**Support Vector Machine Classiferimplementation in R with caret**

**for SVM classier implementation in R programming language using caretpackage, we are going to examine a tidy dataset of customer purches.**

Caret Package Installation: install.packages(“caret”)

caret package provides us direct access to various functions for training ourmodel with various [machine learning algorithms](https://dataaspirant.com/2016/09/24/classification-clustering-alogrithms/) like Knn, SVM, decision tree,[linear regression](https://dataaspirant.com/2014/10/02/linear-regression/), etc

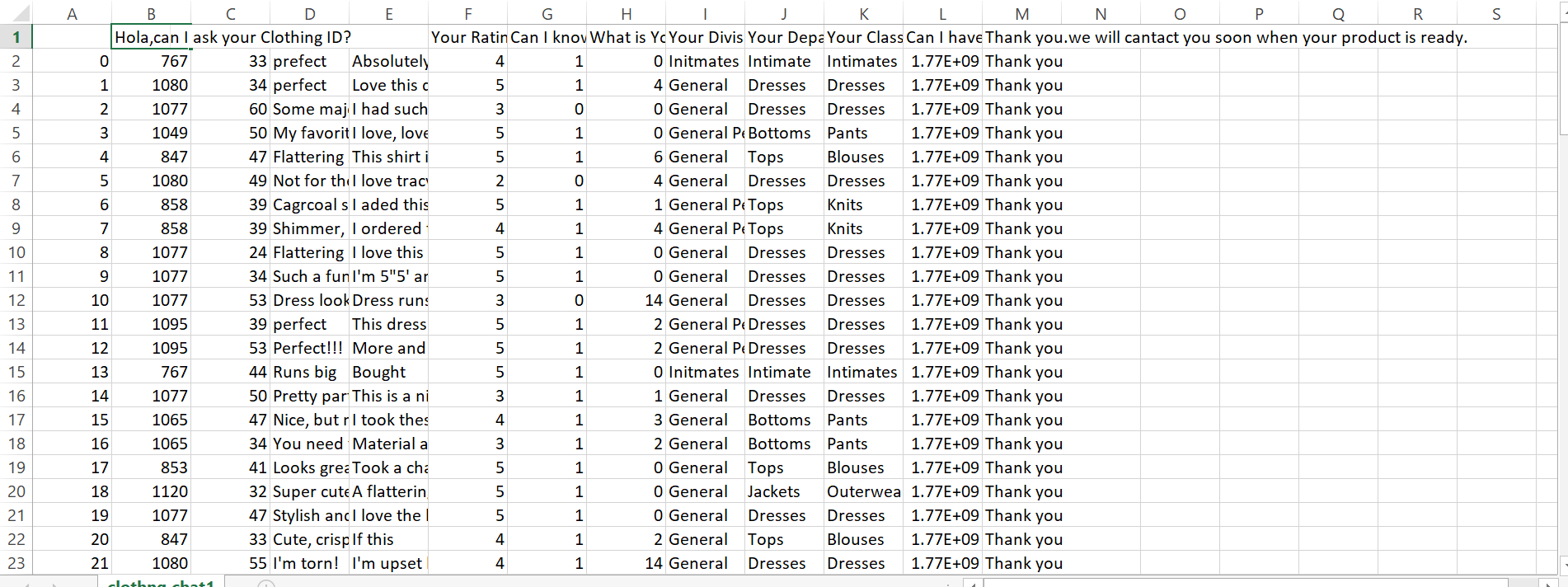
customer puches Recognition Data Set Description:data contain 107 attribute and 7 vaiable.

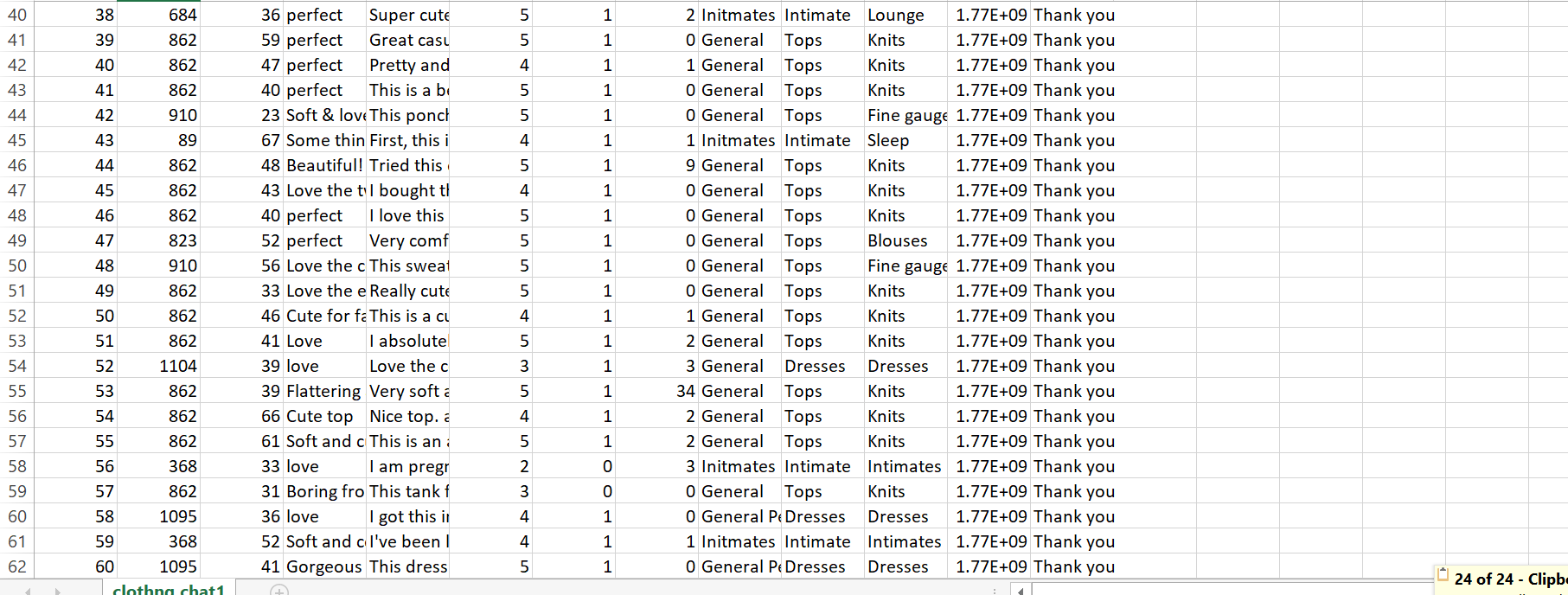
he target variable is at index 3.

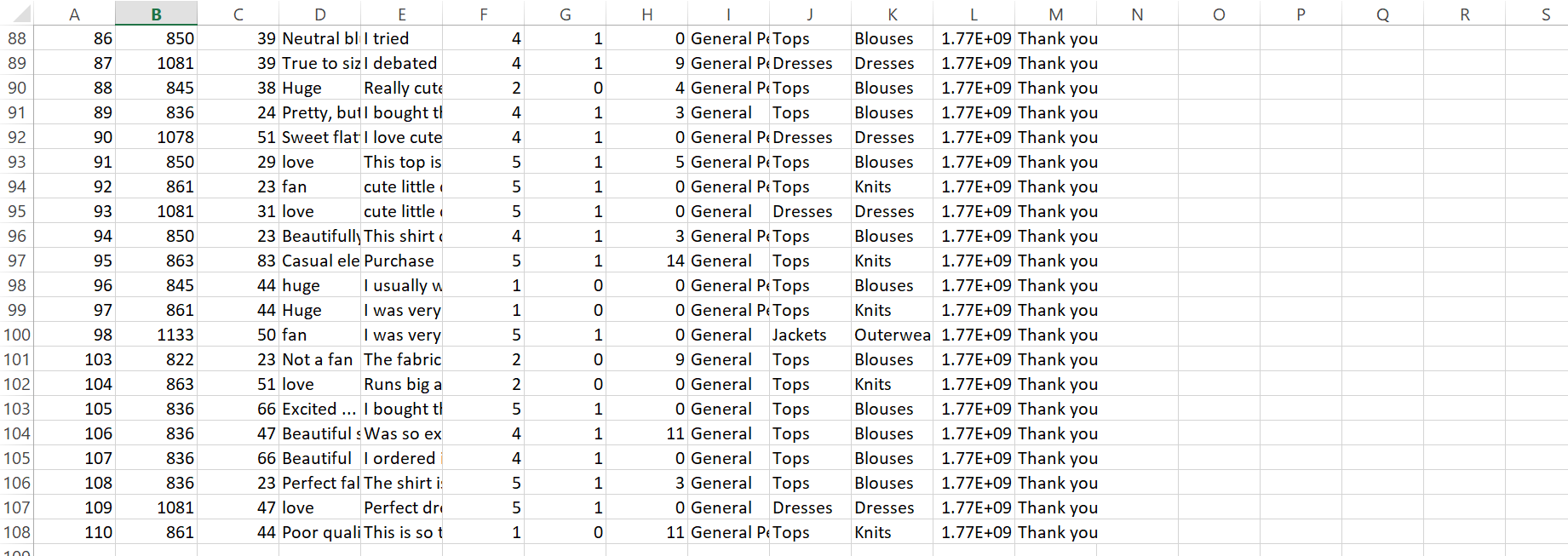
**R caret Library:**

[**For implementing SVM in r, we only need to import caret package. As we**](http://void(0);) **mentioned above, it helps to perform various tasks to perform our machinelearning work. Just past the below command in R console to import r machinelearning package Caret.**

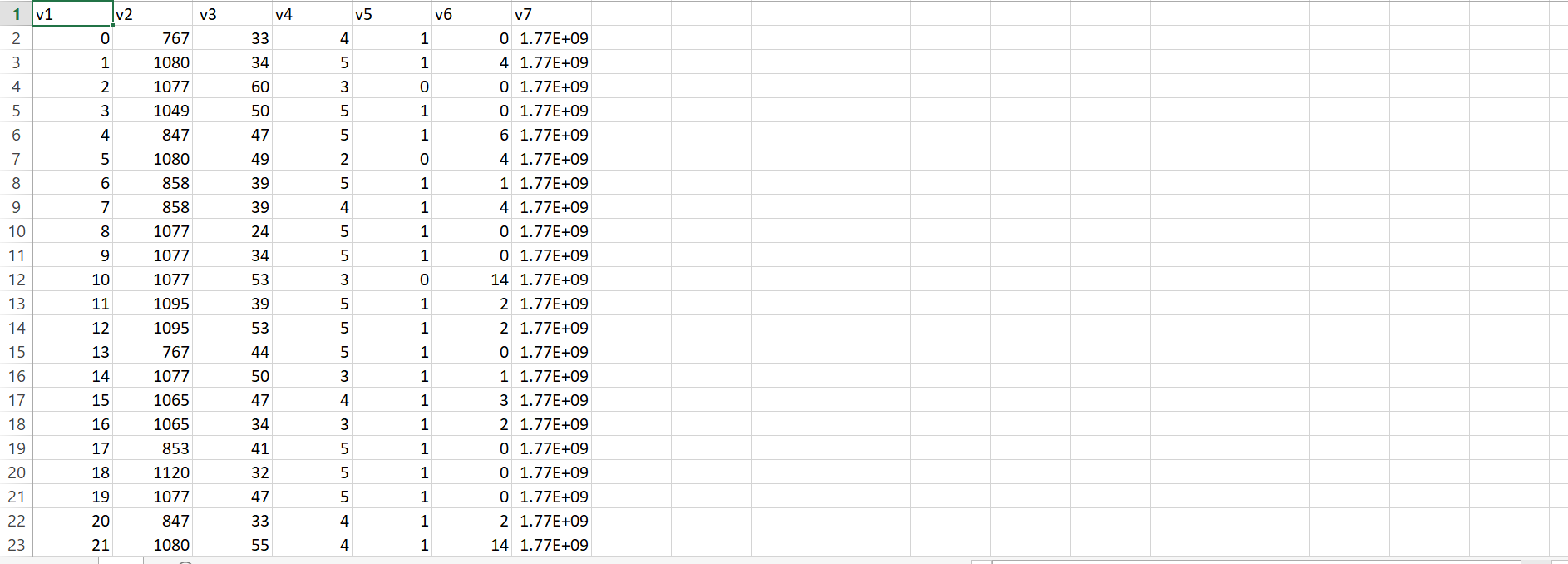
1. library(caret)
2. **Data Import**
3. **For importing the data and manipulating it, we are going to use data frames.First of all, we need to download the dataset. You can download the datasetour repository. It’s a CSV le i.e, Comma Separated Values le. All the datavalues are separated by commas. After downloading the CSV le, you needto set your working directory via console else save the data le in yourcurrent working director**

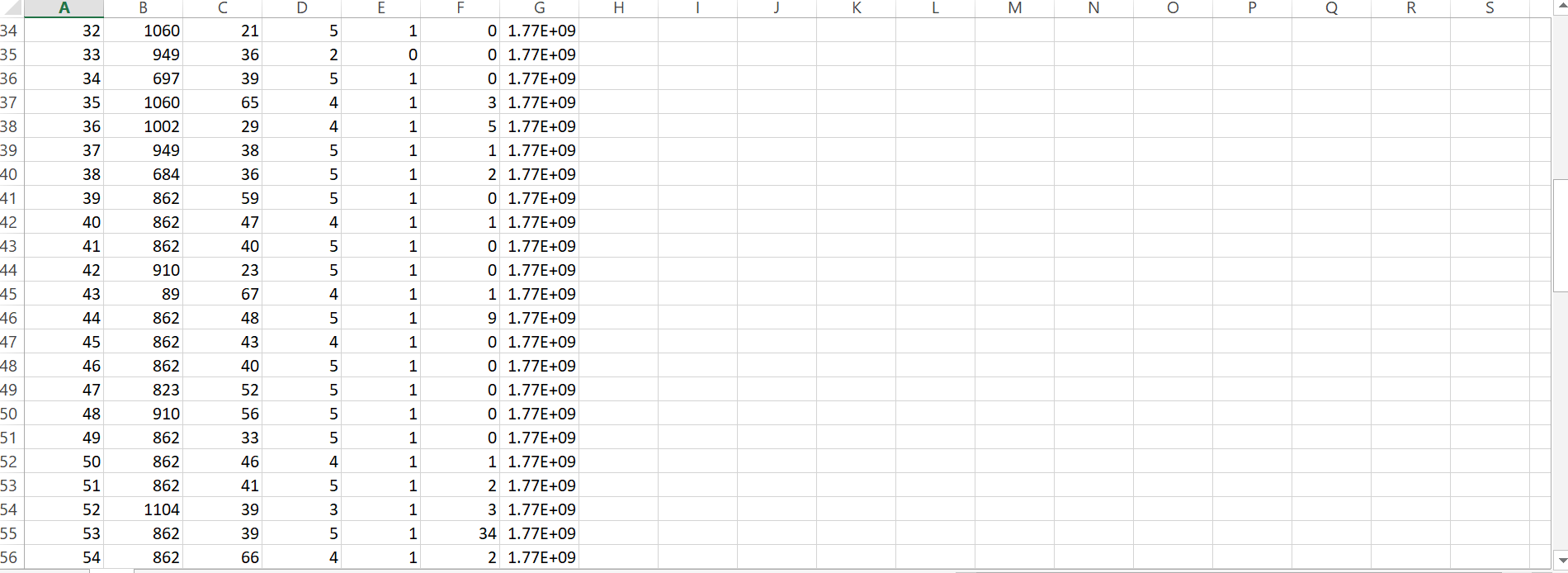
****

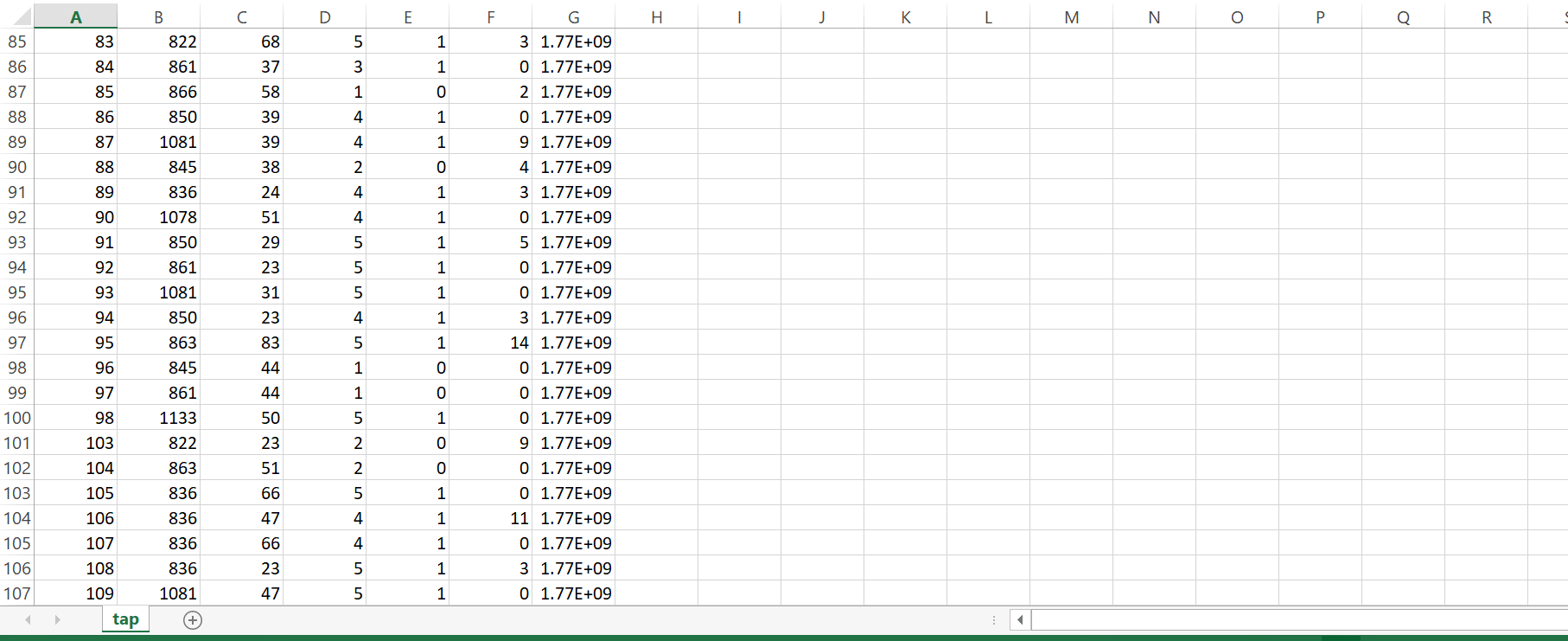
****

****

**Modifiyed numeric data:for svm implement**

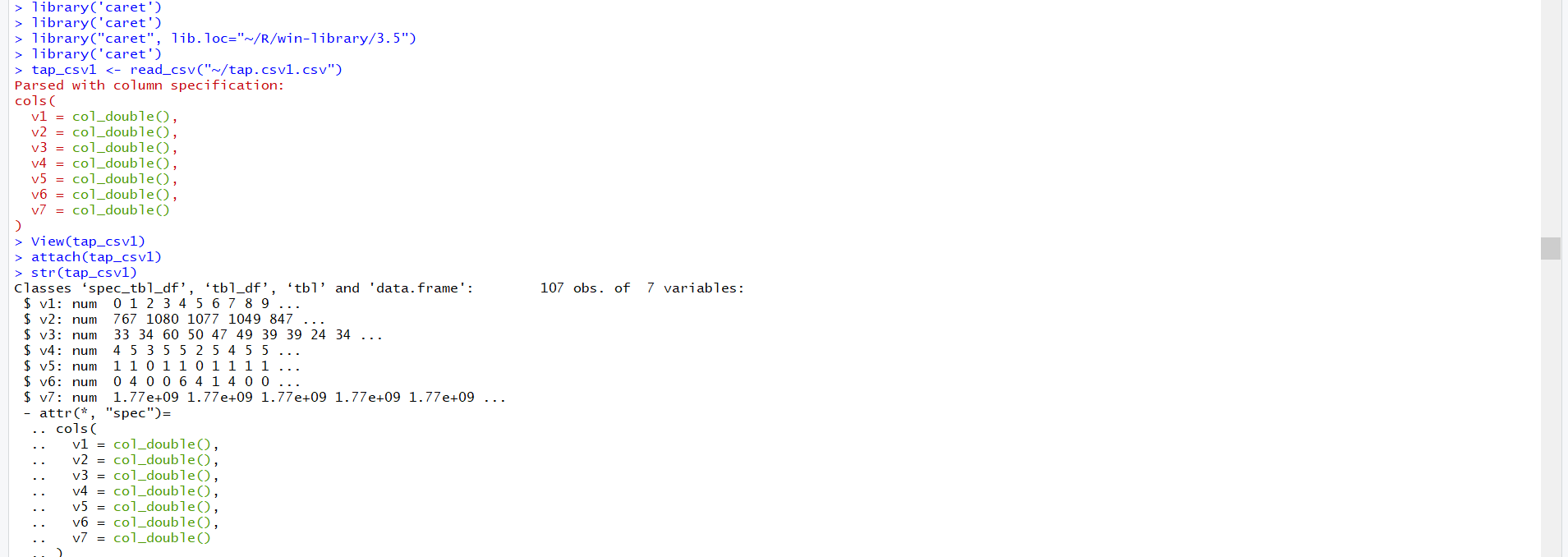
****

****

****

1. **tap\_csv1 <- read\_csv("~/tap.csv1.csv")**

**For importing data into an R data frame, we can useread.csv() method withparameters as a le name and whether our dataset consists of the 1st rowwith a header or not. If a header row exists then, the header should be setTRUE else header should set toFALSE**

****

**5.** Creating test and train datasets:

**intrain <- createDataPartition(y = tap\_csv1$v3,p = 0.7,list = FALSE)**

**training <- tap\_csv1[intrain,]**

**testing <- tap\_csv1[-intrain,]**

**The set.seed()**

**method is used to make our work replicable. As we want ourreaders to learn concepts by coding these snippets. To make our answersreplicable, we need to set a seed value. During partitioning of data, it splitsrandomly but if our readers will pass the same value in the set.seed()**

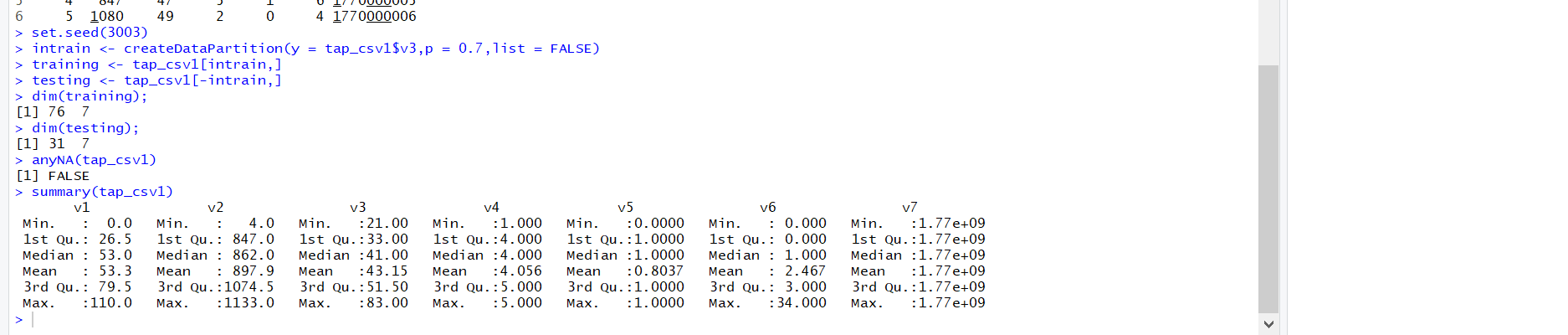
**method. Then we both will get identical results.The caret package provides a method createDataPartition() for partitioningour data into train and test set. We are passing 3 parameters.**

**6.** **dim(training);**

**dim(testing);**

**anyNA(tap\_csv1)**

**summary(tap\_csv1)**

**output:**

**7.** **training[["v3"]]=factor(training[["v3"]])**

The above line of code will convert training data frame’s “V3” column to factor variable

**Training the SVM model:**

**trctrl <- trainControl(method = "repeatedcv", number = 10,repeats = 3)**

**set.seed(3233)**

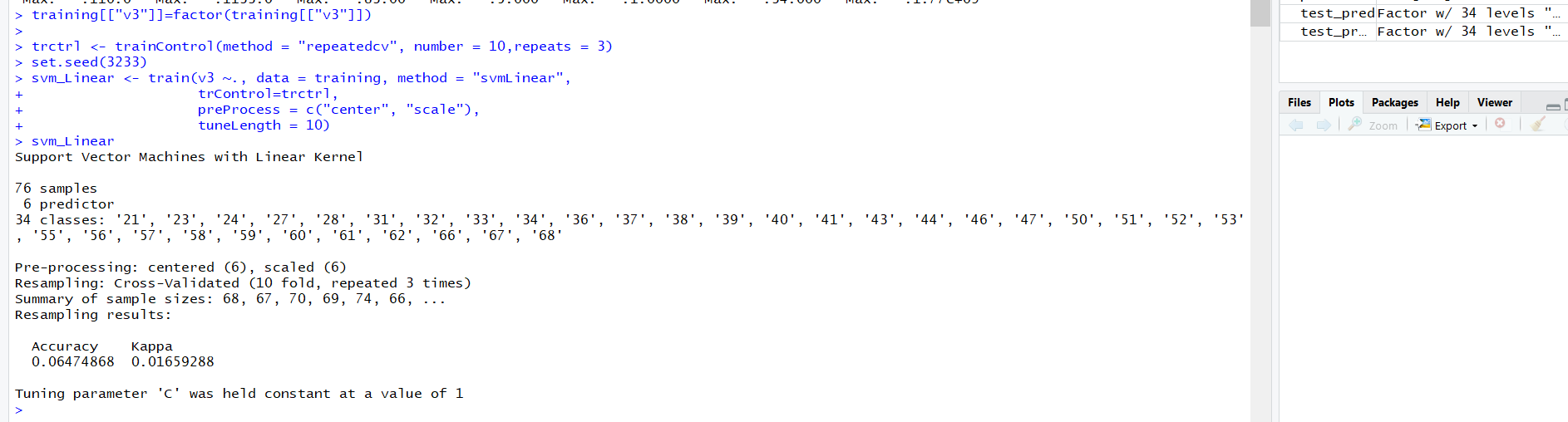
**svm\_Linear <- train(v3 ~., data = training, method = "svmLinear",**

**trControl=trctrl,**

**preProcess = c("center", "scale")**

**tuneLength = 10)**

**svm\_Linear**

**output:** Trained SVM model result****

**We are setting 3 parameters of**

**trainControl()**

**method. The “method”parameter holds the details about resampling method. We can set “method”with many values like “boot”, “boot632”, “cv”, “repeatedcv”, “LOOCV”, “LGOCV”etc. For this tutorial, let’s try to use repeatedcv i.e, repeated cross-validation.The “number” parameter holds the number of resampling iterations. The“repeats ” parameter contains the complete sets of folds to compute for ourrepeated cross-validation. We are using setting number =10 and repeats =3.This**

**trainControl()**

**methods returns a list. We are going to pass this on our**

**train()**

**method.Before training our SVM classier,**

**set.seed()**

**.For training SVM classier,**

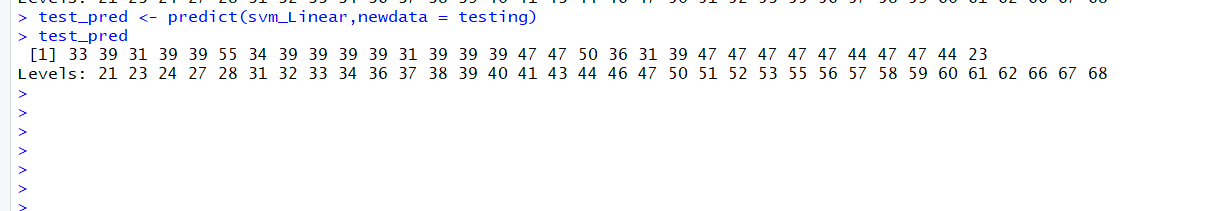
**train()**

**method should be passed with “method”parameter as “svmLinear”. We are passing our target variable V3. The“V14~.” denotes a formula for using all attributes in our classier and 3 asthe target variable. The “trControl” parameter should be passed with resultsfrom our**

**trianControl()**

**method. The “preProcess” parameter is forpreprocessing our training data.As discussed earlier for our data, preprocessing is a mandatory task. We arepassing 2 values in our “preProcess” parameter “center” & “scale”. These twohelp for centering and scaling the data. After preProcessing these convert ourtraining data with mean value as approximately “0” and standard deviation as“1”. The “tuneLength” parameter holds an integer value. This is for tuning ouralgorithm**

**8.** Test Set Prediction: Now, our model is trained with C value as 1. We are ready to predict classesfor our test set. We can use predict() method

****

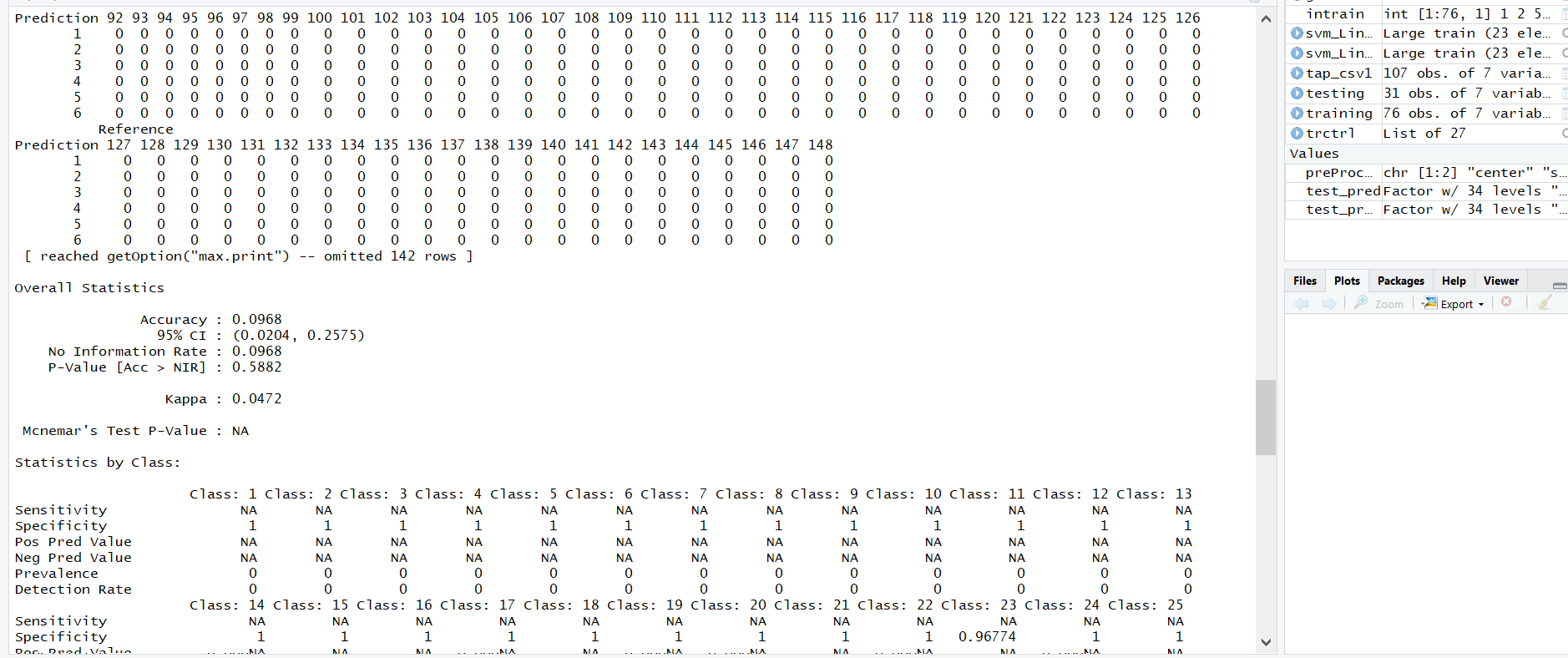
**How Accurately our model is working?:** Using confusion matrix, we can print statistics of our results. It shows that ourmodel accuracy for test set is.9.68%

9. confusionMatrix(

factor(test\_pred, levels = 1:148),

factor(testing$v3, levels = 1:148)

)

Output:****

We can also do some customizations for selecting C value(Cost) in Linearclassier. This can be done by inputting values in grid search. The next codesnippet will show you, building & tuning of an SVM classier with dierentvalues of C. We are going to put some values of C using expand.grid() into“grid” dataframe. Next step is to use this dataframe for testing our classierat specic C values. It needs to be put in train() method with tuneGridparameter

10. grid <- expand.grid(C = c(0,1,3,4,6,0.2,0.3))

set.seed(3233)

svm\_Linear\_Grid <- train(v3 ~., data = training, method = "svmLinear",

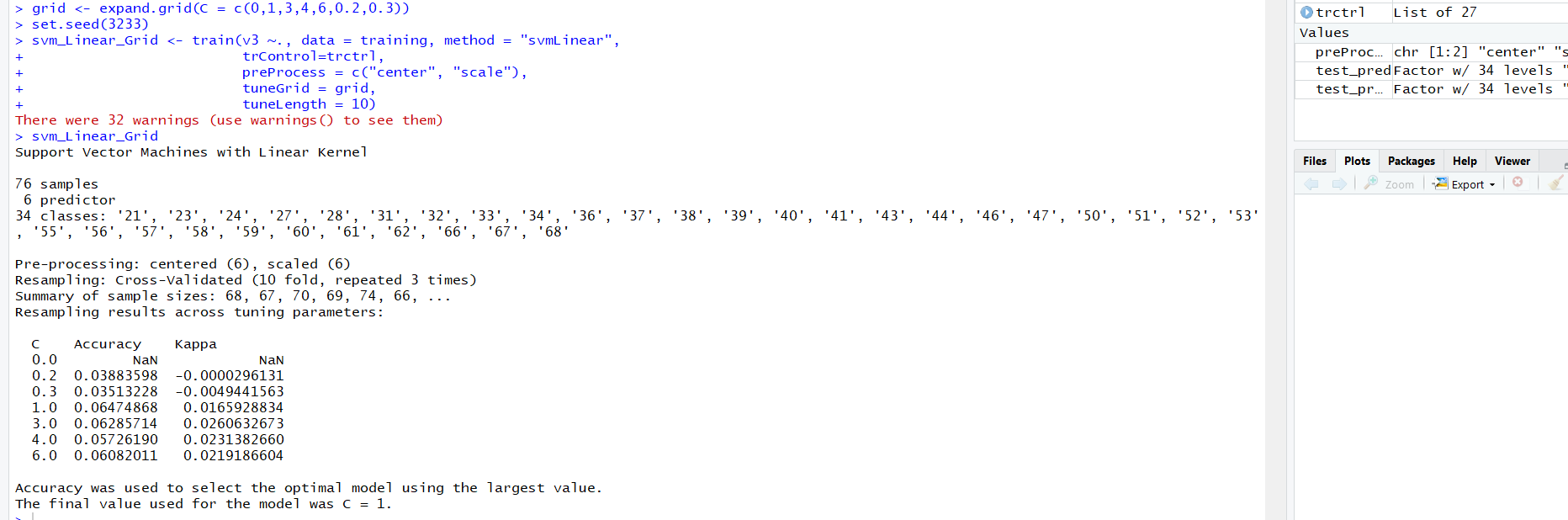
trControl=trctrl,

preProcess = c("center", "scale"),

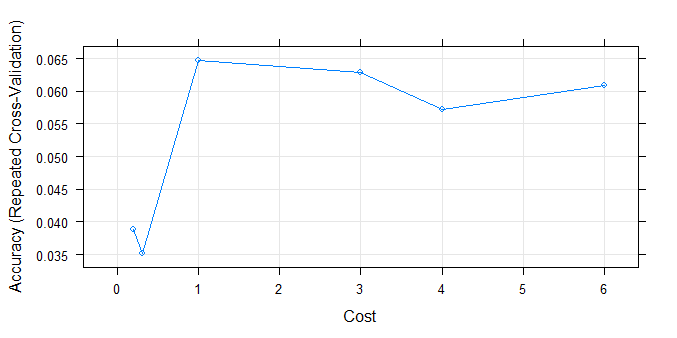
tuneGrid = grid,

tuneLength = 10)

> svm\_Linear\_Grid

0utput:****

11. plot(svm\_Linear\_Grid)

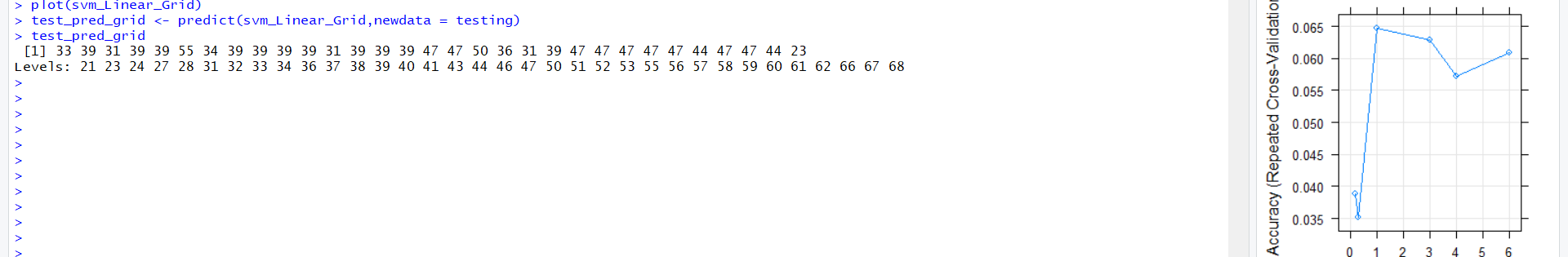
Output:

accuracy was used to select the optimal model using the largest value.

The final value used for the model was C = 1.

12. test\_pred\_grid <- predict(svm\_Linear\_Grid,newdata = testing)

test\_pred\_grid

output:

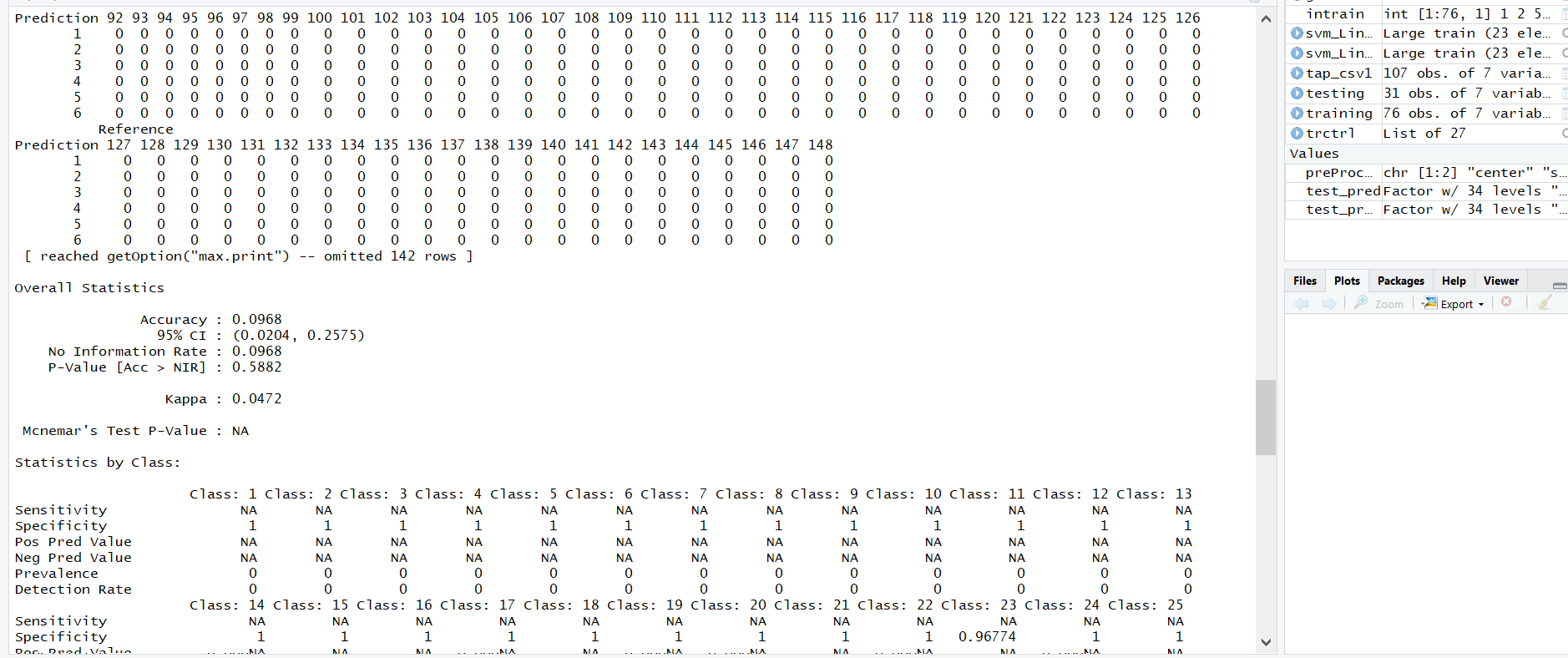
13. Let’s check its accuracy using confusion -matrix.

confusionMatrix(

factor(test\_pred, levels = 1:148),

factor(testing$v3, levels = 1:148)

)

Output: ****

It shows that ourmodel accuracy for test set is.9.68%