7)
```

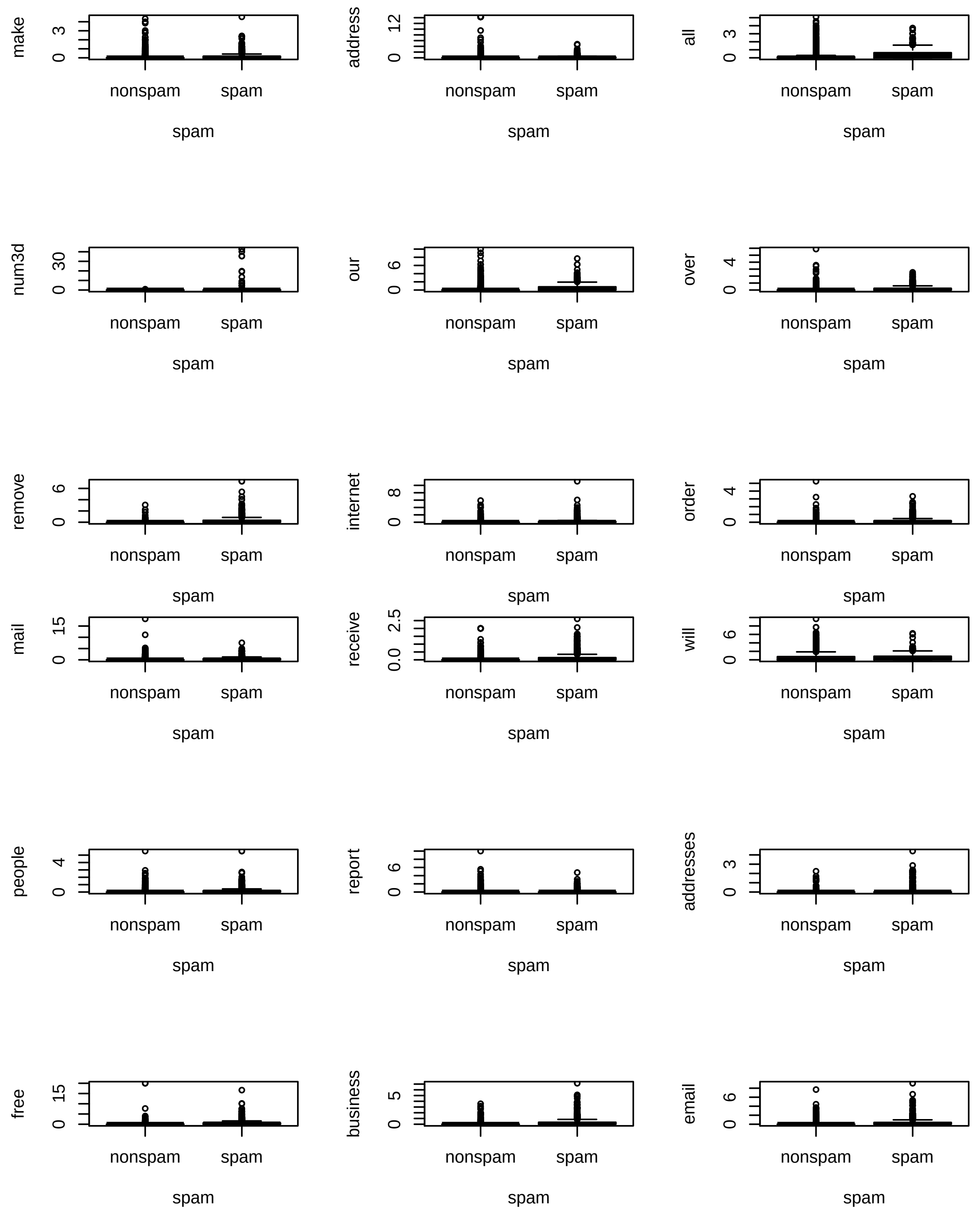


Comment on the correlation

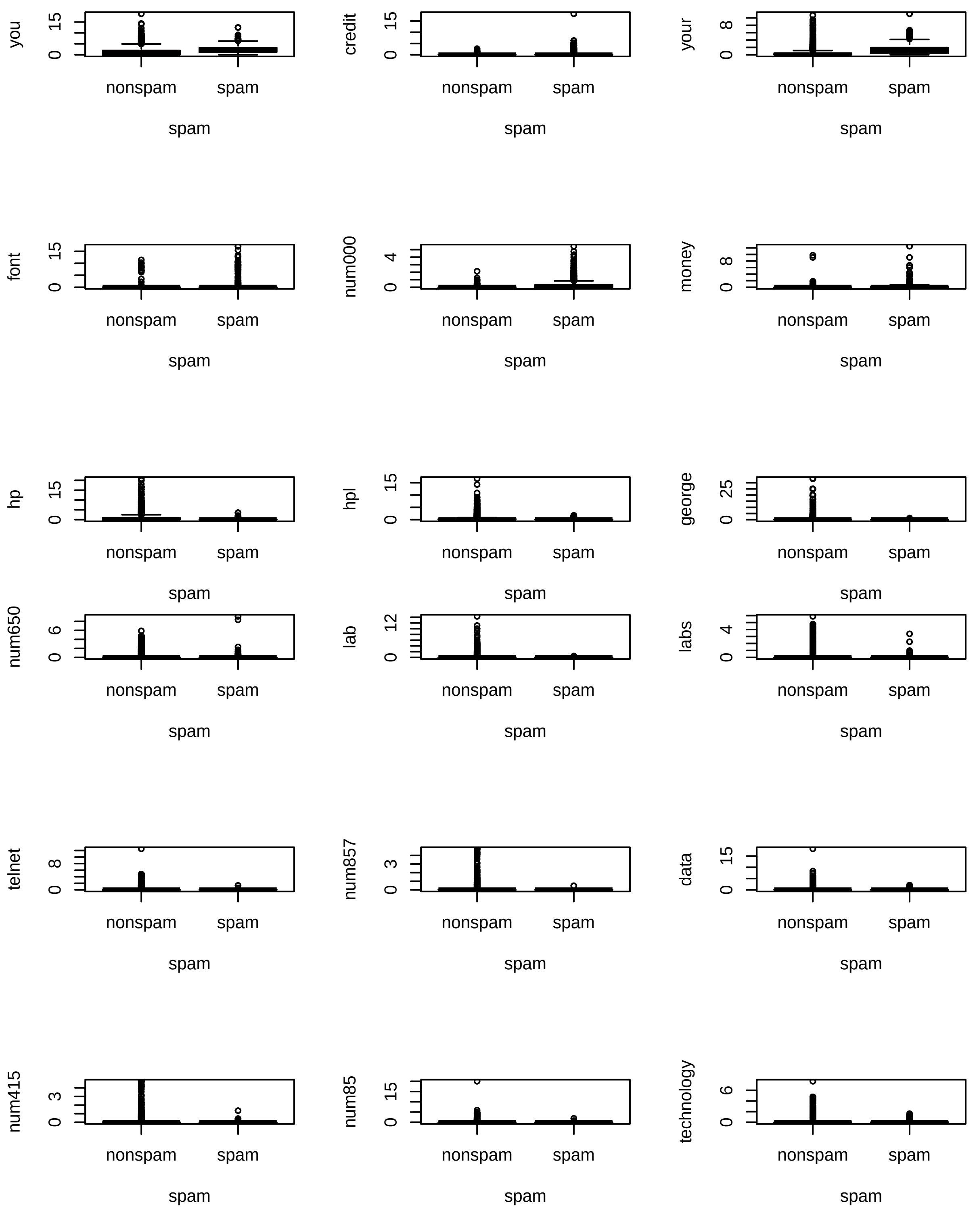
The Correlation figure above show us that must of the variable are strongly correlated.

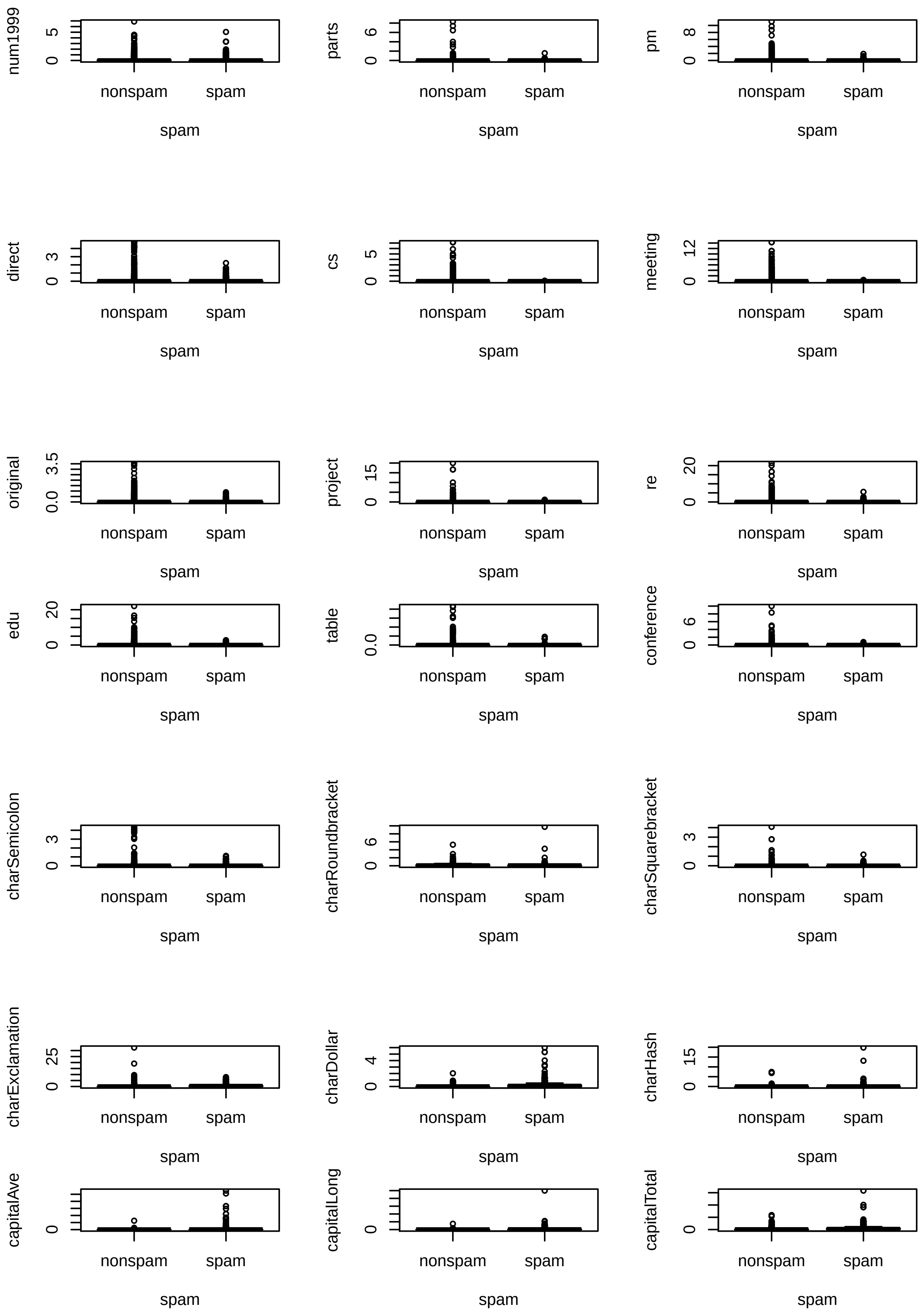
```
#plot(X, col=as.numeric(y)+2)
```

```
 y = data_spam$type
par(mfrow=c(3,3))
 for(j in 1:p)
 {
   boxplot(X[,j]~y, col=2:3, ylab=colnames(X)[j], xlab='spam')
}
```

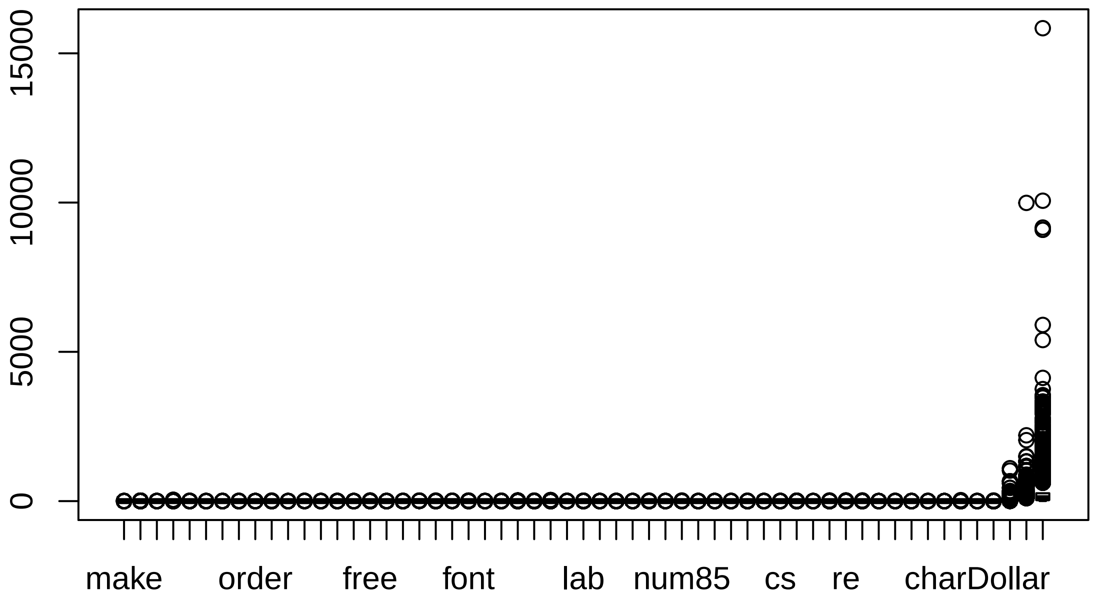


make
0
3
address
0
12
all
0
3
num3d
0
30
our
0
6
over
0
4
remove
0
6
internet
0
8
order
0
4
mail
0
15
receive
0.0
2.5
will
0
6
people
0
4
report
0
6
addresses
0
3
free
0
15
business
0
5
email
0
6
nonspam
spam
spam


you
credit
your
font
num000
money
hp
hpl
george
num650
lab
labs
telnet
num857
data
num415
num85
technology
nonspam
spam
spam


num1999
0
5
parts
0
6
pm
0
8
direct
0
3
cs
0
5
meeting
0
12
original
0.0
3.5
project
0
15
re
0
20
edu
0
20
table
0.0
conference
0
6
charSemicolon
0
3
charRoundbracket
0
6
charSquarebracket
0
3
charExclamation
0
25
charDollar
0
4
charHash
0
15
capitalAve
0
capitalLong
0
capitalTotal
0
nonspam
spam
spam

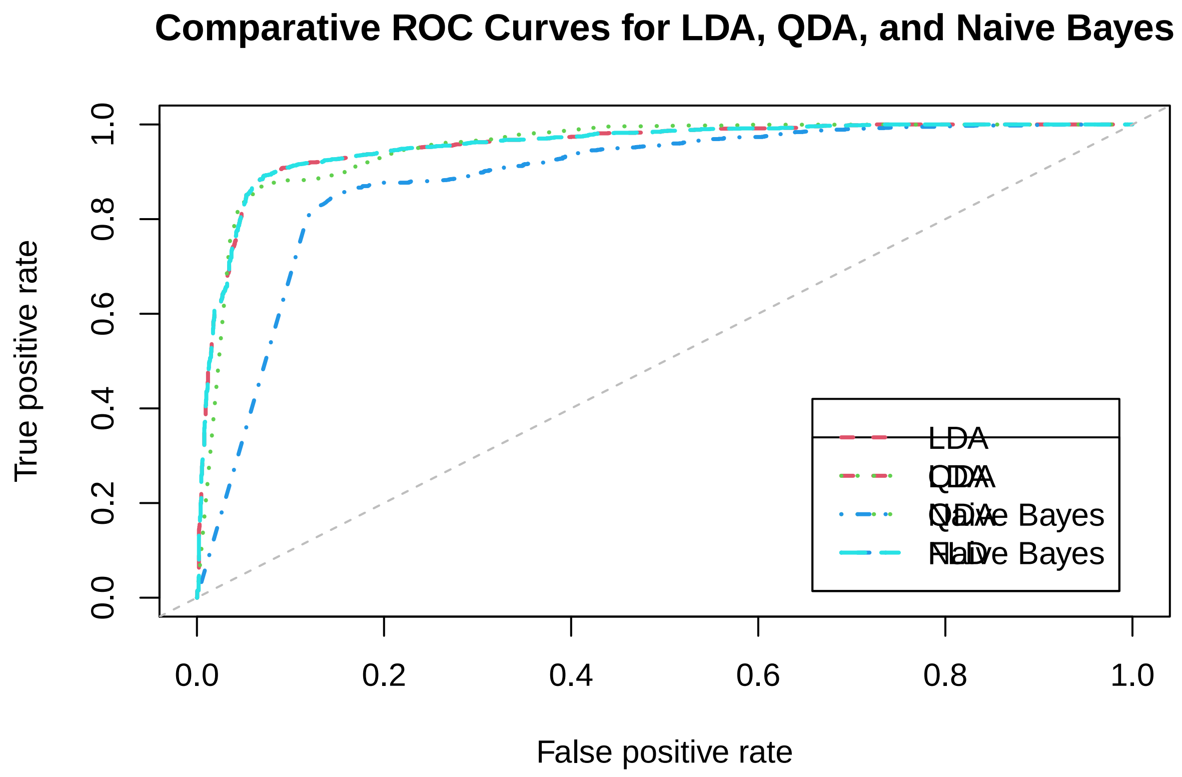
```
boxplot(X)
```



4. Using the whole data for training and the whole data for test, building the above four learning machines, then plot the comparative ROC curves on the same grid

```
y   <- data_spam$type      # Response vector
y_binary <- ifelse(y=="spam", 1 , 0)



# LDA Model (using entire dataset)
lda_model <- lda(type ~ ., data = data_spam)
prob_lda <- predict(lda_model, newdata = X)$posterior[, 2]  # Probabilities for class 1
pred_lda <- prediction(prob_lda, y)
perf_lda <- performance(pred_lda, measure = "tpr", x.measure = "fpr")

# QDA Model (using entire dataset)
qda_model <- qda(type ~ ., data = data_spam)
prob_qda <- predict(qda_model, newdata = X)$posterior[, 2]  # Probabilities for class 1
pred_qda <- prediction(prob_qda, y)
perf_qda <- performance(pred_qda, measure = "tpr", x.measure = "fpr")

# Naive Bayes Model (using entire dataset)
nb_model <- naiveBayes(X, y)
prob_nb <- predict(nb_model, newdata = X, type = "raw")[, 2]  # Probabilities for class 1
pred_nb <- prediction(prob_nb, y)
perf_nb <- performance(pred_nb, measure = "tpr", x.measure = "fpr")

# Fisher's Linear Discriminant (FLD) using LDA (since it's essentially FLD in the two-class case)
# This step is already covered by LDA for binary classification, so we can reuse the LDA model
prob_fld <- prob_lda  # Since LDA is equivalent to FLD in the two-class case
pred_fld <- prediction(prob_fld, y_binary)
perf_fld <- performance(pred_fld, measure = "tpr", x.measure = "fpr")
```

```
# Plot ROC curves for all models
plot(perf_lda, col = 2, lwd = 2, lty = 2, main = "Comparative ROC Curves for LDA, QDA, and Naive Bayes")
plot(perf_qda, col = 3, lwd = 2, lty = 3, add = TRUE)
plot(perf_nb, col = 4, lwd = 2, lty = 4, add = TRUE)

# Add diagonal line (random classifier) and legend
abline(a = 0, b = 1, col = "gray", lty = 2)
legend("bottomright", inset = 0.05, legend = c("LDA", "QDA", "Naive Bayes"), col = 2:4, lty = 2:4, lwd =
plot(perf_fld, col = 5, lwd = 2, lty = 5, add = TRUE)

# Add diagonal line (random classifier) and legend
abline(a = 0, b = 1, col = "gray", lty = 2)
legend("bottomright", inset = 0.05, legend = c("LDA", "QDA", "Naive Bayes", "FLD"), col = 2:5, lty = 2:
```



5. Comment succinctly on what the ROC curves reveal for this data and argue in light of the theory whether or not that was to be expected.

The figure of ROC shows us that the Machine FDA and LDA have the AUC must closed to the top-left. That means theses machines have the better performance. The worse algorithm in training in our full data is Naive Bayes.

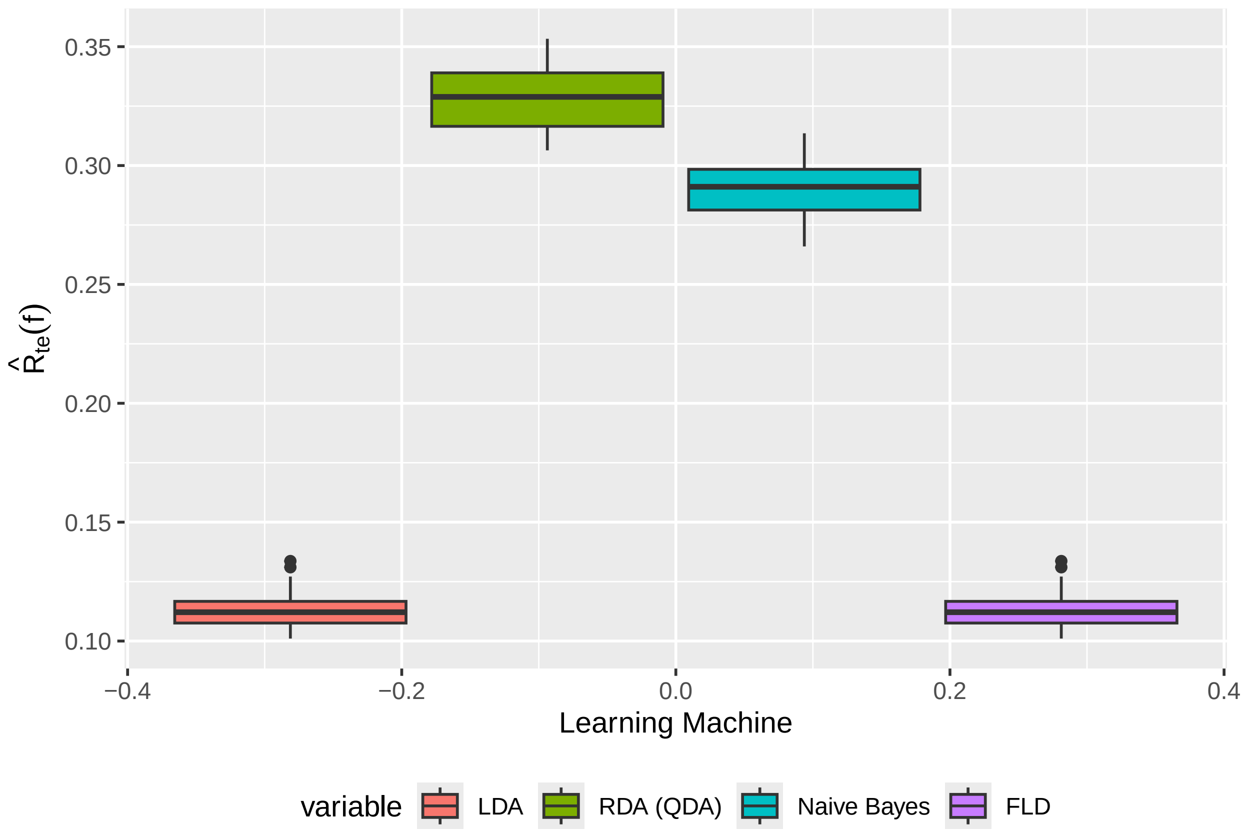
6. Using set.seed(19671210) along with a 2/3 training 1/3 test in the context stratified stochastic holdout split of the data, compute S = 50 replications of the test error for all the above learning machines.

```
set.seed(19671210)

epsilon <- 1/3
n <- nrow(data_spam)
nte <- round(n * epsilon)
ntr <- n - nte
S <- 50
```

```r
test.err <- matrix(0, nrow = S, ncol = 4)

for (s in 1:S) {
  id.tr <- sample(1:n, ntr)
  id.te <- setdiff(1:n, id.tr)

  data.tr <- data_spam[id.tr, ]
  data.te <- data_spam[id.te, ]

  y.te <- data.te$type

  # LDA
  lda_model <- lda(type ~ ., data = data.tr)
  lda_pred <- predict(lda_model, data.te[, 1:57])$class
  test.err[s, 1] <- mean(ifelse(y.te != lda_pred, 1, 0))

  # RDA (Regularized QDA)
  qda_model <- rda(type ~ ., data = data.tr) ## The raison why we used RDA is fix error when the size o
  qda_pred <- predict(qda_model, data.te[, 1:57])$class
  test.err[s, 2] <- mean(ifelse(y.te != qda_pred, 1, 0))

  # Naive Bayes
  nb_model <- naiveBayes(type ~ ., data = data.tr)
  nb_pred <- predict(nb_model, data.te[, 1:57])
  test.err[s, 3] <- mean(ifelse(y.te != nb_pred, 1, 0))

  # FLD (Fisher's Linear Discriminant)
  fld_model <- lda(type ~ ., data = data.tr)
  fld_pred <- predict(fld_model, data.te[, 1:57])$class
  test.err[s, 4] <- mean(ifelse(y.te != fld_pred, 1, 0))
}

# Summary of test errors
test_spam <- as.data.frame(test.err)
colnames(test_spam) <- c('LDA', 'RDA (QDA)', 'Naive Bayes', 'FLD')
```

7. Plot the comparative boxplots (be sure to properly label the plots)

```r
 ggplot(data = melt(test_spam, id.vars = NULL),   aes(x=, y=value)) + geom_boxplot(aes(fill=variable))+
    labs(x='Learning Machine', y=expression(hat(R)[te](f))) +
    theme(legend.position="bottom")
```



8. Comment on the distribution of the test error in light of (implicit) model complexity.

The machine LDA and FDA are similar due we have just two class in response variable , based on the figure above we can see that the Machine LDA and FLD is better for this task.

When we use stratified stochastic holdout Machine Naive Bayes becomes better than QDA which means that Naive Bayes have more ability to generalized than the machine QDA.(when we train in the full dataset D_n QDA have batter performance than Naive Bayes).