**Batch: A - 2 Roll No.: 16014022050**

**Experiment No.: 7**

|  |
| --- |
| **Title: Classification using support vector machine** |

**Aim:** To implement classification using SVM using R libraries

**Expected Outcome of Experiment:**

**CO3:** Understand the basic concept and techniques of Machine Learning regression and classification

**Books/ Journals/ Websites referred:**

1. Data Mining Concepts and Techniques Jiawei Han, Michelin Kamber, Jian Pie, 3rd edition

**Procedure for Implementation in lab:**

1. Select a dataset suitable for classification from UCI data repository or Kaggle. **Students should provide the following details of the dataset:**
   1. Data set used: Heart Disease Dataset.
   2. Title: Heart Disease Dataset.
   3. Source: Dataset available on Kaggle and was uploaded by Yasser H.
   4. Number of instances: Dataset consists of 303 instances or samples.
   5. Number of attributes: Dataset consists of 14 attributes or features.
   6. Attribute information:

* age: Age of the patient in years
* sex: Sex of the patient (1 = male; 0 = female)
* cp: Chest pain type (0 = typical angina; 1 = atypical angina; 2 = non-anginal pain; 3 = asymptomatic)
* trestbps: Resting blood pressure (in mm Hg on admission to the hospital)
* chol: Serum cholesterol level (in mg/dl)
* fbs: Fasting blood sugar > 120 mg/dl (1 = true; 0 = false)
* restecg: Resting electrocardiographic results (0 = normal; 1 = having ST-T wave abnormality; 2 = showing probable or definite left ventricular hypertrophy)
* thalach: Maximum heart rate achieved
* exang: Exercise-induced angina (1 = yes; 0 = no)
* oldpeak: ST depression induced by exercise relative to rest
* slope: The slope of the peak exercise ST segment (0 = upsloping; 1 = flat; 2 = downsloping)
* ca: Number of major vessels (0-3) colored by fluoroscopy
* thal: Thalassemia (3 = normal; 6 = fixed defect; 7 = reversible defect)
* target: Presence of heart disease (1 = heart disease present; 0 = no heart disease)

1. Handle the missing values appropriately
2. Create the SVM model
3. Perform Visualization
4. Calculate prediction accuracy.

**Students should copy their R code and screenshots of output stepwise and paste them here.**

**Applying SVM Model for random data points:**

Rstudio code:

set.seed(100)

# constructing sample dataset that is completely separate

x <- matrix(rnorm(20\*2), ncol = 2)

y <- c(rep(-1, 10), rep(1, 10))

x[y==10, ] <- x[y==1, ] + 3/2

dat <- data.frame(x=x, y=as.factor(y))

View(dat)

library(ggplot2)

ggplot(data = dat, aes(x = x.2, y = x.1, color = y, shape = y)) +

geom\_point(size = 2) + scale\_color\_manual(values = c("#000000", "#FF0000")) +

theme(legend.position = "none")

library(e1071)

svmfit <- svm(y~., data = dat, kernel = "linear", cross = 10)

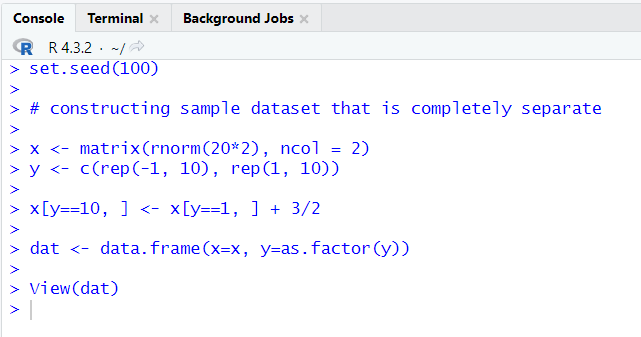
svmfit

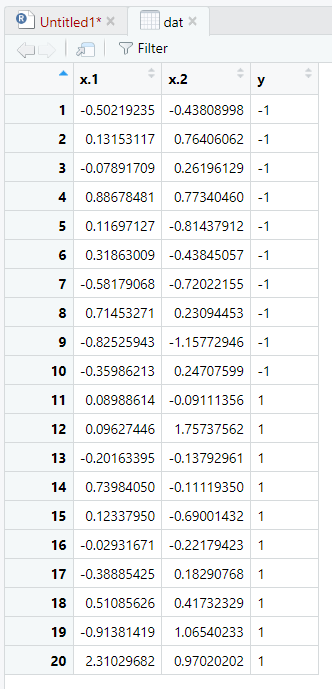
plot(svmfit, dat)

pred <- fitted(svmfit)

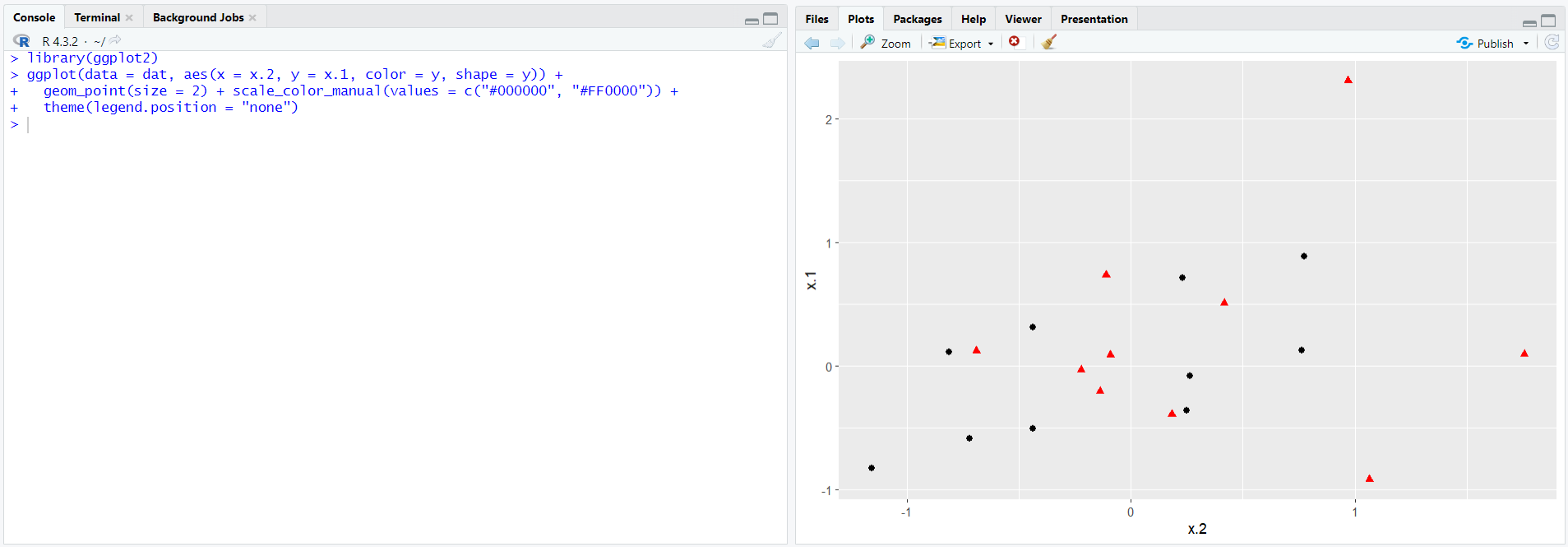
table(pred, y)

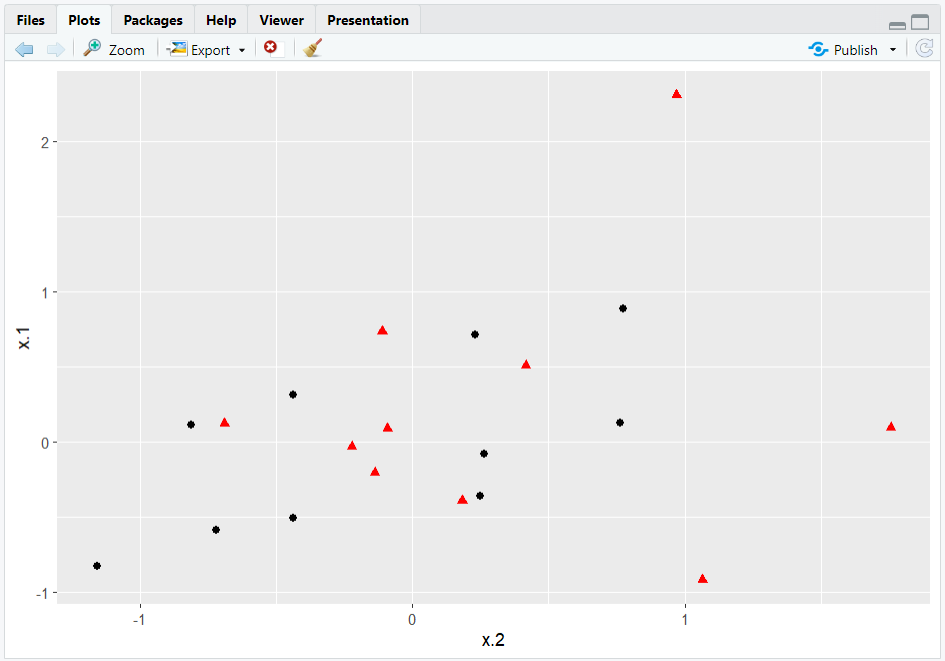
Generating 40 random data points and assigning them to 2 classes:

****

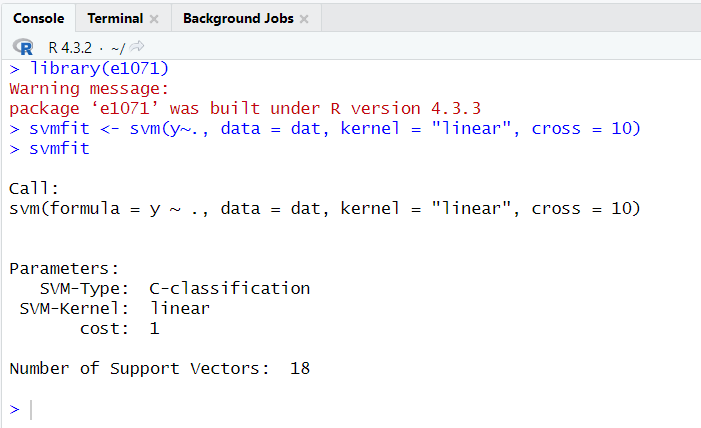
****

Plotting this data:

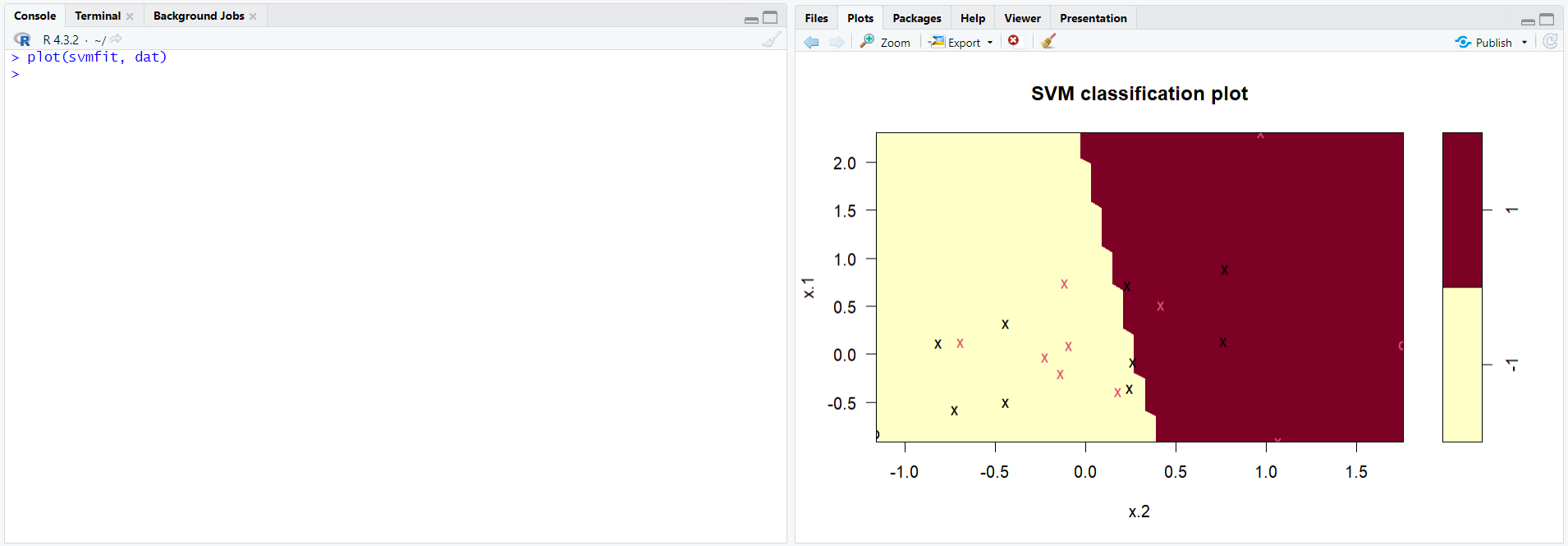


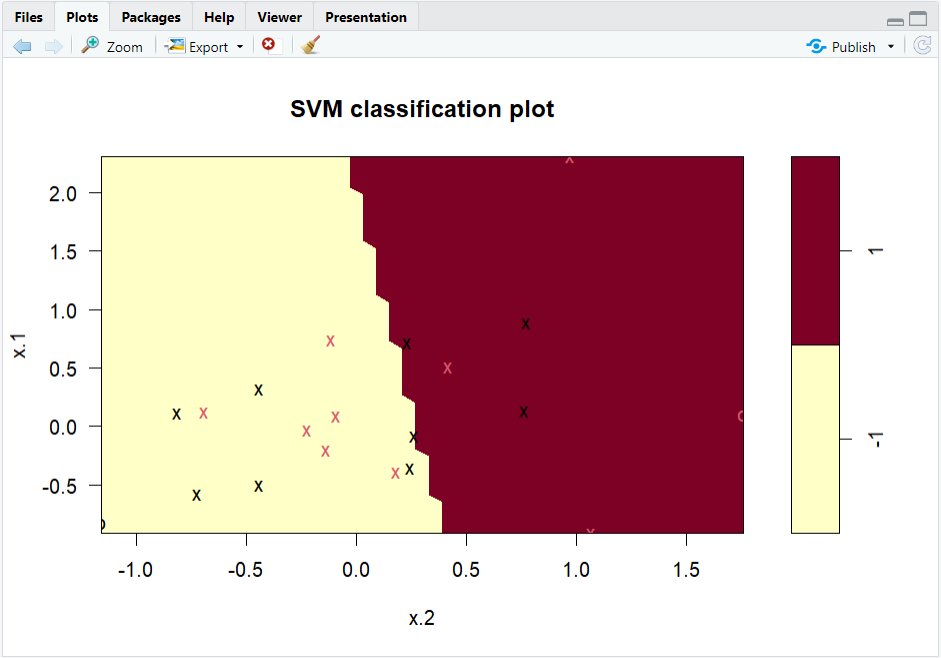


Find line that maximises margin between classes using svm() method and 10-fold cross validation:

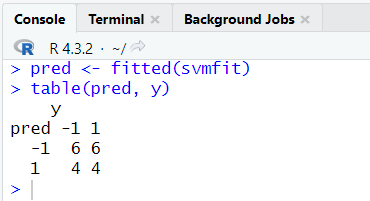


Visualization:



****

Checking accuracy of model:

****

**Applying SVM to Heart Disease Dataset:**

Rstudio code:

install.packages("mlbench")

library(mlbench)

install.packages("e1071")

library(e1071)

file\_path <- "C:\\Users\\admin\\Downloads\\heart.csv"

heart\_data <- read.csv(file\_path)

str(heart\_data)

bc.data <- na.omit(heart\_data)

lin\_svm\_bc <- svm(target ~ ., data = bc.data, kernel = "linear", cross = 10)

lin\_svm\_bc

optimal\_bins <- sapply(heart\_data, function(x) {

if (is.numeric(x)) {

n <- nrow(heart\_data)

range <- diff(range(x))

bin\_width <- 2 \* IQR(x) / (n^(1/3))

num\_bins <- ceiling(range / bin\_width)

return(num\_bins)

} else {

return(0) # Return 0 for non-numeric columns

}

})

print(optimal\_bins)

library(GGally)

library(ggplot2)

heart\_data$target <- as.factor(heart\_data$target)

ggpairs(heart\_data, aes(colour = target, alpha = 0.4), bins = optimal\_bins)

library(ggplot2)

bc.data$target <- as.factor(bc.data$target)

ggplot(data = bc.data, aes(x = age, y = chol, color = target, shape = target)) +

geom\_point(size = 2)

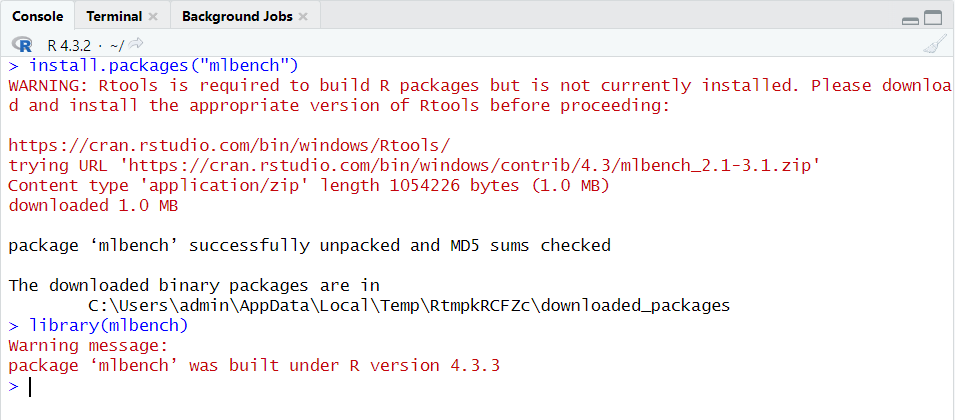
plot(lin\_svm\_bc, data = heart\_data, age ~ chol)

lin\_svm\_bc\_pred <- predict(lin\_svm\_bc, newdata = bc.data)

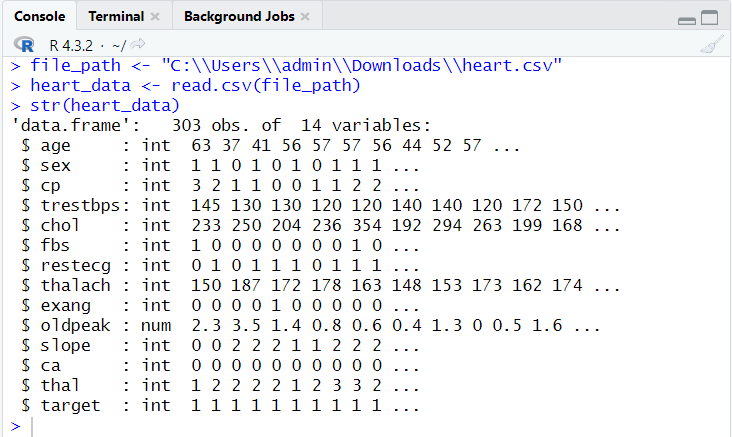
conf\_matrix <- table(Predicted = lin\_svm\_bc\_pred, Actual = bc.data$target)

conf\_matrix

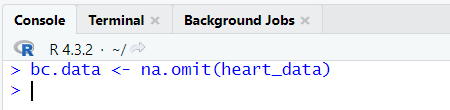
Installing mlbench:



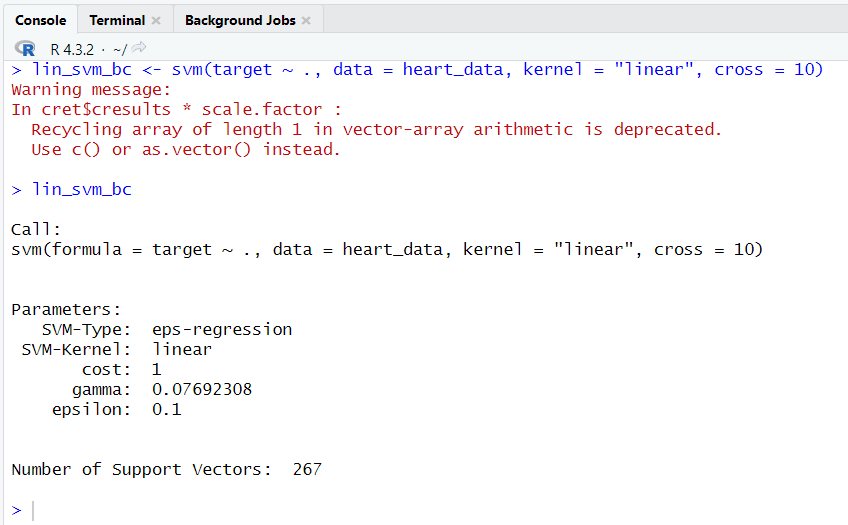
Loading dataset:



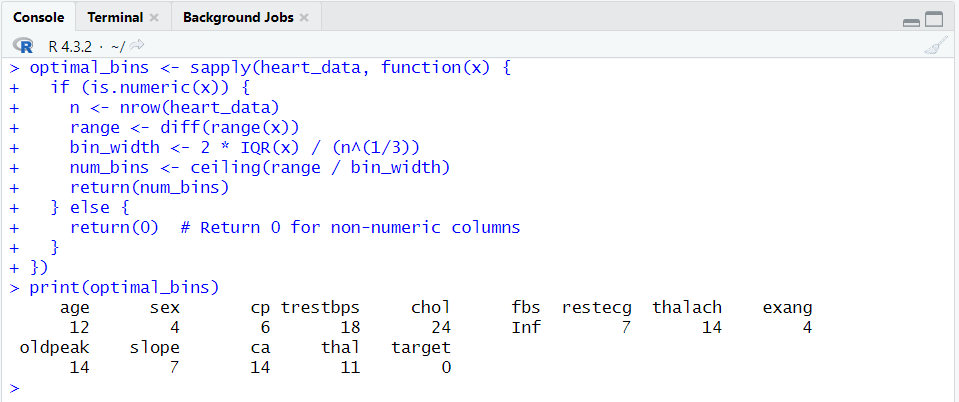
Removing missing or N/A values:

****

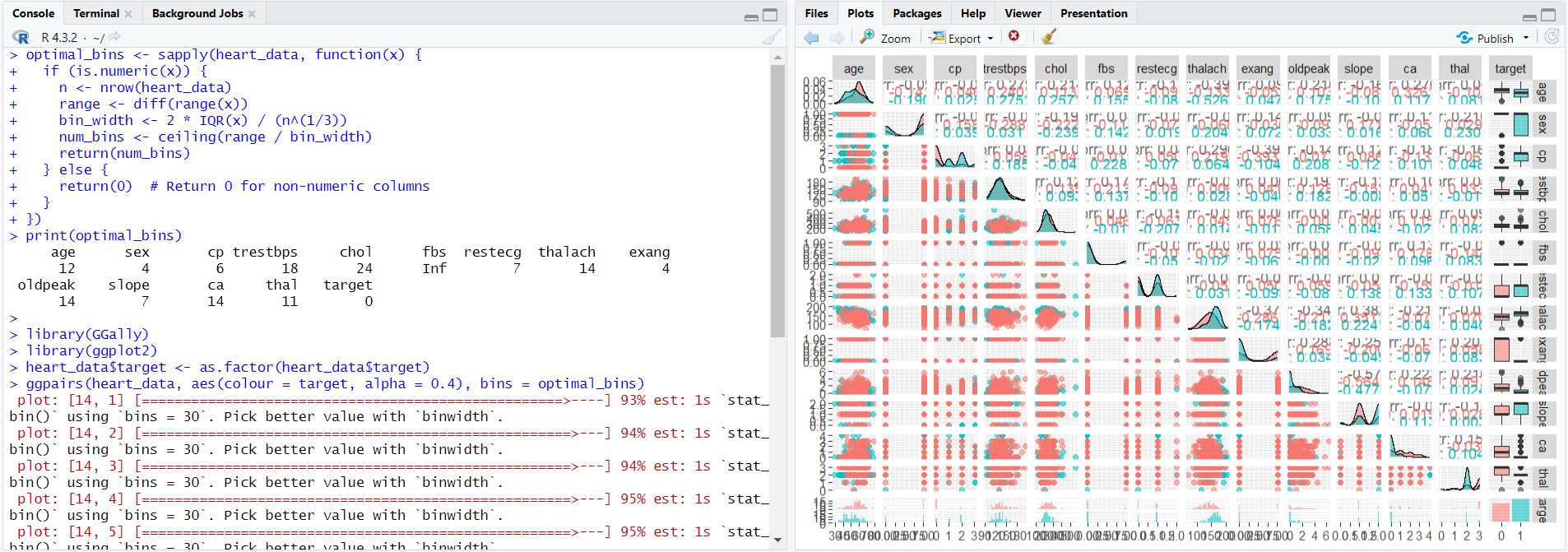
Applying SVM:

****

Creating optimal bins width:

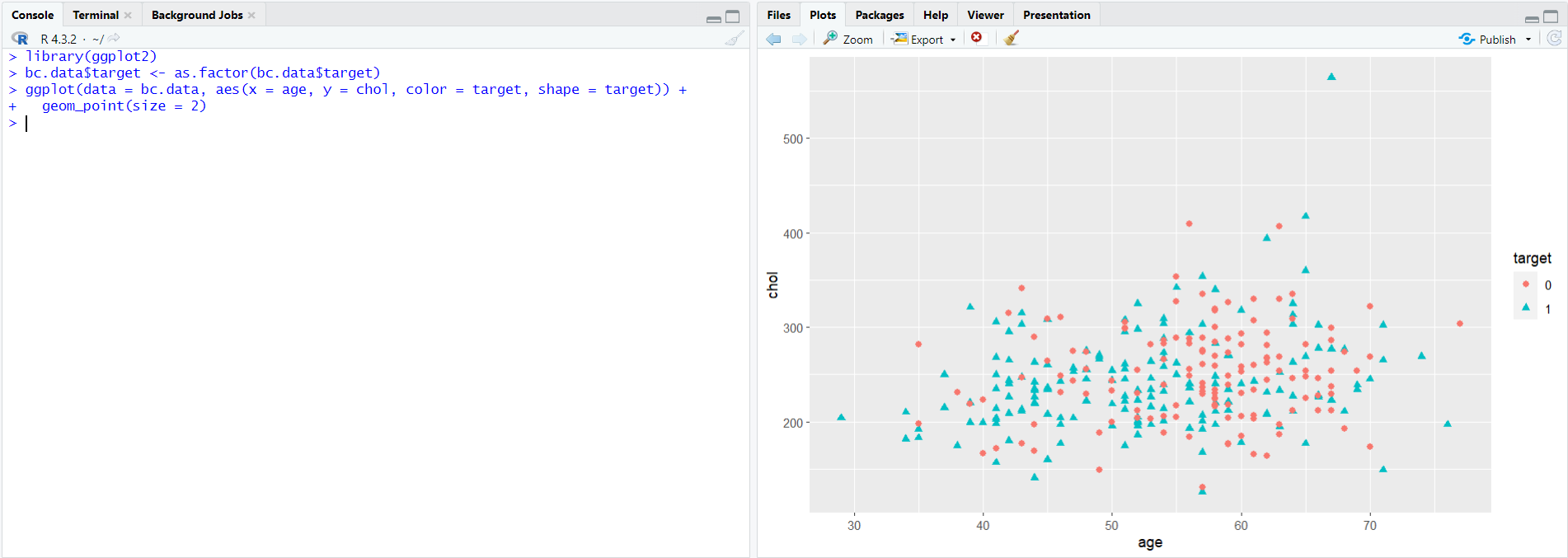
****

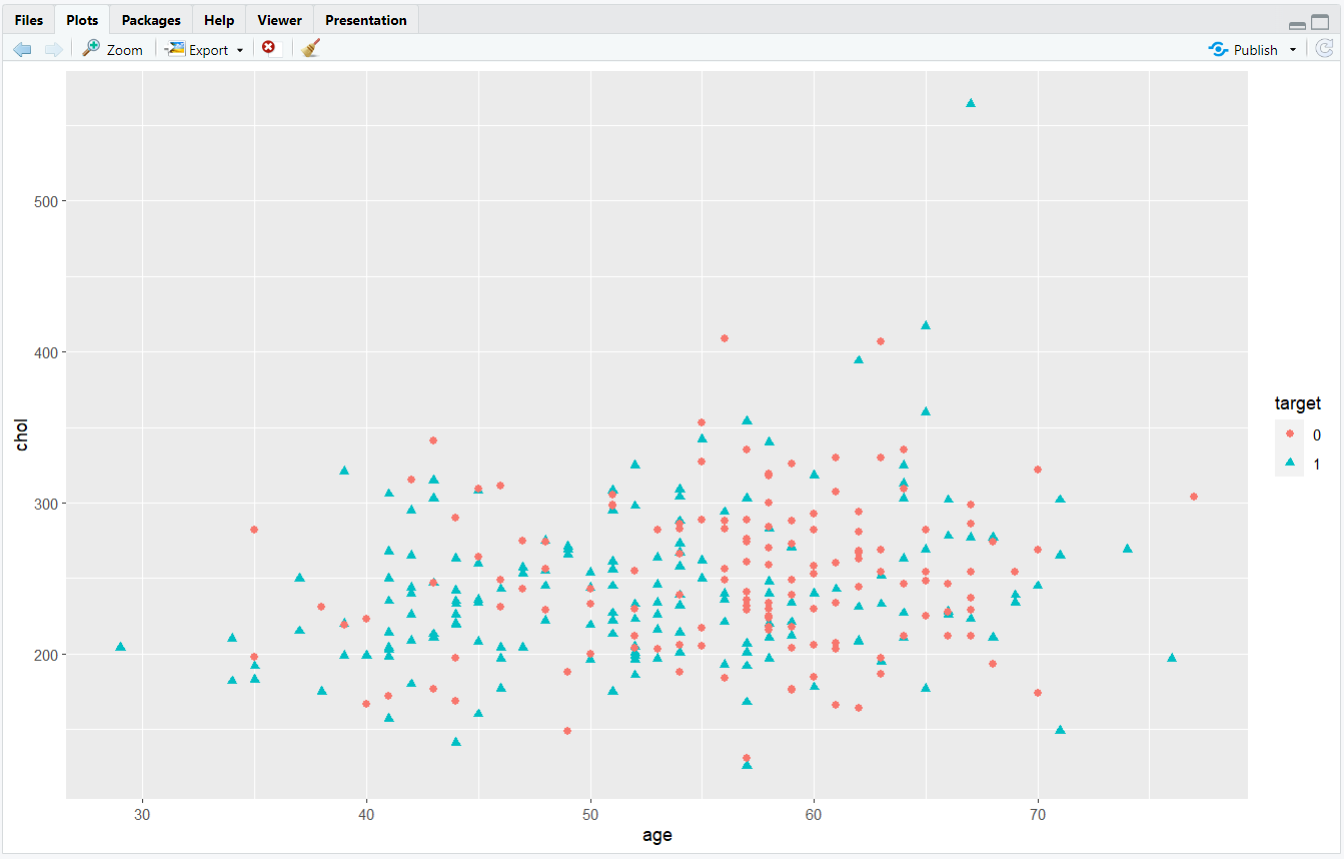
Visualization:



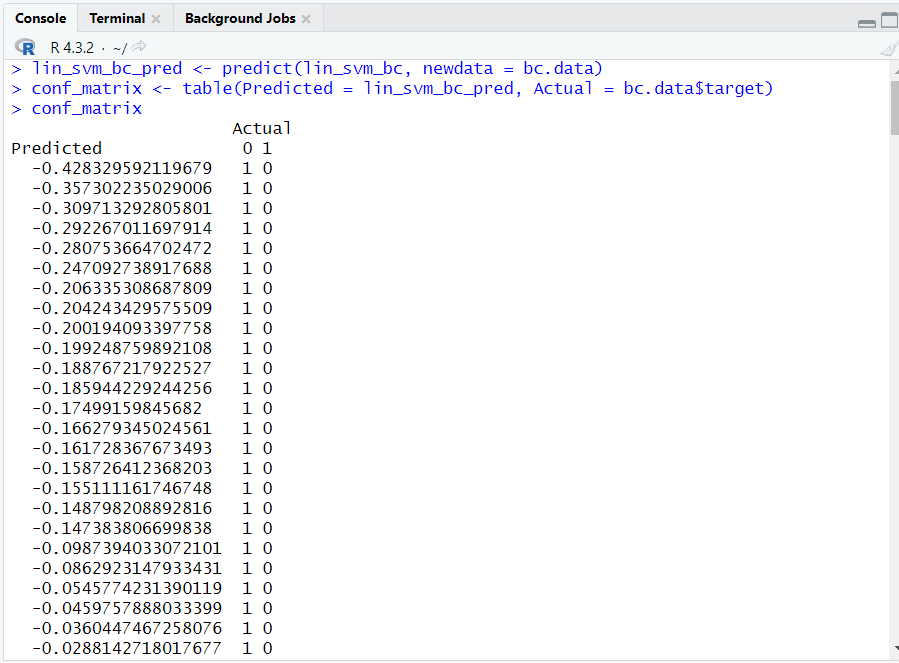


Scatter plot:





Prediction:



**Conclusion:**

**Post lab Questions:**

1. **Evaluating the Impact of Different Kernels:**

**In this experiment, you used a linear kernel for the SVM. Explore non-linear kernels, such as a polynomial kernel or a radial basis function (RBF) kernel. Train the SVM model again using a non-linear kernel of your choice. Evaluate the performance metrics (accuracy, precision, recall, etc.) for both the linear and non-linear models. Explain the observed differences in performance based on the nature of your data and the chosen kernels.**

**Hint: Change the kernel parameter to polynomial / radial basis.**

Rstudio code: (Implementing linear, polynomial and radial kernel to Iris dataset)

# Load necessary libraries

library(e1071) # for SVM model

library(caret) # for performance evaluation

# Load dataset

data(iris)

# Split data into training and testing sets

set.seed(123)

train\_index <- createDataPartition(iris$Species, p = 0.8, list = FALSE)

train\_data <- iris[train\_index, ]

test\_data <- iris[-train\_index, ]

# Train SVM with linear kernel

svm\_linear <- svm(Species ~ ., data = train\_data, kernel = "linear")

# Train SVM with polynomial kernel

svm\_poly <- svm(Species ~ ., data = train\_data, kernel = "polynomial")

# Train SVM with radial basis function (RBF) kernel

svm\_rbf <- svm(Species ~ ., data = train\_data, kernel = "radial")

# Make predictions

linear\_pred <- predict(svm\_linear, test\_data)

poly\_pred <- predict(svm\_poly, test\_data)

rbf\_pred <- predict(svm\_rbf, test\_data)

# Evaluate performance

linear\_metrics <- confusionMatrix(linear\_pred, test\_data$Species)

poly\_metrics <- confusionMatrix(poly\_pred, test\_data$Species)

rbf\_metrics <- confusionMatrix(rbf\_pred, test\_data$Species)

# Print performance metrics

print("Linear Kernel:")

print(linear\_metrics)

print("Polynomial Kernel:")

print(poly\_metrics)

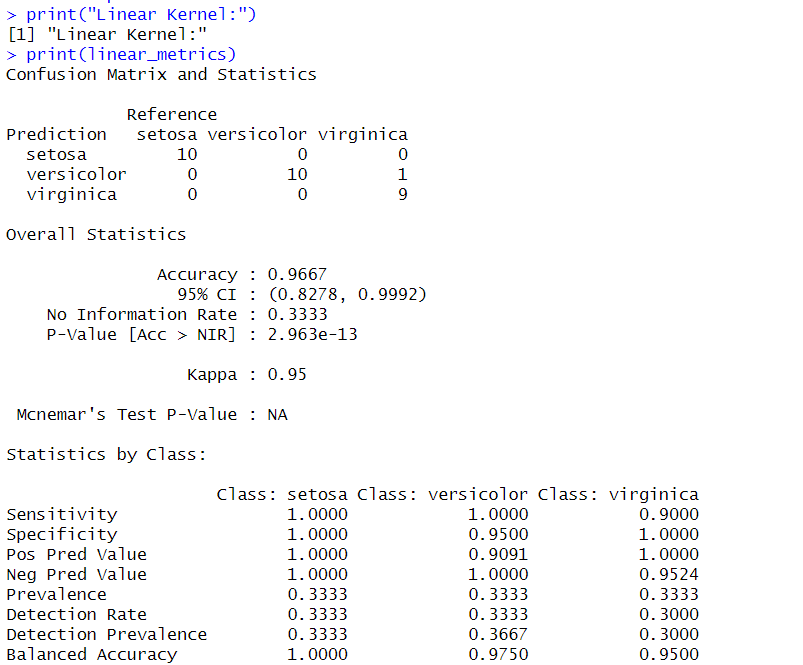
print("RBF Kernel:")

print(rbf\_metrics)

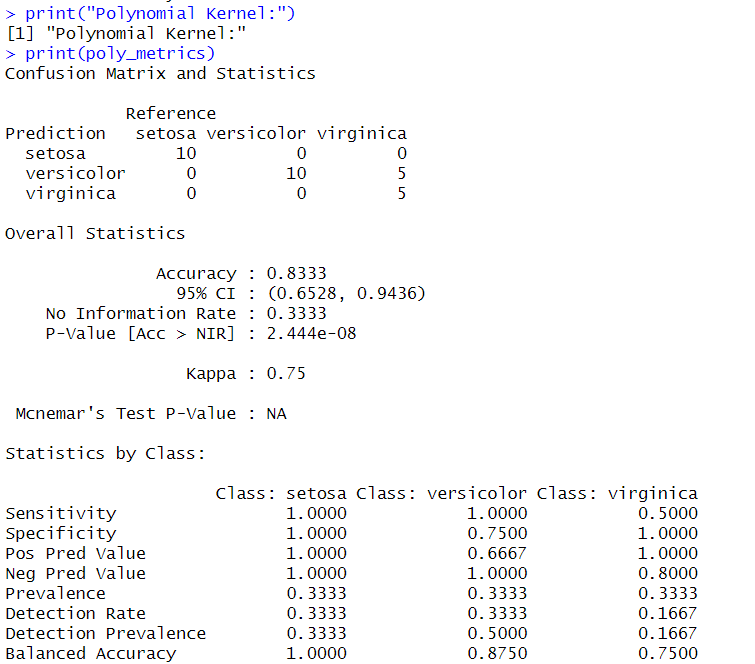
The code displays the following:

* Confusion Matrix: Shows the counts of true positive, true negative, false positive, and false negative predictions for each class.
* Overall Statistics: Includes accuracy, 95% confidence interval, no information rate, kappa statistic, and p-value.
* Statistics by Class: Provides sensitivity, specificity, positive predictive value, negative predictive value, prevalence, detection rate, detection prevalence, and balanced accuracy for each class.

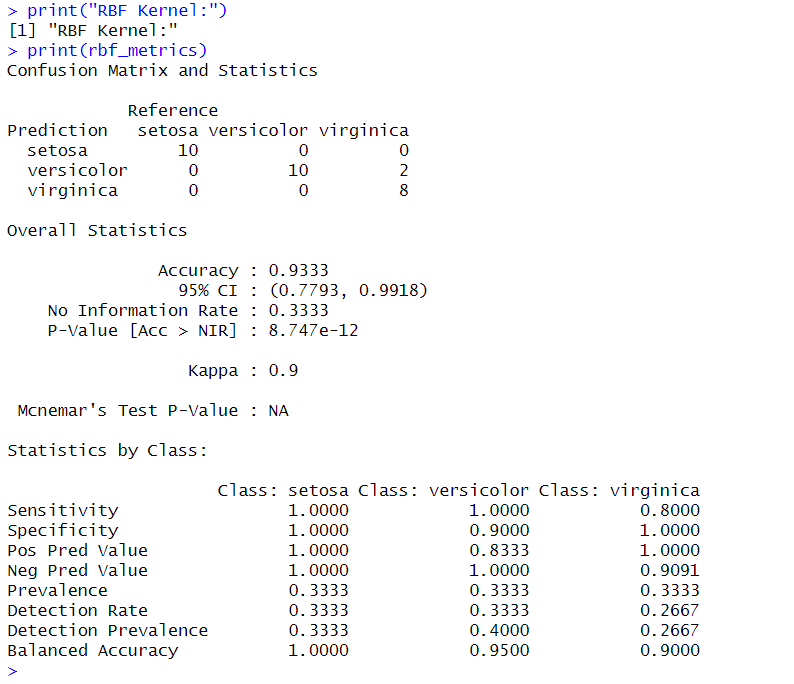
Linear Kernel:



Polynomial Kernel:



RBF Kernel:



Output explanation:

* Accuracy: Overall, the linear kernel achieved the highest accuracy (96.67%), followed by the RBF kernel (93.33%) and the polynomial kernel (83.33%).
* Sensitivity/Recall: Linear kernel and RBF kernel generally performed better in correctly identifying instances of each class compared to the polynomial kernel.
* Specificity: Linear kernel generally had higher specificity for each class, indicating better performance in correctly identifying true negatives.
* Precision: Precision varies across classes and kernel types. Linear kernel generally has higher precision values for each class compared to polynomial and RBF kernels.
* Explanation: Linear kernel performed best likely because the iris dataset may have a linearly separable structure, while polynomial and RBF kernels may overfit due to their higher complexity for this particular dataset. However, for more complex datasets, the non-linear kernels might perform better in capturing the underlying patterns.

1. **Explain underfitting and overfitting in machine learning. Did you observe any of these when you performed the experiment? How do you handle these issues?**

Underfitting and overfitting are two common problems encountered in machine learning models, particularly when dealing with complex datasets. They refer to the model's performance relative to the training data and the ability of the model to generalize to unseen data.

Underfitting:

* Underfitting occurs when the model is too simple to capture the underlying structure of the data.
* The model fails to learn the patterns in the training data and performs poorly on both the training and test datasets.
* In other words, the model has high bias and low variance.
* Signs of underfitting include high training and test errors.
* Underfitting can be addressed by increasing the model complexity, adding more features, or using a more sophisticated algorithm.

Overfitting:

* Overfitting occurs when the model learns the noise and random fluctuations in the training data rather than the underlying patterns.
* The model performs well on the training data but poorly on unseen data because it has essentially memorized the training data.
* Overfitting results in low bias and high variance.
* Signs of overfitting include low training error but high-test error.
* Overfitting can be addressed by reducing the model complexity, regularizing the model, using more training data, or applying techniques such as cross-validation.

To handle these issues:

* Underfitting: Increase the model complexity by adding more features, using a more sophisticated algorithm, or tuning hyperparameters to allow the model to capture more intricate patterns in the data.
* Overfitting: Reduce the model complexity by simplifying the model, regularizing it (e.g., using techniques like L1 or L2 regularization), using more training data, or applying techniques such as cross-validation to evaluate the model's performance on unseen data.