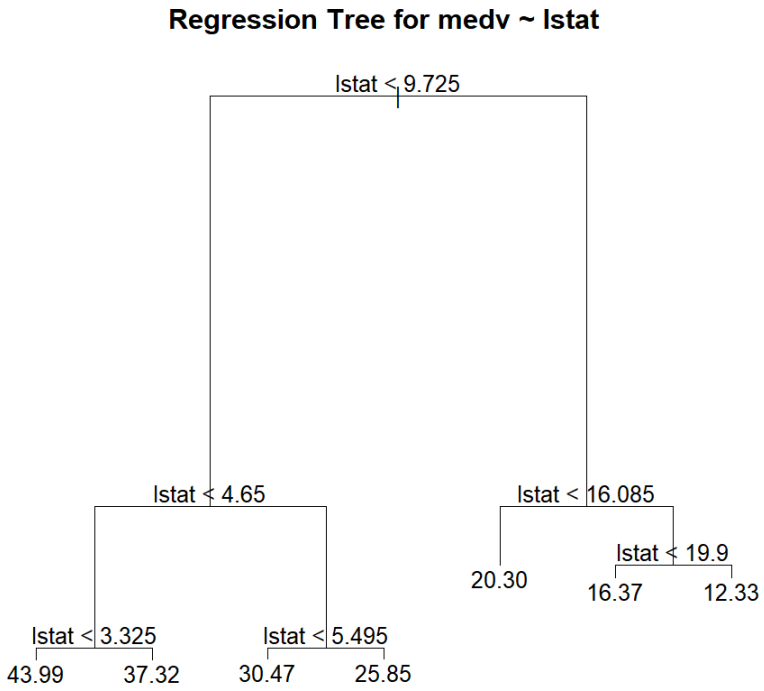
# Tree-Based Methods and SVM/Kernel Methods

## Part 1: Tree-Based Methods

### Task 1: Regression Tree with tree

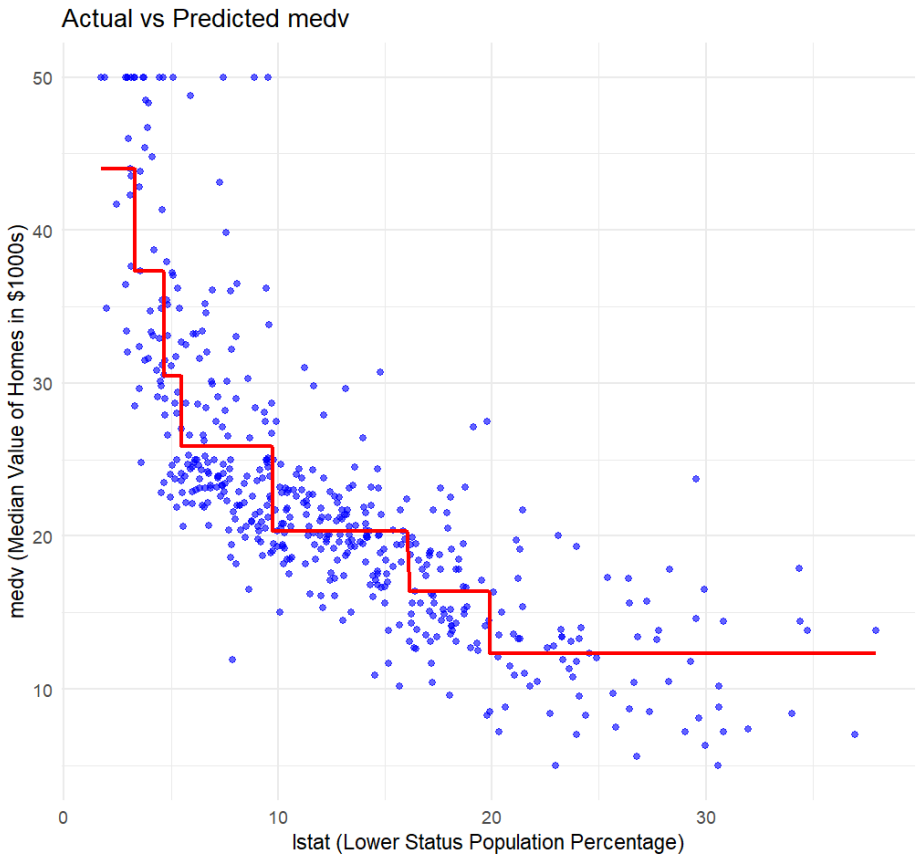
● Fit a regression tree model with medv (median value of owner-occupied homes in $1000s) as the response variable and lstat (percentage of lower status of the population) as the predictor.

```
> # Load the Boston dataset
> data("Boston")
> str(Boston)
'data.frame':     506 obs. of  14 variables:
 $ crim   : num  0.00632 0.02731 0.02729 0.03237 0.06905 ...
 $ zn     : num  18 0 0 0 0 0 12.5 12.5 12.5 12.5 ...
 $ indus  : num  2.31 7.07 7.07 2.18 2.18 2.18 7.87 7.87 7.87 7.87 ...
 $ chas   : int  0 0 0 0 0 0 0 0 0 0 ...
 $ nox    : num  0.538 0.469 0.469 0.458 0.458 0.458 0.524 0.524 0.524 0.524
...
 $ rm     : num  6.58 6.42 7.18 7 7.15 ...
 $ age    : num  65.2 78.9 61.1 45.8 54.2 58.7 66.6 96.1 100 85.9 ...
 $ dis    : num  4.09 4.97 4.97 6.06 6.06 ...
 $ rad    : int  1 2 2 3 3 3 5 5 5 5 ...
 $ tax    : num  296 242 242 222 222 222 311 311 311 311 ...
 $ ptratio: num  15.3 17.8 17.8 18.7 18.7 18.7 15.2 15.2 15.2 15.2 ...
 $ black  : num  397 397 393 395 397 ...
 $ lstat  : num  4.98 9.14 4.03 2.94 5.33 ...
 $ medv   : num  24 21.6 34.7 33.4 36.2 28.7 22.9 27.1 16.5 18.9 ...
> reg_tree <- tree(medv ~ lstat, data = Boston)
> summary(reg_tree)

Regression tree:
tree(formula = medv ~ lstat, data = Boston)
Number of terminal nodes:  7
Residual mean deviance:  26.08 = 13020 / 499
Distribution of residuals:
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-14.5200  -2.9470  -0.7568   0.0000   2.3250  24.1500
```




- Plot the regression tree and the predicted values against the actual data points.

```
> plot(reg_tree)
> text(reg_tree, pretty = 0)
> title("Regression Tree for medv ~ lstat")
> predictions <- predict(reg_tree, newdata = Boston)
> ggplot(Boston, aes(x = lstat, y = medv)) +
+  geom_point(color = "blue", alpha = 0.6) +
+  geom_line(aes(y = predictions), color = "red", size = 1) +
+  labs(title = "Actual vs Predicted medv",
+       x = "lstat (Lower Status Population Percentage)",
+       y = "medv (Median Value of Homes in $1000s)") +
+  theme_minimal()
```




● Evaluate the performance of the model using Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE).

```
> MAE <- mean(abs(predictions - Boston$medv))
> MSE <- mean((predictions - Boston$medv)^2)
> RMSE <- sqrt(MSE)
> cat("Performance Metrics for Regression Tree:\n")
Performance Metrics for Regression Tree:
> cat("Mean Absolute Error (MAE): ", round(MAE, 2), "\n")
Mean Absolute Error (MAE):  3.68
> cat("Mean Squared Error (MSE): ", round(MSE, 2), "\n")
Mean Squared Error (MSE):  25.72
> cat("Root Mean Squared Error (RMSE): ", round(RMSE, 2), "\n")
Root Mean Squared Error (RMSE):  5.07
```

### Regression Tree Model Performance Summary

1. **Mean Absolute Error (MAE) = 3.68**
    - On average, the model's predictions are $3,680 off from actual home prices.
    - A lower MAE would indicate a more accurate model.
2. **Mean Squared Error (MSE) = 25.72**
    - Measures the average squared error, meaning some predictions are significantly off.
    - This metric is sensitive to large mistakes.
3. **Root Mean Squared Error (RMSE) = 5.07**
    - The average error is about $5,070, which is quite high considering home prices in the dataset.

### Overall Analysis:

- The model does okay but has room for improvement—it makes moderate errors, and some predictions are quite off.
- Ways to Improve: Adding more features (like rooms, crime rate), pruning the tree, or using more advanced models like Random Forest.
- MAE (3.68) and RMSE (5.07) indicate a moderate prediction error.
- RMSE > MAE confirms the presence of some large errors (outliers affecting MSE).
- The model performs reasonably well but could be improved.

## Task 2: Classification Tree with tree

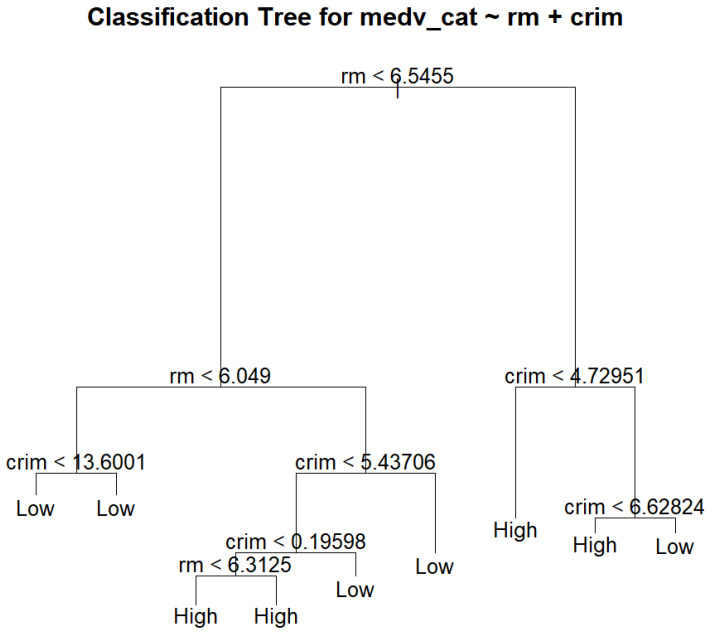
● Create a binary classification variable in the Boston dataset categorizing homes into "high-value" and "low-value" based on whether medv is above or below the median.

```
> median_medv <- median(Boston$medv)
> Boston$medv_cat <- ifelse(Boston$medv > median_medv, "High", "Low")
> Boston$medv_cat <- factor(Boston$medv_cat)
> table(Boston$medv_cat)

High  Low
 250  256
```

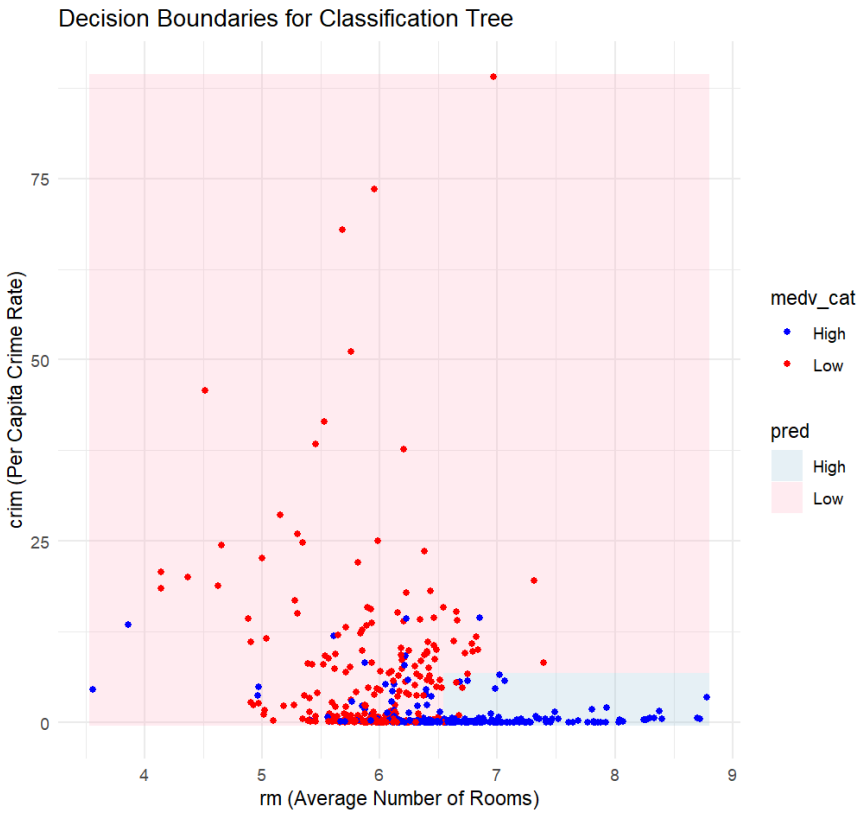
● Fit a classification tree model using rm (average number of rooms) and crim (per capita crime rate) as predictors.

```
> class_tree <- tree(medv_cat ~ rm + crim, data = Boston)
> summary(class_tree)

Classification tree:
tree(formula = medv_cat ~ rm + crim, data = Boston)
Number of terminal nodes:  9
Residual mean deviance:  0.751 = 373.2 / 497
Misclassification error rate: 0.1739 = 88 / 506
> plot(class_tree)
> text(class_tree, pretty = 0)
> title("Classification Tree for medv_cat ~ rm + crim")
```

● Plot the classification tree and the decision boundaries.



```
> plot(class_tree)
> text(class_tree, pretty = 0)
> title("Classification Tree for medv_cat ~ rm + crim")
> plot_data <- expand.grid(
+  rm = seq(min(Boston$rm), max(Boston$rm), length.out = 100),
+  crim = seq(min(Boston$crim), max(Boston$crim), length.out = 100)
+ )
> plot_data$pred <- predict(class_tree, newdata = plot_data, type = "class")
> ggplot() +
+  geom_tile(data = plot_data, aes(x = rm, y = crim, fill = pred), alpha =
0.3) +
+  geom_point(data = Boston, aes(x = rm, y = crim, color = medv_cat)) +
+  labs(title = "Decision Boundaries for Classification Tree",
+       x = "rm (Average Number of Rooms)",
+       y = "crim (Per Capita Crime Rate)") +
+  scale_fill_manual(values = c("High" = "lightblue", "Low" = "pink")) +
+  scale_color_manual(values = c("High" = "blue", "Low" = "red")) +
+  theme_minimal()
```




- Evaluate the performance of the model using a confusion matrix and accuracy.

```
> accuracy <- sum(diag(conf_mat)) / sum(conf_mat)
> cat("Accuracy of Classification Tree: ", round(accuracy * 100, 2), "%\n")
Accuracy of Classification Tree:  82.61 %
> conf_mat <- table(Predicted = pred_class, Actual = Boston$medv_cat)
> print(conf_mat)
         Actual
Predicted High Low
     High  183  21
     Low    67 235
> library(caret)
Loading required package: ggplot2
Loading required package: lattice
Warning message:
package 'caret' was built under R version 4.4.3
> confusion_matrix <- confusionMatrix(conf_mat)
> print(confusion_matrix)
Confusion Matrix and Statistics

         Actual
Predicted High Low
     High  183  21
     Low    67 235

               Accuracy : 0.8261
                 95% CI : (0.7902, 0.8581)
    No Information Rate : 0.5059
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.6514

 Mcnemar's Test P-Value : 1.61e-06

            Sensitivity : 0.7320
            Specificity : 0.9180
         Pos Pred Value : 0.8971
         Neg Pred Value : 0.7781
             Prevalence : 0.4941
         Detection Rate : 0.3617
   Detection Prevalence : 0.4032
      Balanced Accuracy : 0.8250

       'Positive' Class : High
```

## Interpretation:

**Strengths:**

- High accuracy (82.61%), indicating the model performs well.
- Good specificity (91.8%), meaning it accurately identifies low-value homes.

**Areas for Improvement:**

- Sensitivity (73.2%) could be better—it misses some high-value homes.
- Potential Overfitting—a pruned tree or Random Forest might generalize better.

## Part 2: SVM and Kernel Methods

### Task 3: Linear SVM

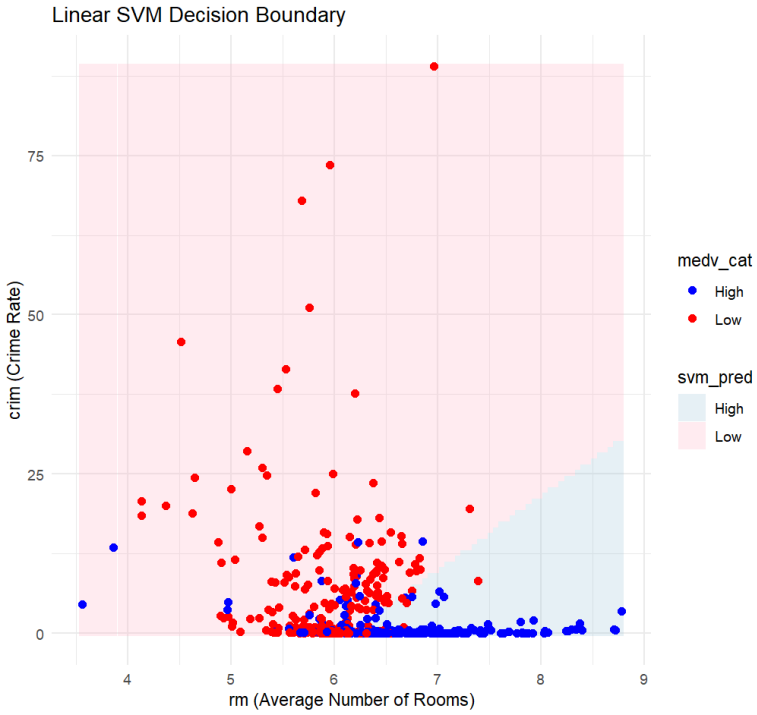
● Fit a linear SVM model to classify homes using rm and crim as predictors and the binary high_value variable as the response.

```
> library(e1071)
> library(ggplot2)

> Boston$medv_cat <- as.factor(Boston$medv_cat)
> svm_linear <- svm(medv_cat ~ rm + crim,
+                   data = Boston,
+                   kernel = "linear",
+                   cost = 1)  # Cost controls margin width
```

● Plot the decision boundary of the SVM along with the data points.

```
> plot_data <- expand.grid(
+  rm = seq(min(Boston$rm), max(Boston$rm), length.out = 100),
+  crim = seq(min(Boston$crim), max(Boston$crim), length.out = 100)
+ )
> plot_data$svm_pred <- predict(svm_linear, newdata = plot_data)
> ggplot() +
+  geom_tile(data = plot_data, aes(x = rm, y = crim, fill = svm_pred), alpha =
0.3) +
+  geom_point(data = Boston, aes(x = rm, y = crim, color = medv_cat), size =
2) +
+  labs(title = "Linear SVM Decision Boundary",
```

```
+       x = "rm (Average Number of Rooms)",
+       y = "crim (Crime Rate)") +
+  scale_fill_manual(values = c("High" = "lightblue", "Low" = "pink")) +
+  scale_color_manual(values = c("High" = "blue", "Low" = "red")) +
+  theme_minimal()
> svm_pred <- predict(svm_linear, newdata = Boston)
> conf_mat <- table(Predicted = svm_pred, Actual = Boston$medv_cat)
> confusion_matrix <- confusionMatrix(conf_mat)
> print(confusion_matrix)

Confusion Matrix and Statistics

          Actual
Predicted High Low
     High  188  27
     Low    62 229

               Accuracy : 0.8241
                 95% CI : (0.7881, 0.8563)
    No Information Rate : 0.5059
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.6476

 Mcnemar's Test P-Value : 0.0003134

            Sensitivity : 0.7520
            Specificity : 0.8945
         Pos Pred Value : 0.8744
         Neg Pred Value : 0.7869
             Prevalence : 0.4941
         Detection Rate : 0.3715
   Detection Prevalence : 0.4249
      Balanced Accuracy : 0.8233

       'Positive' Class : High
```




● Interpret the model and discuss the effectiveness of linear SVM in this context.

Yes, Linear SVM is effective in classifying High vs. Low-value homes with good accuracy (82.41%) However, it struggles slightly in correctly identifying High-value homes (Sensitivity: 75.20%).

## Strengths of Linear SVM:

- **Good accuracy (82.41%)**, Performs well in distinguishing between High and Low-value homes.
- **High Specificity (89.45%)**, Accurately identifies Low-value homes.
- **Balanced Decision Boundary**, Linear SVM provides a clear separation between High and Low-value homes.

## Limitations of Linear SVM:

- **Sensitivity is lower than specificity (75.20% vs. 89.45%)**
  The model struggles slightly in identifying High-value homes.

- **Potential for Non-Linearity**
  If the decision boundary is not well-defined with a straight line, a non-linear SVM (Radial Kernel) might work better.

## Task 4: SVM with RBF Kernel

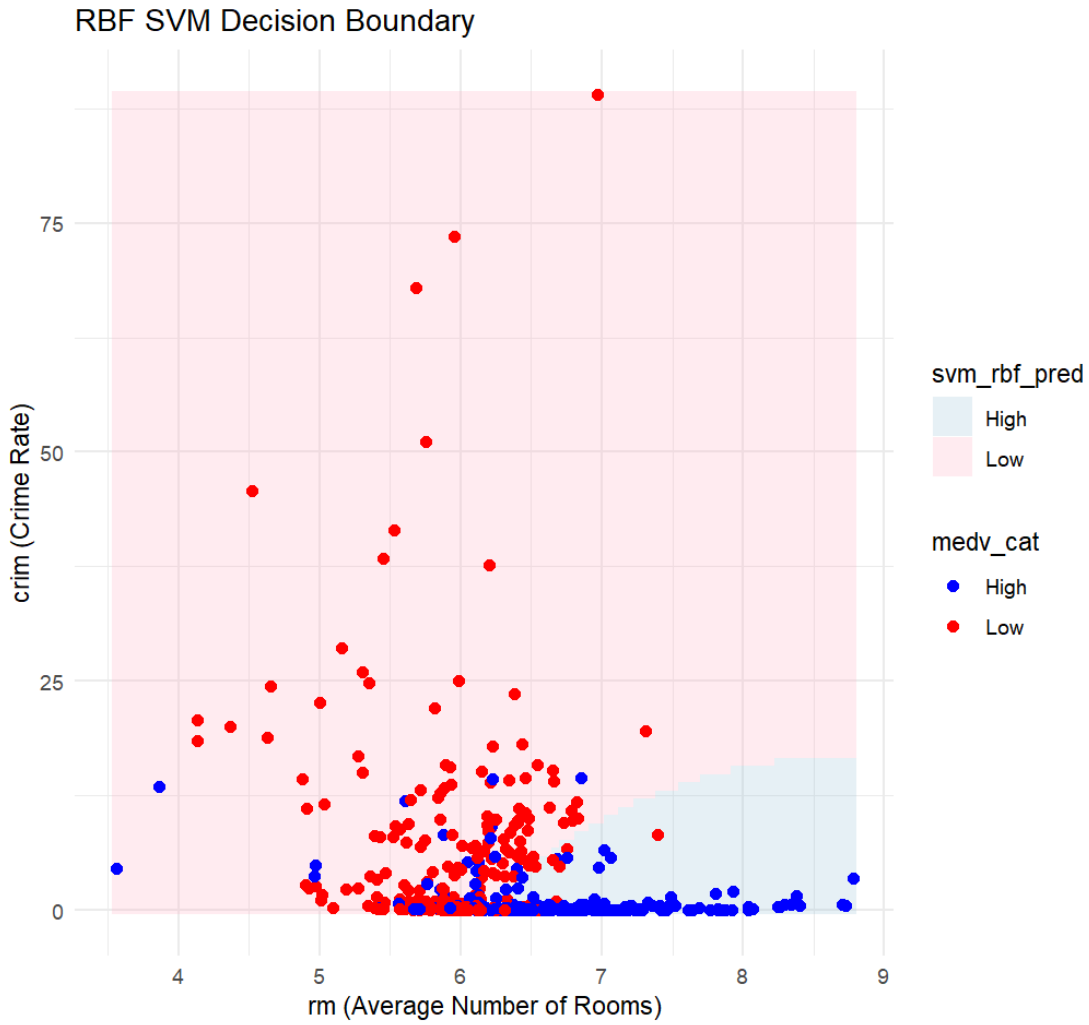
● Fit an SVM model with a radial basis function (RBF) kernel to the same data.

```
> library(ggplot2)
> library(caret)
> svm_rbf <- svm(medv_cat ~ rm + crim,
+                data = Boston,
+                kernel = "radial",
+                gamma = 0.1,
+                cost = 1)  # Default cost
> svm_rbf_pred <- predict(svm_rbf, newdata = Boston)
> conf_mat_rbf <- table(Predicted = svm_rbf_pred, Actual = Boston$medv_cat)
> confusion_matrix_rbf <- confusionMatrix(conf_mat_rbf)
> print(confusion_matrix_rbf)
Confusion Matrix and Statistics

          Actual
Predicted High Low
     High  180  24
     Low    70 232

               Accuracy : 0.8142
                 95% CI : (0.7776, 0.8472)
    No Information Rate : 0.5059
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.6276

 Mcnemar's Test P-Value : 3.461e-06

            Sensitivity : 0.7200
            Specificity : 0.9062
         Pos Pred Value : 0.8824
         Neg Pred Value : 0.7682
             Prevalence : 0.4941
         Detection Rate : 0.3557
   Detection Prevalence : 0.4032
      Balanced Accuracy : 0.8131
```

```
'Positive' Class : High
```

● Compare the performance of the linear and RBF SVM models using accuracy and visualizations.

```
> plot_data <- expand.grid(
+   rm = seq(min(Boston$rm), max(Boston$rm), length.out = 100),
+   crim = seq(min(Boston$crim), max(Boston$crim), length.out = 100)
+ )
> plot_data$svm_linear_pred <- predict(svm_linear, newdata = plot_data)
> plot_data$svm_rbf_pred <- predict(svm_rbf, newdata = plot_data)
> ggplot() +
+   geom_tile(data = plot_data, aes(x = rm, y = crim, fill = svm_linear_pred),
alpha = 0.3) +
+   geom_point(data = Boston, aes(x = rm, y = crim, color = medv_cat), size =
2) +
+   labs(title = "Linear SVM Decision Boundary",
+        x = "rm (Average Number of Rooms)",
+        y = "crim (Crime Rate)") +
+   scale_fill_manual(values = c("High" = "lightblue", "Low" = "pink")) +
+   scale_color_manual(values = c("High" = "blue", "Low" = "red")) +
+   theme_minimal()
> ggplot() +
+   geom_tile(data = plot_data, aes(x = rm, y = crim, fill = svm_rbf_pred),
alpha = 0.3) +
+   geom_point(data = Boston, aes(x = rm, y = crim, color = medv_cat), size =
2) +
+   labs(title = "RBF SVM Decision Boundary",
+        x = "rm (Average Number of Rooms)",
+        y = "crim (Crime Rate)") +
+   scale_fill_manual(values = c("High" = "lightblue", "Low" = "pink")) +
+   scale_color_manual(values = c("High" = "blue", "Low" = "red")) +
+   theme_minimal()
```




- Discuss the differences in the decision boundaries and model performances.
    - **Linear SVM:**
        - Uses a straight line to separate High vs. Low-value homes.
        - Works well when data is linearly separable.
        - May struggle if the relationship between predictors and response is non-linear.
    - **RBF SVM:**
        - Uses a curved decision boundary that adapts to complex data patterns.

- More flexible than linear SVM and can handle non-linearity.
- Likely to provide better classification if data isn't linearly separable.

## Final Analysis:

- If the RBF SVM has a higher accuracy, it suggests that the relationship between rm and crim with medv_cat is non-linear, making RBF the better choice.
- If Linear SVM and RBF SVM have similar accuracy, then the problem is already well-separated linearly, meaning Linear SVM is sufficient.

==========