**Batch: H\_ADS (H2-3)**

**Roll No.: 16010122221**

**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.
   1. Data set used:
   2. Title:
   3. Source:
   4. Number of instances:
   5. Number of attributes:
   6. Attribute information **:**
2. Handle the missing values appropriately
3. Create the SVM model
4. Perform Visualization
5. Calculate prediction accuracy.

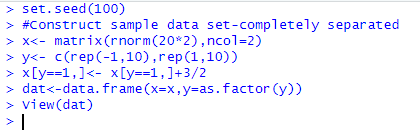
**What are SVMs?**

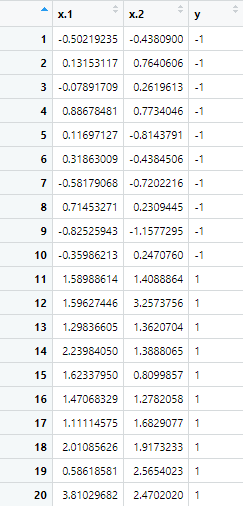
A support vector machine (SVM) is a supervised machine learning algorithm that classifies data by finding an optimal line or hyperplane that maximizes the distance between each class in an N-dimensional space.

**Maximal Margin Classifier**

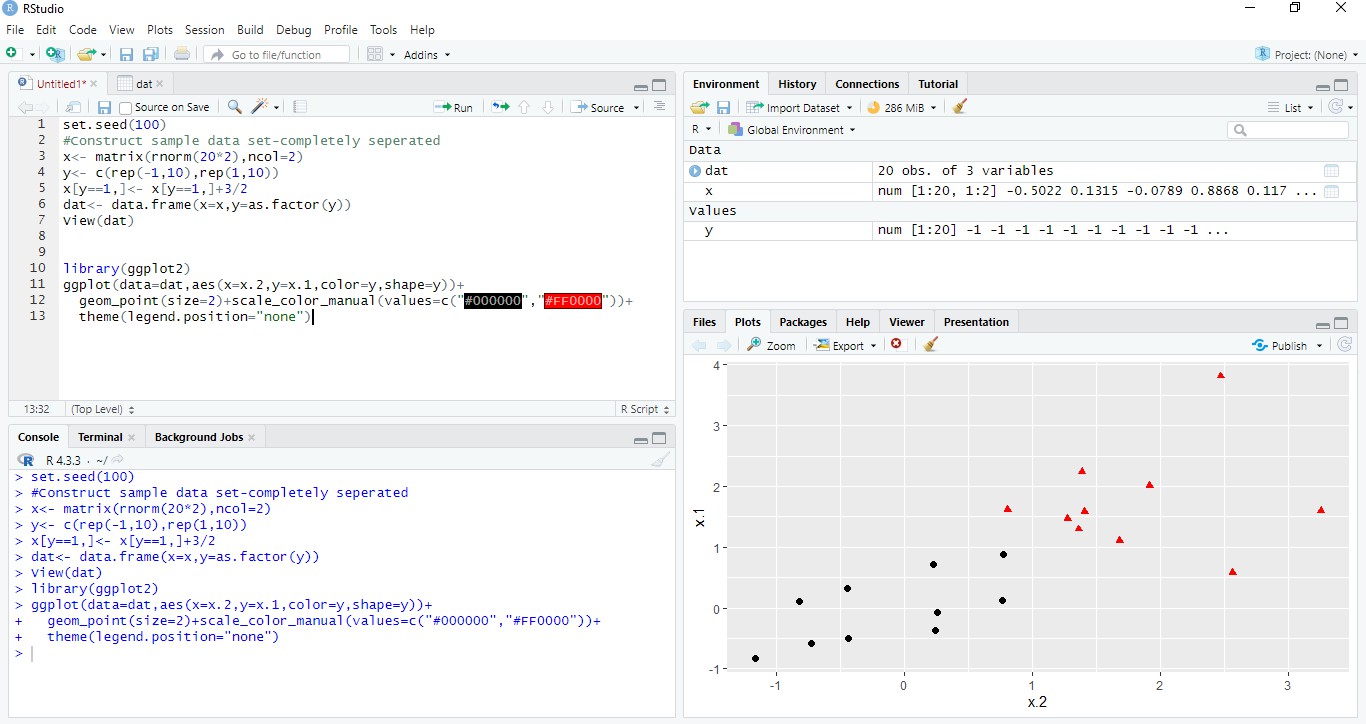
There are several types of SVM’s. The simplest is the\_maximal margin classifier (MMC). Though the MMC is elegant and simple, it cannot be applied to most data sets, since it requires the classes to be separable by a linear boundary.

To visualize an example of separated data, we will generate 40 random data points and assign them to two classes.

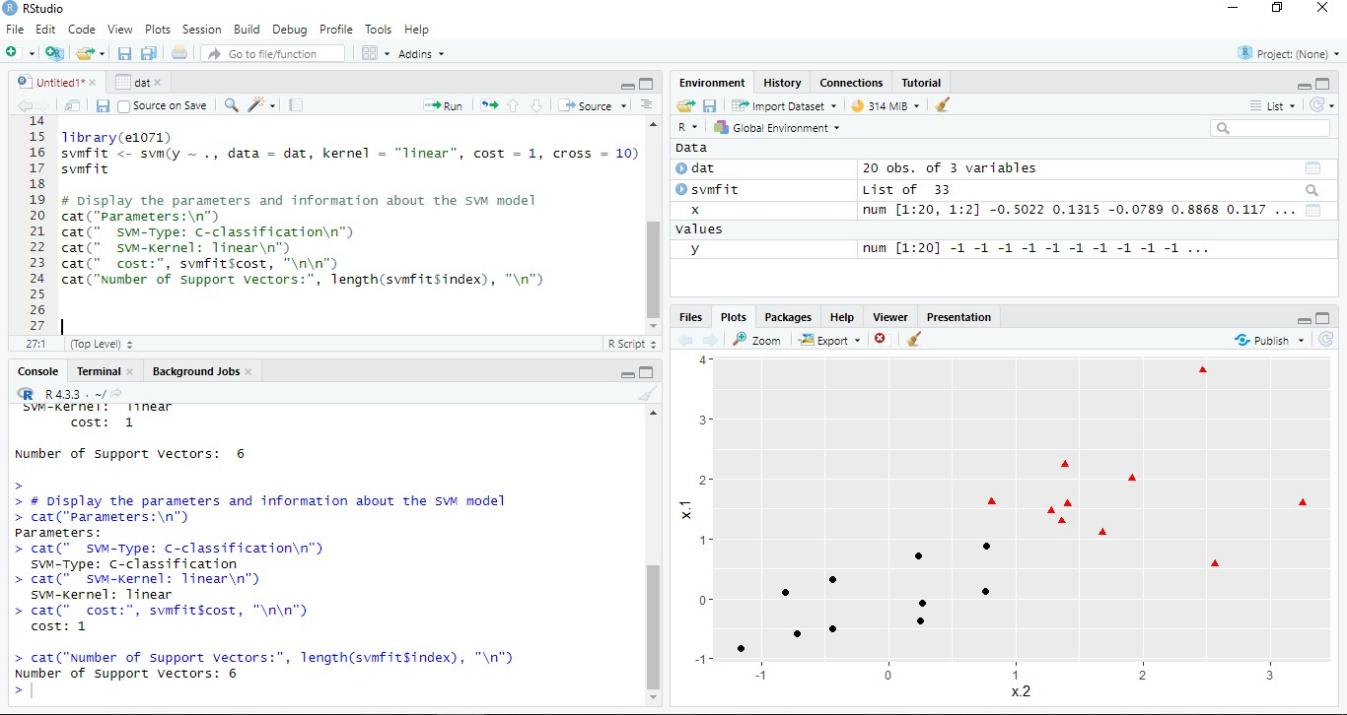




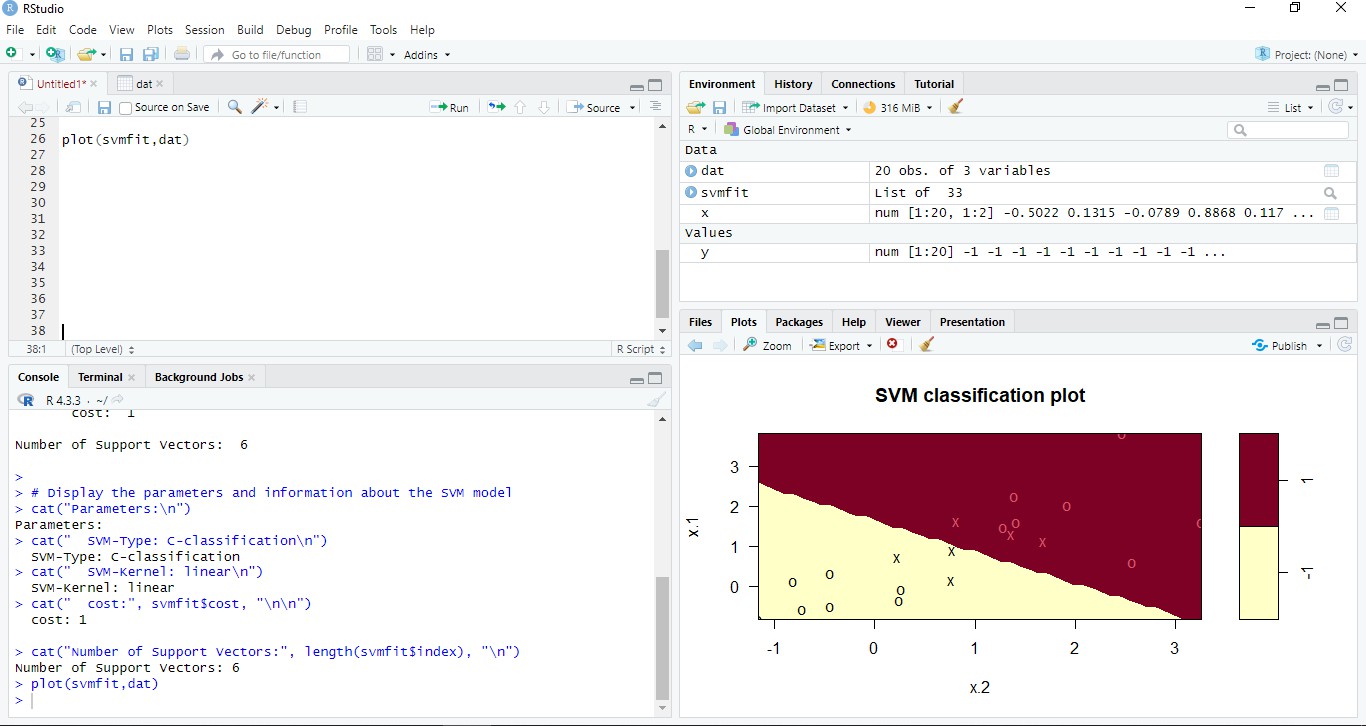
**Now we will plot this data**

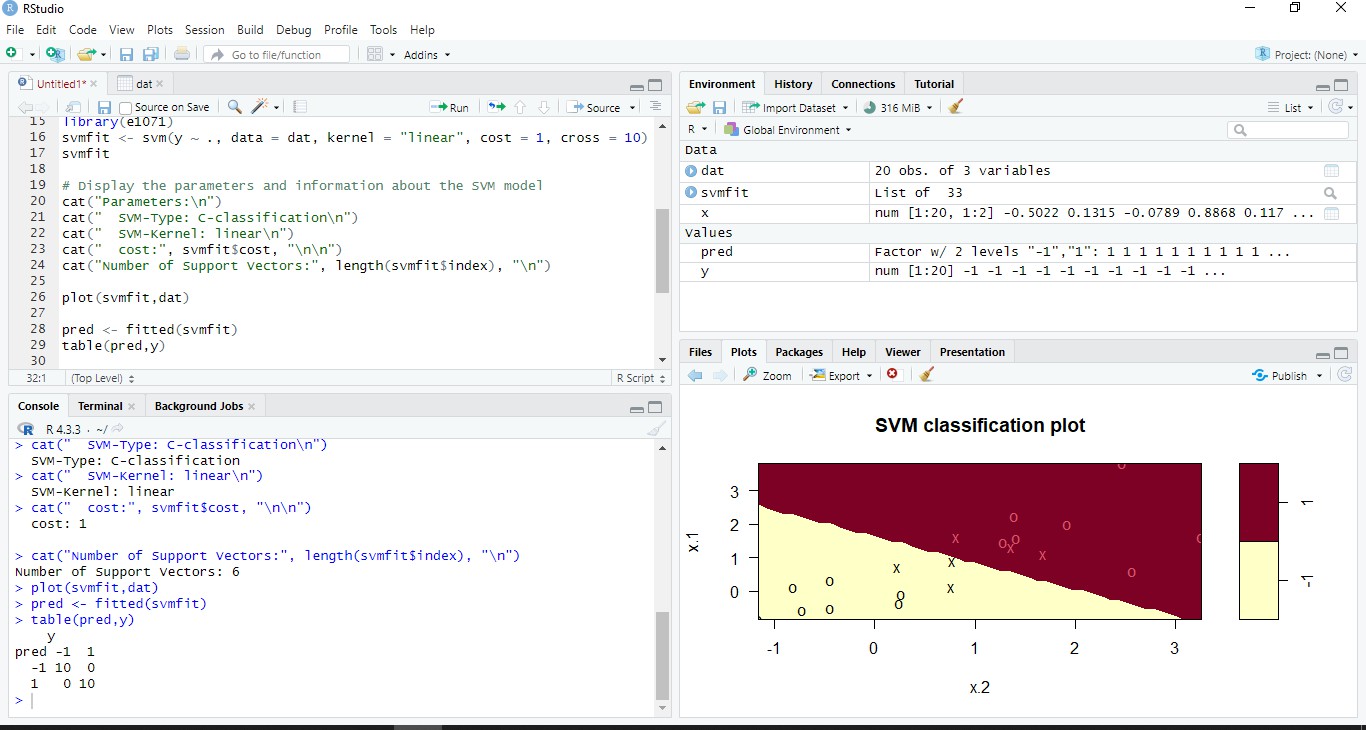


**Upon visual inspection, we can see that infinitely many lines exist that split the two classes. To find the line that maximizes the margin between the classes, we use the svm() method from the e1071 library. We will use 10-fold cross validation.**



**Visualization:**

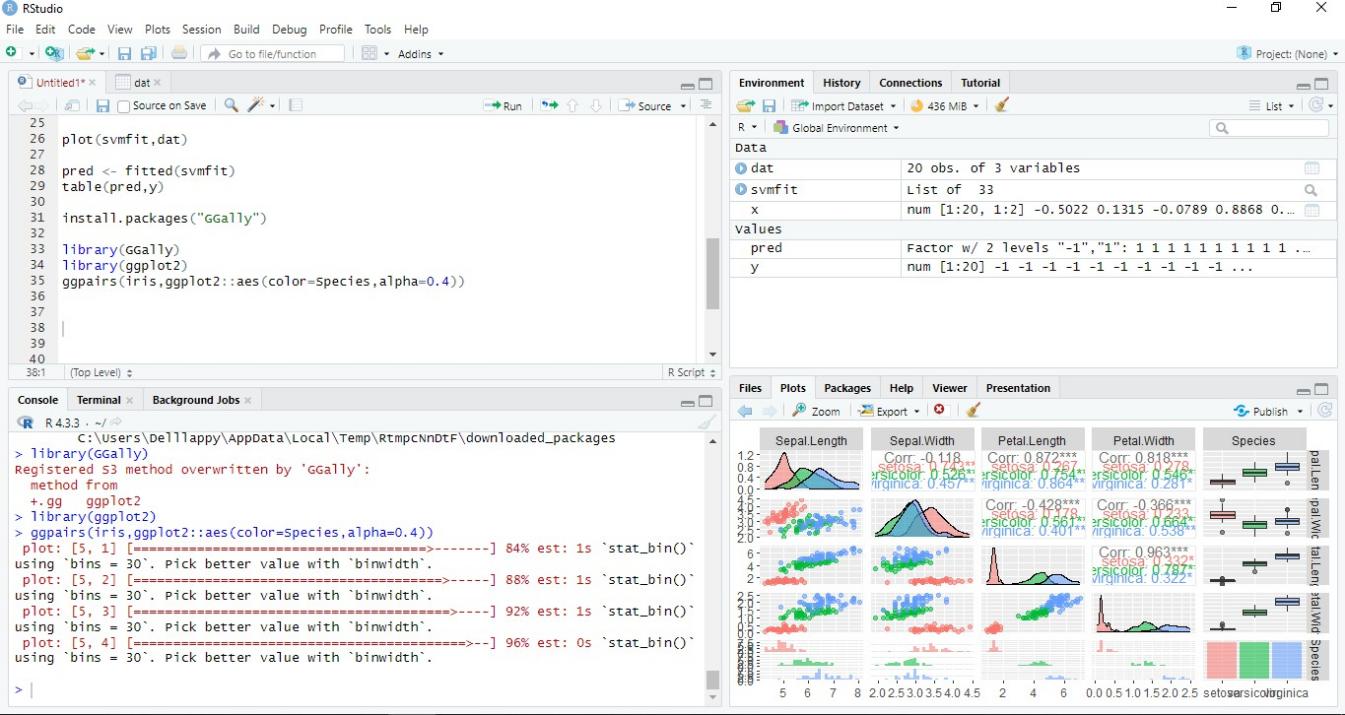
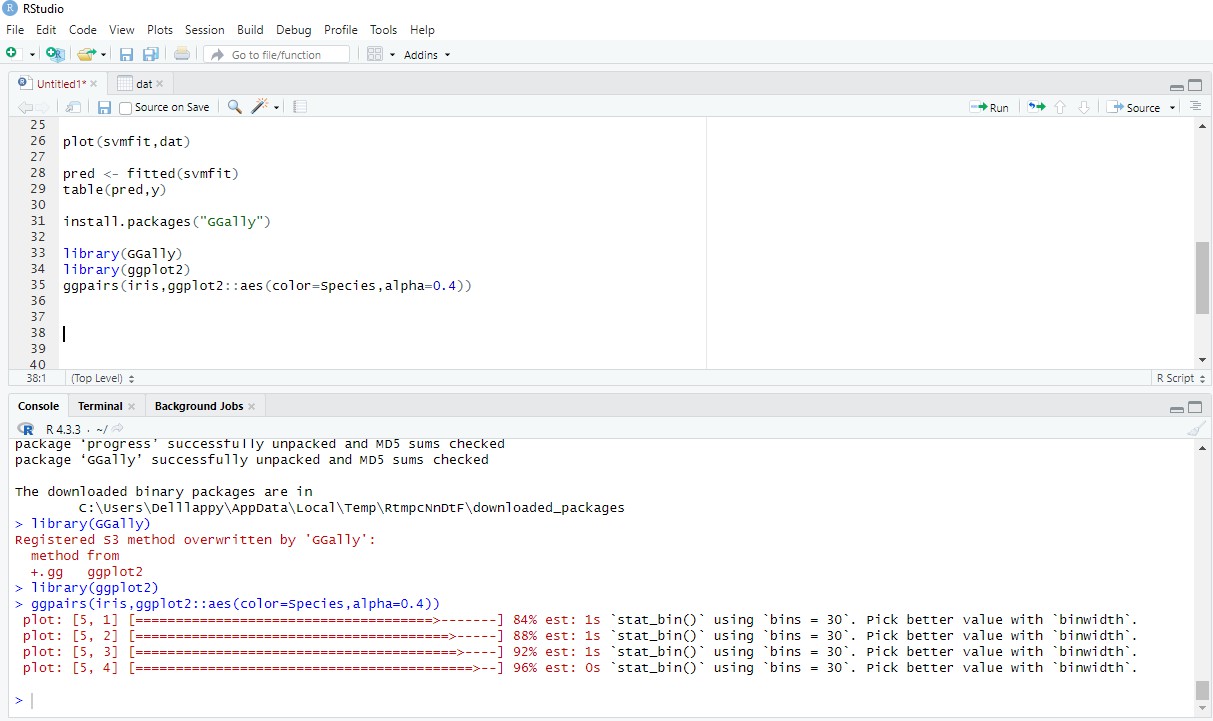


**Finding the accuracy of the model:**

**Accuracy = 100% (dataset is too small)**

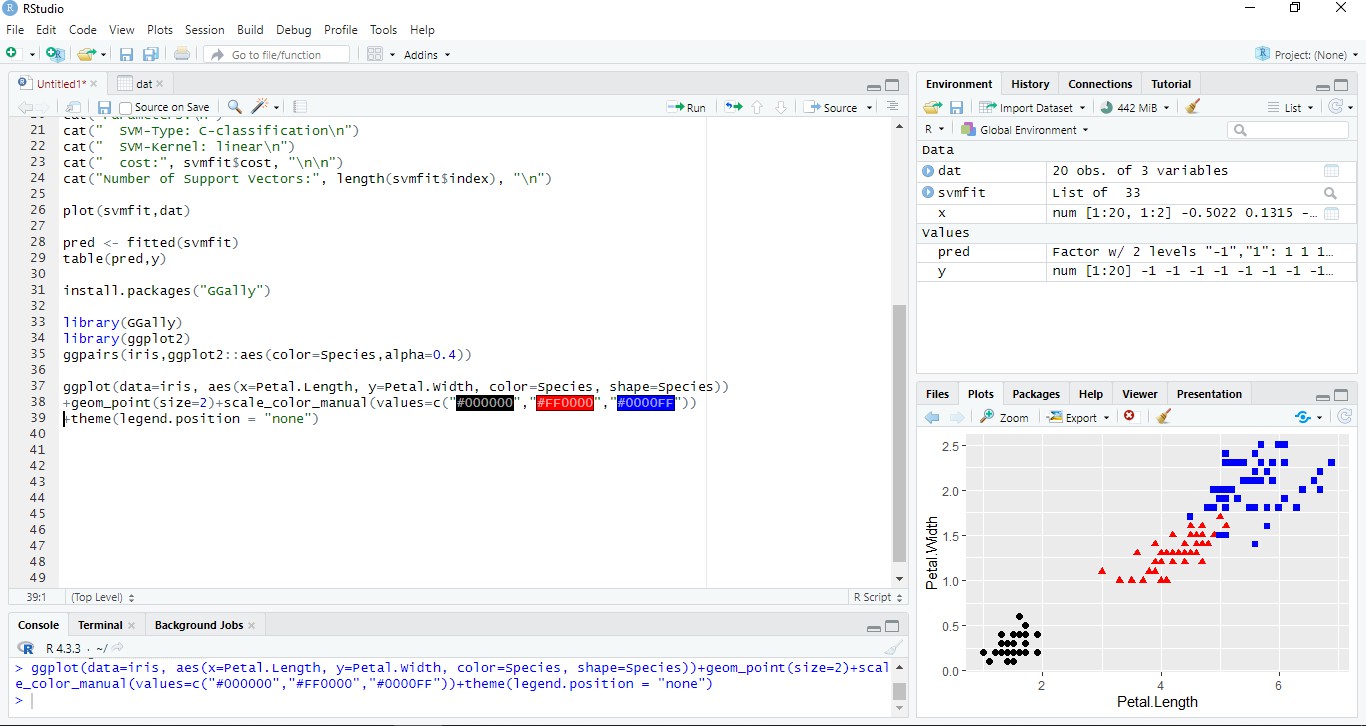
**Applying SVM to Iris Dataset (4 predictors, 3 classes)**

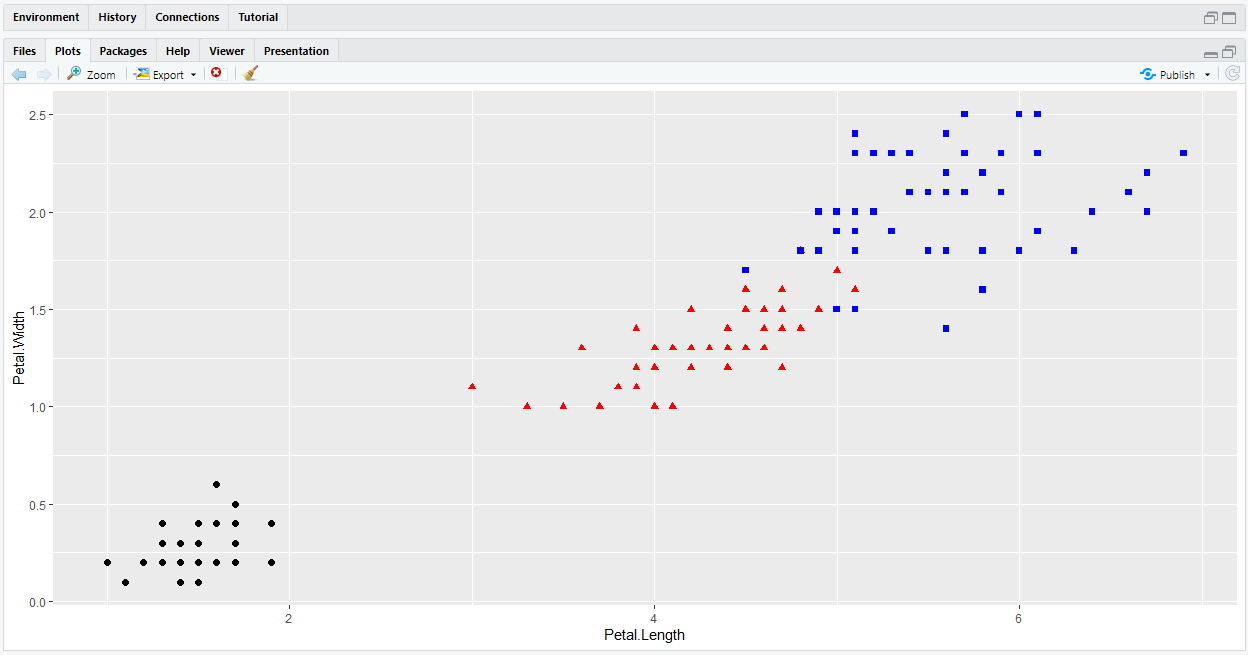
Since there are 4 predictors, it's not possible to have a simple 2D visualization. Hence we will select only 2 features, just for the sake of visualization. We will use all the four features for training the SVM model.

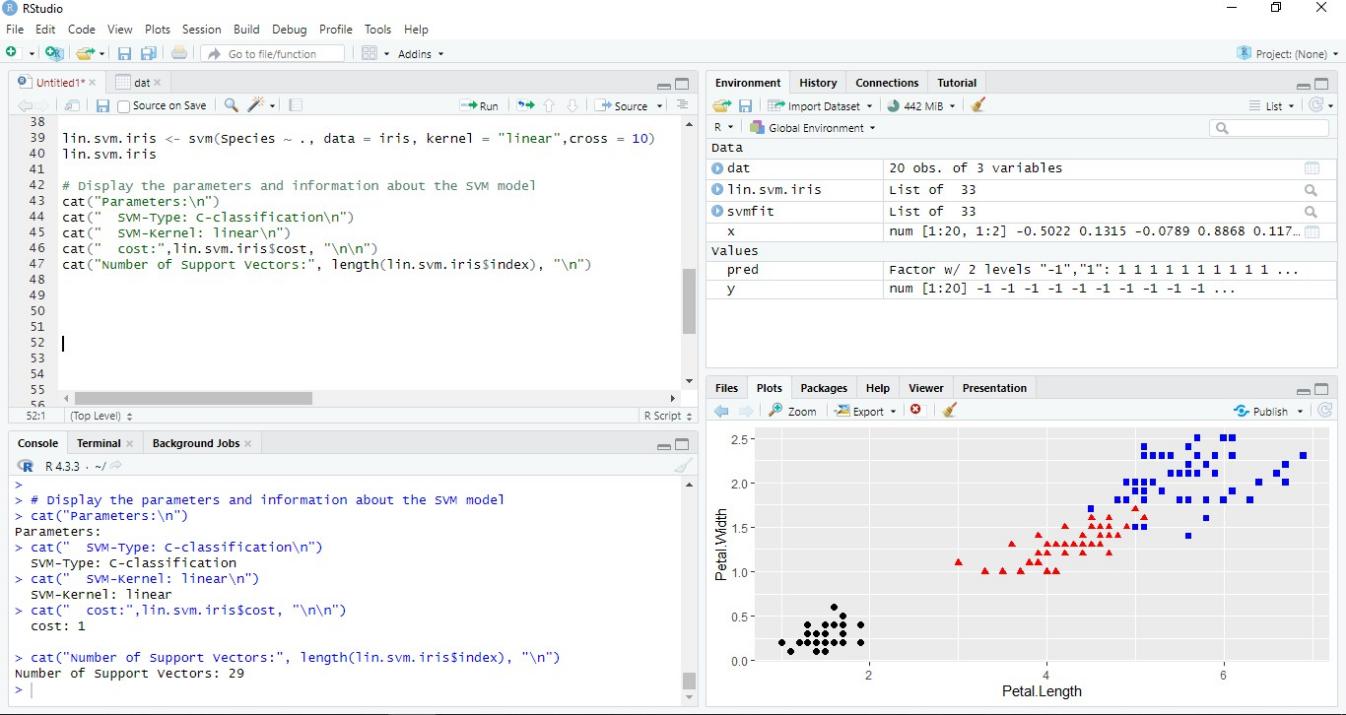




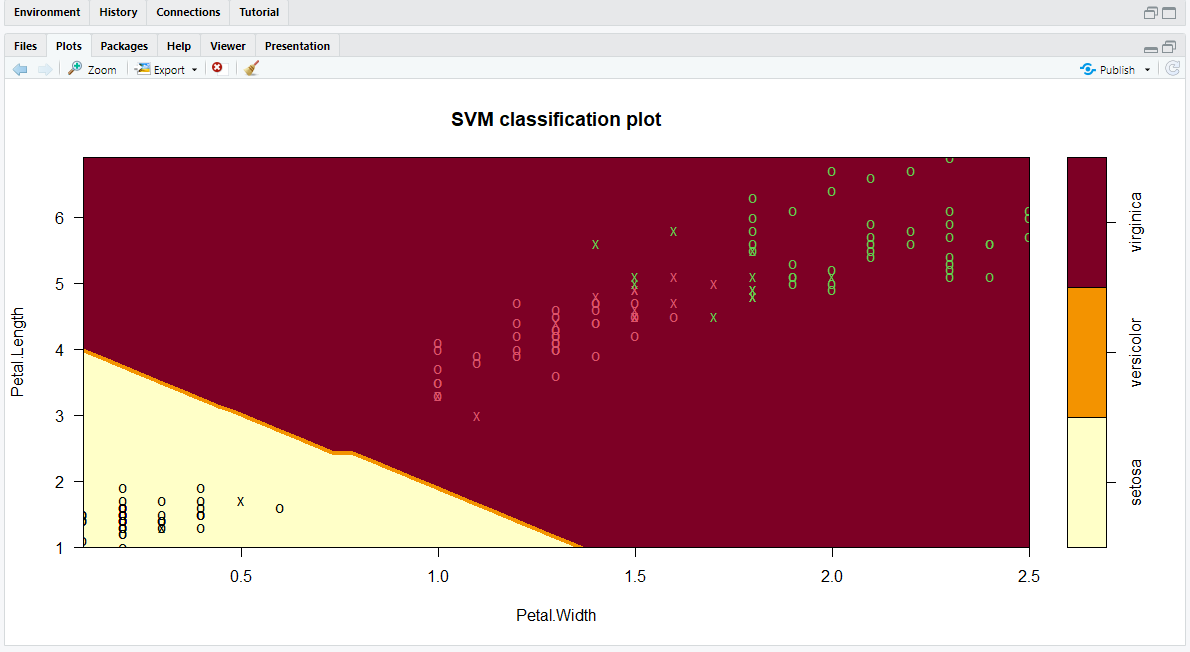
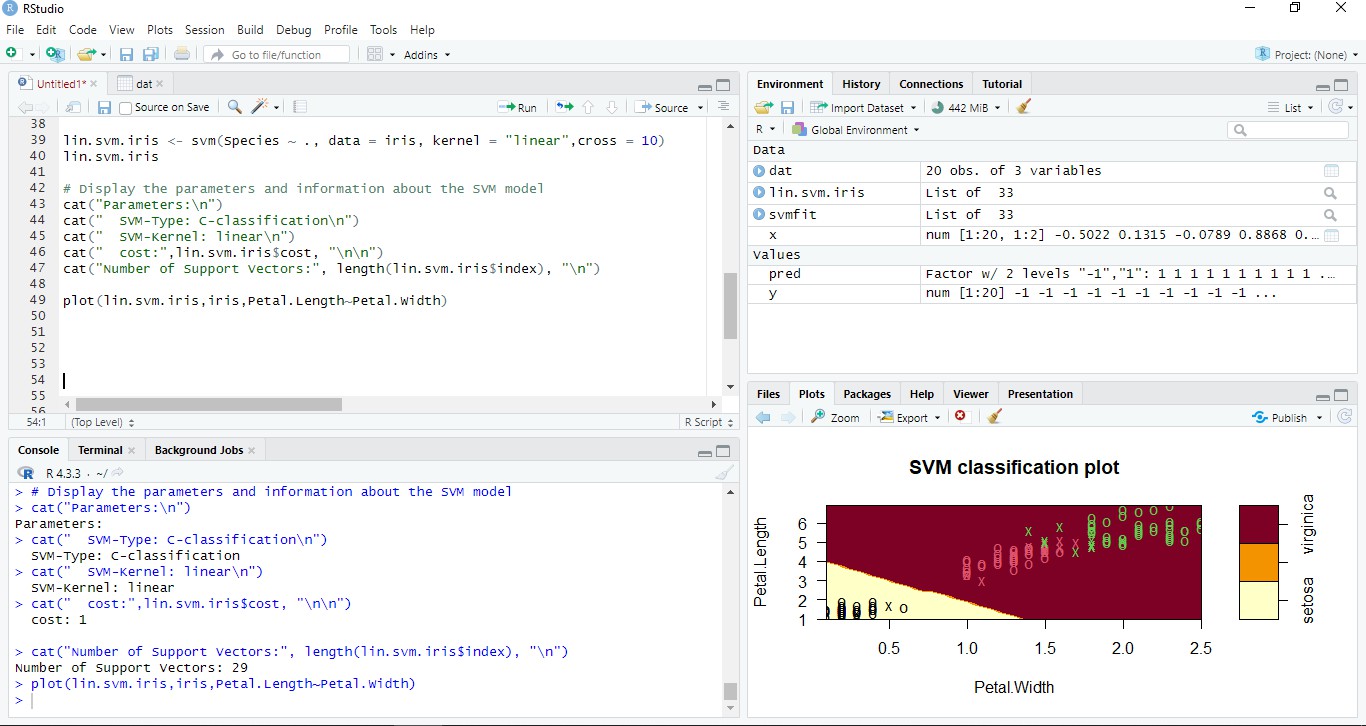
Clearly from the Histograms of Petal.length and Petal.width that we can clearly separate out Setosa species with very high confidence. However, Versicolor and Virginica Species are overlapped. If we look at the scatterplot of Petal.Width vs Petal.Length, we can distinctly see a separator that can be drawn between the groups of Species. Looks like we can just use Petal.Width and Petal.Length as parameters for visualization.

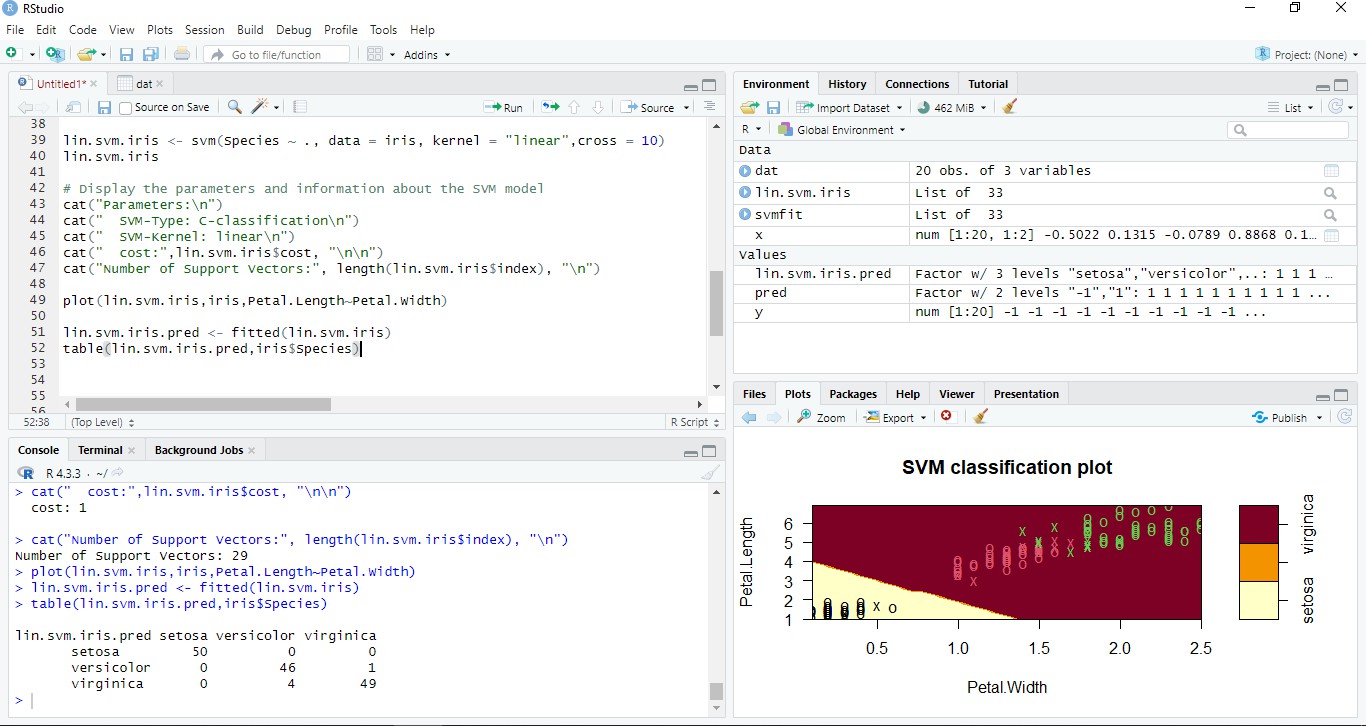




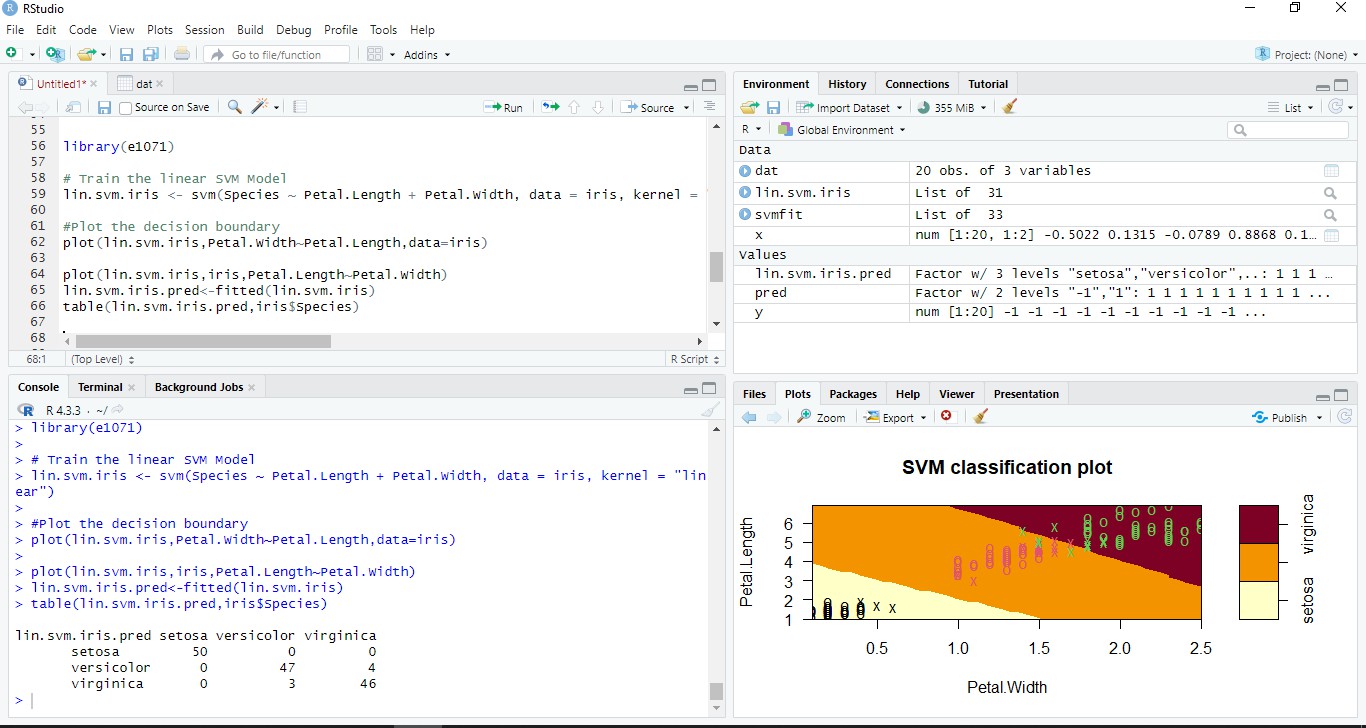


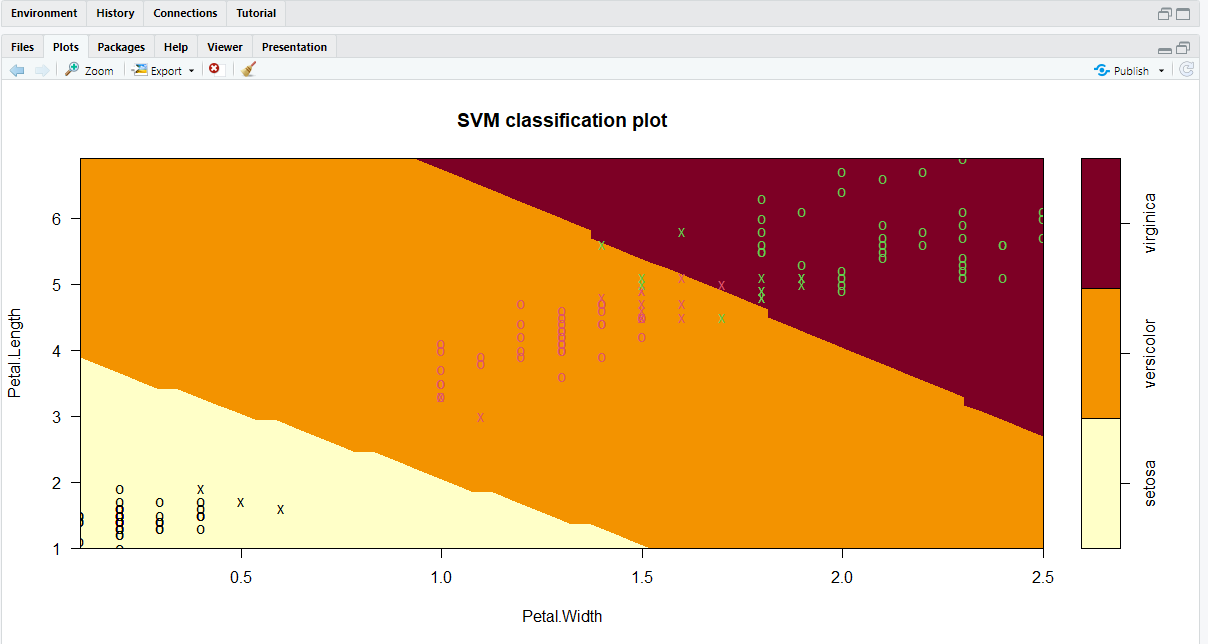
**Visualization:**



**Note: The hyperplane here will be 3-D (n predictors = (n-1) D hyperplane) Finding the accuracy of the model:**

**Accuracy = (50+46+49)/150 = 0.967**





**Conclusion:**

In this experiment, our aim was to implement classification using Support Vector Machines (SVM) utilizing R libraries. We successfully trained SVM models on our dataset, leveraging the capabilities of R libraries such as 'e1071' and 'svm'. By fine-tuning model parameters and exploring different kernel functions, we gained insights into SVM's effectiveness in classifying data and its versatility in handling various classification tasks. This experiment underscores the practical utility of SVM algorithms and the robustness of R libraries for machine learning applications.

**Post lab Questions:**

**Q1. 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.

**Q2. 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 occurs when a model is too simple to capture the underlying patterns in the data, resulting in poor performance on both training and test datasets. Overfitting happens when a model is excessively complex, capturing noise instead of true patterns, leading to excellent performance on the training set but poor generalization to unseen data. Regularization techniques like L1/L2 regularization, cross-validation, and using simpler models can help mitigate these issues.