se182947-mai-tu225186165n-anh

October 29, 2024

1 Part 1. Overfit - Underfit Models

2 Generate Training and Test Data

This script generates synthetic training and test datasets for a regression problem. The target variable `y` is a polynomial function of `X` with added random noise. The training set contains 30 data points, while the test set contains 20 data points. A seed is set for reproducibility.

[ ]:
```r
# Tạo dữ liệu huấn luyện và kiểm tra
set.seed(123)  # Đặt seed để kết quả tái lập được

N <- 30
N_test <- 20

# Tạo dữ liệu huấn luyện
X <- runif(N, 0, 5)
y <- 3 * (X - 2) * (X - 3) * (X - 4) + 10 * rnorm(N)

# Tạo dữ liệu kiểm tra
X_test <- (runif(N_test, 0, 1) - 1/8) * 10
y_test <- 3 * (X_test - 2) * (X_test - 3) * (X_test - 4) + 10 * rnorm(N_test)

# Xem trước dữ liệu
head(X)
head(y)
head(X_test)
head(y_test)
```

1. 1.43788760062307 2. 3.94152567721903 3. 2.0448846090585 4. 4.41508702002466 5. 4.70233642146923 6. 0.227782496949658

1. 11.1198912720452 2. 4.65783224907151 3. -19.4147242877303 4. 11.2693112800203 5. 4.964930539629 6. -66.2766382846067

1. 0.0569569156505167 2. 5.28101925039664 3. 2.18516472261399 4. 5.31758127966896 5. 1.95373242488131 6. 0.626911192666739

1. -52.4796906513331 2. 13.274177768682 3. 6.66759772041212 4. 31.6302580265448 5. 1.86224690676235 6. -29.1768470397883

3 Feature Matrix Construction Function

This function `buildX` constructs a feature matrix for polynomial regression. The matrix includes a bias term (column of ones) and columns of powers of the input vector `X` up to degree `d`. The result is a matrix that can be used for fitting a polynomial model.

[ ]:
```r
# Hàm xây dựng ma trận đặc trưng
buildX <- function(X, d = 2) {
  res <- matrix(1, nrow = length(X), ncol = 1)  # Tạo cột đầu tiên toàn giá trị 1 (bias)

  for (i in 1:d) {  # Vòng lặp từ 1 đến d
    res <- cbind(res, X^i)  # Nối các cột với nhau
  }

  return(res)  # Trả về ma trận đặc trưng
}
```

4 Polynomial Regression Function

This script defines the function `myfit` to perform polynomial regression using the feature matrix generated from `buildX`. It fits a linear model to the polynomial features, retrieves the model coefficients, and returns values for plotting the predicted curve (`y0`) and the true curve (`ytrue`) over a specified range of input values (`x0`).

[ ]:
```r
library(MASS)


# Hàm thực hiện hồi quy đa thức
myfit <- function(X, y, d) {
  Xbar <- buildX(X, d)  # Xây dựng ma trận đặc trưng với bậc d
  model <- lm(y ~ 0 + Xbar)  # Hồi quy tuyến tính mà không cần bias (intercept = FALSE)

  w <- coef(model)  # Trích xuất trọng số của mô hình
  w_0 <- w[1]
  w_1 <- w[2]

  x0 <- seq(-2, 7, length.out = 200)  # Tạo tập giá trị x0 cho đồ thị
  y0 <- rep(0, length(x0))
  ytrue <- 5 * (x0 - 2) * (x0 - 3) * (x0 - 4)  # Công thức mô hình thật

  for (i in 0:d) {
    y0 <- y0 + w[i+1] * x0^i  # Cộng các thành phần vào y0
  }
```

```
  return(list(x0 = x0, y0 = y0, ytrue = ytrue))  # Trả về các giá trị để vẽ đồ␣
 ↪thị
}
```

5 Plotting Results Function

This script defines the `plot_results` function, which visualizes the training and test data along with the polynomial regression model's predictions and the true underlying model. It uses `ggplot2` for plotting and allows for customization based on the degree of the polynomial.

[ ]:
```
library(ggplot2)

# Vẽ đồ thị
plot_results <- function(X, y, X_test, y_test, x0, y0, ytrue, w, d) {
  # Chuyển đổi dữ liệu thành dataframe cho ggplot
  df_train <- data.frame(X = X, y = y)
  df_test <- data.frame(X_test = X_test, y_test = y_test)
  df_model <- data.frame(x0 = x0, y0 = y0, ytrue = ytrue)

  # Vẽ dữ liệu huấn luyện
  p <- ggplot() +
    geom_point(data = df_train, aes(x = X, y = y), color = "red", size = 3,␣
 ↪label = "Training samples") +
    geom_point(data = df_test, aes(x = X_test, y = y_test), color = "yellow",␣
 ↪size = 3, label = "Test samples") +
    geom_line(data = df_model, aes(x = x0, y = y0), color = "blue", size = 1,␣
 ↪linetype = "solid", label = "Trained model") +
    geom_line(data = df_model, aes(x = x0, y = ytrue), color = "green", size =␣
 ↪1, linetype = "dashed", label = "True model") +
    theme_minimal() +
    theme(
      axis.title.x = element_blank(),
      axis.title.y = element_blank(),
      axis.text.x = element_blank(),
      axis.text.y = element_blank(),
      axis.ticks = element_blank()
    )

  # Tạo tiêu đề và chú thích
  if (d < 3) {
    str1 <- "Underfitting"
  } else if (d > 4) {
    str1 <- "Overfitting"
  } else {
    str1 <- "Good fit"
  }
```

```
  str0 <- paste("Degree =", d, ":", str1)
  p <- p + ggtitle(str0)

  # Điều chỉnh phạm vi của biểu đồ
  p <- p + xlim(-4, 10) + ylim(min(y_test) - 100, max(y) + 100)

  # Hiển thị chú thích
  p <- p + labs(color = "Legend") + theme(legend.position = "bottom")

  # Lưu hình ảnh
  fn <- paste0('linreg_', d, '.png')
  ggsave(fn, plot = p, dpi = 600, width = 8, height = 6, units = "in")

  # Hiển thị hình ảnh
  print(p)

  # In trọng số của mô hình
  print(w)
}
```

6 Running the Model with Different Degrees

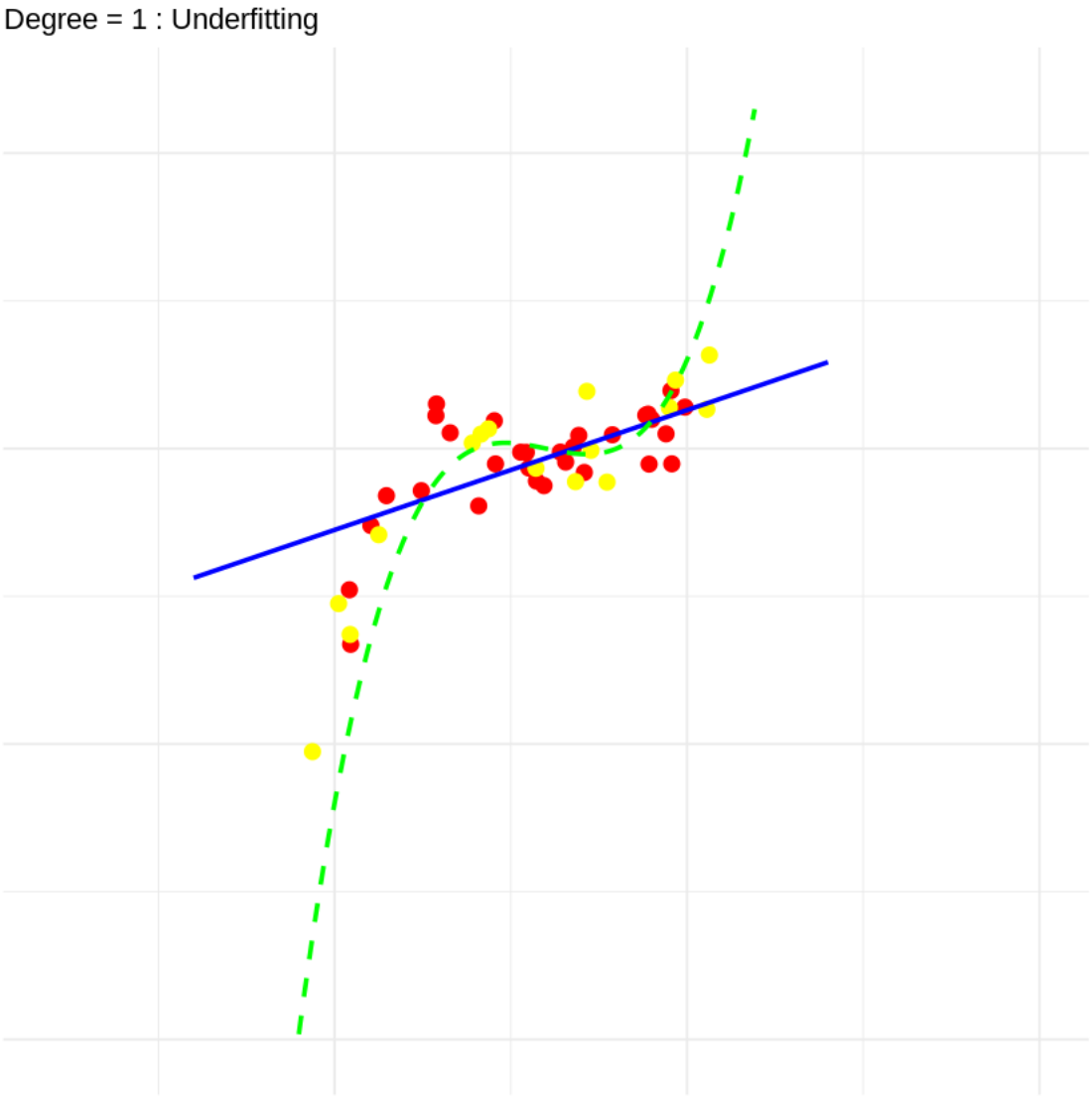
This code snippet runs the polynomial regression model with a degree of 1 and visualizes the results using the `plot_results` function. It fits the model to the training data and then generates a plot to compare the predicted values against the training and test data.

[ ]:
```
# Chạy mô hình với các bậc khác nhau
fit1 <- myfit(X, y, 1)  # Hồi quy bậc 1
plot_results(X, y, X_test, y_test, fit1$x0, fit1$y0, fit1$ytrue, w = coef(lm(y␣
 ↪~ 0 + buildX(X, 1))), 1)
```

```
Warning message in geom_point(data = df_train, aes(x = X, y = y), color = "red",
:
"Ignoring unknown parameters: `label`"
Warning message in geom_point(data = df_test, aes(x = X_test, y = y_test), color
= "yellow", :
"Ignoring unknown parameters: `label`"
Warning message in geom_line(data = df_model, aes(x = x0, y = y0), color =
"blue", :
"Ignoring unknown parameters: `label`"
Warning message in geom_line(data = df_model, aes(x = x0, y = ytrue), color =
"green", :
"Ignoring unknown parameters: `label`"
Warning message:
"Removed 4 rows containing missing values or values outside the scale
range
```

```
(`geom_point()`)."
Warning message:
"Removed 56 rows containing missing values or values outside the scale
range
(`geom_line()`)."
Warning message:
"Removed 4 rows containing missing values or values outside the scale
range
(`geom_point()`)."
Warning message:
"Removed 56 rows containing missing values or values outside the scale
range
(`geom_line()`)."
```

```
buildX(X, 1)1 buildX(X, 1)2
   -27.548565      8.110575
```



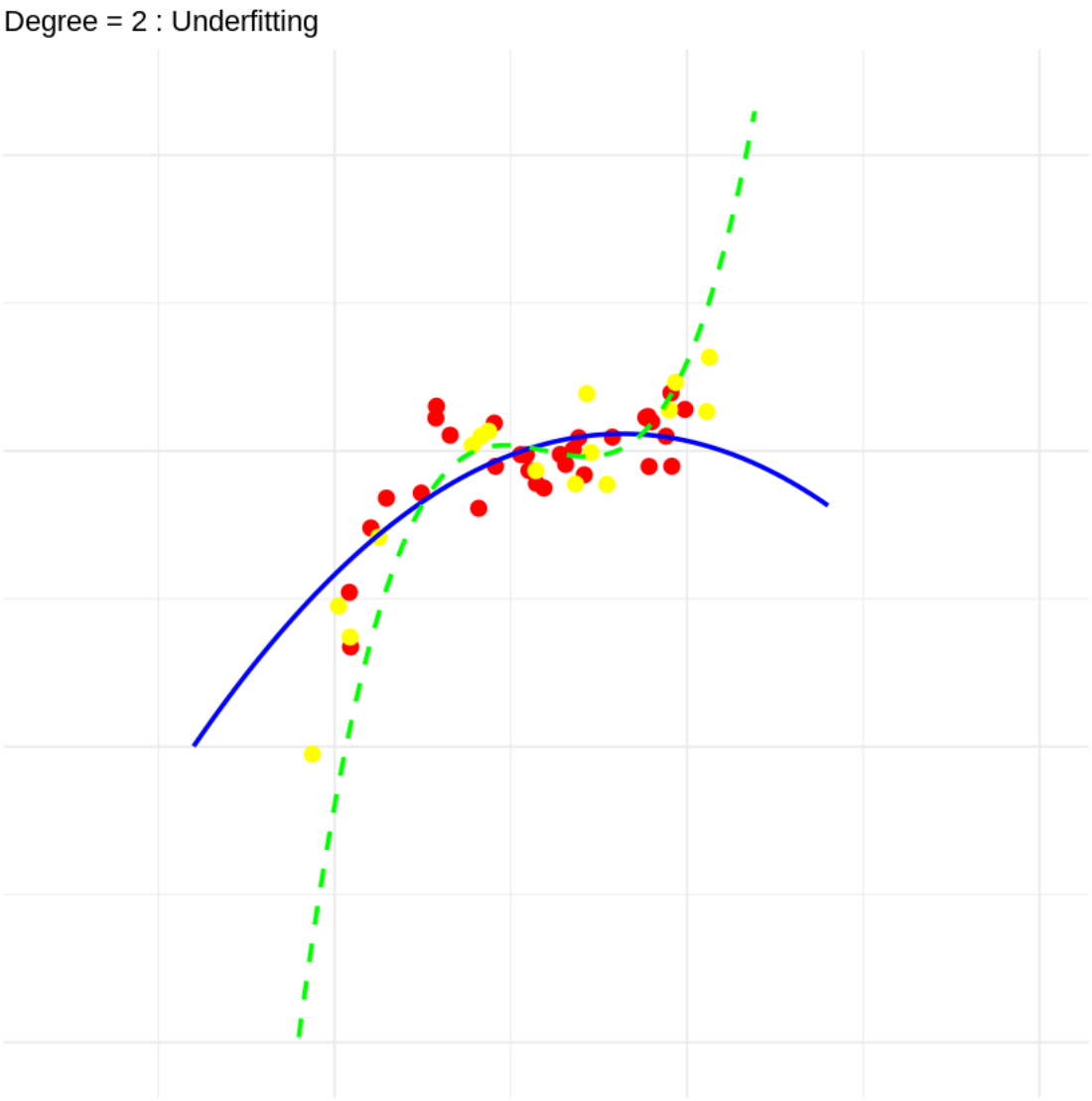
7 Running the Model with Degree 2

This code snippet fits the polynomial regression model with a degree of 2 and visualizes the results using the `plot_results` function. It compares the predicted values from the polynomial regression against the training and test datasets.

[ ]:
```
fit2 <- myfit(X, y, 2)  # Hồi quy bậc 2
plot_results(X, y, X_test, y_test, fit2$x0, fit2$y0, fit2$ytrue, w = coef(lm(y␣
  ↪~ 0 + buildX(X, 2))), 2)
```

```
Warning message in geom_point(data = df_train, aes(x = X, y = y), color = "red",
:
"Ignoring unknown parameters: `label`"
Warning message in geom_point(data = df_test, aes(x = X_test, y = y_test), color
= "yellow", :
"Ignoring unknown parameters: `label`"
Warning message in geom_line(data = df_model, aes(x = x0, y = y0), color =
"blue", :
"Ignoring unknown parameters: `label`"
Warning message in geom_line(data = df_model, aes(x = x0, y = ytrue), color =
"green", :
"Ignoring unknown parameters: `label`"
Warning message:
"Removed 4 rows containing missing values or values outside the scale
range
(`geom_point()`)."
Warning message:
"Removed 56 rows containing missing values or values outside the scale
range
(`geom_line()`)."
Warning message:
"Removed 4 rows containing missing values or values outside the scale
range
(`geom_point()`)."
Warning message:
"Removed 56 rows containing missing values or values outside the scale
range
(`geom_line()`)."
```

```
buildX(X, 2)1 buildX(X, 2)2 buildX(X, 2)3
   -41.835912     23.316910     -2.854078
```



8 Running the Model with Degree 3

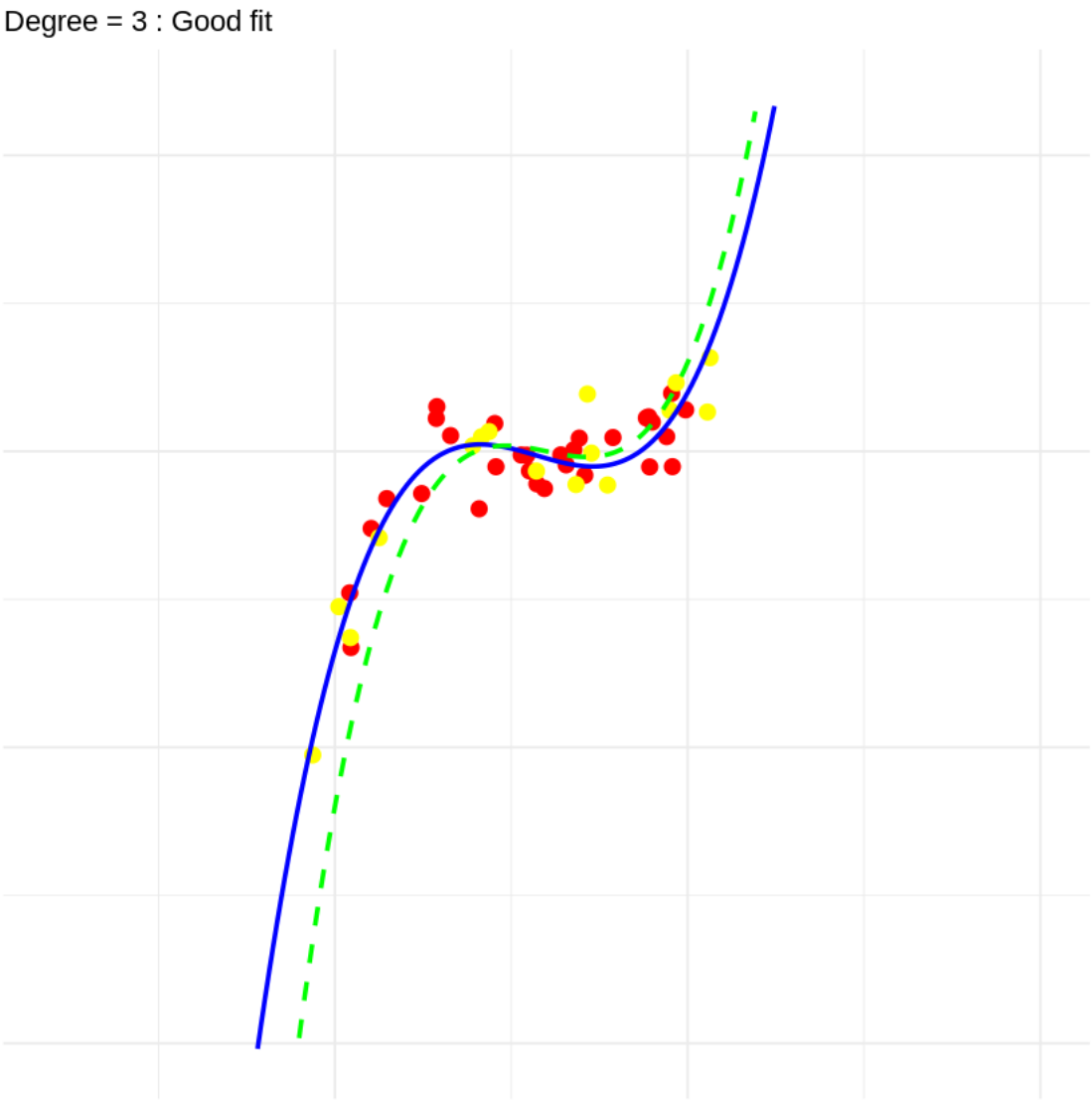
This code snippet fits the polynomial regression model with a degree of 3 and visualizes the results using the `plot_results` function. The plot will show how well the model captures the underlying trend compared to the training and test datasets.

[ ]:
```
fit3 <- myfit(X, y, 3)  # Hồi quy bậc 3
plot_results(X, y, X_test, y_test, fit3$x0, fit3$y0, fit3$ytrue, w = coef(lm(y␣
↪~ 0 + buildX(X, 3))), 3)
```

```
Warning message in geom_point(data = df_train, aes(x = X, y = y), color = "red",
:
"Ignoring unknown parameters: `label`"
```

```
Warning message in geom_point(data = df_test, aes(x = X_test, y = y_test), color
= "yellow", :
"Ignoring unknown parameters: `label`"
Warning message in geom_line(data = df_model, aes(x = x0, y = y0), color =
"blue", :
"Ignoring unknown parameters: `label`"
Warning message in geom_line(data = df_model, aes(x = x0, y = ytrue), color =
"green", :
"Ignoring unknown parameters: `label`"
Warning message:
"Removed 4 rows containing missing values or values outside the scale
range
(`geom_point()`)."
Warning message:
"Removed 37 rows containing missing values or values outside the scale
range
(`geom_line()`)."
Warning message:
"Removed 56 rows containing missing values or values outside the scale
range
(`geom_line()`)."
Warning message:
"Removed 4 rows containing missing values or values outside the scale
range
(`geom_point()`)."
Warning message:
"Removed 37 rows containing missing values or values outside the scale
range
(`geom_line()`)."
Warning message:
"Removed 56 rows containing missing values or values outside the scale
range
(`geom_line()`)."
```

```
buildX(X, 3)1 buildX(X, 3)2 buildX(X, 3)3 buildX(X, 3)4
   -67.554156     83.489897    -31.668300      3.693283
```



9 Running the Model with Degree 4

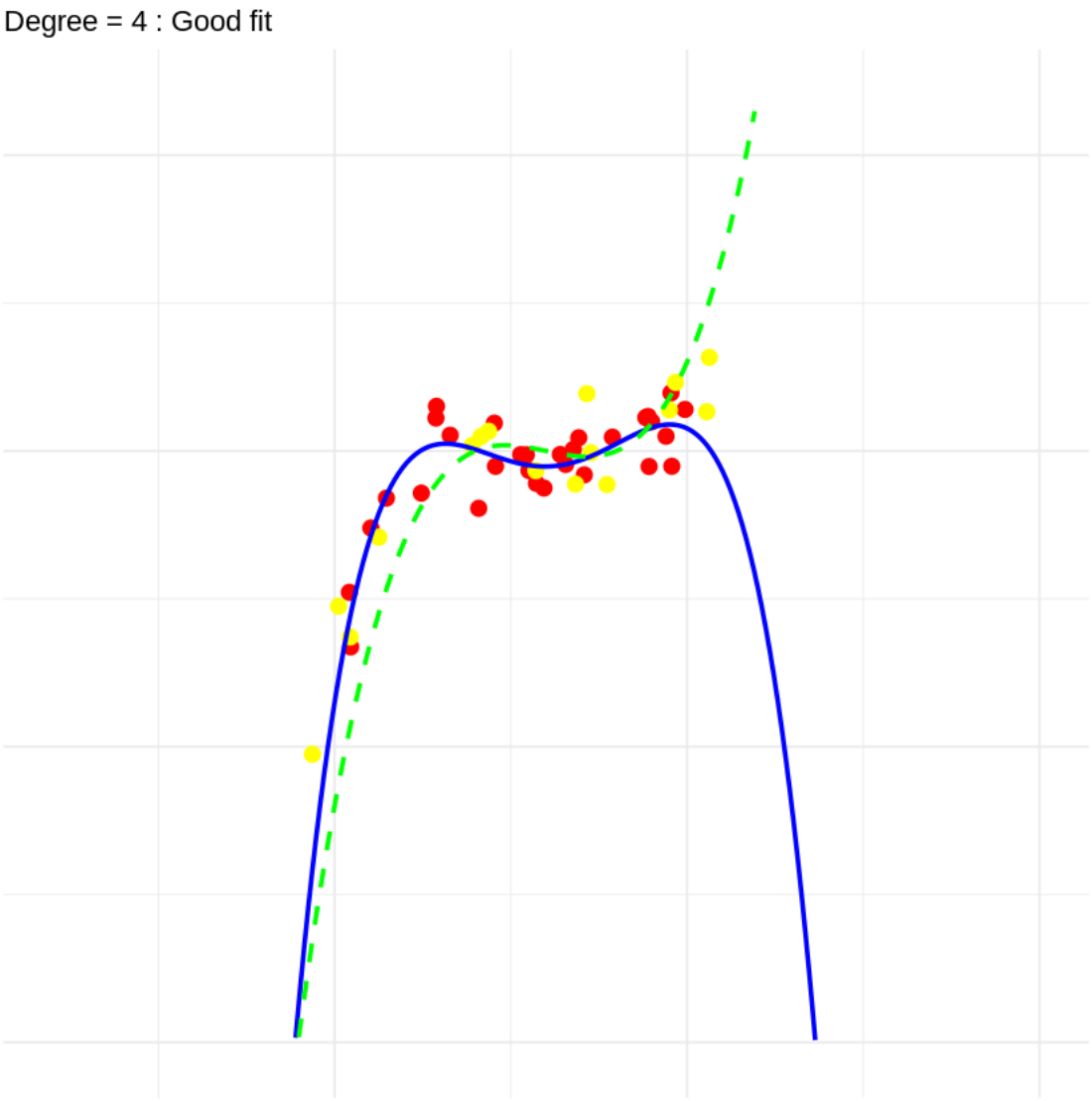
This code snippet fits the polynomial regression model with a degree of 4 and visualizes the results using the `plot_results` function. The resulting plot will illustrate the model's fit compared to both the training and test datasets.

[ ]:
```
fit4 <- myfit(X, y, 4)  # Hồi quy bậc 4
plot_results(X, y, X_test, y_test, fit4$x0, fit4$y0, fit4$ytrue, w = coef(lm(y␣
↪~ 0 + buildX(X, 4))), 4)
```

```
Warning message in geom_point(data = df_train, aes(x = X, y = y), color = "red",
:
"Ignoring unknown parameters: `label`"
```

```
Warning message in geom_point(data = df_test, aes(x = X_test, y = y_test), color
= "yellow", :
"Ignoring unknown parameters: `label`"
Warning message in geom_line(data = df_model, aes(x = x0, y = y0), color =
"blue", :
"Ignoring unknown parameters: `label`"
Warning message in geom_line(data = df_model, aes(x = x0, y = ytrue), color =
"green", :
"Ignoring unknown parameters: `label`"
Warning message:
"Removed 4 rows containing missing values or values outside the scale
range
(`geom_point()`)."
Warning message:
"Removed 36 rows containing missing values or values outside the scale
range
(`geom_line()`)."
Warning message:
"Removed 56 rows containing missing values or values outside the scale
range
(`geom_line()`)."
Warning message:
"Removed 4 rows containing missing values or values outside the scale
range
(`geom_point()`)."
Warning message:
"Removed 36 rows containing missing values or values outside the scale
range
(`geom_line()`)."
Warning message:
"Removed 56 rows containing missing values or values outside the scale
range
(`geom_line()`)."
```

```
buildX(X, 4)1 buildX(X, 4)2 buildX(X, 4)3 buildX(X, 4)4 buildX(X, 4)5
   -85.076154    149.622947    -88.308307     20.790457     -1.674414
```

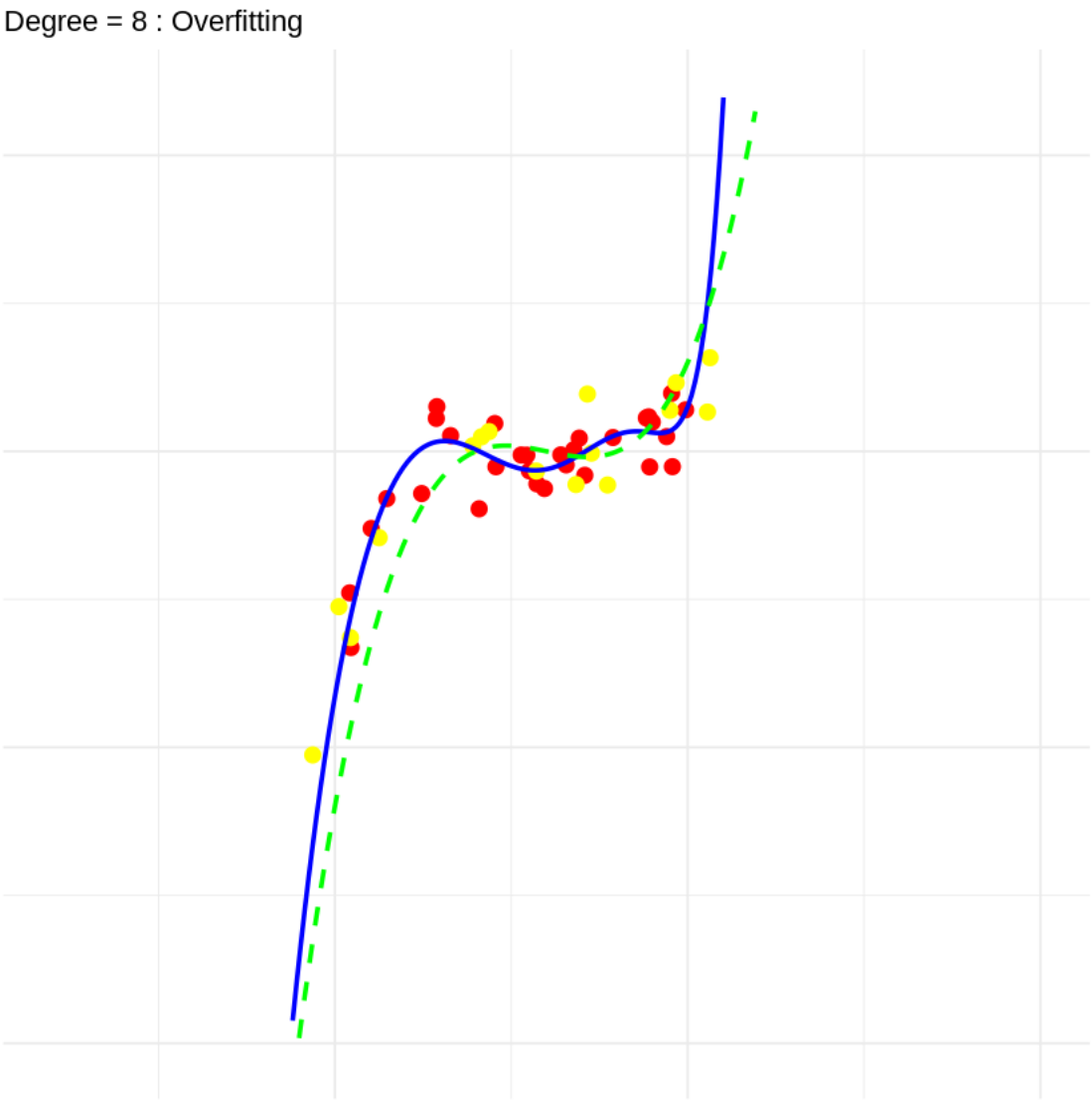


[ ]:
```
fit8 <- myfit(X, y, 8)  # Hồi quy bậc 8
plot_results(X, y, X_test, y_test, fit8$x0, fit8$y0, fit8$ytrue, w = coef(lm(y␣
  ↪~ 0 + buildX(X, 8))), 8)
```

```
Warning message in geom_point(data = df_train, aes(x = X, y = y), color = "red",
:
"Ignoring unknown parameters: `label`"
Warning message in geom_point(data = df_test, aes(x = X_test, y = y_test), color
= "yellow", :
"Ignoring unknown parameters: `label`"
Warning message in geom_line(data = df_model, aes(x = x0, y = y0), color =
"blue", :
"Ignoring unknown parameters: `label`"
```

```
Warning message in geom_line(data = df_model, aes(x = x0, y = ytrue), color =
"green", :
"Ignoring unknown parameters: `label`"
Warning message:
"Removed 4 rows containing missing values or values outside the scale
range
(`geom_point()`)."
Warning message:
"Removed 64 rows containing missing values or values outside the scale
range
(`geom_line()`)."
Warning message:
"Removed 56 rows containing missing values or values outside the scale
range
(`geom_line()`)."
Warning message:
"Removed 4 rows containing missing values or values outside the scale
range
(`geom_point()`)."
Warning message:
"Removed 64 rows containing missing values or values outside the scale
range
(`geom_line()`)."
Warning message:
"Removed 56 rows containing missing values or values outside the scale
range
(`geom_line()`)."
```

```
buildX(X, 8)1 buildX(X, 8)2 buildX(X, 8)3 buildX(X, 8)4 buildX(X, 8)5
 -82.81991817  136.21659090  -70.39466198   13.81402962   -0.61054042
buildX(X, 8)6 buildX(X, 8)7 buildX(X, 8)8 buildX(X, 8)9
  -1.41106981    0.95291875   -0.22265680    0.01727155
```



10 Running the Model with Degree 11

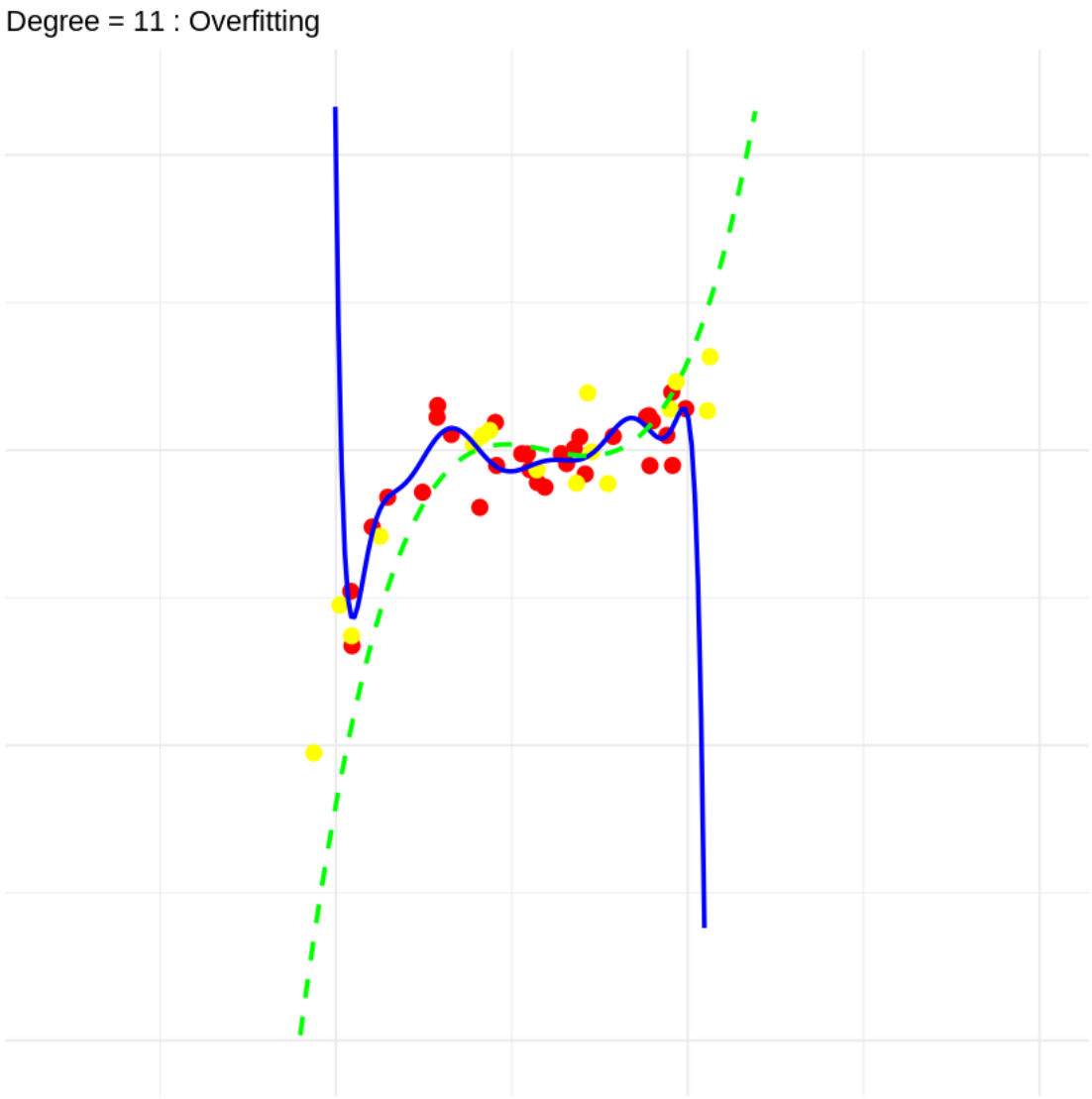
This code snippet fits the polynomial regression model with a degree of 11 and visualizes the results using the `plot_results` function. A degree this high may lead to overfitting, which will be reflected in the resulting plot.

[ ]:
```
fit11 <- myfit(X, y, 11) # Hồi quy bậc 11
plot_results(X, y, X_test, y_test, fit11$x0, fit11$y0, fit11$ytrue, w =␣
↪coef(lm(y ~ 0 + buildX(X, 11))), 11)
```

```
Warning message in geom_point(data = df_train, aes(x = X, y = y), color = "red",
:
"Ignoring unknown parameters: `label`"
```

```
Warning message in geom_point(data = df_test, aes(x = X_test, y = y_test), color
= "yellow", :
"Ignoring unknown parameters: `label`"
Warning message in geom_line(data = df_model, aes(x = x0, y = y0), color =
"blue", :
"Ignoring unknown parameters: `label`"
Warning message in geom_line(data = df_model, aes(x = x0, y = ytrue), color =
"green", :
"Ignoring unknown parameters: `label`"
Warning message:
"Removed 4 rows containing missing values or values outside the scale
range
(`geom_point()`)."
Warning message:
"Removed 83 rows containing missing values or values outside the scale
range
(`geom_line()`)."
Warning message:
"Removed 56 rows containing missing values or values outside the scale
range
(`geom_line()`)."
Warning message:
"Removed 4 rows containing missing values or values outside the scale
range
(`geom_point()`)."
Warning message:
"Removed 83 rows containing missing values or values outside the scale
range
(`geom_line()`)."
Warning message:
"Removed 56 rows containing missing values or values outside the scale
range
(`geom_line()`)."
```

buildX(X, 11)1	buildX(X, 11)2	buildX(X, 11)3	buildX(X, 11)4	buildX(X, 11)5
9.711857e+01	-1.826322e+03	8.032828e+03	-1.719411e+04	2.136714e+04
buildX(X, 11)6	buildX(X, 11)7	buildX(X, 11)8	buildX(X, 11)9	buildX(X, 11)10
-1.664913e+04	8.458041e+03	-2.842790e+03	6.267464e+02	-8.712374e+01
buildX(X, 11)11	buildX(X, 11)12			
6.923647e+00	-2.397252e-01			



- Bias đề cập đến các lỗi xảy ra khi chúng ta cố gắng điều chỉnh mô hình thống kê trên dữ liệu thực tế mà không phù hợp hoàn toàn với một số mô hình toán học. Nếu chúng ta sử dụng một mô hình quá đơn giản để điều chỉnh dữ liệu thì chúng ta có nhiều khả năng phải đối mặt với tình huống Bias cao , tức là trường hợp mô hình không thể học được các mẫu trong dữ liệu đang xét và do đó hoạt động kém.
- Phương sai có nghĩa là giá trị của lỗi khi chúng ta cố gắng đưa ra dự đoán bằng cách sử dụng dữ liệu mà mô hình chưa từng thấy trước đó. Có một tình huống được gọi là phương sai cao xảy ra khi mô hình học được nhiễu có trong dữ liệu.

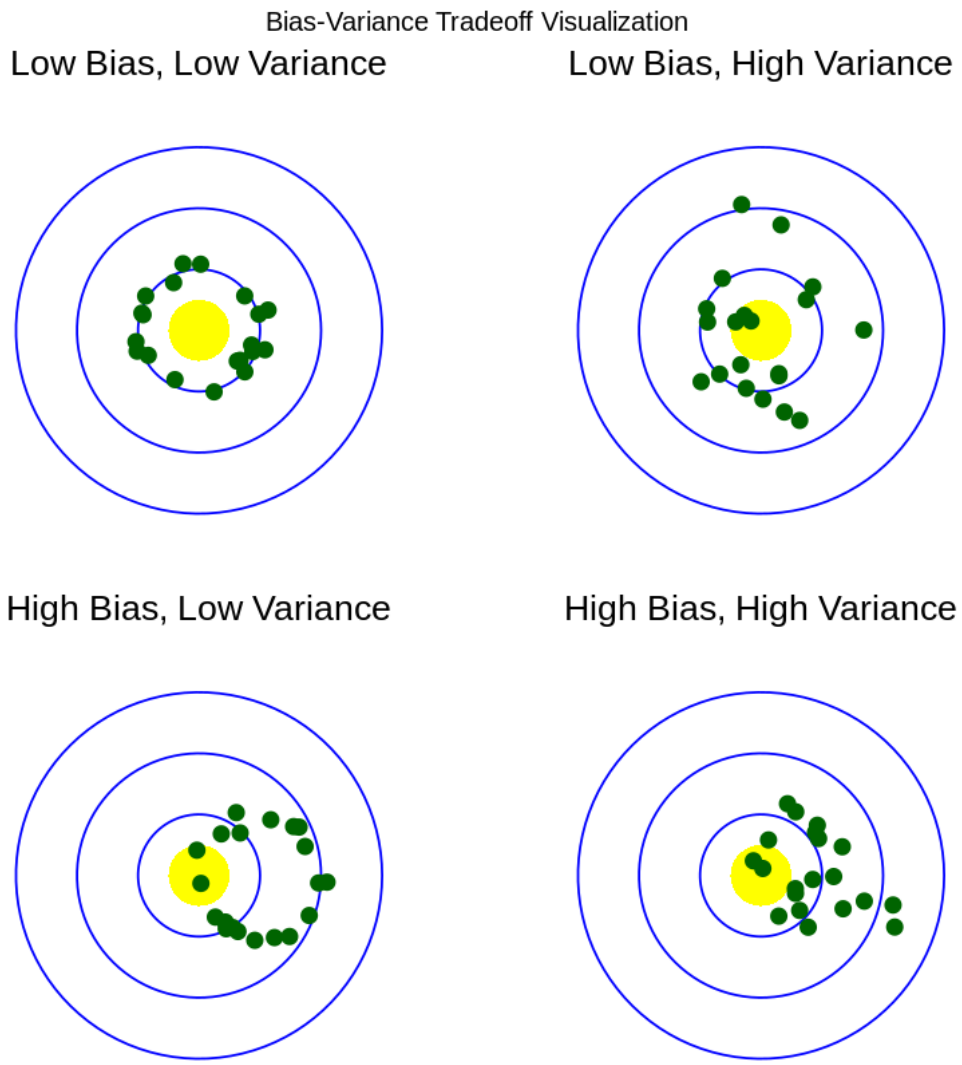
[ ]:
```
install.packages("gridExtra")
```

```
Installing package into ‘/usr/local/lib/R/site-library’
```

```
(as 'lib' is unspecified)
```

[ ]:
```r
library(ggplot2)
library(gridExtra)

# Hàm tạo dữ liệu mô phỏng cho từng trường hợp
generate_target_data <- function(n_points = 20, spread = 0.2, center_bias = 0,␣
↪radius_bias = 0) {
  # Tạo góc ngẫu nhiên
  angles <- runif(n_points, 0, 2 * pi)

  # Tạo bán kính với nhiễu
  r <- rnorm(n_points, 1 + radius_bias, spread)

  # Chuyển đổi từ tọa độ cực sang tọa độ Descartes
  x <- r * cos(angles) + center_bias
  y <- r * sin(angles)

  data.frame(x = x, y = y)
}

# Hàm để vẽ vòng tròn
draw_circle <- function(center = c(0, 0), radius = 1, npoints = 100) {
  tt <- seq(0, 2 * pi, length.out = npoints)
  data.frame(x = center[1] + radius * cos(tt), y = center[2] + radius * sin(tt))
}

# Tạo plot cho một scenario
create_target_plot <- function(data, title, spread) {
  circles <- rbind(
    draw_circle(radius = 3),
    draw_circle(radius = 2),
    draw_circle(radius = 1),
    draw_circle(radius = 0.5)
  )

  ggplot() +
    geom_polygon(data = circles[1:100, ], aes(x = x, y = y), color = "blue",␣
↪fill = NA) +
    geom_polygon(data = circles[101:200, ], aes(x = x, y = y), color = "blue",␣
↪fill = NA) +
    geom_polygon(data = circles[201:300, ], aes(x = x, y = y), color = "blue",␣
↪fill = NA) +
    geom_polygon(data = circles[301:400, ], aes(x = x, y = y), color = NA, fill␣
↪= "yellow") +
    geom_point(data = data, aes(x = x, y = y), color = "darkgreen", size = 3) +
```

```
    coord_fixed() +
    theme_minimal() +
    theme(
      plot.title = element_text(hjust = 0.5, size = 16),
      axis.title = element_blank(),
      axis.text = element_blank(),
      axis.ticks = element_blank(),
      panel.grid = element_blank()
    ) +
    xlim(-3.5, 3.5) +
    ylim(-3.5, 3.5) +
    ggtitle(title)
}

# Tạo dữ liệu cho 4 trường hợp
set.seed(123)
low_bias_low_var <- generate_target_data(n_points = 20, spread = 0.1,␣
 ↪center_bias = 0)
low_bias_high_var <- generate_target_data(n_points = 20, spread = 0.5,␣
 ↪center_bias = 0)
high_bias_low_var <- generate_target_data(n_points = 20, spread = 0.1,␣
 ↪center_bias = 1)
high_bias_high_var <- generate_target_data(n_points = 20, spread = 0.5,␣
 ↪center_bias = 1)

# Tạo 4 plots
p1 <- create_target_plot(low_bias_low_var, "Low Bias, Low Variance", 0.1)
p2 <- create_target_plot(low_bias_high_var, "Low Bias, High Variance", 0.5)
p3 <- create_target_plot(high_bias_low_var, "High Bias, Low Variance", 0.1)
p4 <- create_target_plot(high_bias_high_var, "High Bias, High Variance", 0.5)

# Kết hợp các plots
combined_plot <- grid.arrange(p1, p2, p3, p4, ncol = 2,
                              top = "Bias-Variance Tradeoff Visualization")

# Lưu plot
ggsave("bias_variance_tradeoff.png", combined_plot, width = 12, height = 12,␣
 ↪dpi = 300)
```



- Độ lệch (Bias) cao, phương sai (variance) thấp: Một mô hình có độ lệch cao và phương sai
- Độ phương sai (variance) cao, độ lệch (Bias) thấp: Một mô hình có độ phương sai cao và đ
- Độ lệch (Bias) cao, phương sai (variance) cao: Một mô hình có độ lệch cao và phương sai ca
- Độ lệch (Bias) thấp, phương sai (variance) thấp: Một mô hình có độ lệch thấp và phương sai

11 LAB: Model Evaluation and Refinement

11.1 Objectives

After completing this lab, you will be able to:

- Evaluate and refine prediction models.

11.2 Table of Contents

1. **Model Evaluation**
   - Learn how to evaluate models to measure prediction performance.
2. **Over-fitting, Under-fitting, and Model Selection**
   - Understand the difference between over-fitting and under-fitting.
   - Explore how to select the right model based on data and evaluation metrics.
3. **Ridge Regression**
   - Apply Ridge Regression to reduce multicollinearity and improve model accuracy.
4. **Grid Search**
   - Use Grid Search to optimize hyperparameters and enhance model performance.

11.3 Setup

In this section, you will set up the working environment and download the sample dataset.
- Instructions for downloading data directly via the browser.
- Setting up the necessary libraries to perform the lab.

Note:
Follow the step-by-step instructions in the lab document to complete the tasks and compare prediction model results before and after refinement.

Setup

This function will download the dataset into your browser

This dataset was hosted on IBM Cloud object. Click HERE for free storage.

[ ]:
```
#  URL tải file dữ liệu
path <- 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/
↪IBMDeveloperSkillsNetwork-DA0101EN-SkillsNetwork/labs/Data%20files/
↪module_5_auto.csv'
```

you will need to download the dataset; if you are running locally, please comment out the following

[ ]:
```
# Đặt tên file mới khi lưu về local
local_file <- "auto.csv"

# Hàm download file
download_data <- function(url, filename) {
  tryCatch({
    download.file(url = url,
                  destfile = filename,
                  mode = "wb")

    if (file.exists(filename)) {
      message("File downloaded successfully!")
```

```
      return(TRUE)
    } else {
      warning("Download failed - File does not exist")
      return(FALSE)
    }
  }, error = function(e) {
    warning(paste("Error downloading file:", e$message))
    return(FALSE)
  })
}

# Tải file về máy
download_data(path, local_file)

# Cập nhật đường dẫn để trỏ đến file local
path <- local_file

# Đọc dữ liệu nếu file tồn tại
if (file.exists(path)) {
  df <- read.csv(path)
  print(head(df))  # Hiển thị vài dòng đầu tiên của dữ liệu
} else {
  warning("File không tồn tại sau khi tải về. Vui lòng kiểm tra URL và thử lại.
  ↪")
}
```

```
Warning message in download.file(url = url, destfile = filename, mode = "wb"):
"URL 'http://auto.csv/': status was 'Couldn't resolve host name'"
Warning message in value[[3L]](cond):
"Error downloading file: cannot open URL 'auto.csv'"
```

FALSE

```
Warning message:
"File không tồn tại sau khi tải về. Vui lòng kiểm tra URL và thử lại."
```

First, let's only use numeric data:

[ ]:
```
str(df)
```

```
'data.frame':   201 obs. of  31 variables:
 $ X                : int  0 1 2 3 4 5 6 7 8 9 …
 $ Unnamed..0       : int  0 1 2 3 4 5 6 7 8 9 …
 $ symboling        : int  3 3 1 2 2 2 1 1 1 2 …
 $ normalized.losses: int  122 122 122 164 164 122 158 122 158 192 …
 $ make             : chr  "alfa-romero" "alfa-romero" "alfa-romero" "audi" …
 $ aspiration       : chr  "std" "std" "std" "std" …
 $ num.of.doors     : chr  "two" "two" "two" "four" …
 $ body.style       : chr  "convertible" "convertible" "hatchback" "sedan" …
```

```
 $ drive.wheels     : chr  "rwd" "rwd" "rwd" "fwd" …
 $ engine.location  : chr  "front" "front" "front" "front" …
 $ wheel.base       : num  88.6 88.6 94.5 99.8 99.4 …
 $ length           : num  0.811 0.811 0.823 0.849 0.849 …
 $ width            : num  0.89 0.89 0.91 0.919 0.922 …
 $ height           : num  48.8 48.8 52.4 54.3 54.3 53.1 55.7 55.7 55.9 54.3 …
 $ curb.weight      : int  2548 2548 2823 2337 2824 2507 2844 2954 3086 2395 …
 $ engine.type      : chr  "dohc" "dohc" "ohcv" "ohc" …
 $ num.of.cylinders : chr  "four" "four" "six" "four" …
 $ engine.size      : int  130 130 152 109 136 136 136 136 131 108 …
 $ fuel.system      : chr  "mpfi" "mpfi" "mpfi" "mpfi" …
 $ bore             : num  3.47 3.47 2.68 3.19 3.19 3.19 3.19 3.19 3.13 3.5 …
 $ stroke           : num  2.68 2.68 3.47 3.4 3.4 3.4 3.4 3.4 3.4 2.8 …
 $ compression.ratio: num  9 9 9 10 8 8.5 8.5 8.5 8.3 8.8 …
 $ horsepower       : num  111 111 154 102 115 110 110 110 140 101 …
 $ peak.rpm         : num  5000 5000 5000 5500 5500 5500 5500 5500 5500 5800 …
 $ city.mpg         : int  21 21 19 24 18 19 19 19 17 23 …
 $ highway.mpg      : int  27 27 26 30 22 25 25 25 20 29 …
 $ price            : num  13495 16500 16500 13950 17450 …
 $ city.L.100km     : num  11.19 11.19 12.37 9.79 13.06 …
 $ horsepower.binned: chr  "Medium" "Medium" "Medium" "Medium" …
 $ diesel           : int  0 0 0 0 0 0 0 0 0 0 …
 $ gas              : int  1 1 1 1 1 1 1 1 1 1 …
```

[ ]:
```
# Kiểm tra dữ liệu
head(df)
```

A data.frame: 6 × 31

	X <int>	Unnamed..0 <int>	symboling <int>	normalized.losses <int>	make <chr>	aspiration <chr>	num. <chr
1	0	0	3	122	alfa-romero	std	two
2	1	1	3	122	alfa-romero	std	two
3	2	2	1	122	alfa-romero	std	two
4	3	3	2	164	audi	std	four
5	4	4	2	164	audi	std	four
6	5	5	2	122	audi	std	two

Libraries for plotting:

[ ]:
```
install.packages("manipulate")
```

```
Installing package into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)
```

Functions for Plotting

```
DistributionPlot <- function(RedFunction, BlueFunction, RedName, BlueName, Title) {
 width <- 12
 height <- 10
```

```
 # Tạo dataframe cho plotting
df_red <- data.frame(value = RedFunction, group = RedName)
df_blue <- data.frame(value = BlueFunction, group = BlueName)
df_combined <- rbind(df_red, df_blue)

plot <- ggplot(df_combined, aes(x = value, fill = group)) +
  geom_density(alpha = 0.5) +
  scale_fill_manual(values = c("red", "blue")) +
  labs(title = Title,
       x = "Price (in dollars)",
       y = "Proportion of Cars") +
  theme_minimal() +
  theme(plot.title = element_text(size = 14, face = "bold"),
        legend.title = element_blank())

print(plot)
}
```

Part 1: Training and Testing

An important step in testing your model is to split your data into training and testing data. We will place the target data price in a separate dataframe y_data:

[ ]:
```
# Tạo dataframe y_data chứa biến price
y_data <- data.frame(price = df$price)

# Kiểm tra kết quả
head(y_data)
```

A data.frame: 6 × 1

	price <dbl>
1	13495
2	16500
3	16500
4	13950
5	17450
6	15250

Drop price data in dataframe **x_data**:

[ ]:
```
#x_data <- subset(df, select = -price)

# Sử dụng toán tử [-]
x_data <- df[, !names(df) %in% c("price")]

# Kiểm tra kết quả
head(x_data)
dim(x_data)
```

A data.frame: 6 × 30

	X <int>	Unnamed..0 <int>	symboling <int>	normalized.losses <int>	make <chr>	aspiration <chr>	num.<chr
1	0	0	3	122	alfa-romero	std	two
2	1	1	3	122	alfa-romero	std	two
3	2	2	1	122	alfa-romero	std	two
4	3	3	2	164	audi	std	four
5	4	4	2	164	audi	std	four
6	5	5	2	122	audi	std	two

1. 201 2. 30

Now, we randomly split our data into training and testing data using the function train_test_split.

- Chia dữ liệu thành 90% cho training và 10% cho testing
- Sử dụng set.seed(1) để đảm bảo kết quả có thể tái tạo lại được
- Tạo ra 4 dataframe: x_train, x_test, y_train, y_test

[ ]:
```
install.packages("caret")
library(caret)
```

```
Installing package into '/usr/local/lib/R/site-library'
(as 'lib' is unspecified)
```

[ ]:
```
set.seed(123)

# Tạo chỉ số cho việc chia dữ liệu
training_index <- createDataPartition(y_data$price, p = 0.9, list = FALSE)

# Chia dữ liệu thành training và testing sets
x_train <- x_data[training_index, ]
x_test <- x_data[-training_index, ]
y_train <- y_data[training_index, ]
y_test <- y_data[-training_index, ]

# In ra số lượng mẫu trong mỗi tập
cat("number of test samples:", nrow(x_test), "\n")
cat("number of training samples:", nrow(x_train))

# Kiểm tra kích thước của các tập dữ liệu
dim(x_train)
dim(x_test)
```

```
number of test samples: 20
number of training samples: 181
```

1. 181 2. 30

1. 20 2. 30

The test_size parameter sets the proportion of data that is split into the testing set. In the above,

the testing set is 10% of the total dataset.

Question #1):

Use the function "train_test_split" to split up the dataset such that 40% of the data samples will be utilized for testing. Set the parameter "random_state" equal to zero. The output of the function should be the following: "x_train1" , "x_test1", "y_train1" and "y_test1".

[ ]:
```r
# Thiết lập random seed
set.seed(123)

# Tạo chỉ số cho việc chia dữ liệu (60% cho training)
training_index1 <- createDataPartition(y_data$price, p = 0.6, list = FALSE)

# Chia dữ liệu thành training và testing sets
x_train1 <- x_data[training_index1, ]
x_test1 <- x_data[-training_index1, ]
y_train1 <- y_data[training_index1, ]
y_test1 <- y_data[-training_index1, ]

# In ra số lượng mẫu trong mỗi tập
cat("number of test samples:", nrow(x_test1), "\n")
cat("number of training samples:", nrow(x_train1))
```

```
number of test samples: 80
number of training samples: 121
```

[ ]:
```r
# Kiểm tra kích thước các tập dữ liệu
dim(x_train1)
dim(x_test1)
```

1. 121 2. 30

1. 80 2. 30

Let's import LinearRegression from the module linear_model.

We create a Linear Regression object, fit the model using the feature "horsepower":

[ ]:
```r
# Tạo mô hình hồi quy tuyến tính với lm() trong R
lre <- lm(y_train ~ horsepower, data = x_train)

# Hiển thị tóm tắt mô hình
summary(lre)
```

```
Call:
lm(formula = y_train ~ horsepower, data = x_train)

Residuals:
    Min      1Q  Median      3Q     Max
```

```
-9606.3 -2149.7  -399.6  1808.0 17691.3

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -4989.523   1007.615  -4.952 1.69e-06 ***
horsepower    177.708      9.197  19.323  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4646 on 179 degrees of freedom
Multiple R-squared:  0.6759,        Adjusted R-squared:  0.6741
F-statistic: 373.4 on 1 and 179 DF,  p-value: < 2.2e-16
```

Let's calculate the R^2 on the test data:

[ ]:
```
# Dự đoán giá trị của y_test dựa trên mô hình đã huấn luyện
y_pred <- predict(lre, newdata = x_test)

# Tính R^2 bằng cách so sánh y_test và y_pred
SSE <- sum((y_test - y_pred)^2)  # Tổng bình phương sai số
SST <- sum((y_test - mean(y_test))^2)  # Tổng bình phương toàn phần
R_squared <- 1 - (SSE/SST)

# Hiển thị giá trị R^2
print(paste("R^2:", R_squared))
```

```
[1] "R^2: 0.27149977469876"
```

We can see the R^2 is much smaller using the test data compared to the training data.

[ ]:
```
# Dự đoán giá trị của y_train dựa trên mô hình đã huấn luyện
y_train_pred <- predict(lre, newdata = x_train)

# Tính R^2 cho tập dữ liệu huấn luyện
SSE_train <- sum((y_train - y_train_pred)^2)  # Tổng bình phương sai số cho dữ␣
↪liệu huấn luyện
SST_train <- sum((y_train - mean(y_train))^2)  # Tổng bình phương toàn phần cho␣
↪dữ liệu huấn luyện
R_squared_train <- 1 - (SSE_train / SST_train)

# Hiển thị giá trị R^2 cho dữ liệu huấn luyện
print(paste("R^2 (Training Data):", R_squared_train))
```

```
[1] "R^2 (Training Data): 0.675941422146126"
```

Question #2):

Find the R^2 on the test data using 40% of the dataset for testing.

[ ]:
```
library(dplyr)
```

```
Attaching package: ‘dplyr’


The following object is masked from ‘package:MASS’:

    select


The following objects are masked from ‘package:stats’:

    filter, lag


The following objects are masked from ‘package:base’:

    intersect, setdiff, setequal, union
```

[ ]:
```
# Thiết lập random seed
set.seed(123)

# Chia dữ liệu thành training và testing sets (40% cho testing)
training_index1 <- createDataPartition(y_data$price, p = 0.6, list = FALSE)

# Chia dữ liệu
x_train1 <- x_data[training_index1, ]
x_test1 <- x_data[-training_index1, ]
y_train1 <- y_data[training_index1, ]
y_test1 <- y_data[-training_index1, ]

# Tạo mô hình hồi quy tuyến tính
lre <- lm(price ~ horsepower, data = data.frame(x_train1, price = y_train1))

# Dự đoán giá trị trên tập kiểm tra
y_pred1 <- predict(lre, newdata = x_test1)

# Tính R^2 cho tập dữ liệu kiểm tra
SSE_test1 <- sum((y_test1 - y_pred1)^2)  # Tổng bình phương sai số cho dữ liệu␣
 ↪kiểm tra
SST_test1 <- sum((y_test1 - mean(y_test1))^2)  # Tổng bình phương toàn phần cho␣
 ↪dữ liệu kiểm tra
R_squared_test1 <- 1 - (SSE_test1 / SST_test1)
```

```r
# Hiển thị giá trị R^2 cho dữ liệu kiểm tra
cat("R^2 (Test Data):", R_squared_test1, "\n")
```

```
R^2 (Test Data): 0.5537754
```

Sometimes you do not have sufficient testing data; as a result, you may want to perform cross-validation. Let's go over several methods that you can use for cross-validation.

Cross-Validation Score

Let's import model_selection from the module cross_val_score.

We input the object, the feature ("horsepower"), and the target data (y_data). The parameter 'cv' determines the number of folds. In this case, it is 4.

[ ]:
```r
# Tạo một data frame chứa chỉ biến 'horsepower'
x_data_horsepower <- data.frame(horsepower = x_data$horsepower)

# Thiết lập kiểm soát cho cross-validation
ctrl <- trainControl(method = "cv", number = 4)

# Thực hiện cross-validation
set.seed(123)  # Để kết quả có thể tái tạo được
lm_model <- train(x = x_data_horsepower,  # Sử dụng x_data_horsepower đã tạo
                  y = y_data$price,
                  method = "lm",
                  trControl = ctrl)

# Lấy kết quả R-squared
Rcross <- lm_model$resample$Rsquared
```

The default scoring is R^2. Each element in the array has the average R^2 value for the fold:

[ ]:
```r
# In kết quả
print(Rcross)
```

```
[1] 0.6773254 0.7456737 0.5443686 0.6269395
```

We can calculate the average and standard deviation of our estimate:

[ ]:
```r
# Tính và in giá trị trung bình và độ lệch chuẩn
cat("The mean of the folds is", mean(Rcross), "and the standard deviation is",␣
 ↪sd(Rcross), "\n")
```

```
The mean of the folds is 0.6485768 and the standard deviation is 0.08481708
```

We can use negative squared error as a score by setting the parameter 'scoring' metric to 'neg_mean_squared_error'.

- sử dụng metric = "RMSE" trong hàm train(). RMSE (Root Mean Squared Error) là căn bậc hai của MSE.
- Sau khi huấn luyện mô hình, lấy giá trị RMSE từ lm_model_mse*resample*RMSE.

- Tiếp theo bình phương RMSE để có MSE: mse_results <- lm_model_mse*resample*RMSE^2.
- Sau đó, nhân MSE với -1 để có negative MSE: neg_mse_results <- -1 * mse_results.

[ ]:
```r
# Lấy kết quả MSE (bình phương của RMSE)
mse_results <- lm_model$resample$RMSE^2

# Nhân với -1 để có negative MSE
neg_mse_results <- -1 * mse_results

# In kết quả
print(neg_mse_results)

# Tính và in giá trị trung bình và độ lệch chuẩn của negative MSE
cat("The mean of the negative MSE is", mean(neg_mse_results),
    "and the standard deviation is", sd(neg_mse_results), "\n")
```

```
[1] -21675218 -19348398 -24144482 -21900673
The mean of the negative MSE is -21767193 and the standard deviation is 1960302
```

Question #3):

Calculate the average R^2 using two folds, then find the average R^2 for the second fold utilizing the "horsepower" feature:

[ ]:
```r
# Lấy kết quả R-squared từ kết quả cross-validation
Rcross <- lm_model$resample$Rsquared

# Tính và in giá trị trung bình
mean_Rcross <- mean(Rcross)
print(mean_Rcross)
```

```
[1] 0.6485768
```

You can also use the function 'cross_val_predict' to predict the output. The function splits up the data into the specified number of folds, with one fold for testing and the other folds are used for training. First, import the function:

We input the object, the feature "horsepower", and the target data y_data. The parameter 'cv' determines the number of folds. In this case, it is 4. We can produce an output:

[ ]:
```r
# Thiết lập kiểm soát cho cross-validation
ctrl <- trainControl(method = "cv", number = 4)

# Thực hiện cross-validation và dự đoán giá trị
set.seed(123)  # Để kết quả có thể tái tạo được
lm_model <- train(x = x_data_horsepower,  # Sử dụng x_data_horsepower đã tạo
                  y = y_data$price,
                  method = "lm",
                  trControl = ctrl)
```

```r
# Dự đoán giá trị trên tập dữ liệu
yhat <- predict(lm_model, newdata = x_data_horsepower)

# In ra 5 giá trị đầu tiên
yhat[1:5]
```

1 14514.7682344177 **2** 14514.7682344177 **3** 21918.6424766641 **4** 12965.1201372033 **5** 15203.5007220685

Part 2: Overfitting, Underfitting and Model Selection

It turns out that the test data, sometimes referred to as the "out of sample data", is a much better measure of how well your model performs in the real world. One reason for this is overfitting.

Let's go over some examples. It turns out these differences are more apparent in Multiple Linear Regression and Polynomial Regression so we will explore overfitting in that context.

Let's create Multiple Linear Regression objects and train the model using 'horsepower', 'curb-weight', 'engine-size' and 'highway-mpg' as features.

[ ]:
```r
# Tạo mô hình hồi quy tuyến tính
lre_multi <- lm(price ~ horsepower + curb.weight + engine.size + highway.mpg,␣
 ↪data = data.frame(x_train, price = y_train))

# Hiển thị tóm tắt mô hình
summary(lre_multi)
```

```
Call:
lm(formula = price ~ horsepower + curb.weight + engine.size +
    highway.mpg, data = data.frame(x_train, price = y_train))

Residuals:
    Min      1Q  Median      3Q     Max
-9296.6 -1702.5     9.8  1266.9 13158.7

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -16827.622   4702.107  -3.579 0.000446 ***
horsepower      62.443     15.778   3.958 0.000110 ***
curb.weight      4.753      1.198   3.967 0.000106 ***
engine.size     78.146     14.624   5.344 2.79e-07 ***
highway.mpg     55.466     78.565   0.706 0.481125
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3533 on 176 degrees of freedom
Multiple R-squared:  0.8158,        Adjusted R-squared:  0.8116
F-statistic: 194.8 on 4 and 176 DF,  p-value: < 2.2e-16
```

Prediction using training data:

[ ]:
```
# Dự đoán với dữ liệu training
y_train_pred <- predict(lre_multi, newdata = x_train)

# In ra 5 giá trị đầu tiên
y_train_pred[1:5]
```

2 13871.7721376915 **3** 19527.7457581174 **4** 10832.149794084 **5** 15625.0259097114 **6** 13972.3846189165

Prediction using test data:

[ ]:
```
# Dự đoán với dữ liệu test
y_test_pred <- predict(lre_multi, newdata = x_test)

# In ra 5 giá trị đầu tiên
y_test_pred[1:5]
```

1 13871.7721376915 **19** 5869.44280524041 **23** 8979.0823345349 **43** 9434.34862185079 **60** 11182.0100707821

Let's perform some model evaluation using our training and testing data separately. First, we import the seaborn and matplotlib library for plotting.

[ ]:
```
library(ggplot2)
```

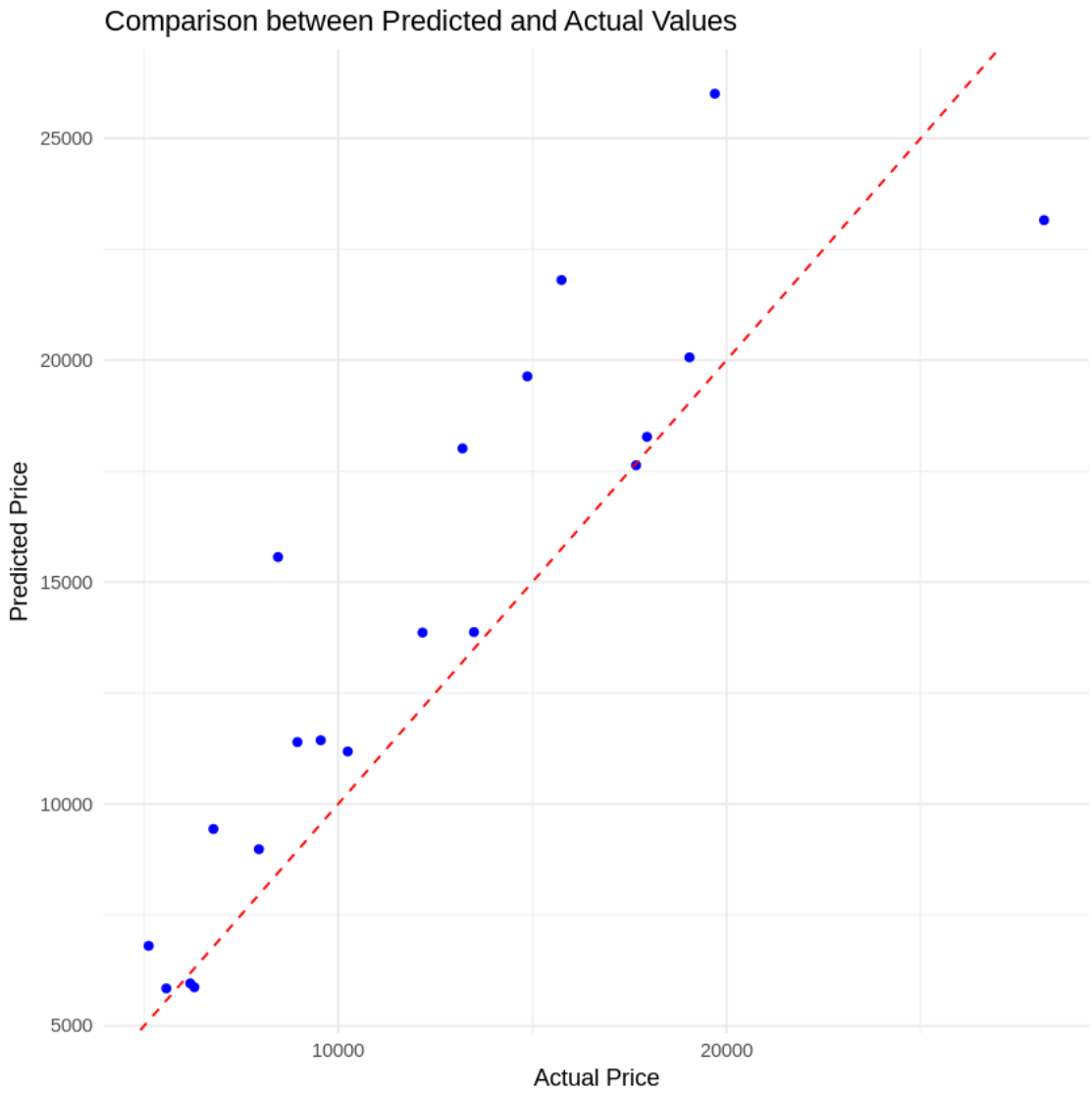
Let's examine the distribution of the predicted values of the training data.

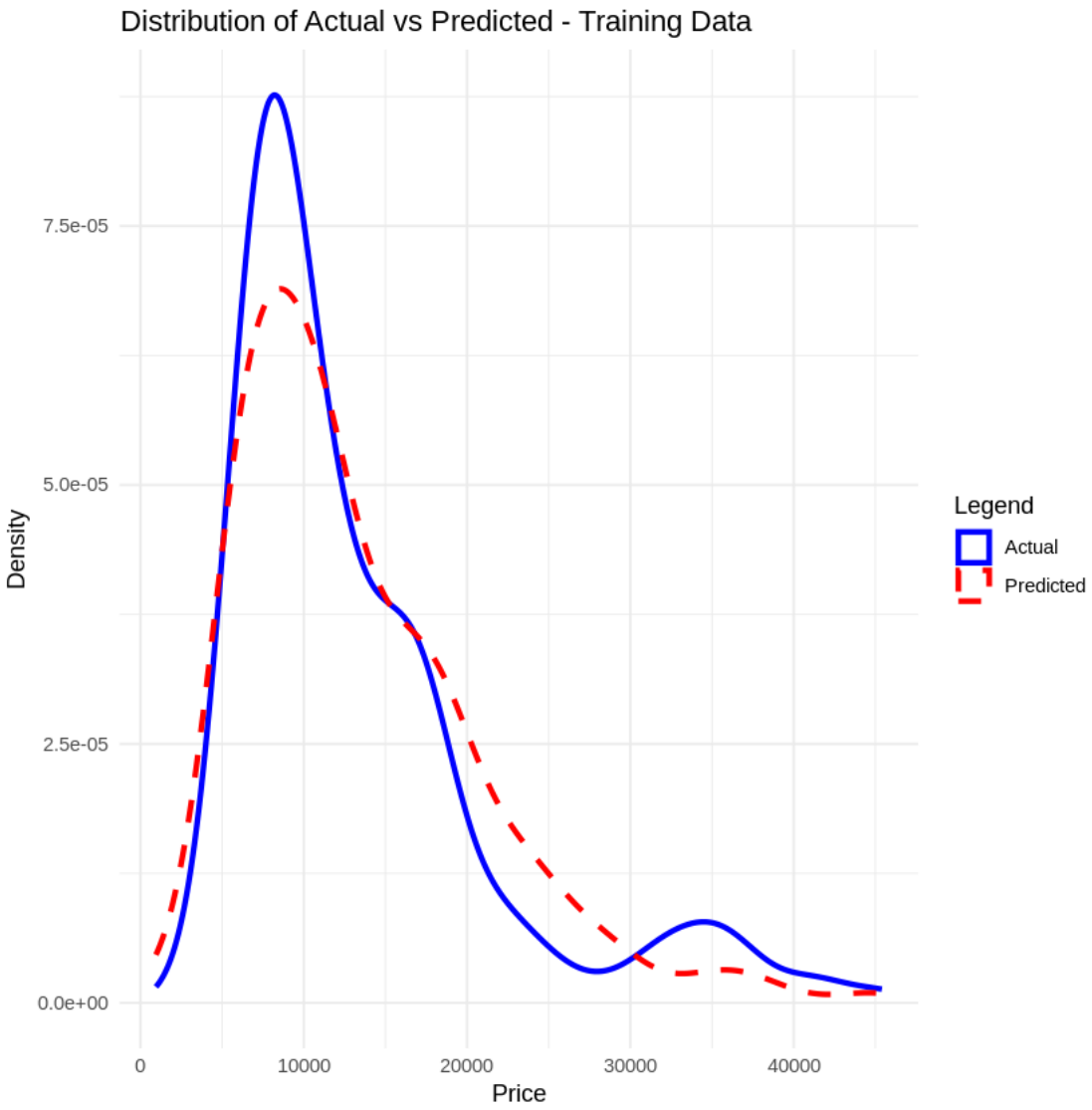
[ ]:
```
# Đánh giá mô hình trên dữ liệu training
train_rmse <- sqrt(mean((y_train - y_train_pred)^2))
train_r2 <- summary(lre_multi)$r.squared

# Đánh giá mô hình trên dữ liệu test
test_rmse <- sqrt(mean((y_test - y_test_pred)^2))
test_r2 <- 1 - sum((y_test - y_test_pred)^2) / sum((y_test - mean(y_test))^2)

# In các chỉ số đánh giá
cat("Training Data:\n")
cat("RMSE:", train_rmse, "\n")
cat("R-squared:", train_r2, "\n")

cat("\nTest Data:\n")
cat("RMSE:", test_rmse, "\n")
cat("R-squared:", test_r2, "\n")

# Vẽ đồ thị so sánh dự đoán và giá trị thực tế trên tập test
library(ggplot2)
```

```r
ggplot() +
  geom_point(aes(x = y_test, y = y_test_pred), color = 'blue') +
  geom_abline(slope = 1, intercept = 0, linetype = 'dashed', color = 'red') +
  labs(x = 'Actual Price', y = 'Predicted Price',
       title = 'Comparison between Predicted and Actual Values') +
  theme_minimal()

# Distribution plot function
distribution_plot <- function(actual, predicted, dataset_type) {
  ggplot() +
    geom_density(aes(x = actual, color = "Actual"), size = 1.2) +
    geom_density(aes(x = predicted, color = "Predicted"), size = 1.2, linetype␣
 ↪= "dashed") +
    labs(title = paste("Distribution of Actual vs Predicted -", dataset_type),
         x = "Price", y = "Density") +
    scale_color_manual(name = "Legend", values = c("Actual" = "blue",␣
 ↪"Predicted" = "red")) +
    theme_minimal()
}

# Vẽ DistributionPlot cho training data
distribution_plot(y_train, y_train_pred, "Training Data")

# Vẽ DistributionPlot cho test data
distribution_plot(y_test, y_test_pred, "Test Data")
```

```
Training Data:
RMSE: 3483.854
R-squared: 0.815763

Test Data:
RMSE: 3345.171
R-squared: 0.6761707
```

```
Warning message:
"Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
 Please use `linewidth` instead."
```


Comparison between Predicted and Actual Values
Predicted Price
Actual Price
25000
20000
15000
10000
5000
10000
20000


Distribution of Actual vs Predicted - Training Data
Density
7.5e-05
5.0e-05
2.5e-05
0.0e+00
0
10000
20000
30000
40000
Price
Legend
Actual
Predicted

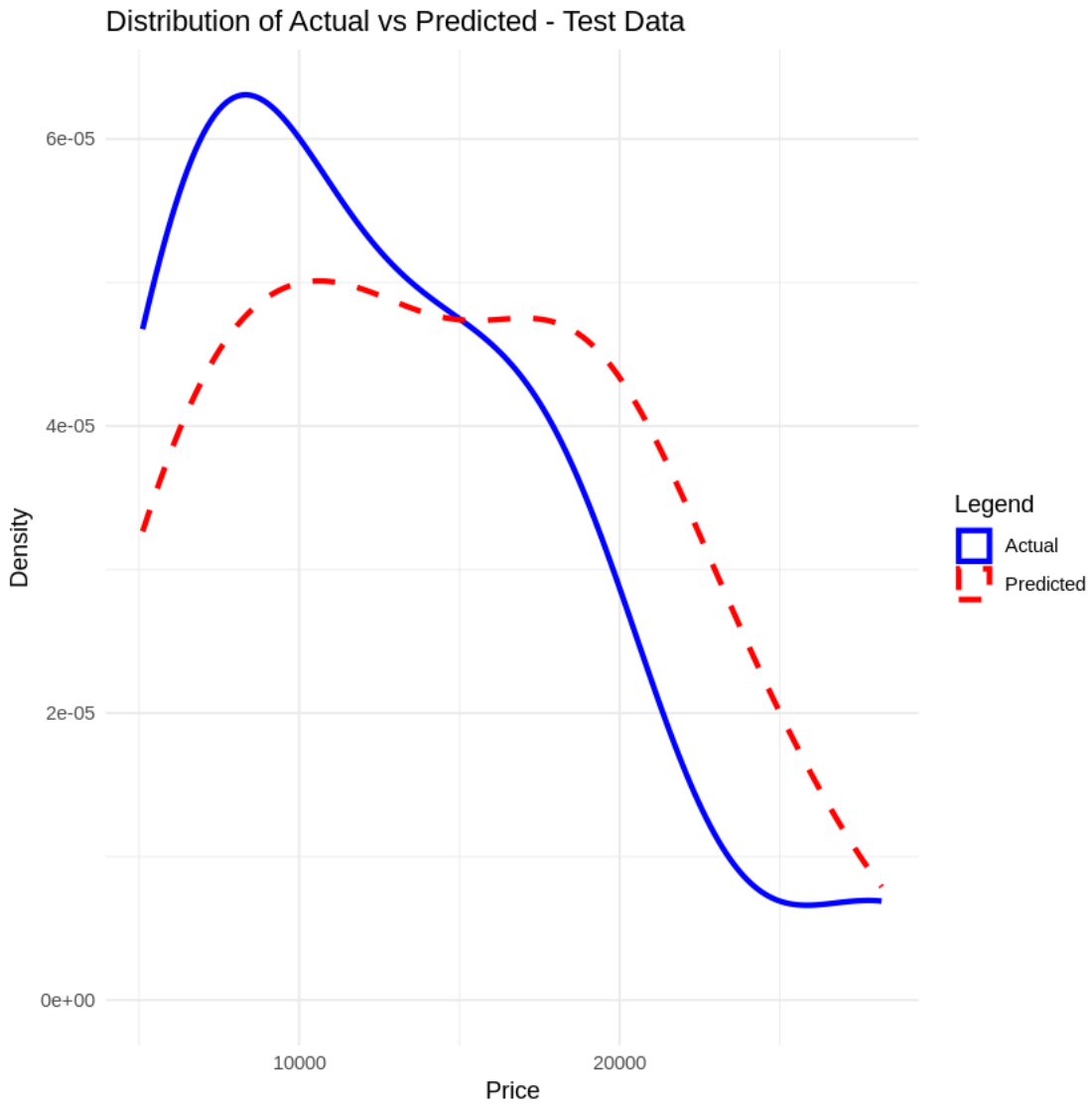


Figure 1: Plot of predicted values using the training data compared to the actual values of the training data.

So far, the model seems to be doing well in learning from the training dataset. But what happens when the model encounters new data from the testing dataset? When the model generates new values from the test data, we see the distribution of the predicted values is much different from the actual target values.

Figure 2: Plot of predicted value using the test data compared to the actual values of the test data.

Comparing Figure 1 and Figure 2, it is evident that the distribution of the test data in Figure 1 is much better at fitting the data. This difference in Figure 2 is apparent in the range of 5000 to 15,000. This is where the shape of the distribution is extremely different. Let's see if polynomial regression also exhibits a drop in the prediction accuracy when analysing the test dataset.

So sánh Hình 1 và Hình 2, rõ ràng là phân phối dữ liệu thử nghiệm trong Hình 1 phù hợp với dữ liệu tốt hơn nhiều. Sự khác biệt này trong Hình 2 thể hiện rõ trong phạm vi từ 5000 đến 15.000. Đây là nơi hình dạng của phân phối cực kỳ khác biệt. Chúng ta hãy xem liệu hồi quy đa thức có biểu hiện giảm độ chính xác dự đoán khi phân tích tập dữ liệu thử nghiệm hay không.

Overfitting

Overfitting occurs when the model fits the noise, but not the underlying process. Therefore, when testing your model using the test set, your model does not perform as well since it is modelling noise, not the underlying process that generated the relationship. Let's create a degree 5 polynomial model.

Let's use 55 percent of the data for training and the test for testing:

[ ]:
```
install.packages("polycor")
```

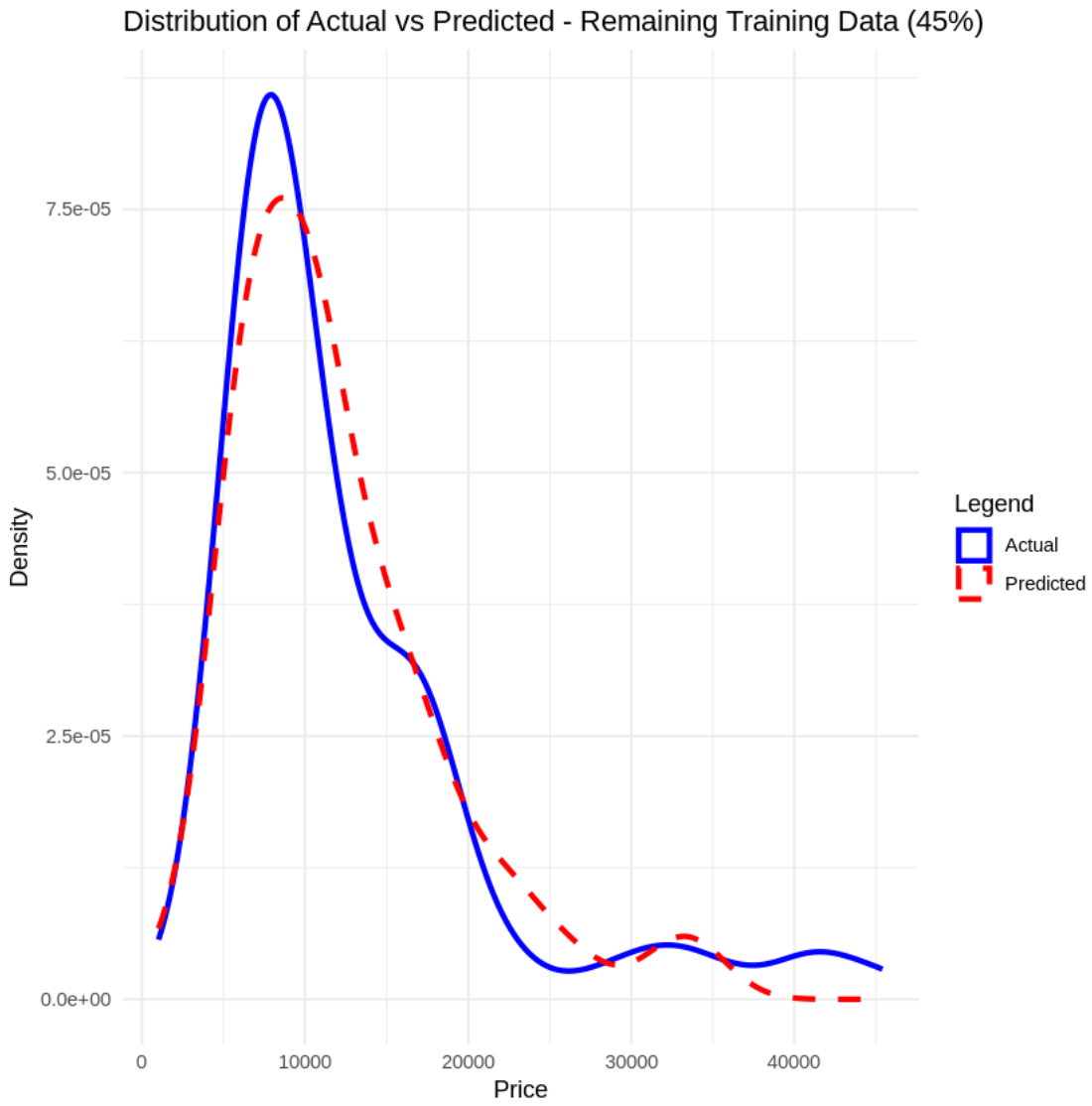
```
Installing package into '/usr/local/lib/R/site-library'
(as 'lib' is unspecified)

also installing the dependencies 'mvtnorm', 'admisc'
```

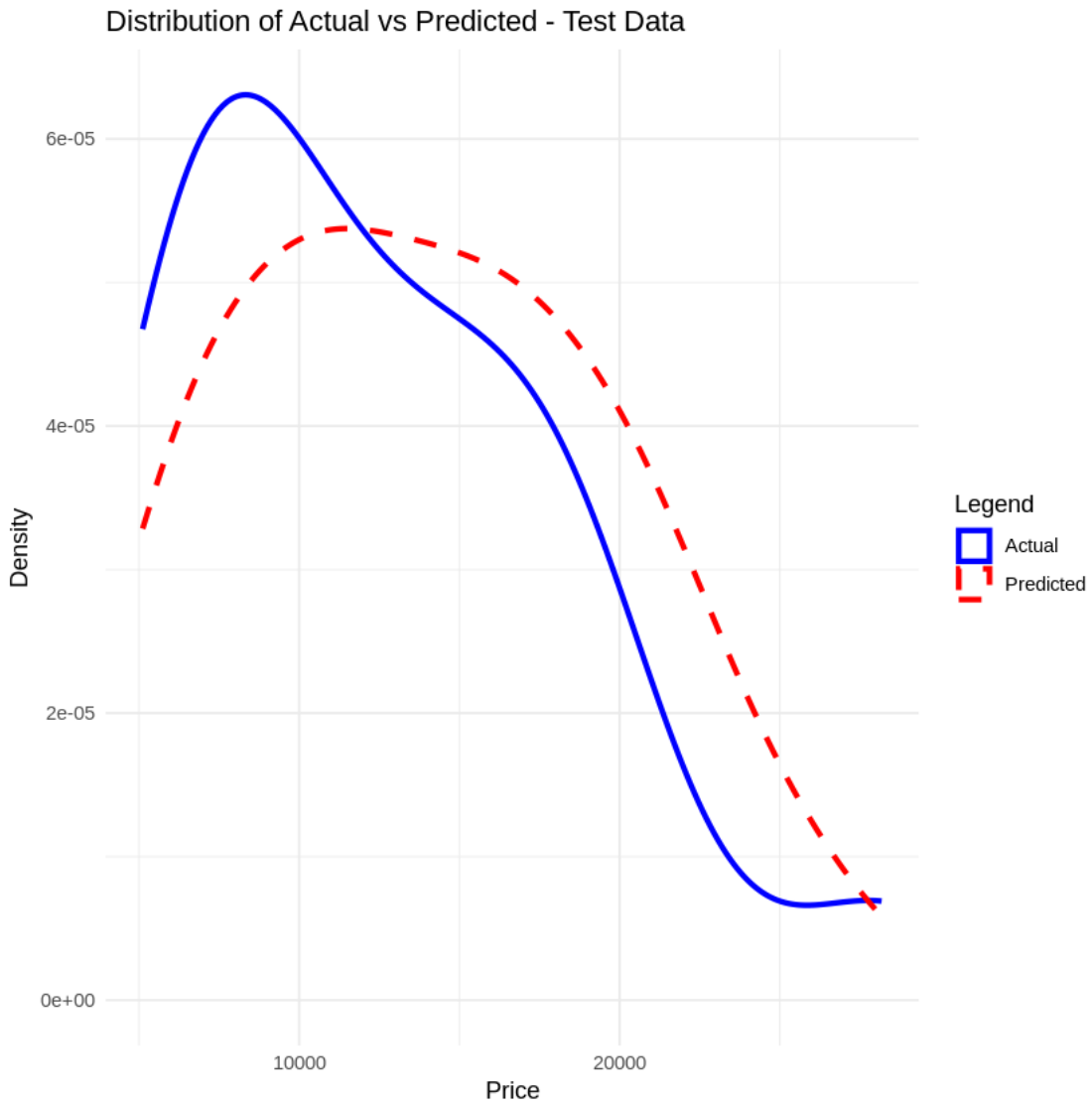
[ ]:
```
# Chia 55% dữ liệu từ training set ban đầu
set.seed(123)  # Đặt seed để tái lập kết quả
train_index_55 <- sample(1:nrow(x_train), size = 0.55 * nrow(x_train))

x_train_55 <- x_train[train_index_55, ]
y_train_55 <- y_train[train_index_55]

# Dữ liệu 45% còn lại để kiểm tra
x_train_remain <- x_train[-train_index_55, ]
y_train_remain <- y_train[-train_index_55]

# Tạo mô hình hồi quy tuyến tính với 55% dữ liệu training
lre_multi_55 <- lm(price ~ horsepower + curb.weight + engine.size + highway.
 ↪mpg, data = data.frame(x_train_55, price = y_train_55))

# Hiển thị tóm tắt mô hình
summary(lre_multi_55)

# Dự đoán với 45% dữ liệu còn lại của training
y_train_remain_pred <- predict(lre_multi_55, newdata = x_train_remain)

# Dự đoán với dữ liệu test ban đầu
y_test_pred_55 <- predict(lre_multi_55, newdata = x_test)

# Đánh giá mô hình trên 45% dữ liệu còn lại của training
train_remain_rmse <- sqrt(mean((y_train_remain - y_train_remain_pred)^2))
```

```r
train_remain_r2 <- 1 - sum((y_train_remain - y_train_remain_pred)^2) /␣
 ↪sum((y_train_remain - mean(y_train_remain))^2)

# Đánh giá mô hình trên dữ liệu test
test_rmse_55 <- sqrt(mean((y_test - y_test_pred_55)^2))
test_r2_55 <- 1 - sum((y_test - y_test_pred_55)^2) / sum((y_test -␣
 ↪mean(y_test))^2)

# In các chỉ số đánh giá
cat("Remaining Training Data (45%):\n")
cat("RMSE:", train_remain_rmse, "\n")
cat("R-squared:", train_remain_r2, "\n")

cat("\nTest Data:\n")
cat("RMSE:", test_rmse_55, "\n")
cat("R-squared:", test_r2_55, "\n")

# Vẽ DistributionPlot cho remaining training data (45%)
distribution_plot(y_train_remain, y_train_remain_pred, "Remaining Training Data␣
 ↪(45%)")

# Vẽ DistributionPlot cho test data
distribution_plot(y_test, y_test_pred_55, "Test Data")
```

```
Call:
lm(formula = price ~ horsepower + curb.weight + engine.size +
    highway.mpg, data = data.frame(x_train_55, price = y_train_55))

Residuals:
    Min      1Q  Median      3Q     Max
-8370.2 -1855.6   142.8  1462.6 11679.1

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -12550.540   7146.396  -1.756  0.08231 .
horsepower      73.915     23.387   3.161  0.00212 **
curb.weight      4.367      1.722   2.536  0.01287 *
engine.size     52.732     21.945   2.403  0.01823 *
highway.mpg      5.681    124.617   0.046  0.96374
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3690 on 94 degrees of freedom
Multiple R-squared:  0.782,        Adjusted R-squared:  0.7727
F-statistic:  84.3 on 4 and 94 DF,  p-value: < 2.2e-16
```

```
Remaining Training Data (45%):
RMSE: 3559.781
R-squared: 0.8265403

Test Data:
RMSE: 3242.91
R-squared: 0.6956667
```





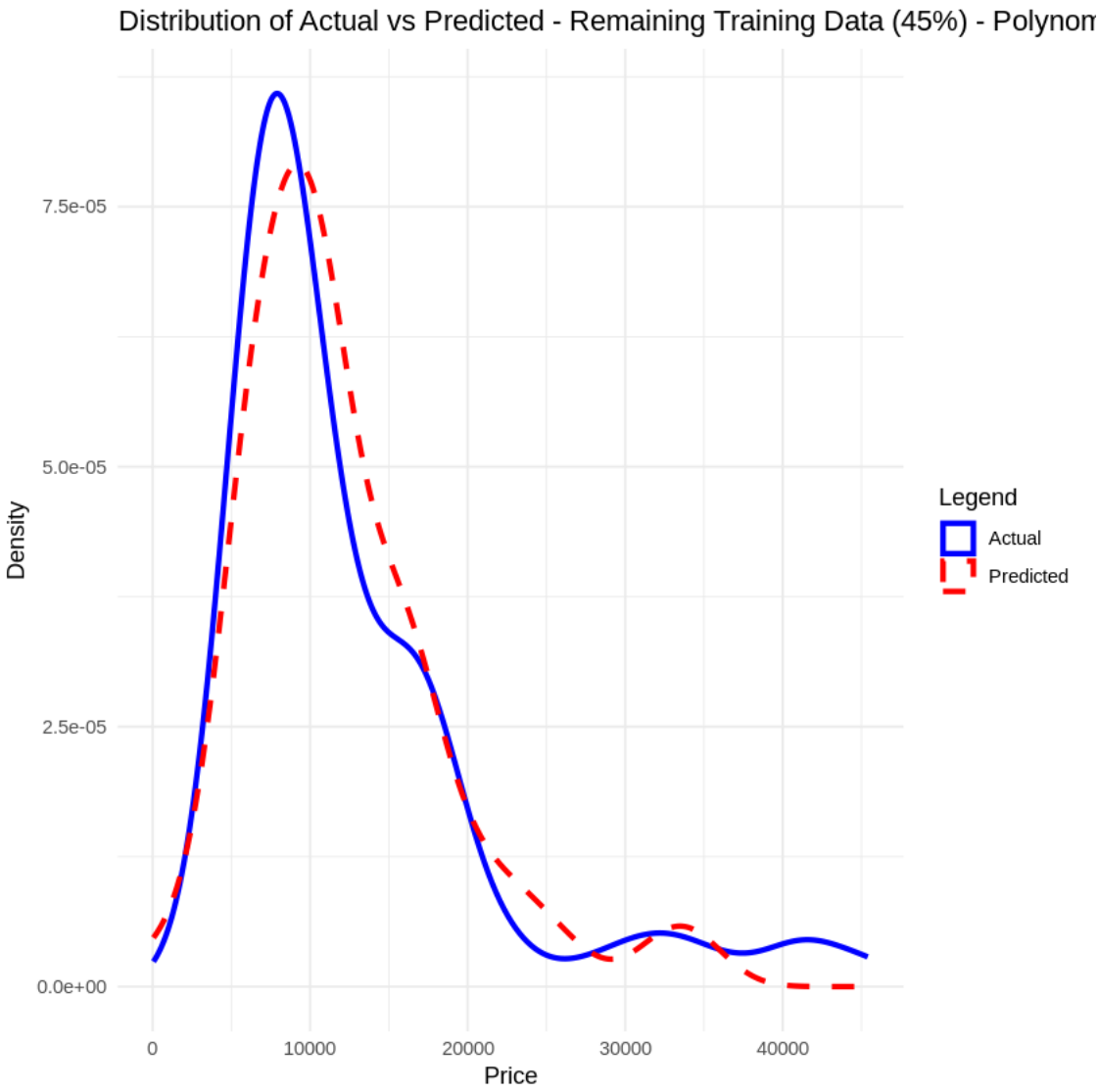
We will perform a degree 3 polynomial transformation on the feature 'horsepower'.

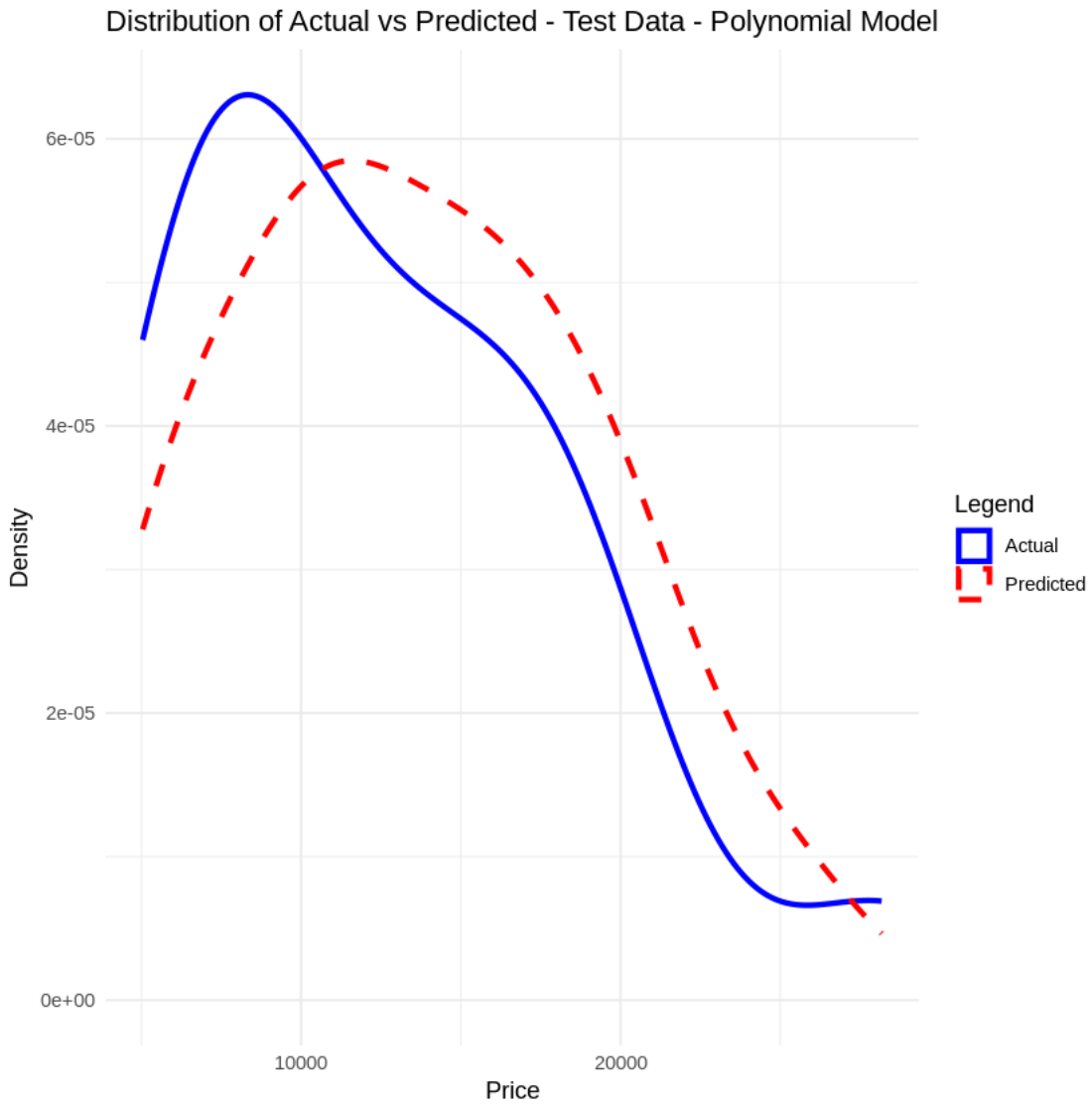
[ ]:
```r
# Thư viện
library(dplyr)
library(ggplot2)

# Tạo polynomial features bậc 3 cho horsepower
x_train_55_poly <- x_train_55 %>%
  mutate(horsepower_poly = horsepower^3)

x_train_remain_poly <- x_train_remain %>%
  mutate(horsepower_poly = horsepower^3)
```

```r
x_test_poly <- x_test %>%
  mutate(horsepower_poly = horsepower^3)

# Tạo mô hình hồi quy đa thức bậc 3
poly_model <- lm(price ~ horsepower_poly + curb.weight + engine.size + highway.
 ↪mpg,
                 data = data.frame(x_train_55_poly, price = y_train_55))

# Hiển thị tóm tắt mô hình
summary(poly_model)

# Dự đoán với dữ liệu còn lại (45%) của training
y_train_remain_poly_pred <- predict(poly_model, newdata = x_train_remain_poly)

# Dự đoán với dữ liệu test
y_test_poly_pred <- predict(poly_model, newdata = x_test_poly)

# Đánh giá mô hình trên 45% dữ liệu còn lại của training
train_remain_poly_rmse <- sqrt(mean((y_train_remain -␣
 ↪y_train_remain_poly_pred)^2))
train_remain_poly_r2 <- 1 - sum((y_train_remain - y_train_remain_poly_pred)^2) /
 ↪ sum((y_train_remain - mean(y_train_remain))^2)

# Đánh giá mô hình trên dữ liệu test
test_poly_rmse <- sqrt(mean((y_test - y_test_poly_pred)^2))
test_poly_r2 <- 1 - sum((y_test - y_test_poly_pred)^2) / sum((y_test -␣
 ↪mean(y_test))^2)

# In các chỉ số đánh giá
cat("Remaining Training Data (45%) with Polynomial Model:\n")
cat("RMSE:", train_remain_poly_rmse, "\n")
cat("R-squared:", train_remain_poly_r2, "\n")

cat("\nTest Data with Polynomial Model:\n")
cat("RMSE:", test_poly_rmse, "\n")
cat("R-squared:", test_poly_r2, "\n")

# Vẽ DistributionPlot cho remaining training data (45%)
distribution_plot(y_train_remain, y_train_remain_poly_pred, "Remaining Training␣
 ↪Data (45%) - Polynomial Model")

# Vẽ DistributionPlot cho test data
distribution_plot(y_test, y_test_poly_pred, "Test Data - Polynomial Model")
```

```
Attaching package: ‘dplyr’
```

```
The following objects are masked from 'package:stats':

    filter, lag


The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union



Call:
lm(formula = price ~ horsepower_poly + curb.weight + engine.size +
    highway.mpg, data = data.frame(x_train_55_poly, price = y_train_55))

Residuals:
     Min       1Q   Median       3Q      Max
-10150.1  -1947.7     59.1   1354.9  12156.7

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)     -2.060e+03  5.930e+03  -0.347  0.72903
horsepower_poly  8.882e-04  2.892e-04   3.071  0.00279 **
curb.weight      4.742e+00  1.766e+00   2.686  0.00856 **
engine.size      4.935e+01  2.302e+01   2.144  0.03464 *
highway.mpg     -1.519e+02  1.016e+02  -1.495  0.13819
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3700 on 94 degrees of freedom
Multiple R-squared:  0.7808,        Adjusted R-squared:  0.7715
F-statistic: 83.73 on 4 and 94 DF,  p-value: < 2.2e-16


Remaining Training Data (45%) with Polynomial Model:
RMSE: 3573.54
R-squared: 0.8251969

Test Data with Polynomial Model:
RMSE: 3064.006
R-squared: 0.7283193
```


Distribution of Actual vs Predicted - Remaining Training Data (45%) - Polynom
Density
7.5e-05
5.0e-05
2.5e-05
0.0e+00
0
10000
20000
30000
40000
Price
Legend
Actual
Predicted



We will perform a degree 5 polynomial transformation on the feature 'horsepower'.

Now, let's create a Linear Regression model "poly" and train it.

We can see the output of our model using the method "predict." We assign the values to "yhat".

Let's take the first five predicted values and compare it to the actual targets.

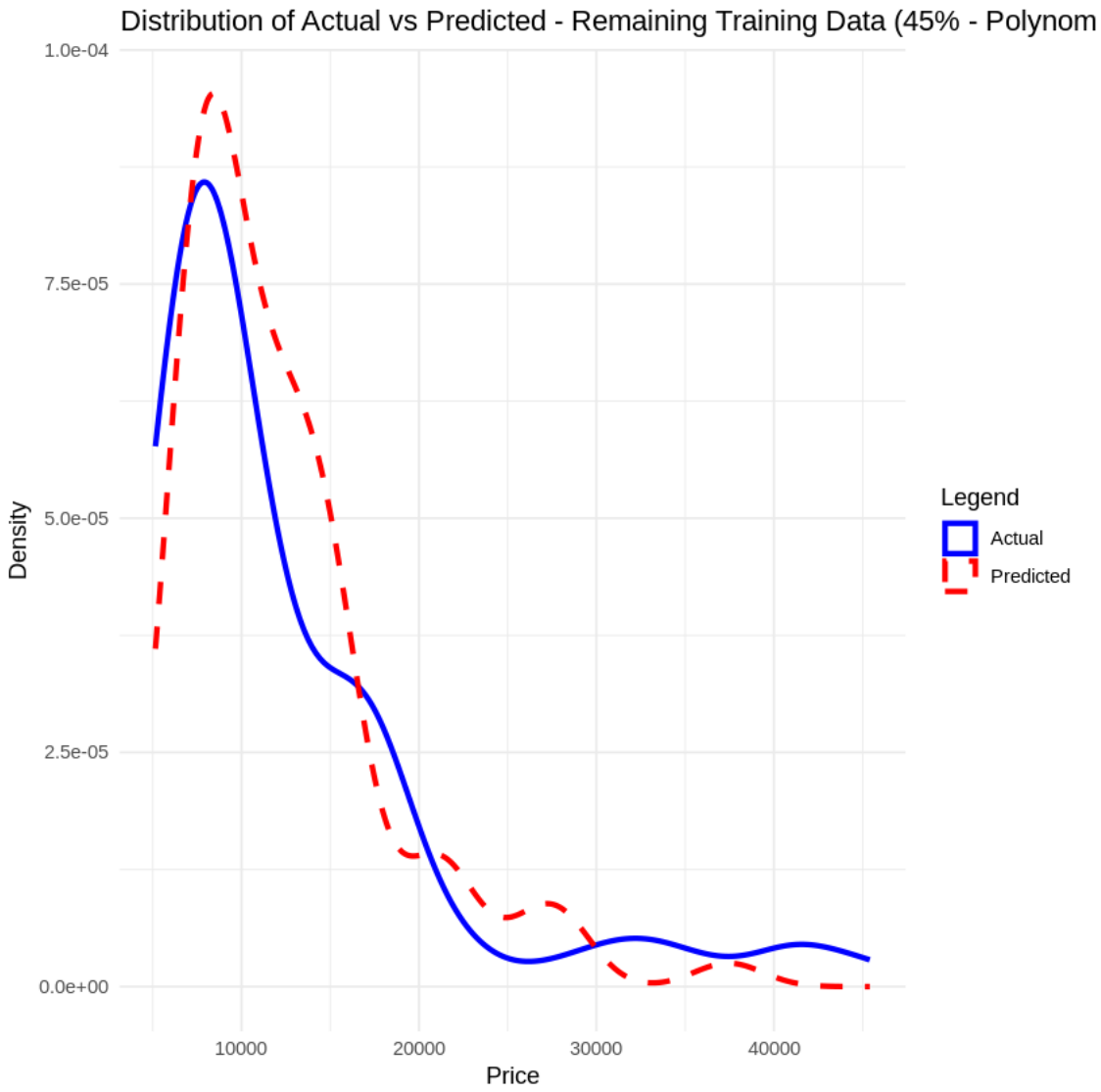
[ ]:
```
install.packages("polycor")
```

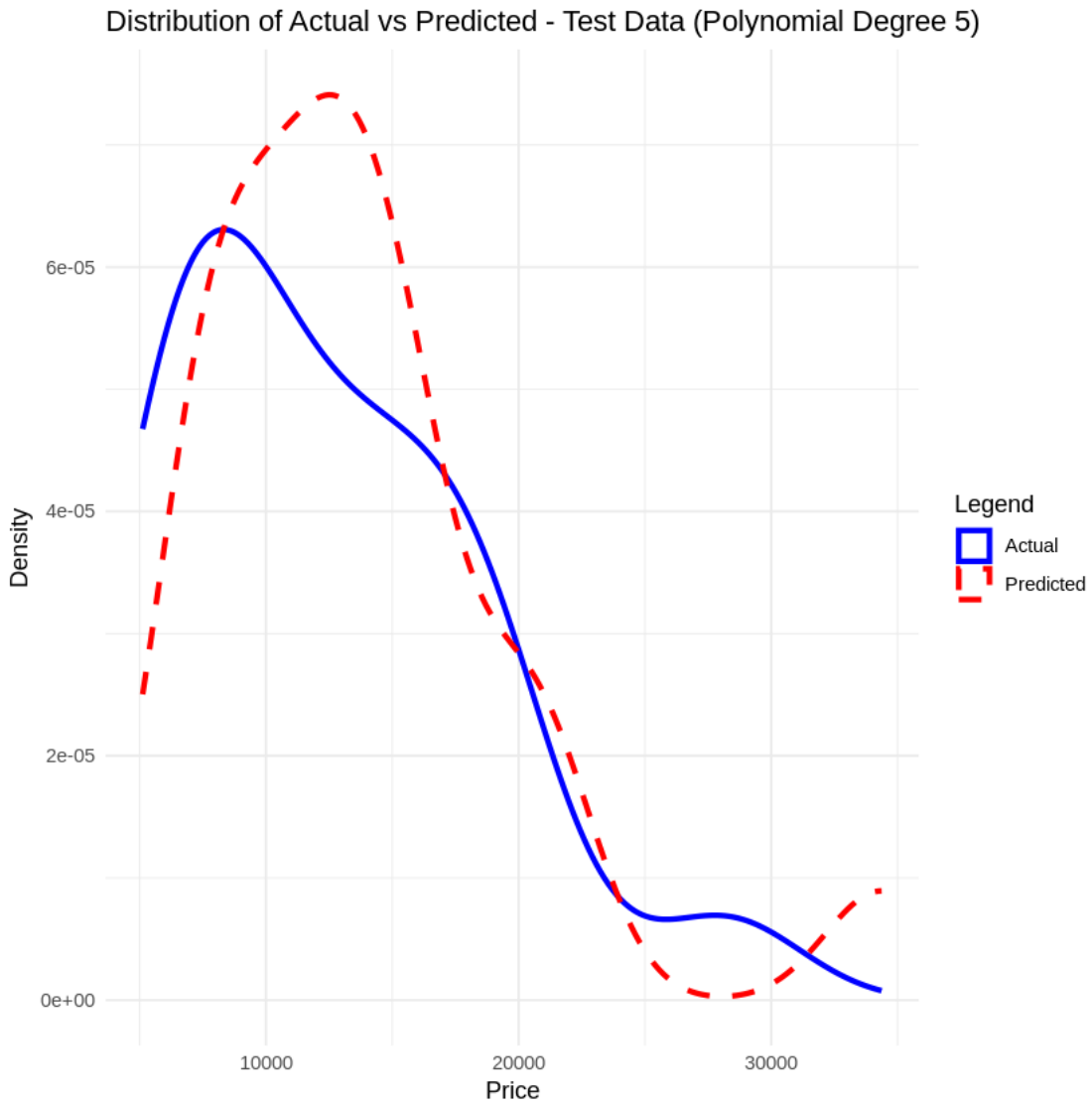
```
Installing package into '/usr/local/lib/R/site-library'
(as 'lib' is unspecified)

also installing the dependencies 'mvtnorm', 'admisc'
```

[ ]:
```r
# Tạo các biến đa thức bậc 5 cho horsepower
library(dplyr)

# Thêm các biến đa thức cho horsepower
x_train_55_poly <- x_train_55 %>%
  mutate(horsepower_poly = poly(horsepower, degree = 5, raw = TRUE))

x_train_remain_poly <- x_train_remain %>%
  mutate(horsepower_poly = poly(horsepower, degree = 5, raw = TRUE))

x_test_poly <- x_test %>%
  mutate(horsepower_poly = poly(horsepower, degree = 5, raw = TRUE))

# Tạo mô hình hồi quy tuyến tính đa thức bậc 5
lre_poly_5 <- lm(price ~ horsepower_poly, data = data.frame(x_train_55_poly,␣
 ↪price = y_train_55))

# Hiển thị tóm tắt mô hình
summary(lre_poly_5)

# Dự đoán với 45% dữ liệu còn lại của training
y_train_remain_poly_pred <- predict(lre_poly_5, newdata = x_train_remain_poly)

# Dự đoán với dữ liệu test
y_test_poly_pred <- predict(lre_poly_5, newdata = x_test_poly)

# Đánh giá mô hình trên 45% dữ liệu còn lại của training
train_remain_poly_rmse <- sqrt(mean((y_train_remain -␣
 ↪y_train_remain_poly_pred)^2))
train_remain_poly_r2 <- 1 - sum((y_train_remain - y_train_remain_poly_pred)^2) /
 ↪ sum((y_train_remain - mean(y_train_remain))^2)

# Đánh giá mô hình trên dữ liệu test
test_poly_rmse <- sqrt(mean((y_test - y_test_poly_pred)^2))
test_poly_r2 <- 1 - sum((y_test - y_test_poly_pred)^2) / sum((y_test -␣
 ↪mean(y_test))^2)

# In các chỉ số đánh giá
cat("Remaining Training Data (45% - Polynomial Degree 5):\n")
cat("RMSE:", train_remain_poly_rmse, "\n")
cat("R-squared:", train_remain_poly_r2, "\n")

cat("\nTest Data (Polynomial Degree 5):\n")
cat("RMSE:", test_poly_rmse, "\n")
cat("R-squared:", test_poly_r2, "\n")
```

```r
# Vẽ DistributionPlot cho remaining training data (45%) với mô hình đa thức
distribution_plot(y_train_remain, y_train_remain_poly_pred, "Remaining Training␣
 ↪Data (45% - Polynomial Degree 5)")

# Vẽ DistributionPlot cho test data với mô hình đa thức
distribution_plot(y_test, y_test_poly_pred, "Test Data (Polynomial Degree 5)")
```

```
Call:
lm(formula = price ~ horsepower_poly, data = data.frame(x_train_55_poly,
    price = y_train_55))

Residuals:
    Min      1Q  Median      3Q     Max
-8586.2 -2046.7  -383.6  1401.0 14575.8

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)       1.010e+05  5.716e+04   1.767   0.0806 .
horsepower_poly1 -4.348e+03  2.375e+03  -1.831   0.0703 .
horsepower_poly2  7.412e+01  3.733e+01   1.986   0.0500 .
horsepower_poly3 -5.793e-01  2.778e-01  -2.086   0.0397 *
horsepower_poly4  2.146e-03  9.799e-04   2.190   0.0310 *
horsepower_poly5 -3.003e-06  1.313e-06  -2.287   0.0245 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4180 on 93 degrees of freedom
Multiple R-squared:  0.7233,        Adjusted R-squared:  0.7084
F-statistic: 48.62 on 5 and 93 DF,  p-value: < 2.2e-16


Remaining Training Data (45% - Polynomial Degree 5):
RMSE: 4872.838
R-squared: 0.6749757

Test Data (Polynomial Degree 5):
RMSE: 5275.235
R-squared: 0.1946898
```


Distribution of Actual vs Predicted - Remaining Training Data (45% - Polynom
1.0e-04
7.5e-05
5.0e-05
2.5e-05
0.0e+00
Density
10000
20000
30000
40000
Price
Legend
Actual
Predicted

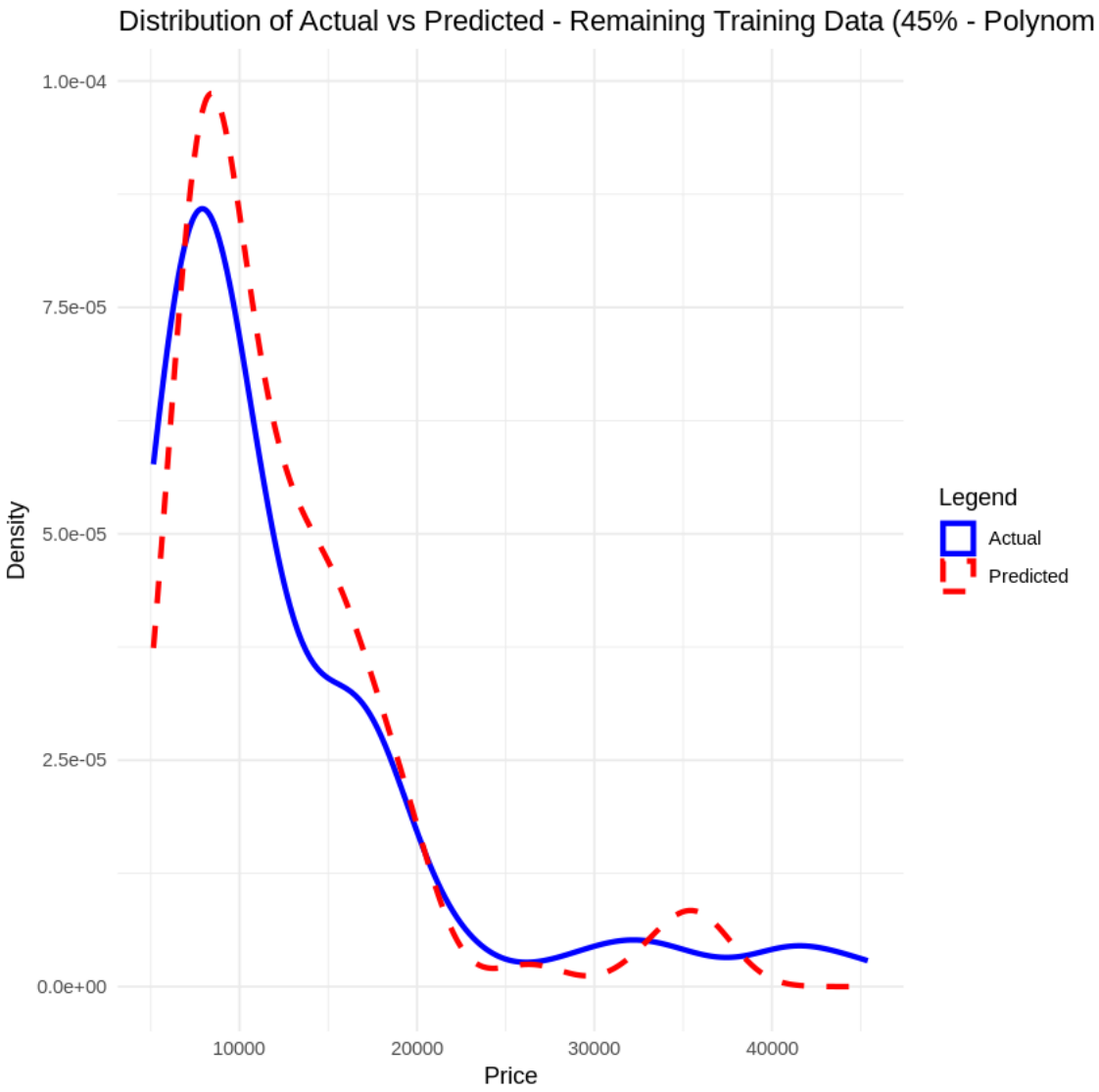


[ ]:
```r
# Tạo các biến đa thức bậc 11 cho horsepower
x_train_55_poly_11 <- x_train_55 %>%
  mutate(horsepower_poly_11 = poly(horsepower, degree = 11, raw = TRUE))

x_train_remain_poly_11 <- x_train_remain %>%
  mutate(horsepower_poly_11 = poly(horsepower, degree = 11, raw = TRUE))

x_test_poly_11 <- x_test %>%
  mutate(horsepower_poly_11 = poly(horsepower, degree = 11, raw = TRUE))

# Tạo mô hình hồi quy tuyến tính đa thức bậc 11
lre_poly_11 <- lm(price ~ horsepower_poly_11, data = data.
  ↪frame(x_train_55_poly_11, price = y_train_55))
```

```r
# Hiển thị tóm tắt mô hình
summary(lre_poly_11)

# Dự đoán với 45% dữ liệu còn lại của training
y_train_remain_poly_11_pred <- predict(lre_poly_11, newdata =␣
 ↪x_train_remain_poly_11)

# Dự đoán với dữ liệu test
y_test_poly_11_pred <- predict(lre_poly_11, newdata = x_test_poly_11)

# Đánh giá mô hình trên 45% dữ liệu còn lại của training
train_remain_poly_11_rmse <- sqrt(mean((y_train_remain -␣
 ↪y_train_remain_poly_11_pred)^2))
train_remain_poly_11_r2 <- 1 - sum((y_train_remain -␣
 ↪y_train_remain_poly_11_pred)^2) / sum((y_train_remain -␣
 ↪mean(y_train_remain))^2)

# Đánh giá mô hình trên dữ liệu test
test_poly_11_rmse <- sqrt(mean((y_test - y_test_poly_11_pred)^2))
test_poly_11_r2 <- 1 - sum((y_test - y_test_poly_11_pred)^2) / sum((y_test -␣
 ↪mean(y_test))^2)

# In các chỉ số đánh giá
cat("Remaining Training Data (45% - Polynomial Degree 11):\n")
cat("RMSE:", train_remain_poly_11_rmse, "\n")
cat("R-squared:", train_remain_poly_11_r2, "\n")

cat("\nTest Data (Polynomial Degree 11):\n")
cat("RMSE:", test_poly_11_rmse, "\n")
cat("R-squared:", test_poly_11_r2, "\n")

# Vẽ DistributionPlot cho remaining training data (45%) với mô hình đa thức bậc␣
 ↪11
distribution_plot(y_train_remain, y_train_remain_poly_11_pred, "Remaining␣
 ↪Training Data (45% - Polynomial Degree 11)")

# Vẽ DistributionPlot cho test data với mô hình đa thức bậc 11
distribution_plot(y_test, y_test_poly_11_pred, "Test Data (Polynomial Degree␣
 ↪11)")
```

```
Call:
lm(formula = price ~ horsepower_poly_11, data = data.frame(x_train_55_poly_11,
    price = y_train_55))

Residuals:
```

```
    Min      1Q  Median      3Q     Max
-8947.1 -1637.1  -332.7  1230.0 16067.8

Coefficients: (1 not defined because of singularities)
                       Estimate Std. Error t value Pr(>|t|)
(Intercept)           7.087e+06  2.012e+07   0.352    0.725
horsepower_poly_111  -6.701e+05  1.854e+06  -0.361    0.719
horsepower_poly_112   2.775e+04  7.462e+04   0.372    0.711
horsepower_poly_113  -6.612e+02  1.727e+03  -0.383    0.703
horsepower_poly_114   1.003e+01  2.541e+01   0.395    0.694
horsepower_poly_115  -1.009e-01  2.479e-01  -0.407    0.685
horsepower_poly_116   6.788e-04  1.616e-03   0.420    0.676
horsepower_poly_117  -2.984e-06  6.881e-06  -0.434    0.666
horsepower_poly_118   7.964e-09  1.777e-08   0.448    0.655
horsepower_poly_119  -1.037e-11  2.240e-11  -0.463    0.644
horsepower_poly_1110         NA         NA      NA       NA
horsepower_poly_1111  1.068e-17  2.164e-17   0.493    0.623

Residual standard error: 4129 on 88 degrees of freedom
Multiple R-squared:  0.7445,        Adjusted R-squared:  0.7155
F-statistic: 25.64 on 10 and 88 DF,  p-value: < 2.2e-16


Remaining Training Data (45% - Polynomial Degree 11):
RMSE: 4059.367
R-squared: 0.7744366

Test Data (Polynomial Degree 11):
RMSE: 8271.134
R-squared: -0.9797472
```


Distribution of Actual vs Predicted - Remaining Training Data (45% - Polynom
1.0e-04
7.5e-05
5.0e-05
2.5e-05
0.0e+00
Density
10000
20000
30000
40000
Price
Legend
Actual
Predicted

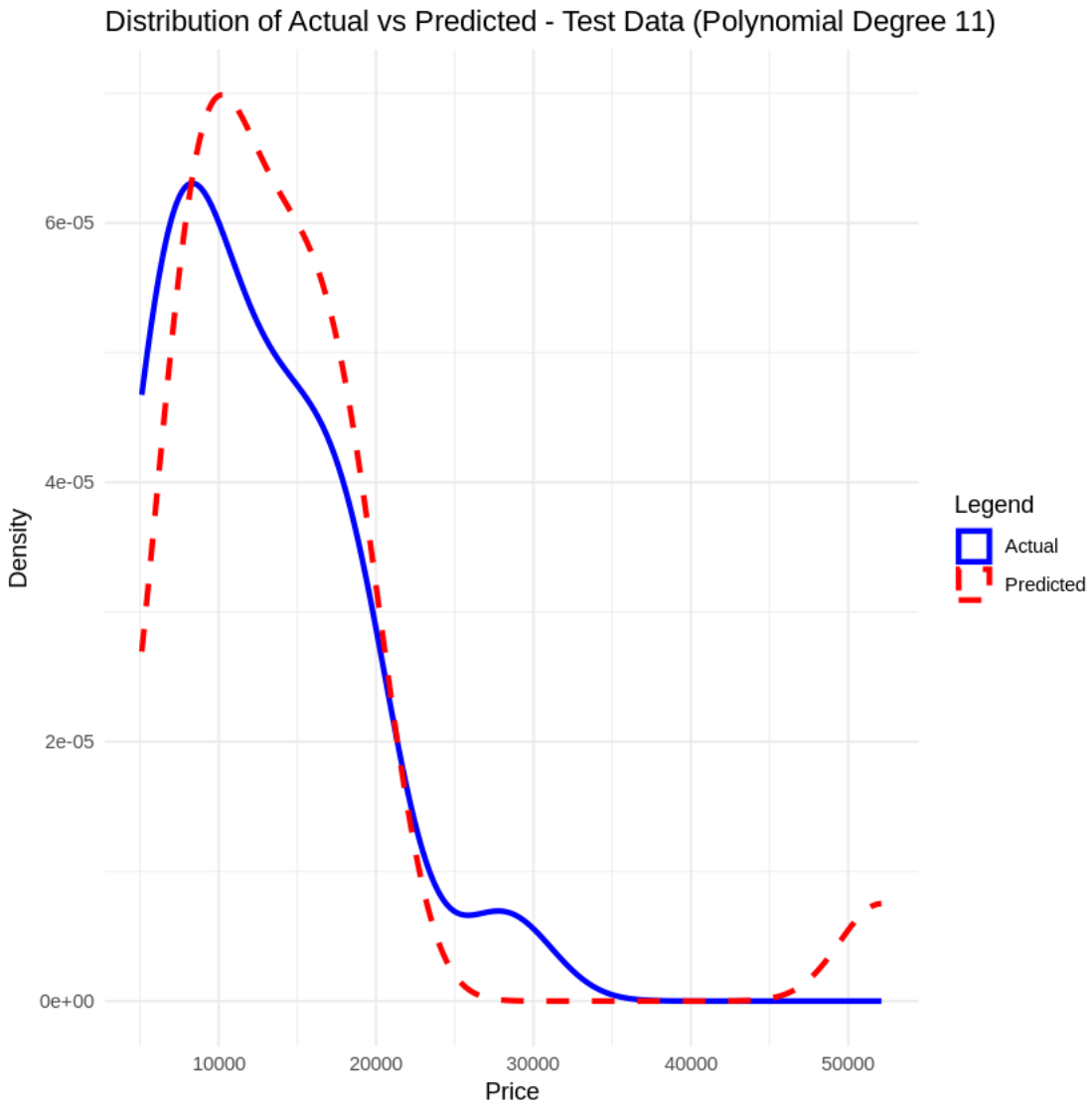


Figure 3: A polynomial regression model where red dots represent training data, green dots represent test data, and the blue line represents the model prediction.

We see that the estimated function appears to track the data but around 200 horsepower, the function begins to diverge from the data points.

R^2 of the training data: R-squared: 0.7744366

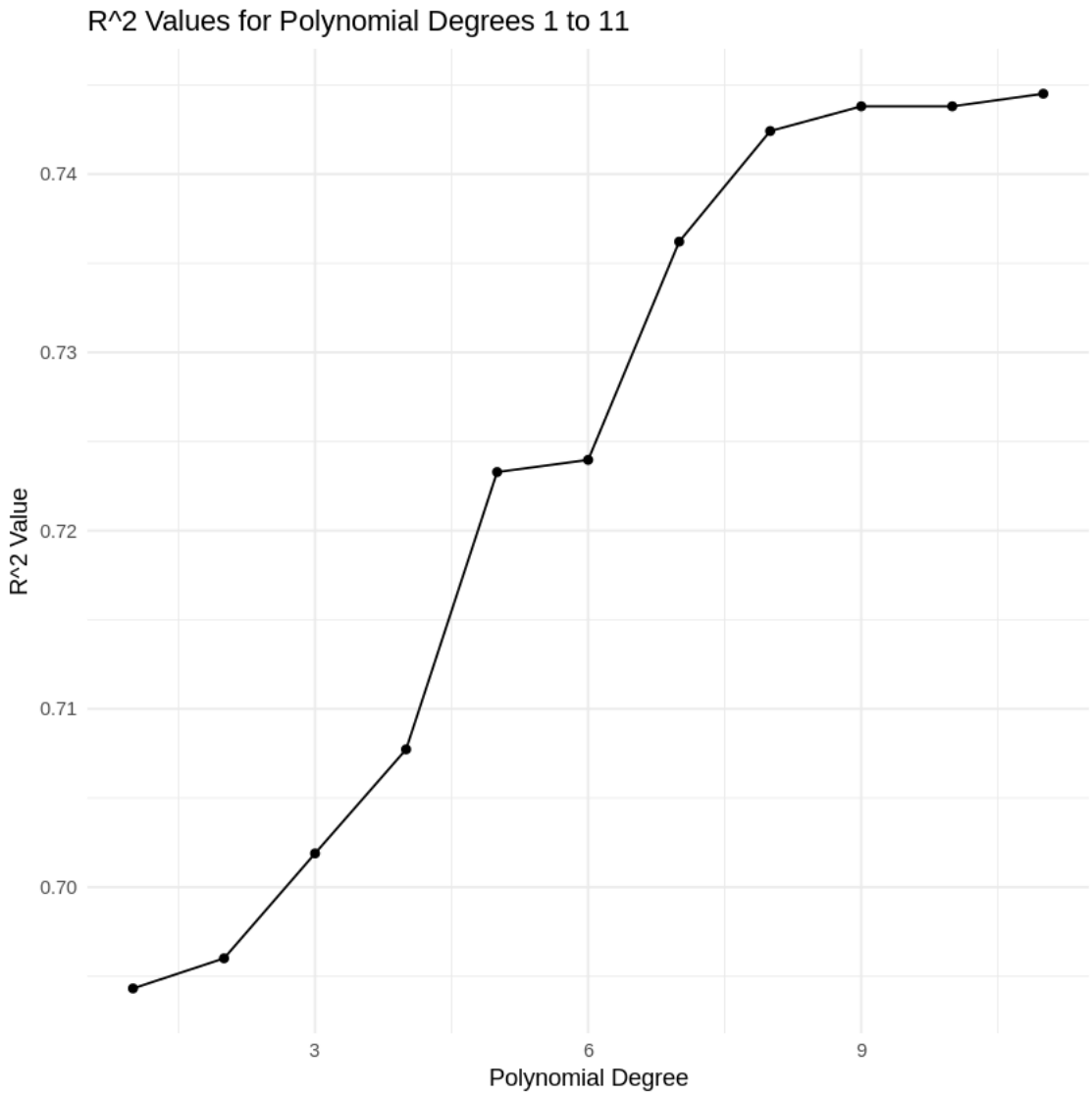
R^2 of the test data: R-squared: -0.9797472

We see the R^2 for the training data is 0.7744366 while the R^2 on the test data was -0.9797472 . The lower the R^2, the worse the model. A negative R^2 is a sign of overfitting.

Let's see how the R^2 changes on the test data for different order polynomials and then plot the results:

[ ]:
```r
# Khởi tạo vector để lưu R^2
r_squared_values <- numeric(11)

# Tạo mô hình hồi quy đa thức cho mỗi bậc từ 1 đến 11
for (degree in 1:11) {
  # Tạo các biến đa thức cho horsepower
  x_train_55_poly <- x_train_55 %>%
    mutate(horsepower_poly = poly(horsepower, degree = degree, raw = TRUE))

  # Tạo mô hình hồi quy tuyến tính
  model <- lm(price ~ horsepower_poly, data = data.frame(x_train_55_poly, price␣
 ↪= y_train_55))

  # Tính R^2
  r_squared_values[degree] <- summary(model)$r.squared
}

# In các giá trị R^2
for (degree in 1:11) {
  cat("Degree:", degree, "R^2:", r_squared_values[degree], "\n")
}

# Vẽ biểu đồ R^2 theo các bậc đa thức
library(ggplot2)

r_squared_df <- data.frame(Degree = 1:11, R_squared = r_squared_values)

ggplot(r_squared_df, aes(x = Degree, y = R_squared)) +
  geom_line() +
  geom_point() +
  labs(title = "R^2 Values for Polynomial Degrees 1 to 11",
       x = "Polynomial Degree",
       y = "R^2 Value") +
  theme_minimal()
```

```
Degree: 1 R^2: 0.6943162
Degree: 2 R^2: 0.6960052
Degree: 3 R^2: 0.701891
Degree: 4 R^2: 0.707729
Degree: 5 R^2: 0.7232873
Degree: 6 R^2: 0.7239691
Degree: 7 R^2: 0.7362115
Degree: 8 R^2: 0.7424176
Degree: 9 R^2: 0.7438045
Degree: 10 R^2: 0.7438045
Degree: 11 R^2: 0.7445109
```



We see the R^2 gradually increases until an order three polynomial is used. Then, the R^2 dramatically decreases at an order four polynomial.

11.4 Đánh Giá R^2

1. **Bậc 1 ($R^2 = 0.6943$)**:
   - Mô hình hồi quy tuyến tính đơn giản với biến `horsepower` có thể giải thích khoảng 69.43% phương sai của biến mục tiêu `price`. Đây là một R^2 tốt cho mô hình đơn giản, nhưng có thể cải thiện.
2. **Bậc 2 ($R^2 = 0.6960$)**:
   - Việc thêm một bậc đa thức đã cải thiện nhẹ độ chính xác, với (R^2) tăng lên 69.60%. Sự cải thiện là nhỏ nhưng cho thấy mô hình đang bắt đầu nắm bắt thêm một số biến thể trong dữ liệu.
3. **Bậc 3 ($R^2 = 0.7019$)**:

- Tăng thêm một bậc nữa dẫn đến việc giải thích khoảng 70.19% phương sai, cho thấy mô hình có thể bắt đầu nắm bắt được các mối quan hệ phi tuyến tính giữa `horsepower` và `price`.

4. **Bậc 4 ($R^2 = 0.7077$)**:
   - Với (R^2) tăng lên 70.77%, mô hình tiếp tục cải thiện độ chính xác. Điều này cho thấy rằng mối quan hệ giữa `horsepower` và `price` có thể phức tạp hơn và một mô hình phi tuyến tính hơn là cần thiết.
5. **Bậc 5 ($R^2 = 0.7233$)**:
   - Mô hình với bậc 5 đã giải thích khoảng 72.33% phương sai, cho thấy một sự cải thiện đáng kể. Điều này cho thấy mô hình đang bắt đầu nắm bắt được các xu hướng quan trọng trong dữ liệu.
6. **Bậc 6 ($R^2 = 0.7240$)**:
   - Chỉ số này rất gần với bậc 5, cho thấy rằng việc thêm bậc thứ 6 không tạo ra sự cải thiện đáng kể trong độ chính xác của mô hình.
7. **Bậc 7 ($R^2 = 0.7362$)**:
   - Với (R^2) là 73.62%, mô hình bắt đầu cho thấy sự cải thiện lại. Điều này có thể cho thấy rằng độ phức tạp của mô hình đang tạo ra lợi ích.
8. **Bậc 8 ($R^2 = 0.7424$)**:
   - Tăng nhẹ (R^2) lên 74.24%, mô hình bắt đầu giải thích một phần lớn hơn của phương sai trong `price`.
9. **Bậc 9 ($R^2 = 0.7438$)**:
   - R^2 chỉ tăng một chút (0.7438), cho thấy mô hình đang đạt đến giới hạn khả năng giải thích của nó.
10. **Bậc 10 ($R^2 = 0.7438$)**:
    - Không có sự cải thiện so với bậc 9. Điều này có thể cho thấy mô hình đã đạt đến một mức độ phức tạp tối ưu.
11. **Bậc 11 ($R^2 = 0.7445$)**:
    - Mặc dù có sự tăng nhẹ, nhưng mức tăng rất nhỏ. do đó khi thêm một bậc đa thức nữa không cải thiện đáng kể độ chính xác và có thể dẫn đến hiện tượng overfitting.

11.4.1 Kết Luận:

- **Tăng Trưởng Đầu Tiên**: Các bậc thấp (từ 1 đến 5) cho thấy sự cải thiện đáng kể về (R^2). Điều này cho thấy mô hình đang học hỏi từ dữ liệu và nắm bắt được các mối quan hệ quan trọng.
- **Khả Năng Tăng Trưởng Giảm Dần**: Sau bậc 5, mặc dù có sự tăng trưởng trong (R^2), nhưng mức độ cải thiện bắt đầu giảm dần và trở nên không đáng kể.
- **Overfitting**: Các mô hình với bậc cao (trên 8) có thể dẫn đến hiện tượng overfitting, vì mô hình học hỏi quá mức từ dữ liệu huấn luyện và không tổng quát tốt cho dữ liệu mới.

Recomment: Có thể cân nhắc sử dụng mô hình với bậc 5 hoặc 6, vì các mô hình này vẫn thể hiện độ chính xác cao mà không quá phức tạp. Sử dụng các phương pháp kiểm định như k-fold cross-validation để xác định độ chính xác của mô hình trên các tập dữ liệu khác nhau.

Part 3: Ridge Regression

In this section, we will review Ridge Regression and see how the parameter alpha changes the model. Just a note, here our test data will be used as validation data.

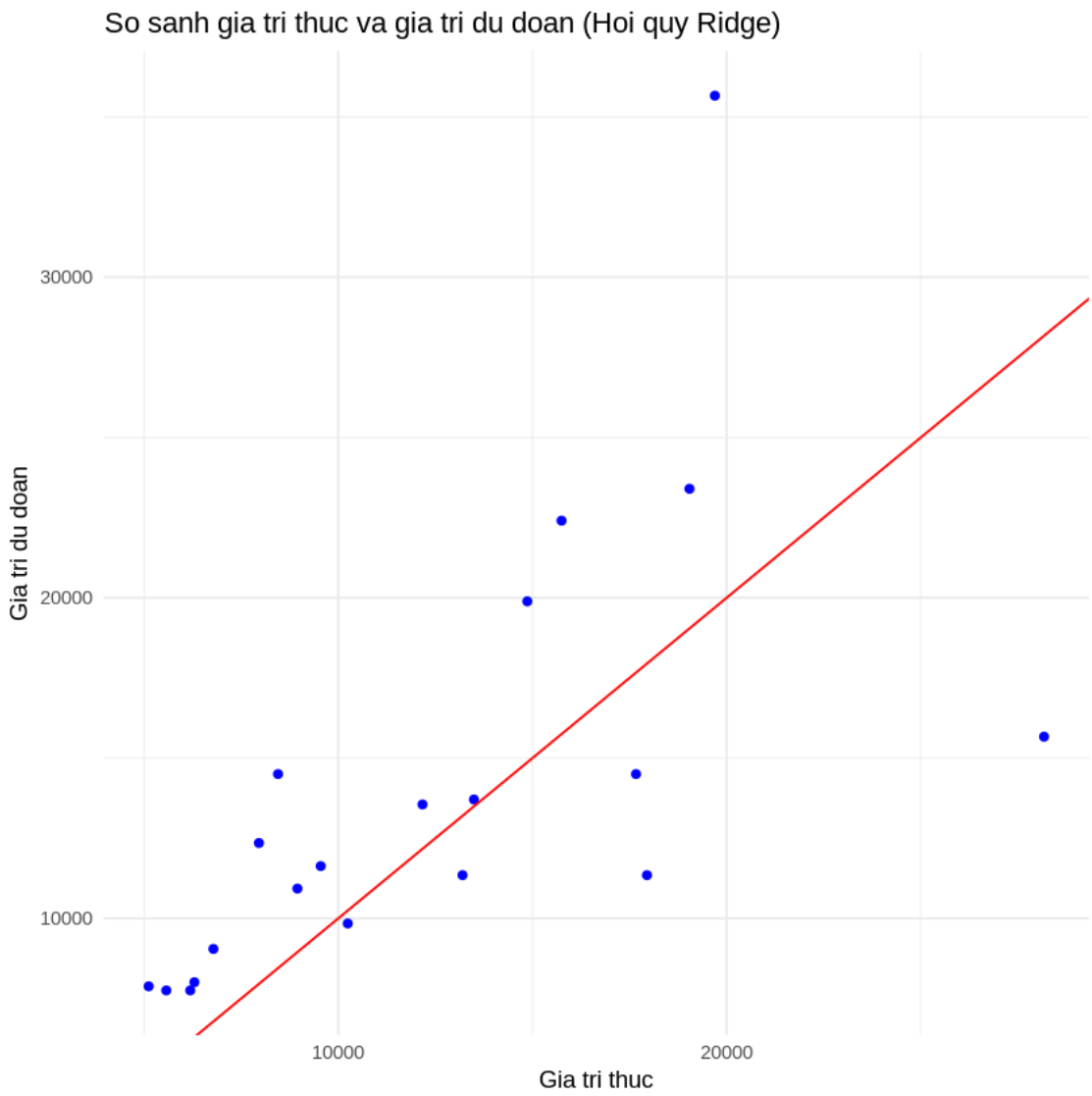
[ ]:
```r
install.packages("MASS")
```

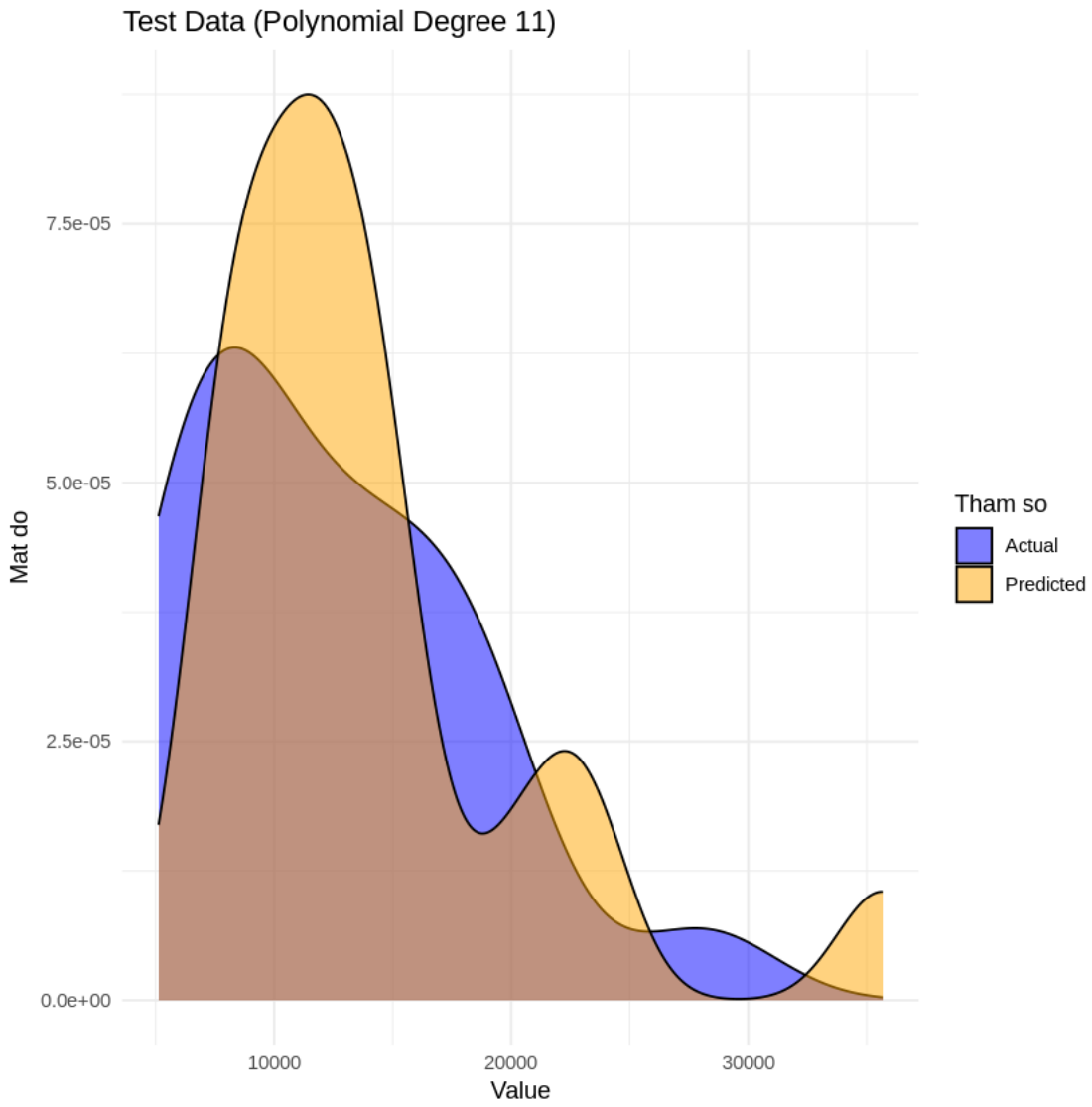
```
Installing package into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)
```

[ ]:
```r
# Thư viện
library(MASS)   # Thư viện cho hồi quy Ridge
library(ggplot2)

# Tạo biến đa thức bậc 11 cho biến horsepower
x_train_poly <- data.frame(horsepower = x_train$horsepower)
x_train_poly <- cbind(x_train_poly, poly(x_train$horsepower, degree = 11, raw =␣
 ↪TRUE))

# Đặt tên cho các biến đa thức
colnames(x_train_poly) <- c("horsepower", paste0("X", 1:11))

# Huấn luyện mô hình hồi quy Ridge với lambda = 0.5
lambda_value <- 0.5
ridge_model <- lm.ridge(price ~ ., data = data.frame(x_train_poly, price =␣
 ↪y_train), lambda = lambda_value)

# Tạo biến đa thức cho dữ liệu test
x_test_poly <- data.frame(horsepower = x_test$horsepower)
x_test_poly <- cbind(x_test_poly, poly(x_test$horsepower, degree = 11, raw =␣
 ↪TRUE))

# Đặt tên cho các biến đa thức trong dữ liệu test
colnames(x_test_poly) <- c("horsepower", paste0("X", 1:11))

# Chuyển đổi x_test_poly thành ma trận số
x_test_matrix <- as.matrix(x_test_poly)

# Dự đoán giá trị với mô hình hồi quy Ridge
y_pred_ridge <- as.vector(cbind(1, x_test_matrix) %*% coef(ridge_model))

# Tính toán R^2 cho dữ liệu test
rss <- sum((y_test - y_pred_ridge) ^ 2)
tss <- sum((y_test - mean(y_test)) ^ 2)
r_squared_ridge <- 1 - (rss / tss)

# In ra kết quả
cat("R^2 cho mo hinh hoi quy Ridge (bac 11) voi lambda =", lambda_value, ":",␣
 ↪r_squared_ridge, "\n")

# Vẽ biểu đồ so sánh giá trị thực và giá trị dự đoán
```

```r
comparison_df <- data.frame(Actual = y_test, Predicted = y_pred_ridge)

ggplot(comparison_df, aes(x = Actual, y = Predicted)) +
  geom_point(color = "blue") +
  geom_abline(slope = 1, intercept = 0, color = "red") +
  labs(title = "So sanh gia tri thuc va gia tri du doan (Hoi quy Ridge)",
       x = "Gia tri thuc",
       y = "Gia tri du doan") +
  theme_minimal()

# Hàm để vẽ Distribution Plot
distribution_plot <- function(actual, predicted, title) {
  data <- data.frame(Actual = actual, Predicted = predicted)

  ggplot(data) +
    geom_density(aes(x = Actual, fill = "Actual"), alpha = 0.5) +
    geom_density(aes(x = Predicted, fill = "Predicted"), alpha = 0.5) +
    labs(title = title, x = "Value", y = "Mat do") +
    scale_fill_manual(name = "Tham so", values = c("Actual" = "blue",␣
 ↪"Predicted" = "orange")) +
    theme_minimal()
}

# Vẽ DistributionPlot cho dữ liệu test
distribution_plot(y_test, y_pred_ridge, "Test Data (Polynomial Degree 11)")
```

```
Attaching package: ‘MASS’


The following object is masked from ‘package:dplyr’:

    select


R^2 cho mo hinh hoi quy Ridge (bac 11) voi lambda = 0.5 : 0.06512191
```


So sanh gia tri thuc va gia tri du doan (Hoi quy Ridge)
Gia tri du doan
30000
20000
10000
10000
20000
Gia tri thuc



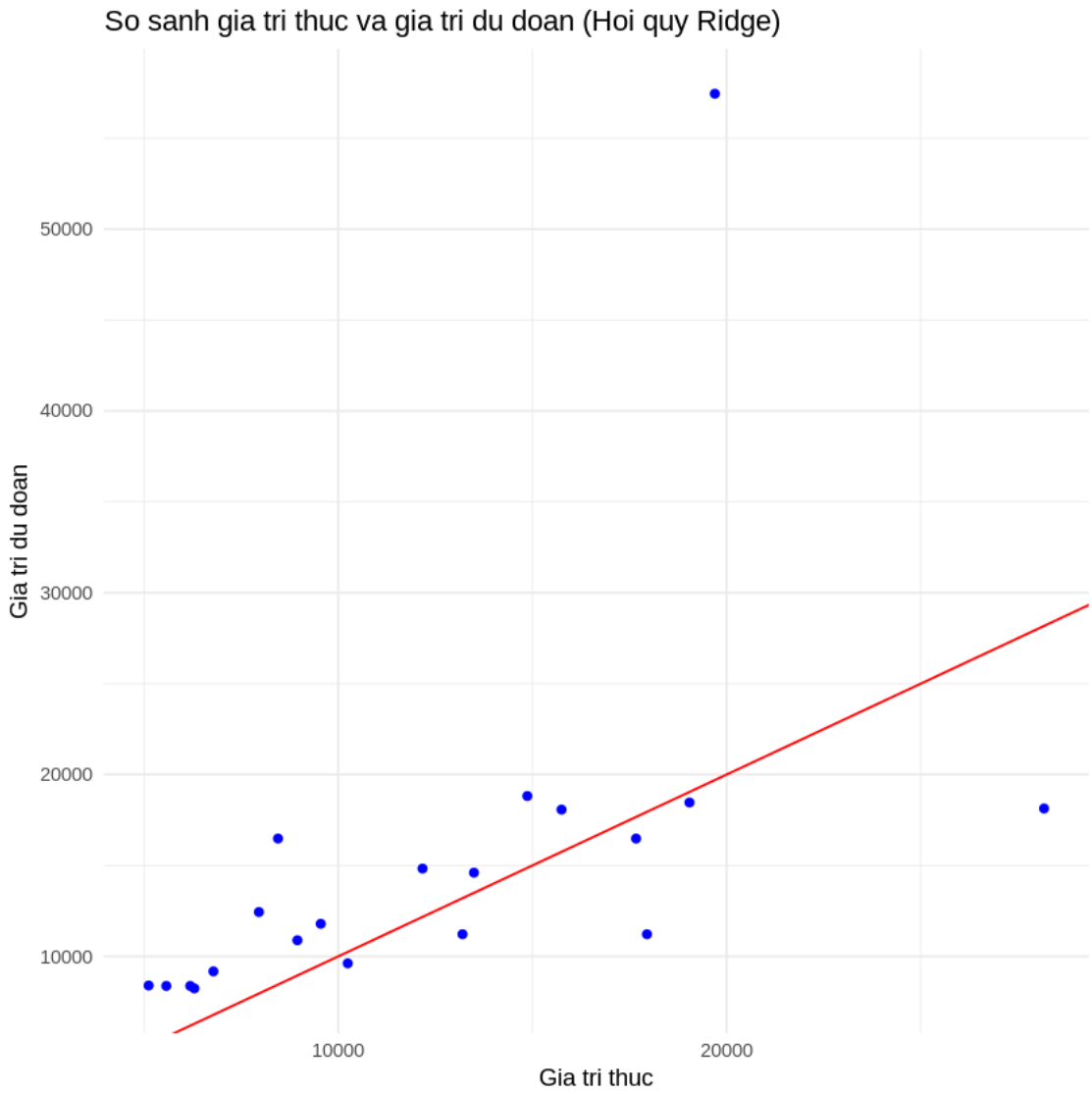
Chọn lamda tối ưu Sử dụng cross-validation thử nhiều giá trị lambda khác nhau tìm lamda tốt nhất

[ ]:
```r
# Thư viện
library(MASS)   # Thư viện cho hồi quy Ridge
library(ggplot2)

# Tạo biến đa thức bậc 11 cho biến horsepower
x_train_poly <- data.frame(horsepower = x_train$horsepower)
x_train_poly <- cbind(x_train_poly, poly(x_train$horsepower, degree = 11, raw =␣
↪TRUE))

# Đặt tên cho các biến đa thức
```

```r
colnames(x_train_poly) <- c("horsepower", paste0("X", 1:11))

# Thực hiện hồi quy Ridge với nhiều giá trị lambda
lambdas <- seq(0, 10, by = 0.1)
ridge_models <- lm.ridge(price ~ ., data = data.frame(x_train_poly, price =␣
 ↪y_train), lambda = lambdas)

# Chọn giá trị lambda tối ưu dựa trên GCV (Generalized Cross-Validation)
optimal_lambda <- ridge_models$lambda[which.min(ridge_models$GCV)]

cat("Gia tri lambda toi uu:", optimal_lambda, "\n")

# Huấn luyện mô hình hồi quy Ridge với lambda tối ưu
ridge_model <- lm.ridge(price ~ ., data = data.frame(x_train_poly, price =␣
 ↪y_train), lambda = optimal_lambda)

# Tạo biến đa thức cho dữ liệu test
x_test_poly <- data.frame(horsepower = x_test$horsepower)
x_test_poly <- cbind(x_test_poly, poly(x_test$horsepower, degree = 11, raw =␣
 ↪TRUE))

# Đặt tên cho các biến đa thức trong dữ liệu test
colnames(x_test_poly) <- c("horsepower", paste0("X", 1:11))

# Chuyển đổi x_test_poly thành ma trận số
x_test_matrix <- as.matrix(x_test_poly)

# Dự đoán giá trị với mô hình hồi quy Ridge
y_pred_ridge <- as.vector(cbind(1, x_test_matrix) %*% coef(ridge_model))

# Tính toán R^2 cho dữ liệu test
rss <- sum((y_test - y_pred_ridge) ^ 2)
tss <- sum((y_test - mean(y_test)) ^ 2)
r_squared_ridge <- 1 - (rss / tss)

# In ra kết quả
cat("R^2 cho mo hinh hoi quy Ridge (bac 11) voi lambda =", optimal_lambda, ":",␣
 ↪r_squared_ridge, "\n")

# Vẽ biểu đồ so sánh giá trị thực và giá trị dự đoán
comparison_df <- data.frame(Actual = y_test, Predicted = y_pred_ridge)

ggplot(comparison_df, aes(x = Actual, y = Predicted)) +
  geom_point(color = "blue") +
  geom_abline(slope = 1, intercept = 0, color = "red") +
  labs(title = "So sanh gia tri thuc va gia tri du doan (Hoi quy Ridge)",
       x = "Gia tri thuc",
```

```
       y = "Gia tri du doan") +
  theme_minimal()

# Hàm để vẽ Distribution Plot
distribution_plot <- function(actual, predicted, title) {
  data <- data.frame(Actual = actual, Predicted = predicted)

  ggplot(data) +
    geom_density(aes(x = Actual, fill = "Actual"), alpha = 0.5) +
    geom_density(aes(x = Predicted, fill = "Predicted"), alpha = 0.5) +
    labs(title = title, x = "Value", y = "Mat do") +
    scale_fill_manual(name = "Tham so", values = c("Actual" = "blue",␣
 ↪"Predicted" = "orange")) +
    theme_minimal()
}

# Vẽ DistributionPlot cho dữ liệu test
distribution_plot(y_test, y_pred_ridge, "Test Data (Polynomial Degree 11)")
```

```
Gia tri lambda toi uu: 0
R^2 cho mo hinh hoi quy Ridge (bac 11) voi lambda = 0 : -1.507141
```


So sanh gia tri thuc va gia tri du doan (Hoi quy Ridge)
Gia tri du doan
50000
40000
30000
20000
10000
10000
20000
Gia tri thuc

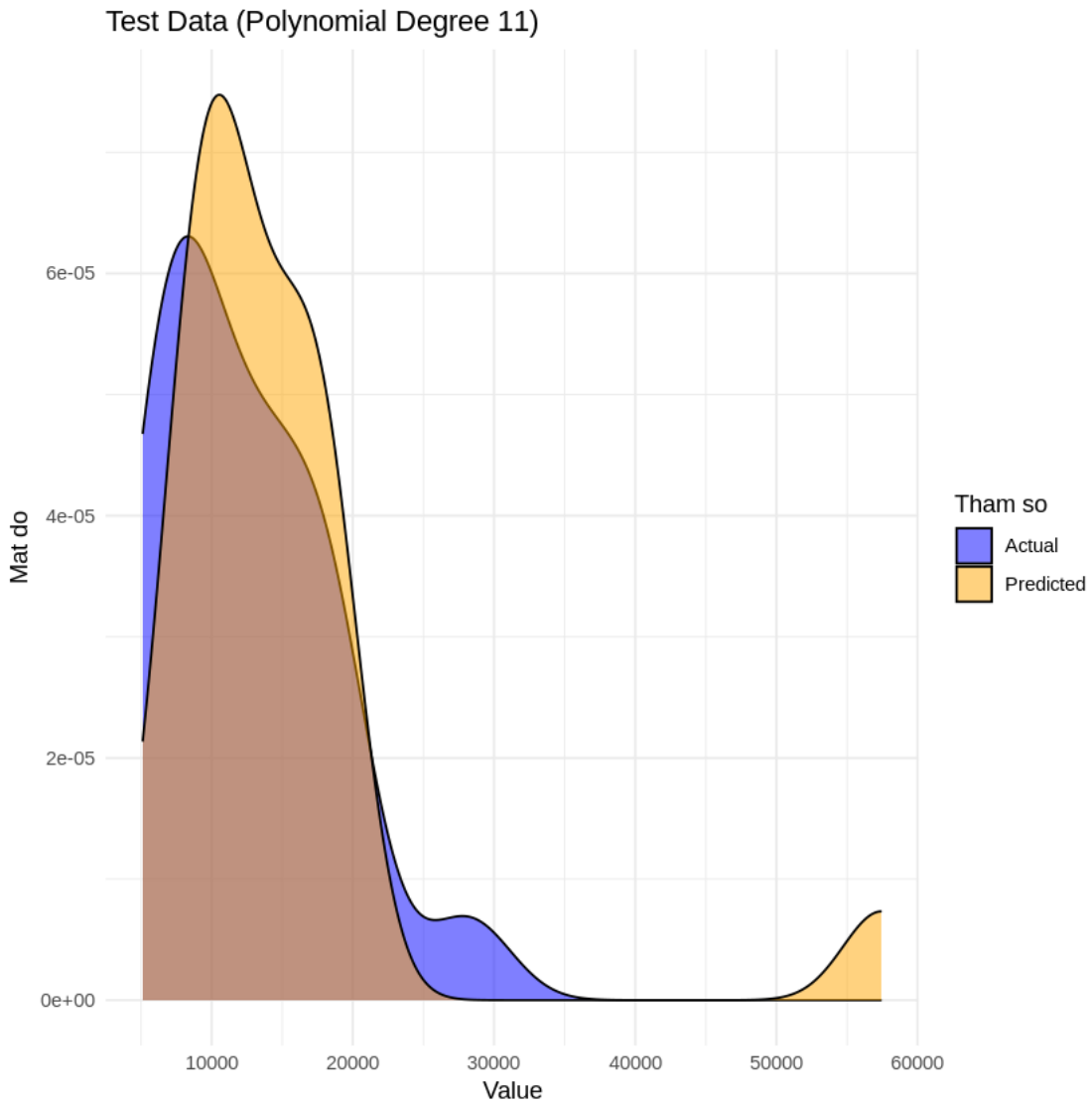


Figure 4: The blue line represents the R^2 of the validation data, and the red line represents the R^2 of the training data. The x-axis represents the different values of Alpha.

Here the model is built and tested on the same data, so the training and test data are the same.

The red line in Figure 4 represents the R^2 of the training data. As alpha increases the R^2 decreases. Therefore, as alpha increases, the model performs worse on the training data

The blue line represents the R^2 on the validation data. As the value for alpha increases, the R^2 decreases and converges at a point.

Question #5):

Perform Ridge regression. Calculate the R^2 using the polynomial features, use the training data to train the model and use the test data to test the model. The parameter alpha should be set to 10.

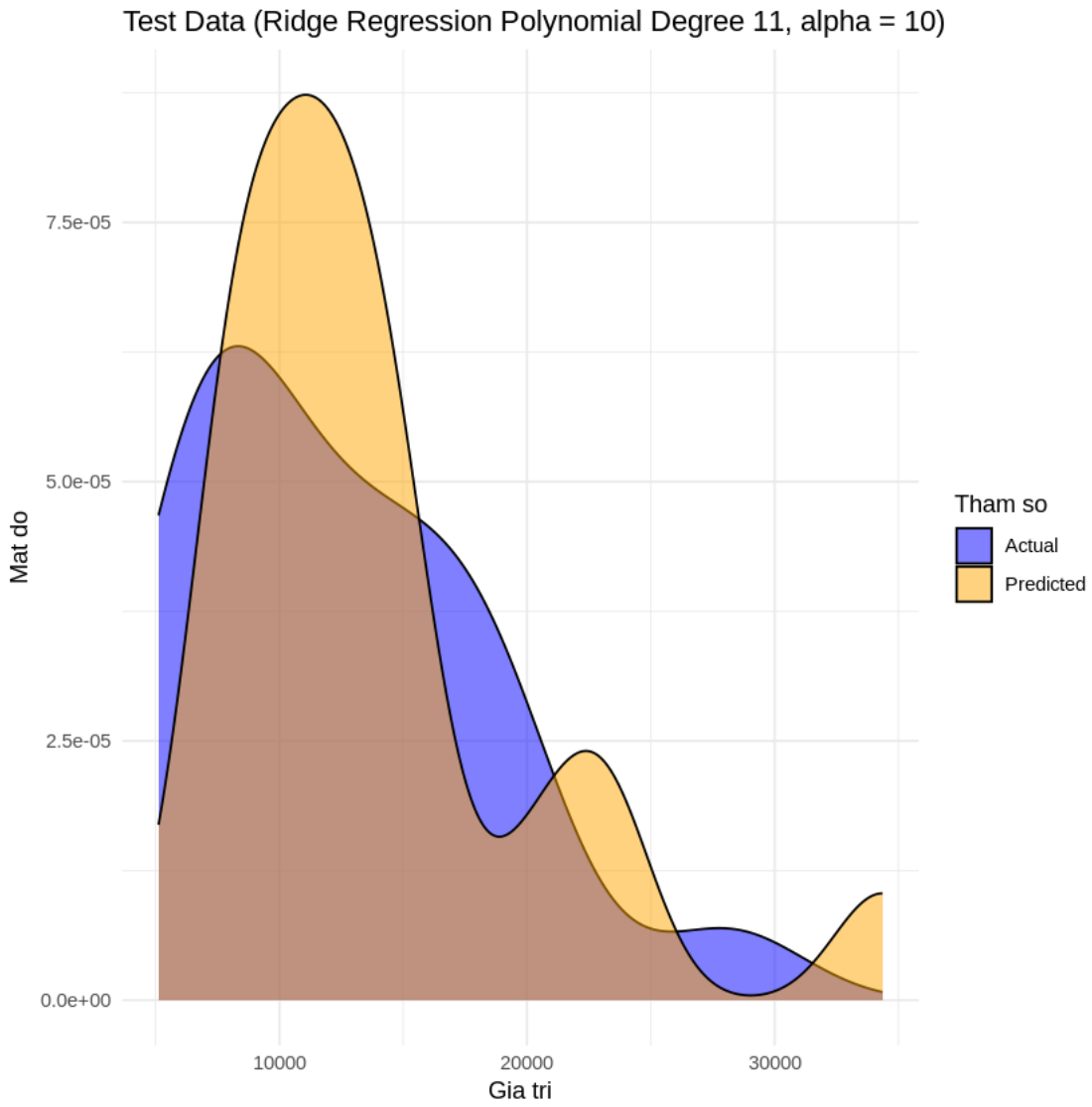
[ ]:
```r
install.packages("glmnet")
```

```
Installing package into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)

also installing the dependency ‘RcppEigen’
```

[ ]:
```r
# Thư viện cần thiết
library(MASS)   # Thư viện cho hồi quy Ridge
library(ggplot2)

# Tạo các đặc trưng đa thức bậc 11 cho biến horsepower
x_train_poly <- data.frame(horsepower = x_train$horsepower)
x_train_poly <- cbind(x_train_poly, poly(x_train$horsepower, degree = 11, raw =␣
 ↪TRUE))

# Đặt tên cho các biến đa thức
colnames(x_train_poly) <- c("horsepower", paste0("X", 1:11))

# Huấn luyện mô hình hồi quy Ridge với alpha = 10
ridge_model <- lm.ridge(price ~ ., data = data.frame(x_train_poly, price =␣
 ↪y_train), lambda = 10)

# Tạo đặc trưng đa thức cho dữ liệu test
x_test_poly <- data.frame(horsepower = x_test$horsepower)
x_test_poly <- cbind(x_test_poly, poly(x_test$horsepower, degree = 11, raw =␣
 ↪TRUE))

# Đặt tên cho các biến đa thức trong dữ liệu test
colnames(x_test_poly) <- c("horsepower", paste0("X", 1:11))

# Chuyển đổi x_test_poly thành ma trận số
x_test_matrix <- as.matrix(x_test_poly)

# Dự đoán giá trị với mô hình hồi quy Ridge
y_pred_ridge <- as.vector(cbind(1, x_test_matrix) %*% coef(ridge_model))

# Tính toán R² cho dữ liệu test
rss <- sum((y_test - y_pred_ridge) ^ 2)
tss <- sum((y_test - mean(y_test)) ^ 2)
r_squared_ridge <- 1 - (rss / tss)

# In ra kết quả
cat("R^2 cho mo hinh hoi quy Ridge (bac 11) voi alpha = 10:", r_squared_ridge,␣
 ↪"\n")
```

```r
# Hàm để vẽ Distribution Plot
distribution_plot <- function(actual, predicted, title) {
  data <- data.frame(Actual = actual, Predicted = predicted)

  ggplot(data) +
    geom_density(aes(x = Actual, fill = "Actual"), alpha = 0.5) +
    geom_density(aes(x = Predicted, fill = "Predicted"), alpha = 0.5) +
    labs(title = title, x = "Gia tri", y = "Mat do") +
    scale_fill_manual(name = "Tham so", values = c("Actual" = "blue",␣
 ↪"Predicted" = "orange")) +
    theme_minimal()
}

# Vẽ DistributionPlot cho dữ liệu test
distribution_plot(y_test, y_pred_ridge, "Test Data (Ridge Regression Polynomial␣
 ↪Degree 11, alpha = 10)")
```

```
R^2 cho mo hinh hoi quy Ridge (bac 11) voi alpha = 10: 0.1167197
```



Part 4: Grid Search

lambda tối ưu và R^2 tốt nhất (0.001,0.1,1, 10, 100, 1000, 10000, 100000, 100000)

[ ]:
```
# Thư viện cần thiết
library(MASS)   # Thư viện cho hồi quy Ridge

# Tạo các đặc trưng đa thức bậc 11 cho biến horsepower
x_train_poly <- data.frame(horsepower = x_train$horsepower)
x_train_poly <- cbind(x_train_poly, poly(x_train$horsepower, degree = 11, raw =␣
 ↪TRUE))

# Đặt tên cho các biến đa thức
colnames(x_train_poly) <- c("horsepower", paste0("X", 1:11))
```

```r
# Tạo đặc trưng đa thức cho dữ liệu test
x_test_poly <- data.frame(horsepower = x_test$horsepower)
x_test_poly <- cbind(x_test_poly, poly(x_test$horsepower, degree = 11, raw =␣
↪TRUE))

# Đặt tên cho các biến đa thức trong dữ liệu test
colnames(x_test_poly) <- c("horsepower", paste0("X", 1:11))

# Chuyển đổi x_test_poly thành ma trận số
x_test_matrix <- as.matrix(x_test_poly)

# Danh sách giá trị lambda cần thử
lambda_values <- c(0.001, 0.1, 1, 10, 100, 1000, 10000, 100000, 1000000)
best_lambda <- NULL
best_r_squared <- -Inf

# Vòng lặp tìm giá trị lambda tối ưu
for (lambda in lambda_values) {

  # Huấn luyện mô hình hồi quy Ridge với lambda hiện tại
  ridge_model <- lm.ridge(price ~ ., data = data.frame(x_train_poly, price =␣
 ↪y_train), lambda = lambda)

  # Dự đoán giá trị với mô hình hồi quy Ridge
  y_pred_ridge <- as.vector(cbind(1, x_test_matrix) %*% coef(ridge_model))

  # Tính toán R² cho dữ liệu test
  rss <- sum((y_test - y_pred_ridge) ^ 2)
  tss <- sum((y_test - mean(y_test)) ^ 2)
  r_squared_ridge <- 1 - (rss / tss)

  # Kiểm tra nếu đây là giá trị R² tốt nhất
  if (r_squared_ridge > best_r_squared) {
    best_r_squared <- r_squared_ridge
    best_lambda <- lambda
  }

  # In kết quả cho mỗi lambda
  cat("Lambda =", lambda, "- R^2 =", r_squared_ridge, "\n")
}

# In ra lambda tối ưu và R^2 tốt nhất
cat("Gia tri lambda tot nhat:", best_lambda, "\n")
cat("R^2 tot nhat cho lambda =", best_lambda, ":", best_r_squared, "\n")
```

```
Lambda = 0.001 - R^2 = 0.04128939
```

```
Lambda = 0.1 - R^2 = 0.05527595
Lambda = 1 - R^2 = 0.07138966
Lambda = 10 - R^2 = 0.1167197
Lambda = 100 - R^2 = 0.2566883
Lambda = 1000 - R^2 = 0.3296223
Lambda = 10000 - R^2 = 0.07998838
Lambda = 1e+05 - R^2 = -0.01297159
Lambda = 1e+06 - R^2 = -0.02450953
Gia tri lambda tot nhat: 1000
R^2 tot nhat cho lambda = 1000 : 0.3296223
```

We now test our model on the test data:

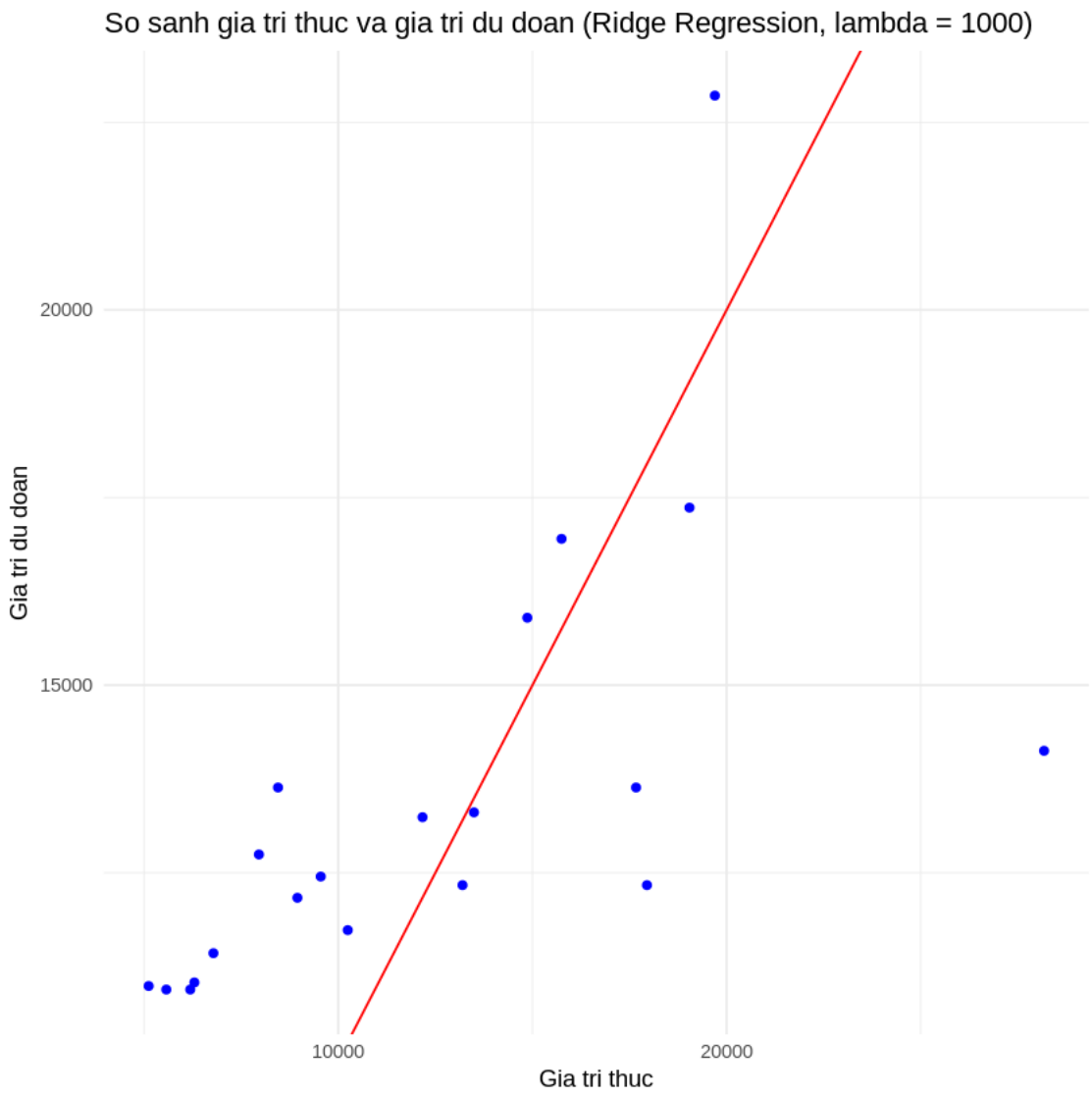
xây dựng mô hình Ridge Regression với lambda tối ưu là 1000 và đánh giá trên tập dữ liệu test

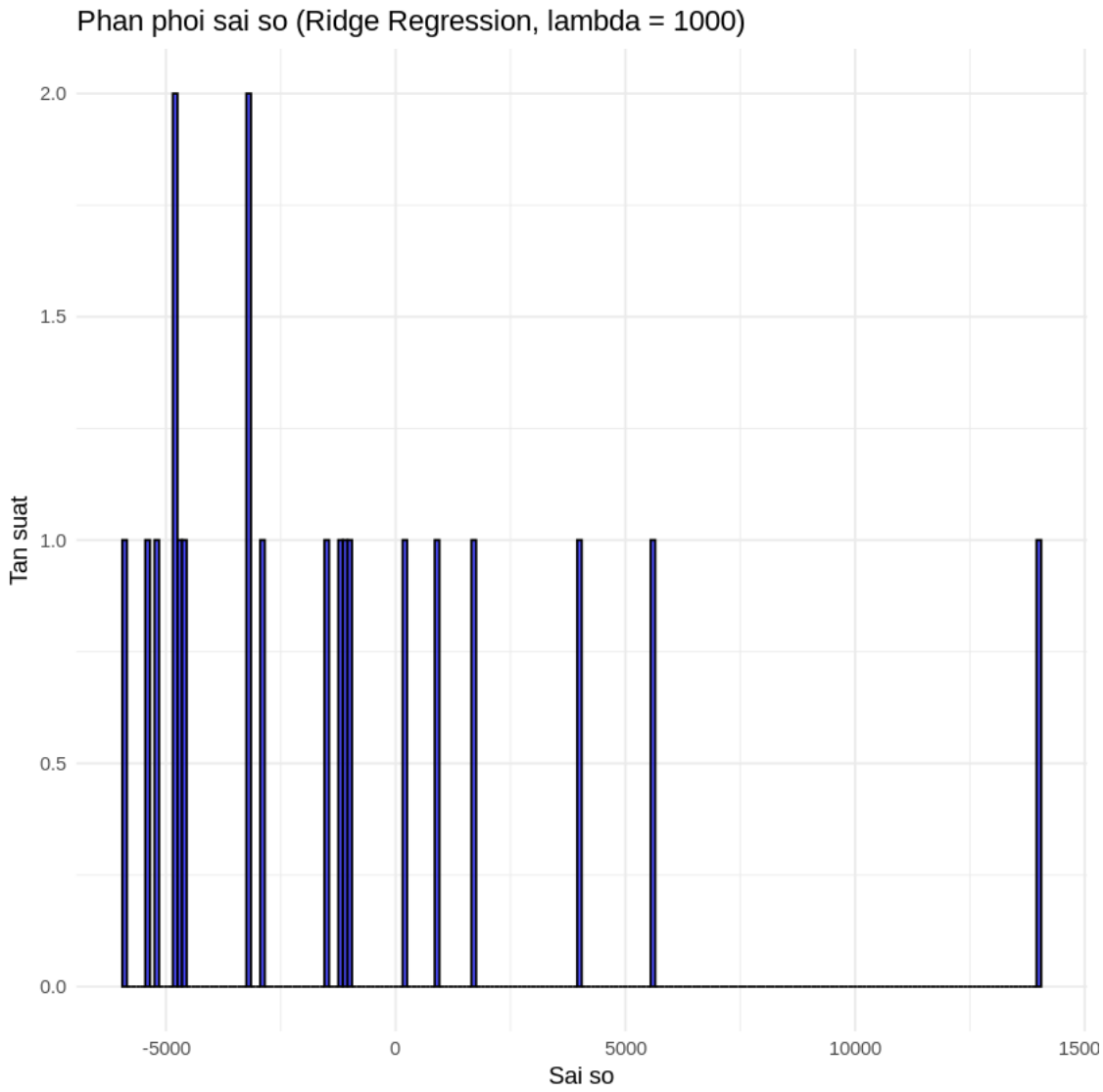
[ ]:
```r
# Thư viện
library(MASS)   # Thư viện cho hồi quy Ridge
library(ggplot2)

# Tạo các đặc trưng đa thức bậc 11 cho biến horsepower trong dữ liệu train
x_train_poly <- data.frame(horsepower = x_train$horsepower)
x_train_poly <- cbind(x_train_poly, poly(x_train$horsepower, degree = 11, raw =␣
 ↪TRUE))

# Đặt tên cho các biến đa thức
colnames(x_train_poly) <- c("horsepower", paste0("X", 1:11))

# Tạo đặc trưng đa thức bậc 11 cho biến horsepower trong dữ liệu test
x_test_poly <- data.frame(horsepower = x_test$horsepower)
x_test_poly <- cbind(x_test_poly, poly(x_test$horsepower, degree = 11, raw =␣
 ↪TRUE))

# Đặt tên cho các biến đa thức trong dữ liệu test
colnames(x_test_poly) <- c("horsepower", paste0("X", 1:11))

# Huấn luyện mô hình hồi quy Ridge với lambda = 1000
lambda_value <- 1000
ridge_model <- lm.ridge(price ~ ., data = data.frame(x_train_poly, price =␣
 ↪y_train), lambda = lambda_value)

# Chuyển đổi x_test_poly thành ma trận số cho dự đoán
x_test_matrix <- as.matrix(x_test_poly)

# Dự đoán giá trị với mô hình hồi quy Ridge
y_pred_ridge <- as.vector(cbind(1, x_test_matrix) %*% coef(ridge_model))

# Tính toán R^2 cho dữ liệu test
rss <- sum((y_test - y_pred_ridge) ^ 2)
```

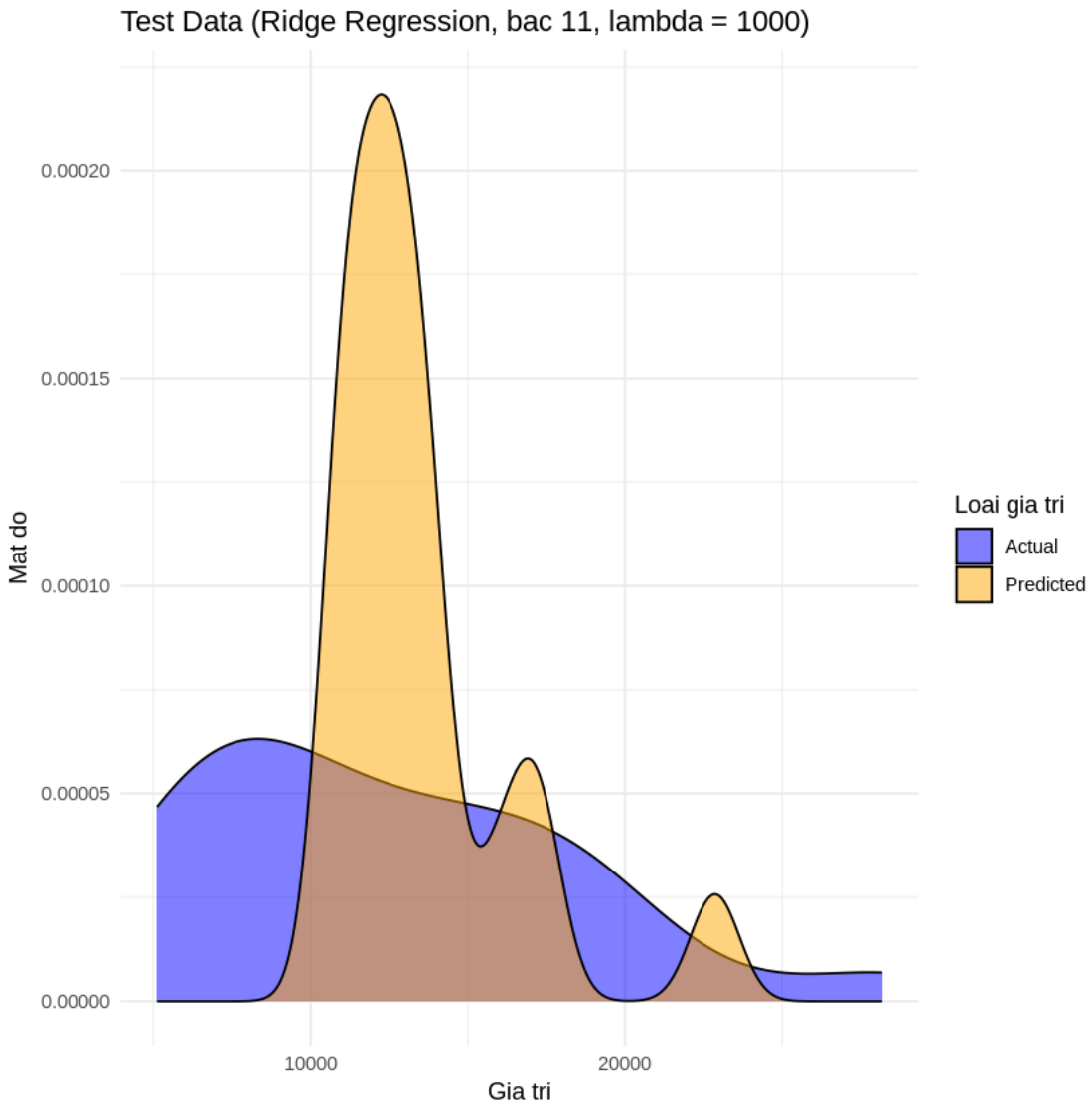
```r
tss <- sum((y_test - mean(y_test)) ^ 2)
r_squared_ridge <- 1 - (rss / tss)

# In ra kết quả R²
cat("R^2 cho mo hinh hoi quy Ridge (bac 11) voi lambda =", lambda_value, ":",␣
 ↪r_squared_ridge, "\n")

# Vẽ biểu đồ so sánh giá trị thực và dự đoán
comparison_df <- data.frame(Actual = y_test, Predicted = y_pred_ridge)

ggplot(comparison_df, aes(x = Actual, y = Predicted)) +
  geom_point(color = "blue") +
  geom_abline(slope = 1, intercept = 0, color = "red") +
  labs(title = "So sanh gia tri thuc va gia tri du doan (Ridge Regression,␣
 ↪lambda = 1000)",
       x = "Gia tri thuc",
       y = "Gia tri du doan") +
  theme_minimal()

# Vẽ DistributionPlot cho sai số
distribution_plot <- function(y_true, y_pred, title) {
  error <- y_true - y_pred
  ggplot(data.frame(error = error), aes(x = error)) +
    geom_histogram(binwidth = 100, fill = "blue", color = "black", alpha = 0.7)␣
 ↪+
    labs(title = title, x = "Sai so", y = "Tan suat") +
    theme_minimal()
}

# Vẽ phân phối sai số cho test data
distribution_plot(y_test, y_pred_ridge, "Phan phoi sai so (Ridge Regression,␣
 ↪lambda = 1000)")

# Vẽ biểu đồ mật độ cho giá trị thực và dự đoán
density_plot <- function(y_true, y_pred, title) {
  data <- data.frame(
    Value = c(y_true, y_pred),
    Type = rep(c("Actual", "Predicted"), each = length(y_true))
  )

  ggplot(data, aes(x = Value, fill = Type)) +
    geom_density(alpha = 0.5) +
    labs(title = title, x = "Gia tri", y = "Mat do") +
    scale_fill_manual(name = "Loai gia tri", values = c("Actual" = "blue",␣
 ↪"Predicted" = "orange")) +
    theme_minimal()
}
```

```
# Vẽ biểu đồ mật độ cho dữ liệu test
density_plot(y_test, y_pred_ridge, "Test Data (Ridge Regression, bac 11, lambda␣
  ↪= 1000)")
```

```
R^2 cho mo hinh hoi quy Ridge (bac 11) voi lambda = 1000 : 0.3296223
```




Phan phoi sai so (Ridge Regression, lambda = 1000)
Tan suat
2.0
1.5
1.0
0.5
0.0
-5000
0
5000
10000
1500
Sai so


Test Data (Ridge Regression, bac 11, lambda = 1000)
0.00020
0.00015
0.00010
0.00005
0.00000
Mat do
10000
20000
Gia tri
Loai gia tri
Actual
Predicted