Statistical Machine Learning Assignment

Agbatan Fiacre Luc KOUDERIN

Master's Course in Statistical Machine Learning

Date: December 9, 2024

EXERCICE 1

Part 1 : Dataset Exploration

1. Load the data

Let's see the first rows of our data.

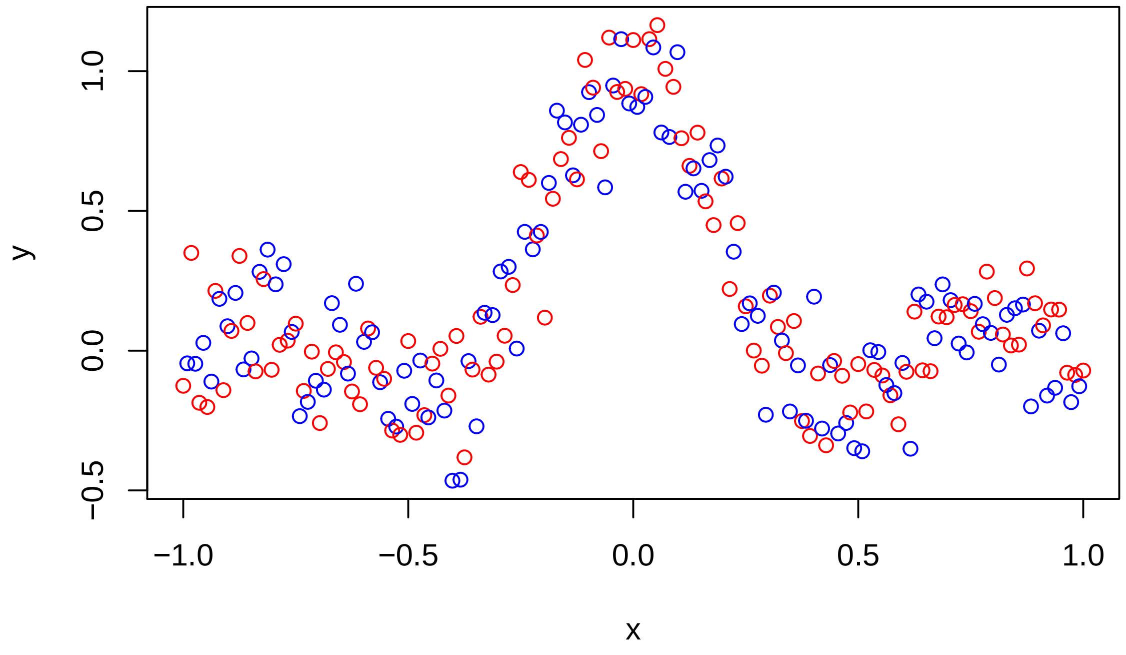
```
## # A tibble: 6 x 2
##          x       y
##      <dbl>   <dbl>
## 1 -1      -0.126
## 2 -0.991 -0.0455
## 3 -0.982  0.350
## 4 -0.973 -0.0469
## 5 -0.964 -0.186
## 6 -0.955  0.0279
```

2. Size of the data

```
## [1] 225   2
```

Our data have two variables with 225 observations

3- Creation of a scatterplot of y versus x



4- Determination wether this is a classification or regression task.

Our task here will be a regression for the following reasons :

Nature of the response variable y : Based on the data, our variable y is a continuous variable and a regression task aim to predict a continuous variable while classification deal with categorical outcomes(which is not our case here).

Scatter plot characteristics : We can remark on the scatter plot that y changes continuously as x varies. We have a clear trend, (a curve to be accurate) which show the dependency between x and y and regression models are more better to capture this kind of relationship. Additionally, in the scatter plot, y shows a spread of points rather than distinct clusers, which supports again the regression hypothesis.

Part 2 : Theoritical Framework

1. Function Space $\mathcal{H}$

For this task, we choose the function space $\mathcal{H}$ which is defined as follows:

$$\mathcal{H} = \left\{ f(x) : f(x) = \sum_{j=0}^{p} \beta_j x^j,\ \beta_j \in \mathbb{R},\ p \in \mathbb{N} \right\}.$$

Thus, this represents the set of polynomial functions of degree p.

2. Loss Function

The **Loss 2** is the most standard loss function utilized in reregression tasks and we'll use it for our task too :

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

Justification:

- It's measures the squared difference between predicted and actual values, penalizing larger errors more heavily.

- It makes the empirical risk smooth and differential, making optimization (e.g. gradient descent which is an optimization algorithm used to find the optimal values of the parameters) computationally efficient.

- It directly aligns with the goal of minimizing prediction error in regression tasks.

3. Theoretical Risk

The theoretical risk $R(f)$ is the expected value of the loss function over the joint distribution of (x, y): Its expression is given by :

$$R(f) = \mathbb{E}\left[(y - f(x))^2\right].$$

This expectation can be expressed as an integral over the entire population:

$$R(f) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} (y - f(x))^2 \, p(x, y) \, dx \, dy,$$

where $p(x, y)$ is the joint probability density function of (x, y). The integration is performed over all possible values of x and y, capturing the entire population.

Let's remark that since the true distribution of (x, y) is unknown, $R(f)$ cannot be computed directly.

4. Bayes Learning Machine

The Bayes predictor $f^*(x)$ minimizes the theoretical risk $R(f)$ and is given by:

$$f^*(x) = \mathbb{E}[y \mid x].$$

This expectation can be expressed as an integral over the conditional distribution of y given x, as follows:

$$f^*(x) = \int_{-\infty}^{\infty} y \, p(y \mid x) \, dy,$$

where $p(y \mid x)$ is the conditional probability density function of y given x. The integral is taken over all possible values of y, reflecting the expected value of y given x.

$f^*(x)$ is the true (or best machine) underlying relationship between x and y.

5. Empirical Risk

The empirical risk $\hat{R}(f)$ is an approximation of the theoretical risk based on the data sample. Using the loss 2, it is defined as:

$$\hat{R}(f) = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \beta_0 - \beta_1 x - \beta_2 x^2 - \cdots - \beta_p x^p\right)^2.$$

Minimizing $\hat{R}(f)$ allows us to find the best candidate f within the function space $\mathcal{H}$. Let's remark that **the degree p controls the complexity of the model**.

Part 3 : Estiimation and Model Complexity

1. Expression for the OLS estimator

Our goal is to derive the Ordinary Least Squares (OLS) estimator for a polynomial regression model of degree p:

$$f(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_p x^p.$$

This consist of the minimization of the empirical risk:

$$\hat{R}(f) = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \sum_{j=0}^{p}\beta_j x_i^j\right)^2.$$

Let:

- $\mathbf{y} = (y_1, y_2, \dots, y_n)^\top$, the vector of observed values,
- $\mathbf{X}$, the design matrix of dimension $n \times (p+1)$:

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

- $\beta = (\beta_0, \beta_1, \dots, \beta_p)^\top$.

The empirical risk can now be written as:

$$\hat{R}(f) = \frac{1}{n}\|\mathbf{y} - \mathbf{X}\beta\|^2.$$

To minimize $\hat{R}(f)$, we have to differentiate $\|\mathbf{y} - \mathbf{X}\beta\|^2$ with respect to β:

$$\|\mathbf{y} - \mathbf{X}\beta\|^2 = (\mathbf{y} - \mathbf{X}\beta)^\top(\mathbf{y} - \mathbf{X}\beta).$$

Expanding this:

$$(\mathbf{y} - \mathbf{X}\beta)^\top(\mathbf{y} - \mathbf{X}\beta) = \mathbf{y}^\top\mathbf{y} - 2\mathbf{y}^\top\mathbf{X}\beta + \beta^\top\mathbf{X}^\top\mathbf{X}\beta.$$

The gradient with respect to β is:

$$\frac{\partial}{\partial\beta}\left(\mathbf{y}^\top\mathbf{y} - 2\mathbf{y}^\top\mathbf{X}\beta + \beta^\top\mathbf{X}^\top\mathbf{X}\beta\right) = -2\mathbf{X}^\top\mathbf{y} + 2\mathbf{X}^\top\mathbf{X}\beta.$$

Setting the gradient to zero:

$$-2\mathbf{X}^\top\mathbf{y} + 2\mathbf{X}^\top\mathbf{X}\beta = 0.$$

which implies finally that :

$$\hat{\beta} = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y}$$

Therefore, the predicted values are:

$$\hat{f}(x) = \mathbf{X}\hat{\beta}.$$

2. Comments on the properties of $\hat{\beta}$

- **Linearity**: $\hat{\beta}$ is a linear in $\mathbf{y}$, making it computationnaly efficient.
- **Unbiasedness**: Under standard assumptions (e.g. $E[\mathbf{y}] = 0$ and $Var[y] = \sigma^2$), $\mathbb{E}[\hat{\beta}] = \beta$. In fact, we have :

$$\begin{aligned}\mathbb{E}[\hat{\beta}] &= \mathbb{E}[(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y}] \\ &= (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbb{E}[\mathbf{y}] \\ &= (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{X}\beta \\ \mathbb{E}[\hat{\beta}] &= \beta\end{aligned}$$

- **Variance**: The variance of $\hat{\beta}$ is proportional to $(\mathbf{X}^\top\mathbf{X})^{-1}$. It's got by this way :

$$Var[\widehat{\beta}] = Var[(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y}]$$

$$= (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\sigma^2\mathbf{X}(\mathbf{X}^\top\mathbf{X})^{-1}$$

$$Var[\widehat{\beta}] = \sigma^2(\mathbf{X}^\top\mathbf{X})^{-1}$$

Thus, the variance increases as p (the degree of the polynomial) increases.

- **Bias-Variance Trade off**: Higher-degree polynomials reduce bias but increase variance, leading to overfitting.
- **Efficiency** : Under the Gauss-Markov theorem, the OLS estimator is the Best Linear Unbiased Estimator (BLUE), meaning that it has the smallest variance among all linear estimators.

3. Let's use V-fold cross-validation to get the optimal complexity.

Optimal complexity is the point where the model is complex enough to capture the underlying patterns in the data but simple enough to avoid overfitting. It's means that the model should not be too complex (which could lead to overfitting) or too simple (which could lead to underfitting)

Let's determne this optimal complexity using V- fold cross validation.

Our algorithm can be described like as follow :

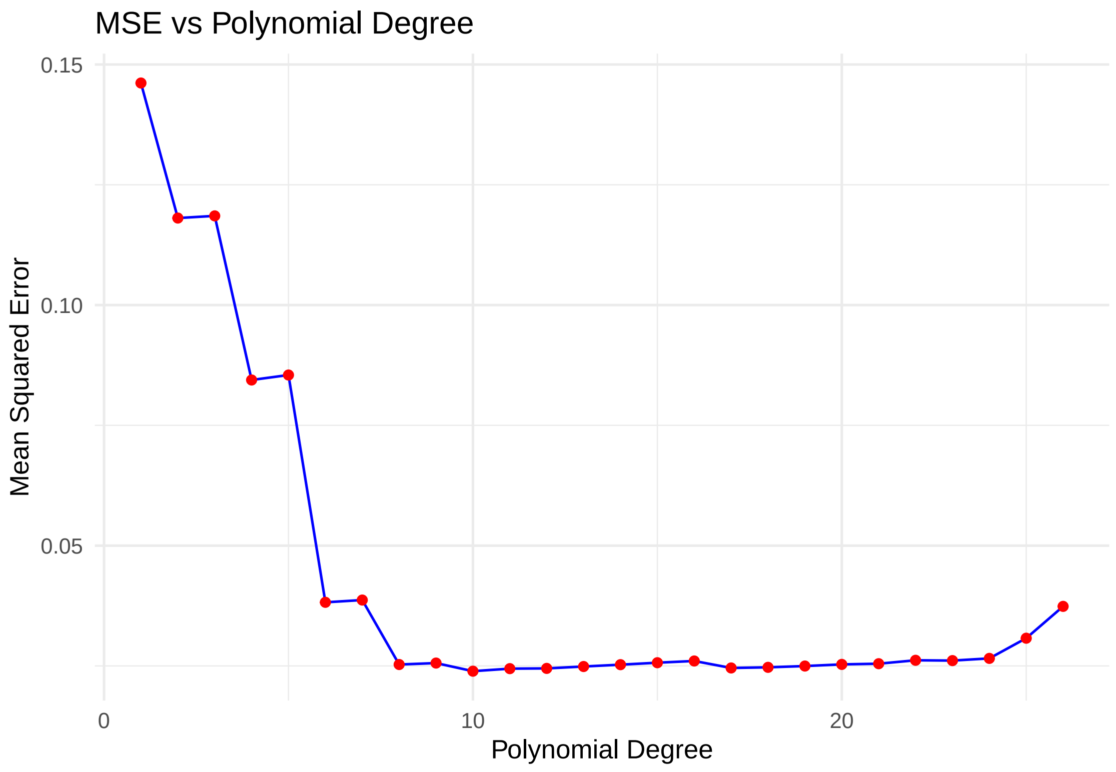
- Divide the dataset into V equally sized folds,
- For each fold :
  - Use $V - 1$ folds to train the model,
  - Evaluate its performance (here, the MSE) on the remainig fold
- Repeat this process for each candidate degree p
- Choose p that minimizes the average MSE across all folds

Let's apply this: As output, we print the MSE for some d degree (from degree 8 to degree 12) and print out which degree generate the smallest MSE.

```
##    Degrees         MSE
## 8         8 0.02529390
## 9         9 0.02559603
## 10       10 0.02390674
## 11       11 0.02443440
## 12       12 0.02448503
```

```
## The optimal degree is 10 with an MSE of 0.02390674
```

At the end, using $V = 15$ and a range of degrees between 1 and 26, we get the degree which give the minimum error is 10. Let's plot the evolution of the MSE as function of degrees.



Remark : As the degree increases, the MSE is decreasing until when the degree is 8. From 8, the MSE become more constant and very close to zero. Then, its making sens that 10, which is not so far to 8, is the best value of degree. However, according to the graph, the difference between the error give us by degree 8 is not very very significant. Let's check between the model with degree 8 and the model with degree 10, which one is the best.

```
##
## Call:
## lm(formula = y ~ poly(x, 8), data = aims_data)
##
```

```
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.42601 -0.09697 -0.00511  0.12137  0.35005
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.16620    0.01039  15.998   <2e-16 ***
## poly(x, 8)1  0.04154    0.15584   0.267   0.7901
## poly(x, 8)2 -2.53339    0.15584 -16.257   <2e-16 ***
## poly(x, 8)3 -0.05565    0.15584  -0.357   0.7214
## poly(x, 8)4  2.80492    0.15584  17.999   <2e-16 ***
## poly(x, 8)5 -0.27326    0.15584  -1.753   0.0809 .
## poly(x, 8)6 -3.15952    0.15584 -20.274   <2e-16 ***
## poly(x, 8)7  0.10617    0.15584   0.681   0.4964
## poly(x, 8)8  1.68296    0.15584  10.799   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1558 on 216 degrees of freedom
## Multiple R-squared:  0.8383, Adjusted R-squared:  0.8323
## F-statistic:   140 on 8 and 216 DF,  p-value: < 2.2e-16
```

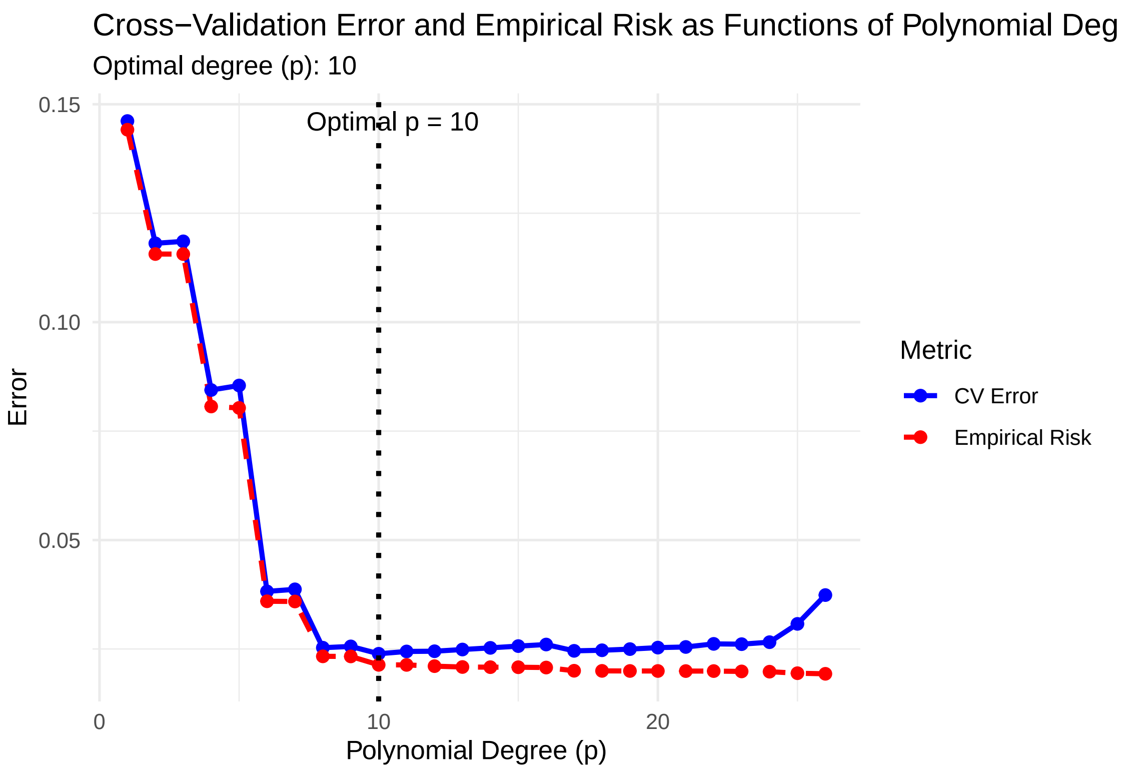
```
##
## Call:
## lm(formula = y ~ poly(x, 10), data = aims_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.39738 -0.09895 -0.00637  0.11672  0.37150
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    0.166202   0.009991  16.636  < 2e-16 ***
## poly(x, 10)1   0.041538   0.149860   0.277   0.7819
## poly(x, 10)2  -2.533391   0.149860 -16.905  < 2e-16 ***
## poly(x, 10)3  -0.055651   0.149860  -0.371   0.7107
## poly(x, 10)4   2.804919   0.149860  18.717  < 2e-16 ***
## poly(x, 10)5  -0.273258   0.149860  -1.823   0.0696 .
## poly(x, 10)6  -3.159519   0.149860 -21.083  < 2e-16 ***
## poly(x, 10)7   0.106174   0.149860   0.708   0.4794
## poly(x, 10)8   1.682964   0.149860  11.230  < 2e-16 ***
## poly(x, 10)9   0.064073   0.149860   0.428   0.6694
## poly(x, 10)10 -0.659976   0.149860  -4.404 1.68e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1499 on 214 degrees of freedom
## Multiple R-squared:  0.8518, Adjusted R-squared:  0.8449
## F-statistic:   123 on 10 and 214 DF,  p-value: < 2.2e-16
```

Based on the summaries on the two models, we can remark that the Residual standard error, the R squared for *model_10* are less than the values of this quantities for *model_8*. This is confirming that the model with 10 degree is the best. The following ANOVA table confirm this statement :

```
## Analysis of Variance Table
##
## Model 1: y ~ poly(x, 8)
## Model 2: y ~ poly(x, 10)
##   Res.Df    RSS Df Sum of Sq      F    Pr(>F)
## 1    216 5.2457
## 2    214 4.8060  2   0.43967 9.7888 8.554e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

So. **We can definitely conclude that the optimal degree for this polynomial regression task is 10**.

4. Let's plot the cross validation error and empirical risk as function of p



Comment: As we can guess, our empirical risk is decreasing when the degree is increasing basically mean more our model is complex, more the empirical risk tend to be zero. However For the cross validation which measure the capacity of prediction of the model on the new data, in a first region , when $p < 8$, it's decreasing, when p increases, its become more stable for $p \in [8, 25]$ but for $p > 25$, its start to increase again.

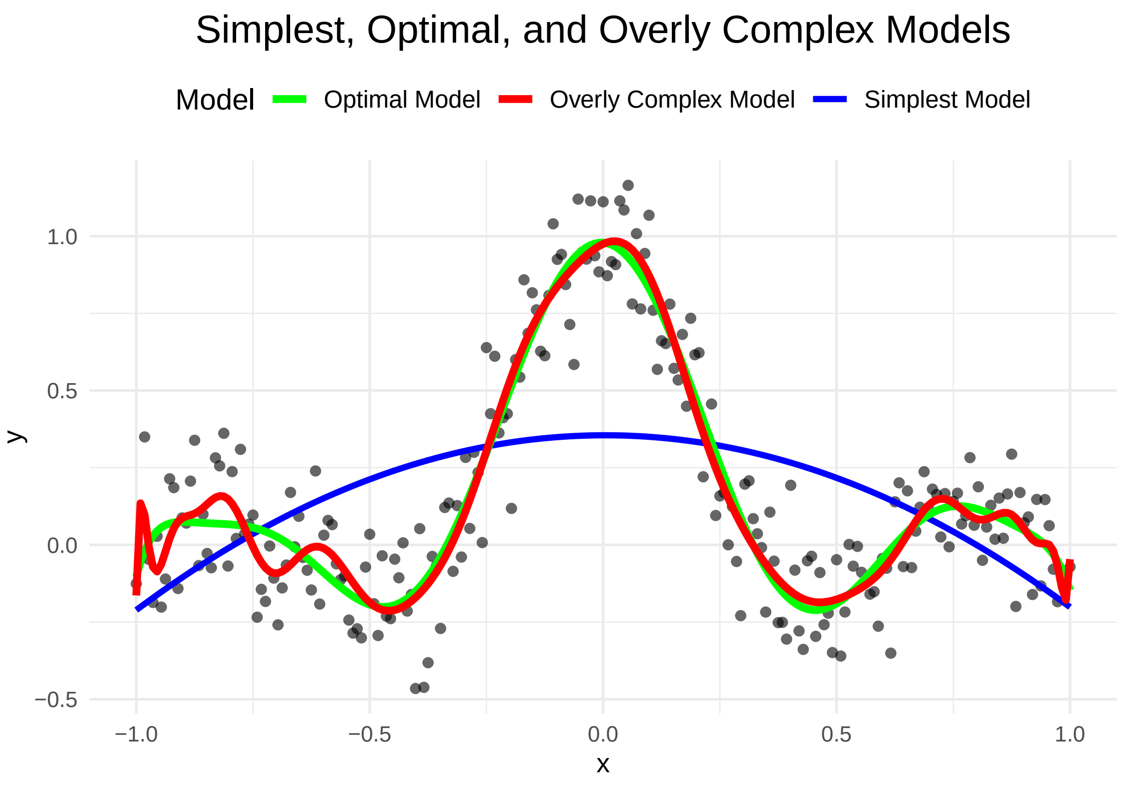
This basically means for small p (small complexity), the model is underfitting (which big error), and when p become too big(in our case > 25), the model become less performant because of risk of overfitting (just memorize the data and not get the signal).

Part 4 : Model Comparison and Evaluation

1. Let's fit and plot all this estimators.

- For the simplest one, the take the degree of polynomial is 2 (as we have already see the pattern of x and y, it will 'inconvenient' to take a simple linear regression even if as simplest model), which mean a quadratic regression

- For the optimal estimator, we take the degree of polynomial is equal 10
- For an overly complex estimator, we take the degree is equal to 25

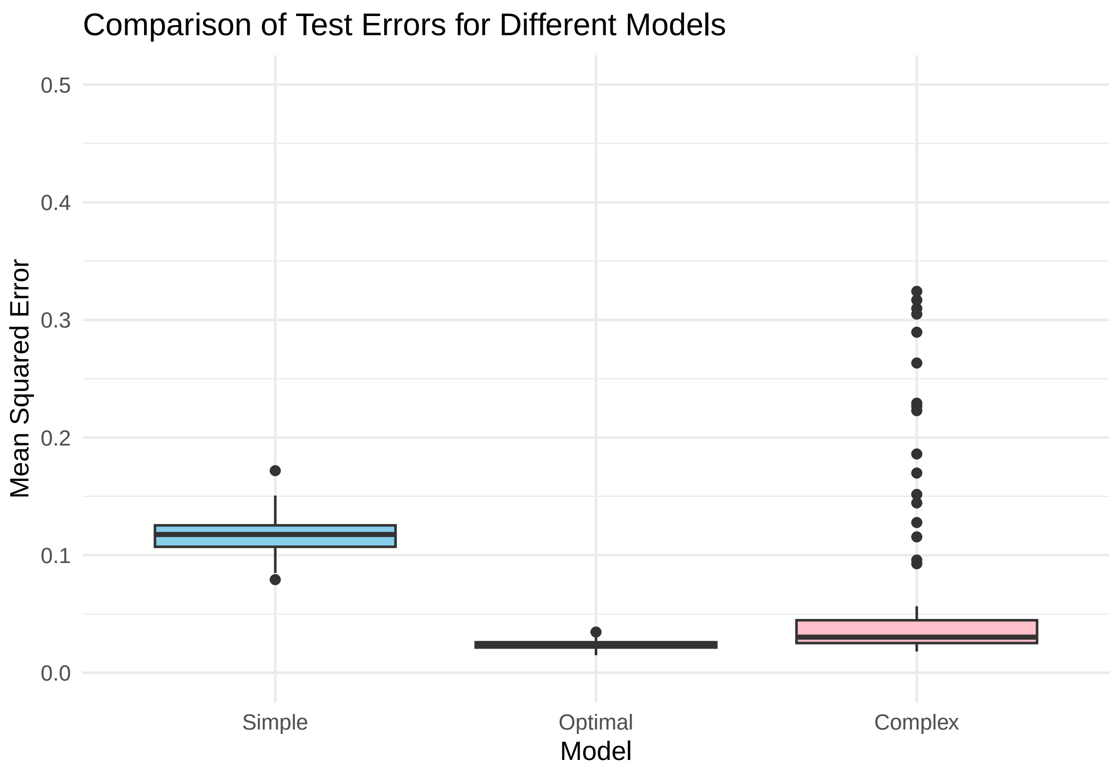


Comments :

- The simplest model (blue curve) give us a capture of the general trend of the data but fails to adapt to the oscillations present in the dataset. It's underfits in regions where the data demonstrates significant variability.
- The optimal model (green line) adapts well to the data's structure, capturing the major trends and oscillations without overfitting to noise. It strikes a balance flexibily and generalization, performing well across the entire range of the data
- The overly complex model (red curve) fits the data too tightly, capturing even minor fluctuations and noise. It suffers from overfitting, particularly in regions with sparse data points, leading to exaggerated oscillations.

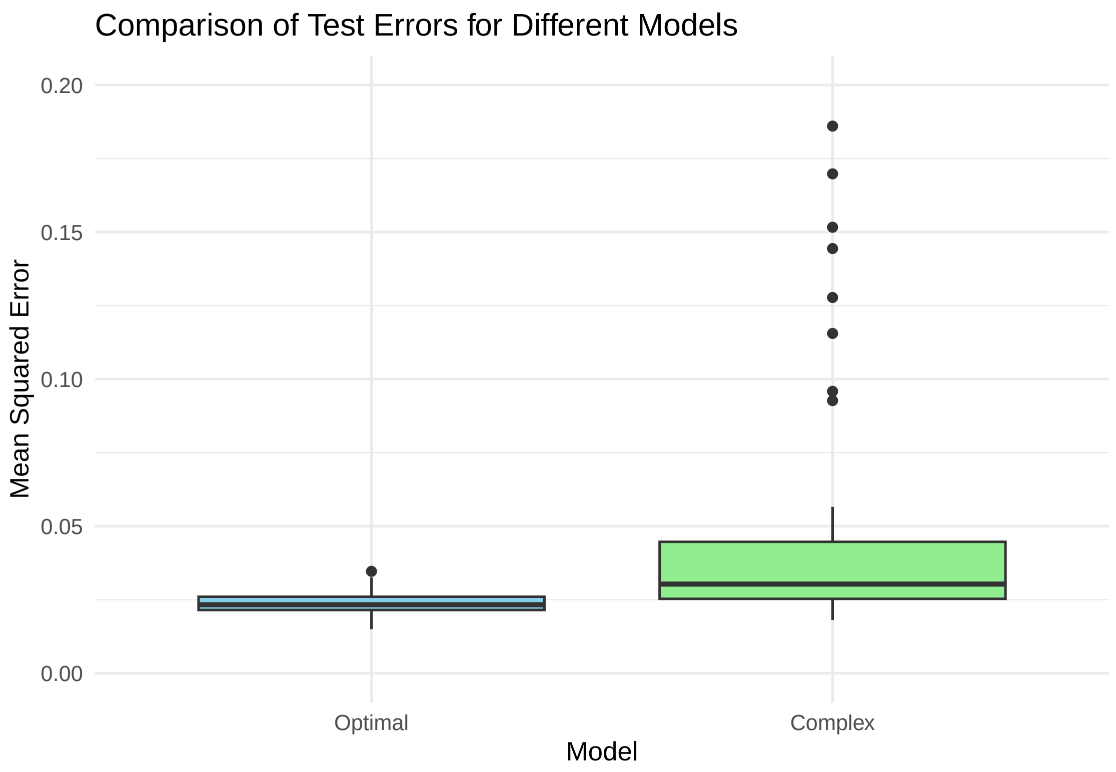
As overall comment, we can deduce that optimal model is the best choice for capturing the key features of the data.

2. Let's perform a hold out validation for each model and lot the box-plots



Based on this boxplot, its clear that the worse model is the simplest one. However, concerning the best model, we might think that it could be the optimal model or the complex, but this is caused by the scale of the errors

Let's plot the boxplot without the simplest model.



This plot highlights a little bit more the difference between the optimal model and the complex one. Even if the difference is not too big, the range of the errors for the complex one is larger than the range of errors for the optimal and additionally there are more outliers in the complex model than outliers in the optimal one

Part 5 : Further Analysis

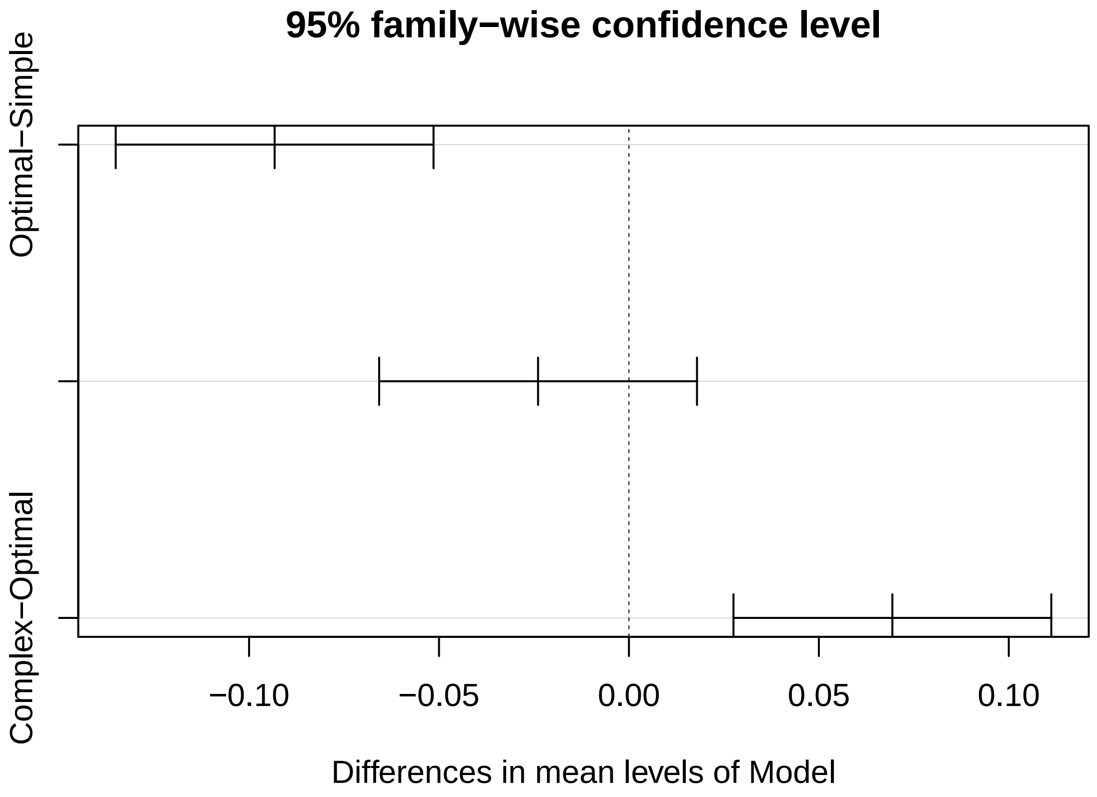
1. Let's perform an ANOVA on the test errors

```
##              Df Sum Sq Mean Sq F value   Pr(>F)    
## Model         2  0.469 0.23471   14.88 6.93e-07 ***
## Residuals   297  4.684 0.01577                     
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Comments : We remark that the p Value is zero which mean that the test errors are significantly different across the models. In other words, there is at least two models which are significant different.

Thus, we can to perform further analysis to get which models produces different results.

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##     factor levels have been ordered
##
## Fit: aov(formula = Error ~ Model, data = error_df_long)
##
## $Model
##                        diff         lwr        upr     p adj
## Complex-Optimal 0.06935722  0.02752217 0.11119227 0.0003424
## Simple-Optimal  0.09327406  0.05143901 0.13510911 0.0000009
## Simple-Complex  0.02391684 -0.01791821 0.06575189 0.3705635
```



This is why the use Tukey Honest Significant Difference (HSD) test result, which do a pairwise comparison. From its graph, we can deduce that :

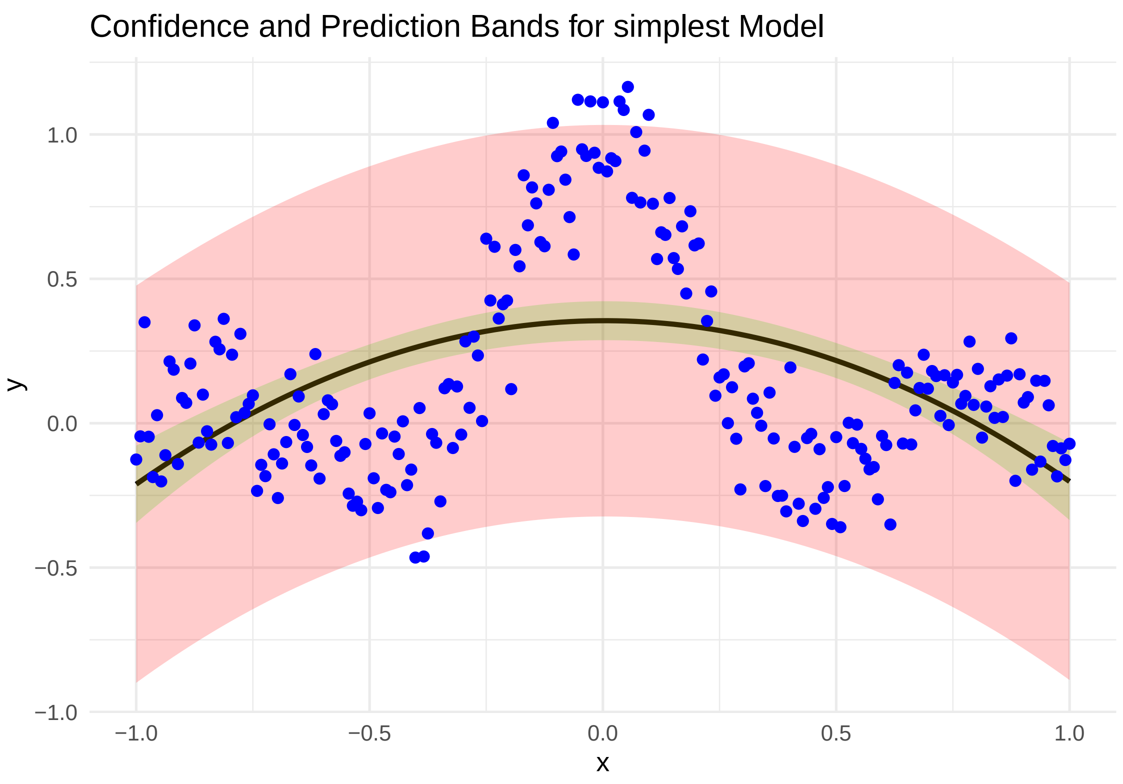
- *Optimal vs. Simple (Optimal-Simple)*: The confidence interval does not cross 0, indicating a statistically significant difference between the “Optimal” and “Simplest” models.
- *Complex vs. Simple (Complex-Simple)*: The confidence interval does not cross 0, indicating a statistically significant difference between the “Complex”

and “Simplest” models.

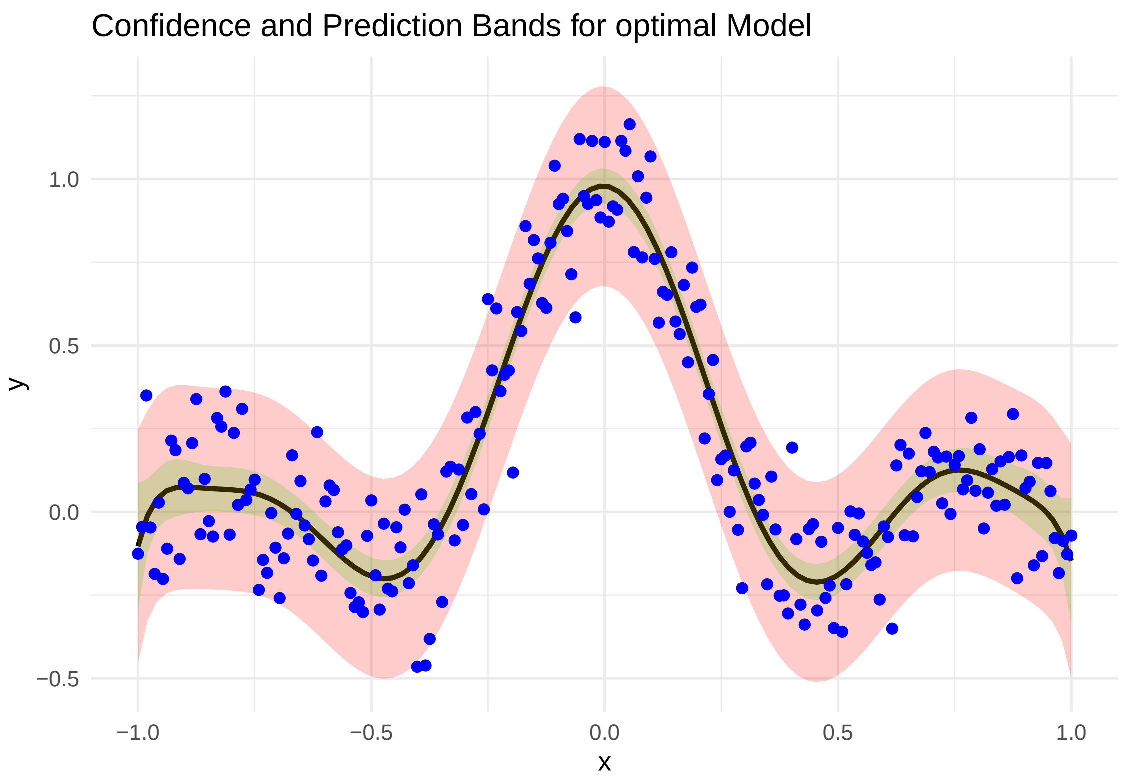
- *Complex vs. Optimal (Complex-Optimal)*: The confidence interval includes 0, meaning there is no statistically significant difference between the “Complex” and “Optimal” models.

2. Let’s obtain and plot the 95% confidence and predictions bands for the dataset D_n

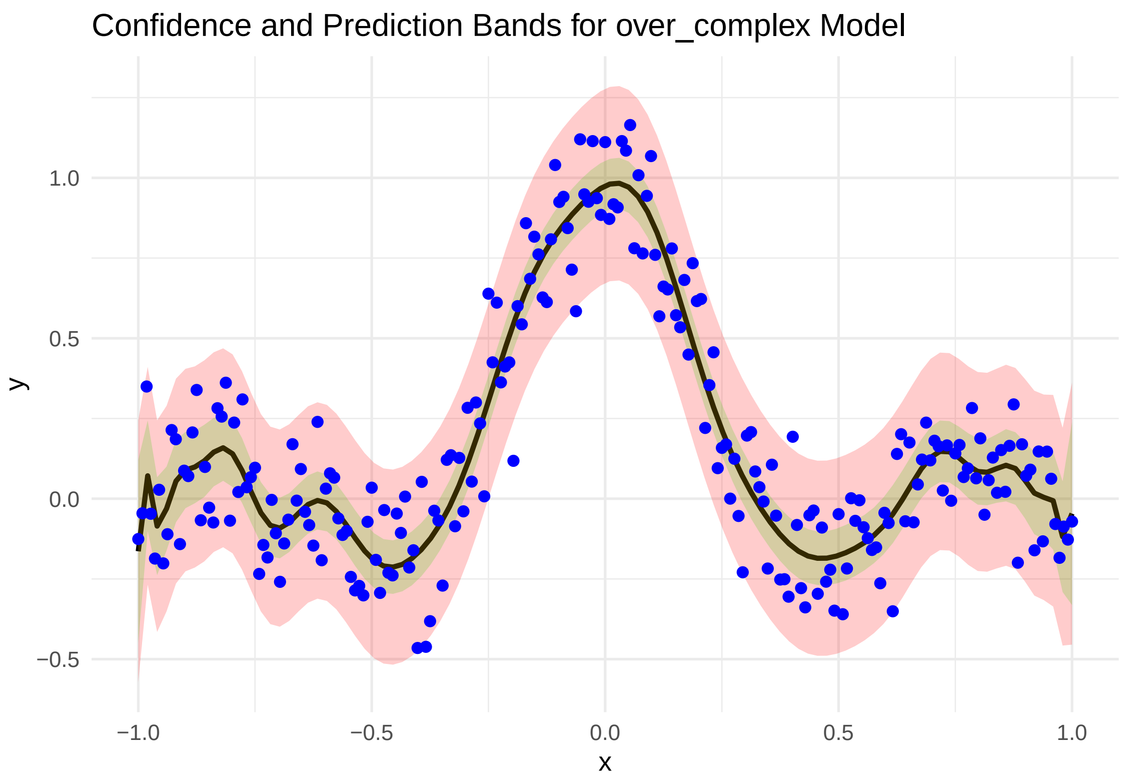
```
## [[1]]
```



```
##
## [[2]]
```


Confidence and Prediction Bands for optimal Model
1.0
0.5
0.0
−0.5
y
−1.0
−0.5
0.0
0.5
1.0
x

```
##
## [[3]]
```



In this plots :

- The confidence band is represented by the green band
- The predictive band is represented by the red band
- The black line represent the predictive values from the model

3. Mathematical Expressions of Confidence and Prediction Bands

For a single observation (X_i, Y_i):

- **Confidence Band**:
  The confidence interval for the mean response at X_i is given by:

$$\hat{y}_i \pm t_{\alpha/2,n-p} \cdot \sqrt{\text{Var}(\hat{y}_i)}$$

where:

- $\hat{y}_i$ is the predicted value of Y_i,
- $t_{\alpha/2,n-p}$ is the critical value from the t-distribution with $n - p$ degrees of freedom,
- $\text{Var}(\hat{y}_i)$ is the variance of the predicted value.

- **Prediction Band**:
  The prediction interval for a single new observation at X_i is given by:

  $$\hat{y}_i \pm t_{\alpha/2,n-p} \cdot \sqrt{\mathrm{Var}(\hat{y}_i) + \sigma^2}$$

  where σ^2 is the residual variance of the model.

The **prediction band** is wider than the **confidence band**, as it includes the variability of the individual observations in addition to the variability of the mean.

4. Comments on the confidence and prediction bands

- **Simplest Model :** For this model, we can remark that the green region (confidence band) is relatively narrow, a narrowness due to the small number of parameters, resulting in lower variability.

About the prediction band, its significantly wider than the confidence bands. However, cause of the simplicity of the model, the bands fail to capture the nuances of the data.

- **Optimal Model:** The black line showing the model's predictions fits the data better than the simplest one. The confidence bands are narrower than those of the simplest model in regions with dense data but widen slighly in areas with sparse data.

About the predictions, they bands are wider than the confidence bands and more adaptive to the data's variability compared to the simplest model. They effectively capture most of the observed points, suggesting the model does a good job balancing between underfitting and overfitting.

- **Complex Model :** In this model, the confidence bands are very narrow around the fitted curve, suggesting the model is very certain about its predictions. However, this high certainty doesn't mean the model will work well on new data. The prediction bands are much wider and follow even the smallest changes in the data, which shows that the model is overfitting the data.

EXERCICE 2 :

1. Let's plot the distribution of the response for this dataset and let's comment

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

Let's have a look on the first observations in the data and for the first variables

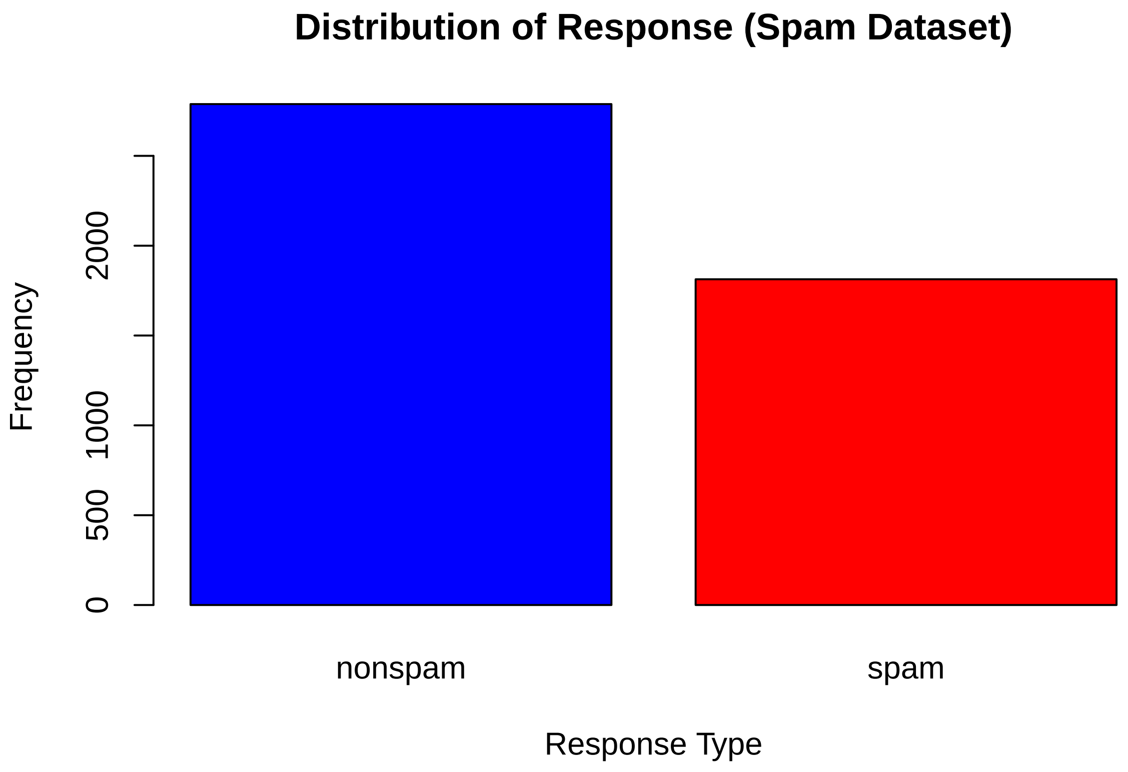
```
##   capitalAve capitalLong capitalTotal type
## 1      3.756          61          278 spam
## 2      5.114         101         1028 spam
## 3      9.821         485         2259 spam
## 4      3.537          40          191 spam
## 5      3.537          40          191 spam
## 6      3.000          15           54 spam
```

Let's plot the distribution of this variable:

Let's first check the proportion of each factors.

```
##
## nonspam    spam
##    2788    1813
```

Let's plot then the barplot of this response variable



Comment : This response variable is called **type**, it is a categorical variable with two factors : "non spam" and "spam" which mean that we will deal with a binary classification problem.

Additionally, we have in our response variable, 2788 classified "non spam" and 1813 classified "spam".

2. Let's comment on the shape of this dataset in terms of the sample size and the dimensionnality of the input space

We use the command 'help' to learn more about how the data is collected.

The dimension of the dataset is :

```
## [1] 4601   58
```

- The dataset contains 4601 samples (rows), where each of them represent an email that is either "spam" or "not spam"
- The dataset has 57 components as input variables + one output variable which represent various metrics on the email. For instance :
- The first 48 variables contain the frequency of the variable name (e.g. business) in the email.
- The variables from 49 to 54 indicate the frequency of some punctuation characters (e.g. ',', ';', etc.)
- The variables from 55 to 57 contain the average, longest and total run-length of capital letters
- All of this variables are numeric.

3. Let's comment from the statistical perspective on the type of data in the input space

- *Dimensionnality vs Sample size :* We have 57 variables with 4601 observations, whch basically means that the number of observations is more 5 times the number of parameters. This indicates the dataset has more than enough observations relatives to the number of variables which is a good thing for the majority of machine learning algorithm.

- *Nature of the predictive variables :* All of the components of the input space are derived from the same text and this can cause significant correlations. IN other words, many variables can be irrelevant or redundant; they can introduce noise and reduce the model's performance.
- *Algorithm and computational costs :* Have 57 variables is a little bit a big

number of parameters to estimate and this increases inevitably the computational cost. So, some dimensionnality reduction techniques might be needed to reduce the number of variables.

4. Let's build the ROC curves for this four machines after used the whole data as a train and a test set.

Let's first give a few definition of each machine :

- **LDA :** This stands for Linear Discriminant Analysis : it's a classification method that assumes normally distributed data for each class and find a linear combination of variables to maximize the separation between classes.
- **QDA :** It's stand for Quadratic Discriminant Analysis : its similar to LDA but allows for class-specific covariance matrices, leading to non linear decision boundaries.
- **Naive Bayes :** It's a probabilistic classifier based on Bayes's theorem, assuming independance between predictors
- **FLD :** Stands for Fisher's Linear Discriminant, its a dimensionnaly reduction technique that projects data onto a line to maximize class separation. **For a binary classification, FLD and LDA are equivalent, which means that we don't need to use both of them (the code to fit the two models is exactly the same)**

Then, we'll use LDA, and for all conclusions drawed about LDA, we can conclude the same thing for FLD

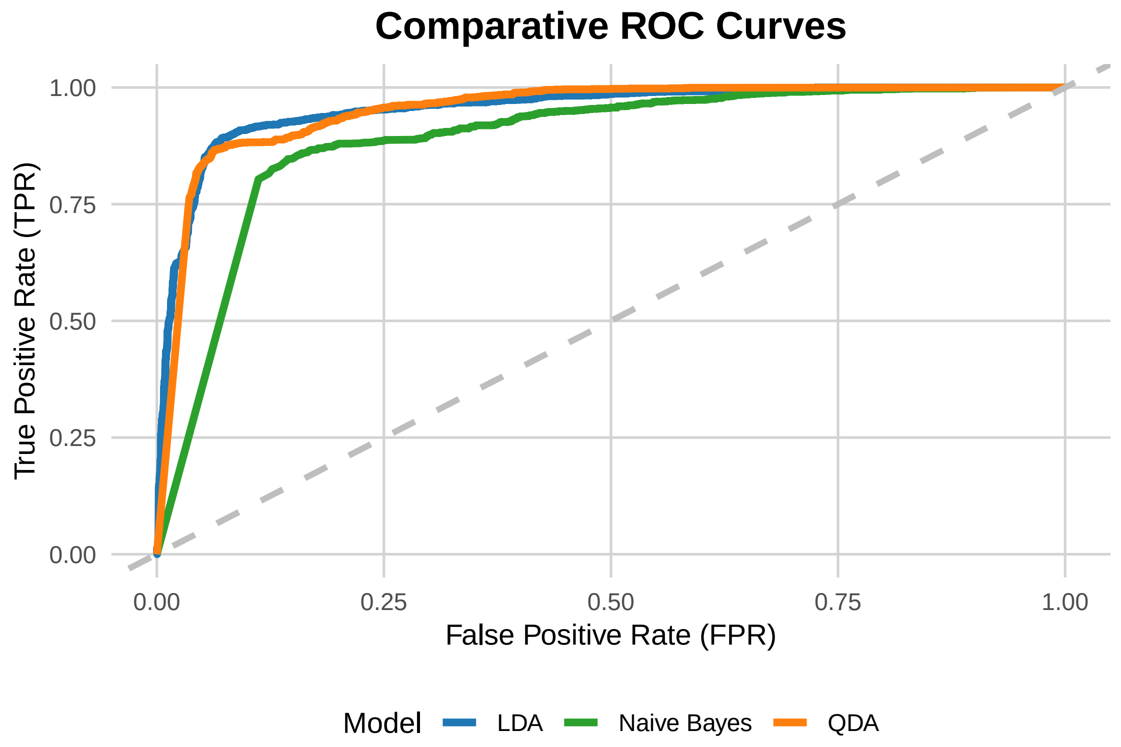
But first of all, let's perform a summary on out input space to see if you face a scale problem or not.

Based on the results, we can say that we have a big scale problem. While the mean of some variables is less than 1, we have another variables which have mean greater than 100. We can remark this on the following variables for exemple :

```
##       make           address
##  Min.   :0.0000   Min.   : 0.000
##  1st Qu.:0.0000   1st Qu.: 0.000
##  Median :0.0000   Median : 0.000
##  Mean   :0.1046   Mean   : 0.213
##  3rd Qu.:0.0000   3rd Qu.: 0.000
##  Max.   :4.5400   Max.   :14.280
```

```
##   capitalLong       capitalTotal
##  Min.   :   1.00   Min.   :    1.0
##  1st Qu.:   6.00   1st Qu.:   35.0
##  Median :  15.00   Median :   95.0
##  Mean   :  52.17   Mean   :  283.3
##  3rd Qu.:  43.00   3rd Qu.:  266.0
##  Max.   :9989.00   Max.   :15841.0
```

We thus normalize our variables before plot the following ROC curves.



5. Comment on the ROC curve

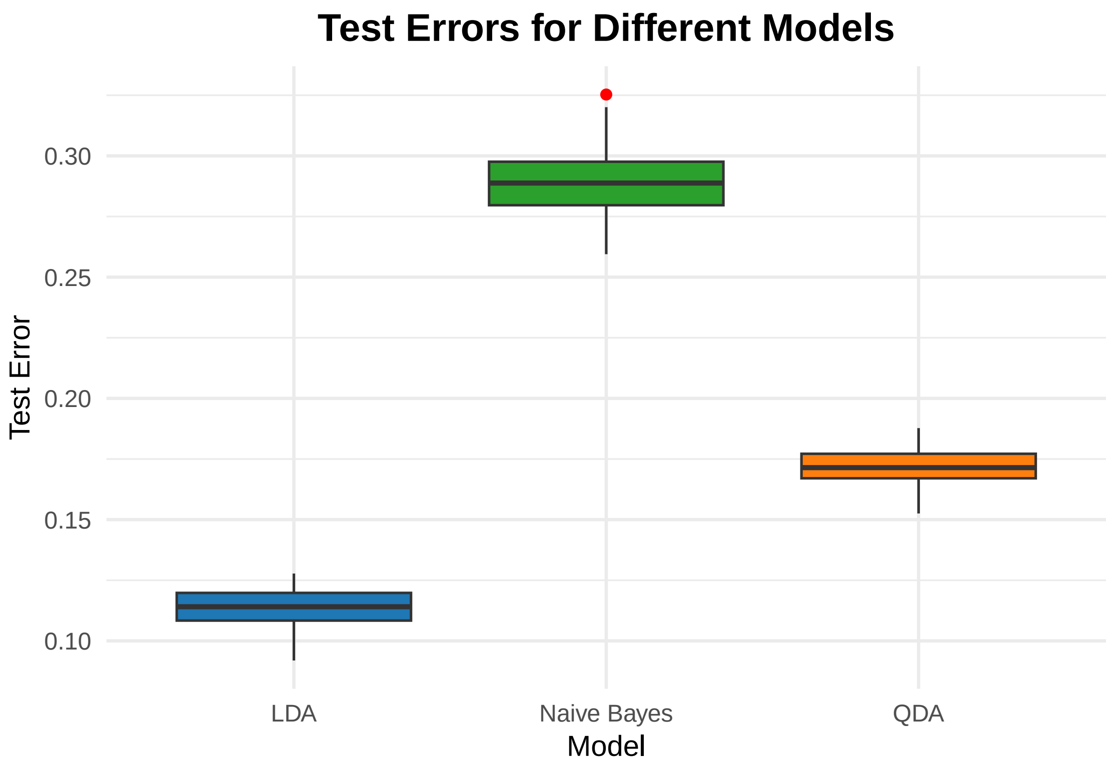
According to the ROC curves, we can say that all of this models seem perform enough well but this is normal because **we use the same base to train and predict**, then, we'll be able to draw best and more **trues** conclusions when we'll test the machine using a data different from which one is used for training because this will give us the real capacity of the model to predict on new data. Anyway, based on this plot, we can draw more conclusion like :

- The worse model is the Naive Bayes and this is understandable because according to our previous definitions, this model assume that the independence between predictors, but we know that for our dataset, all the predictors are collected for the same text, for each observations, which typically in pratice don't verified this assumption.
- The models LDA (or FLD) and QDA appears as the best models for this

task. LDA seem like more better than QDA but the difference likely to not be statistically significant. Again, this make sens, considering how our sample is collected. Work under the assumption that the data from each class share the same covariance matrix could lead to have a good model.

6. Let's compute 50 replications of the test error for all the machines above

7- Let's plot the comparatives boxplots for this machines



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

- **LDA (or FLD) :** We can remark that the test error for this model is consistently low, with minimal variation, as indicated by the small spread of the boxplot. It achieves the *lowest median test error* among the three models, suggesting that it is likely the most robust and accurate model **for this dataset**
- **Naive Bayes :** This machine has a significant high median test error compared to LDA and QDA. We can see a larger spread, which implies its performance is less stable across different test split

- **QDA :** This machine has a median test error lower than Naive Bayes but higher than LDA which suggests that this machine perform well than Naive Bayes but not so well like the LDA machine.
- **As overall conclusion, we can say that LDA appear to perform the best in terms of both accuracy (lowest test error) and stability (least variance)**

Exercice 3

Link of my video