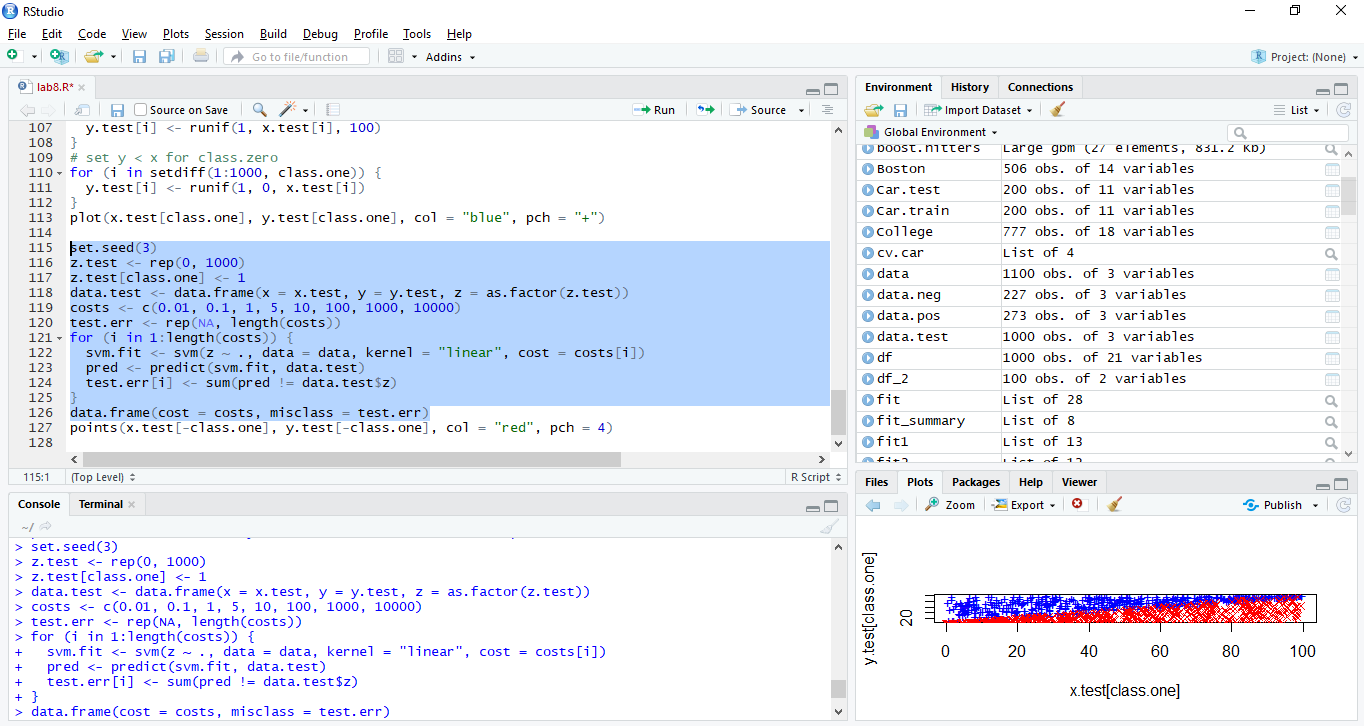
**Introduction to Statistical Learning Lab8 (Support Vector Machines)**

**Name: Sandeep Reddy Salkuti**

**Id: 16296868**

**Email:** [**sswf7@umsystem.edu**](mailto:sswf7@umsystem.edu)

1. **You may download the R Code for Labs and the Data Sets to use from the textbook website.**



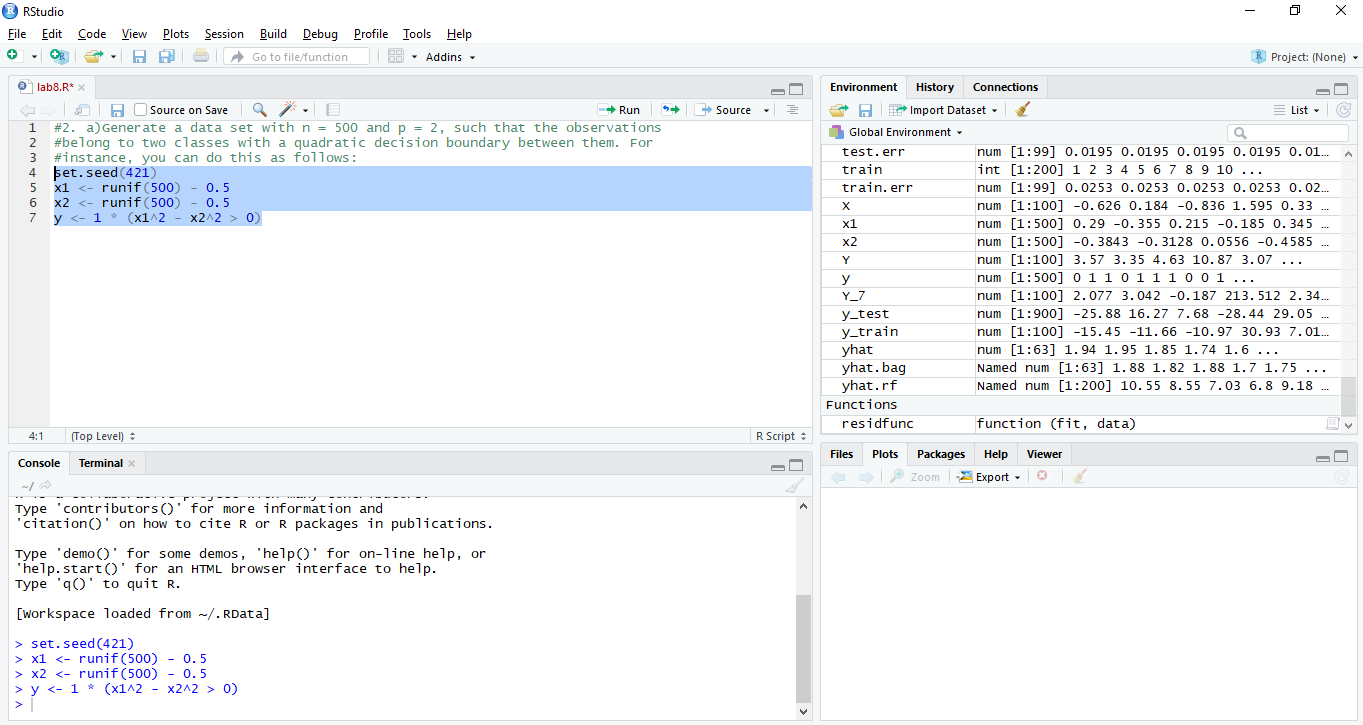
**2. (45 points total) We have seen that we can fit an SVM with a non-linear kernel in order to perform classification using a non-linear decision boundary. We will now see that we can also obtain a non-linear decision boundary by performing logistic regression using non-linear transformations of the features.**

**(a) (5 points) Generate a data set with n = 500 and p = 2, such that the observations belong to two classes with a quadratic decision boundary between them. For instance, you can do this as follows:**

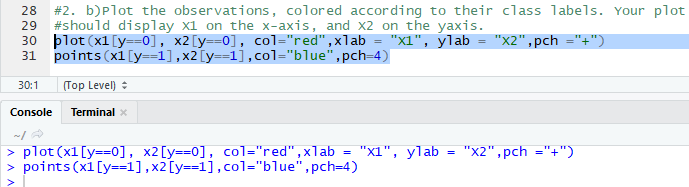
**> x1=runif (500) -0.5**

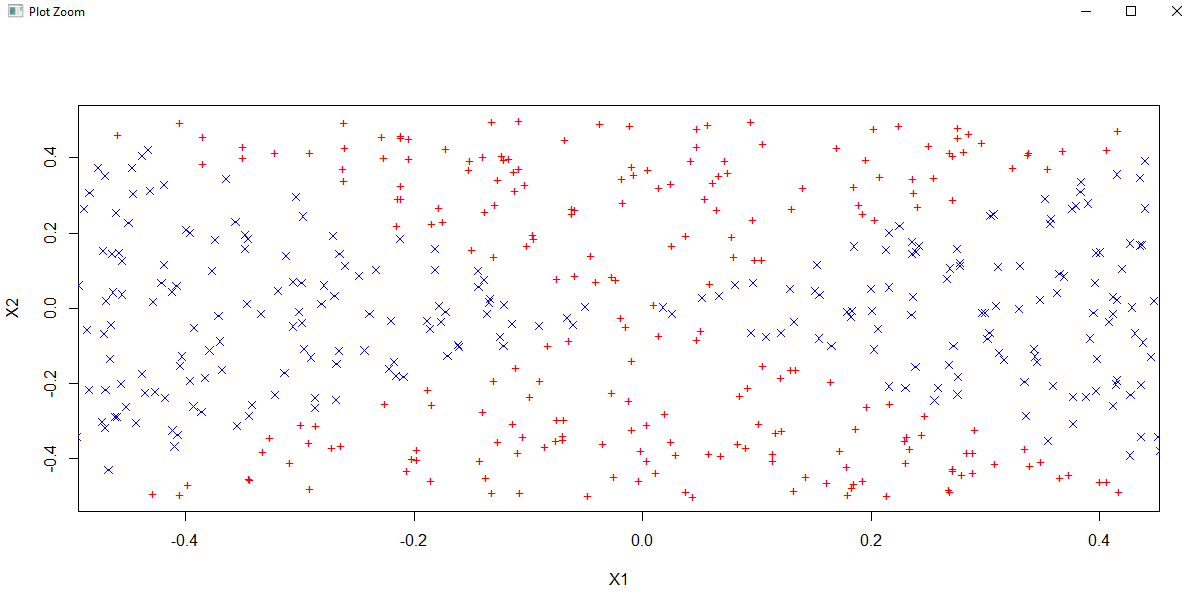
**> x2=runif (500) -0.5**

**> y = 1\*( x1b2-x2b2 > 0)**

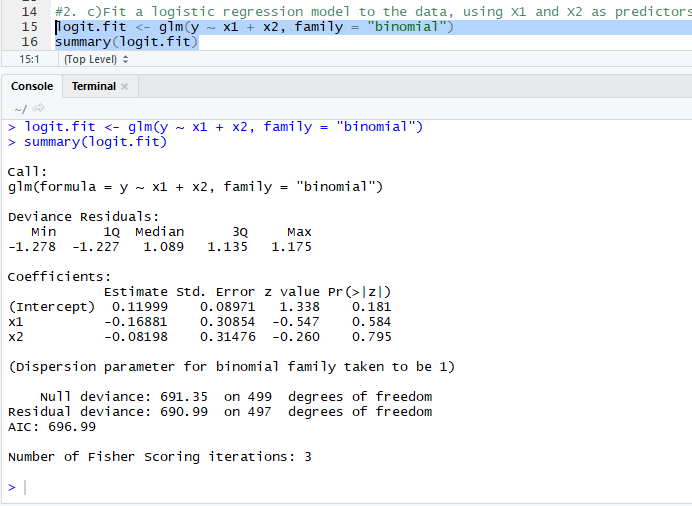


**(b) (5 points) Plot the observations, colored according to their class labels. Your plot should display X1 on the x-axis, and X2 on the yaxis.**

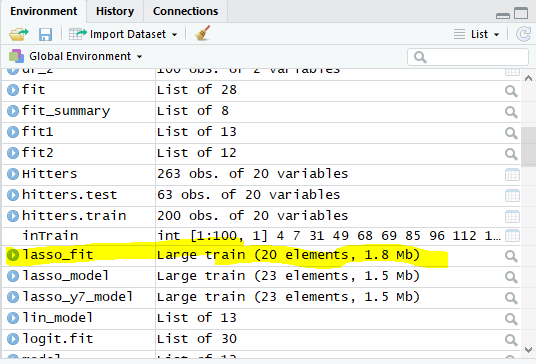




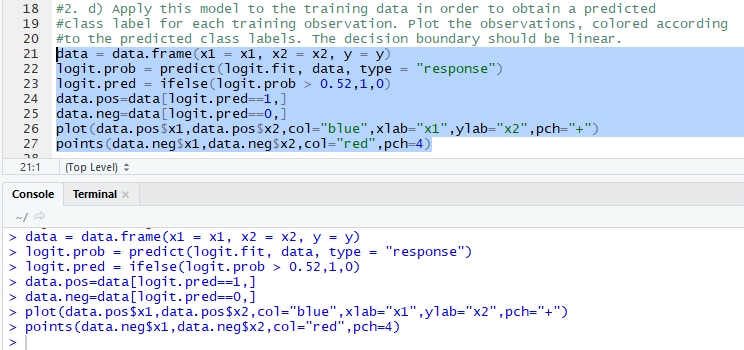
**(c) (5 points) Fit a logistic regression model to the data, using X1 and X2 as predictors.**

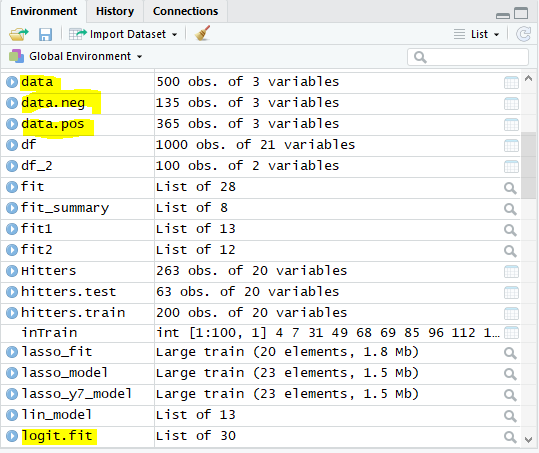


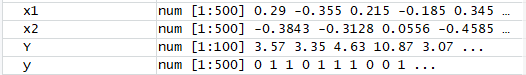
None of the variables are statistically significant.

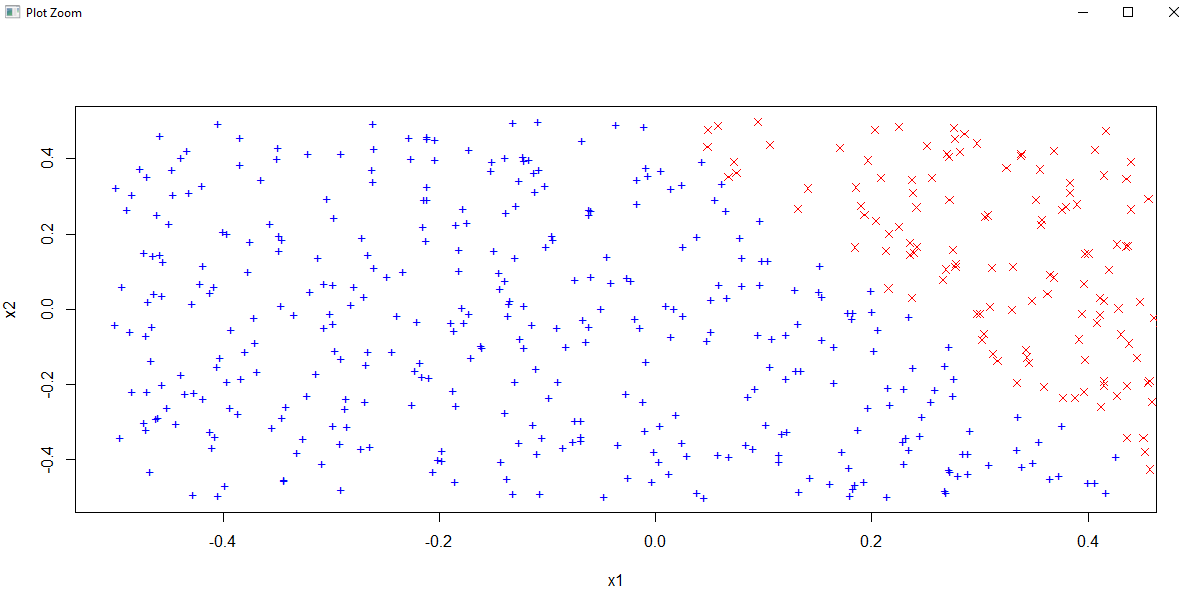


**(d) (5 points) Apply this model to the training data in order to obtain a predicted class label for each training observation. Plot the observations, colored according to the predicted class labels. The decision boundary should be linear.**



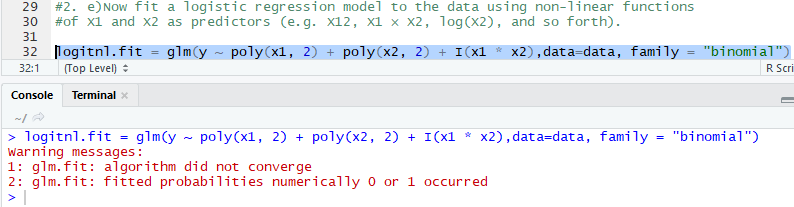


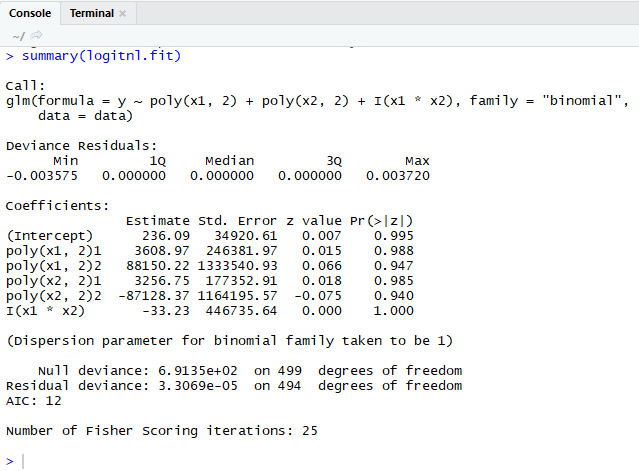




The decision boundary is obviously linear

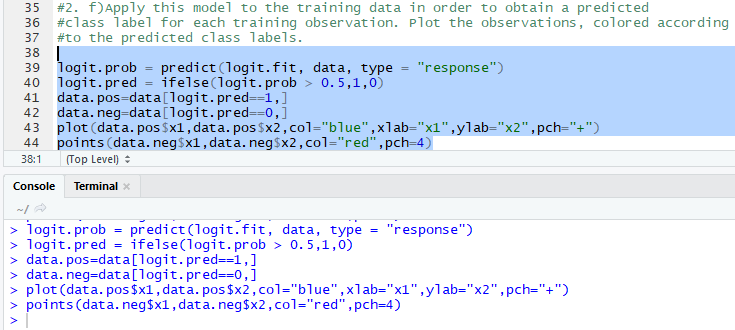
**(e) (5 points) Now fit a logistic regression model to the data using non-linear functions of X1 and X2 as predictors (e.g. X2 1 , X1 × X2, log(X2), and so forth).**

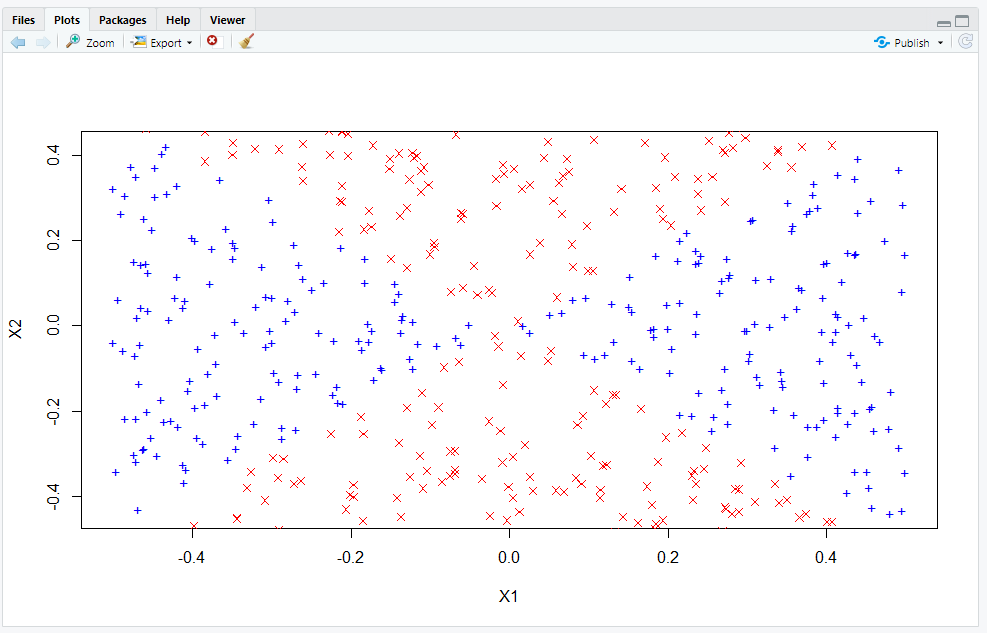




In this we can see that none of the variables are statistically significant.

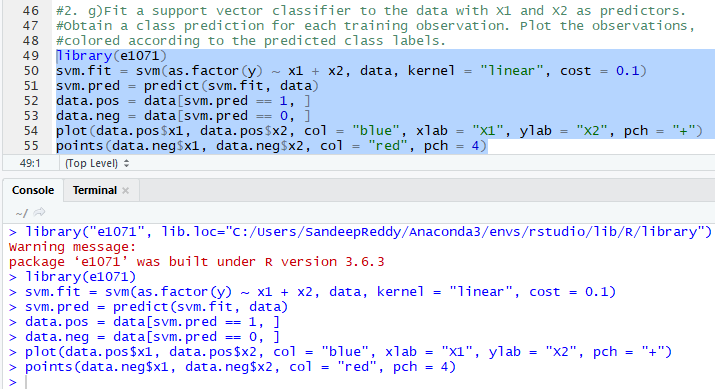
**(f) (5 points) Apply this model to the training data in order to obtain a predicted class label for each training observation. Plot the observations, colored according to the predicted class labels. The decision boundary should be obviously nonlinear. If it is not, then repeat (a)-(e) until you come up with an example in which the predicted class labels are obviously non-linear.**

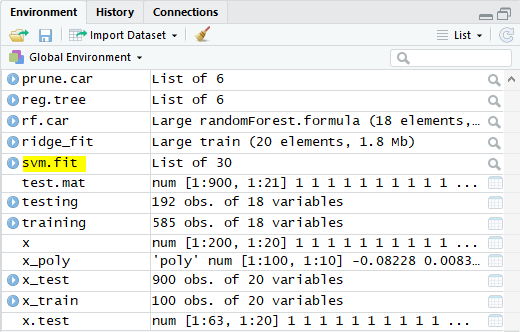


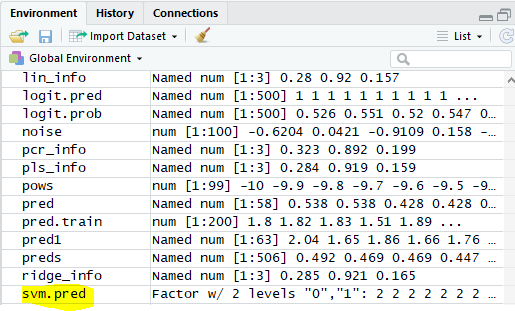


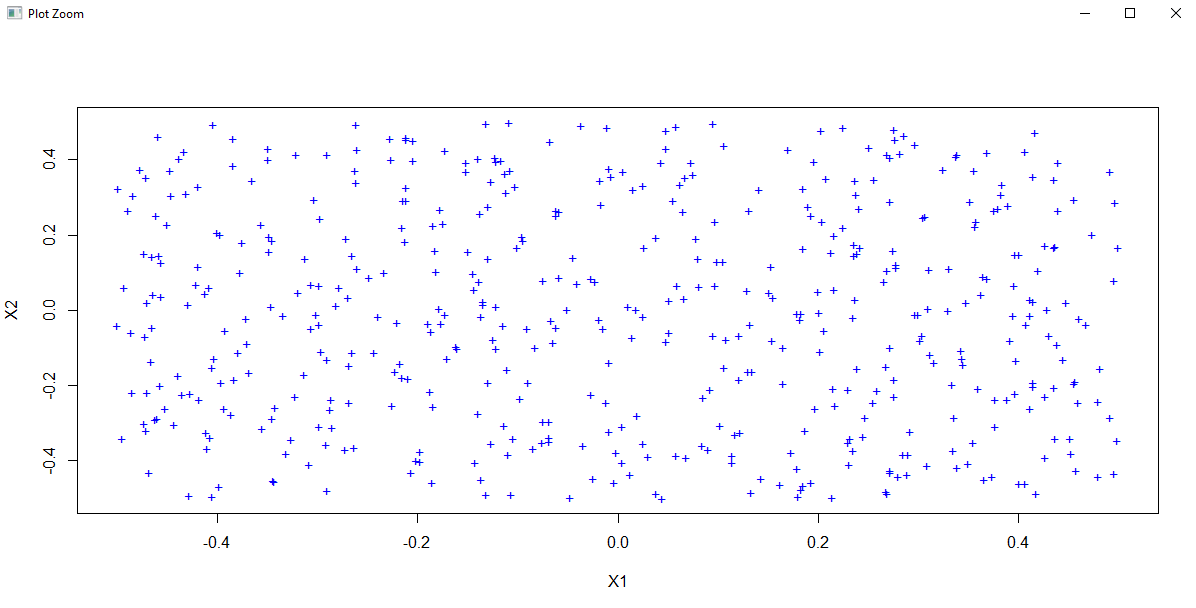
The non linear decision boundary is very similar to true decision boundary.

**(g) (5 points) Fit a support vector classifier to the data with X1 and X2 as predictors. Obtain a class prediction for each training observation. Plot the observations, colored according to the predicted class labels.**



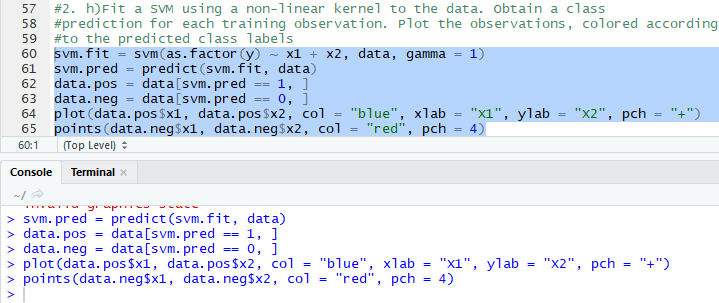


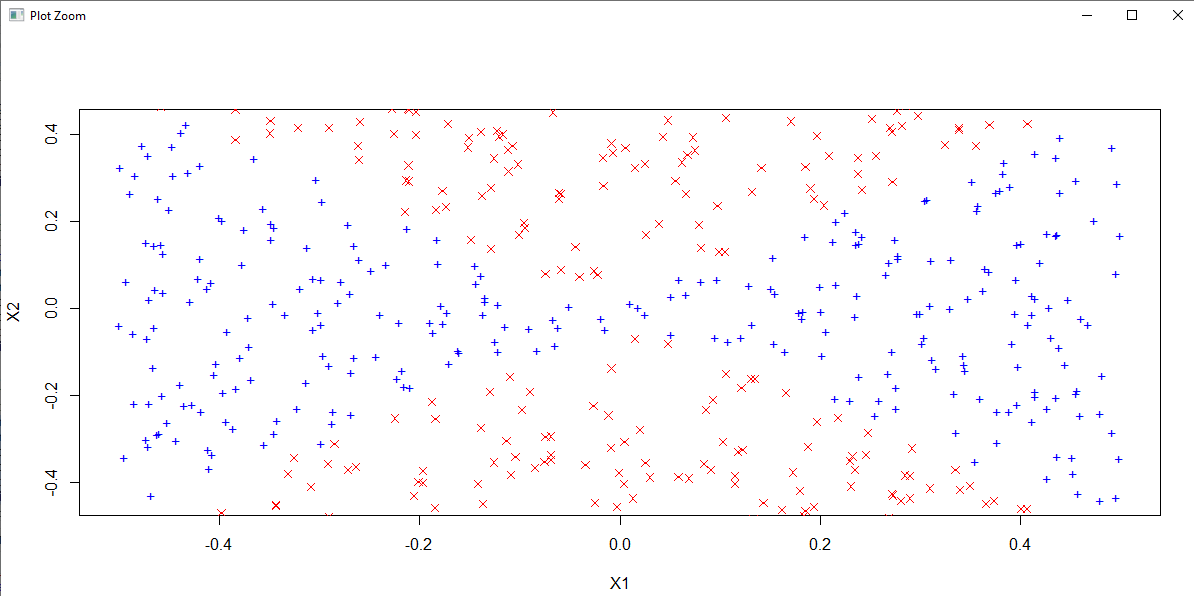




This support vector classifier classifies all points to a single class

**(h) (5 points) Fit a SVM using a non-linear kernel to the data. Obtain a class prediction for each training observation. Plot the observations, colored according to the predicted class labels.**





In this again non linear decision boundary is very similar to true decision boundary

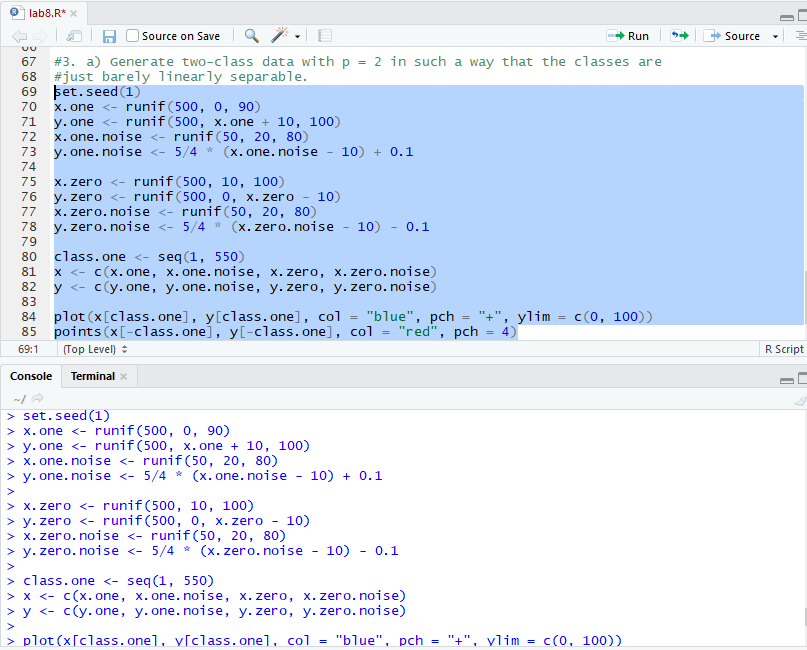
1. **(5 points) Comment on your results.**

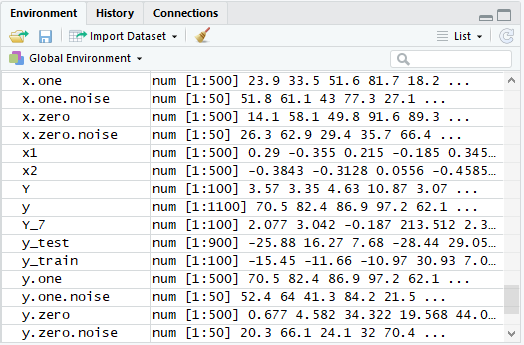
The experiment performed covers the idea of SVMS are important to use for finding non-linear models, using cross validation would be easier with the parameter of gamma

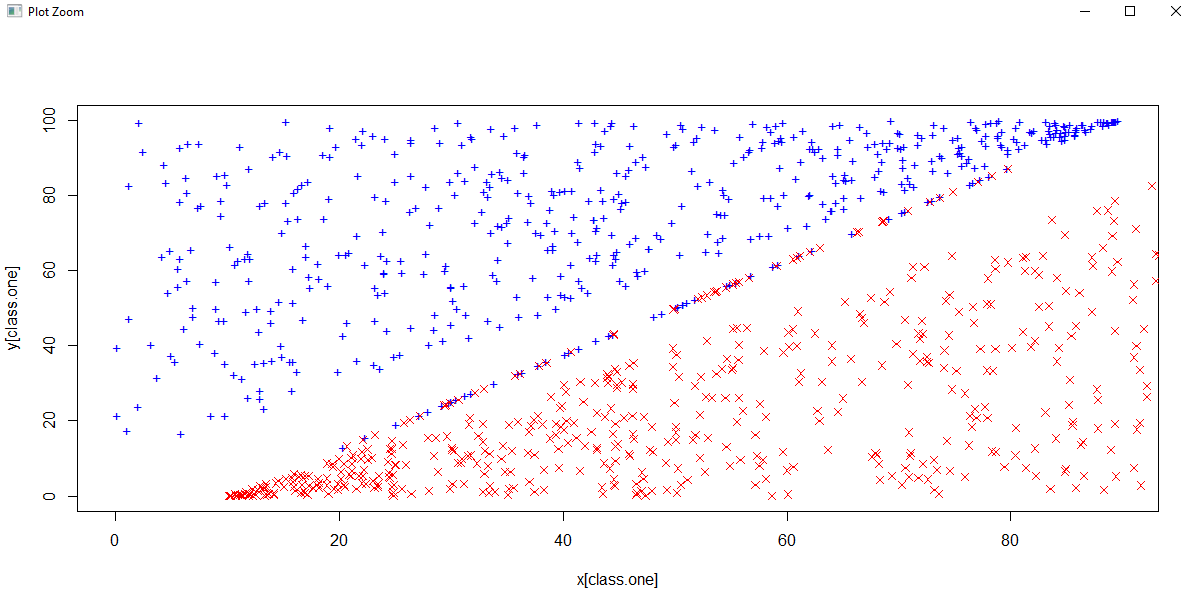
**3. (20 points total) At the end of Section 9.6.1, it is claimed that in the case of data that is just barely linearly separable, a support vector classifier with a small value of cost that misclassifies a couple of training observations may perform better on test data than one with a huge value of cost that does not misclassify any training observations. You will now investigate this claim.**

**(a) (5 points) Generate two-class data with p = 2 in such a way that the classes are just barely linearly separable.**

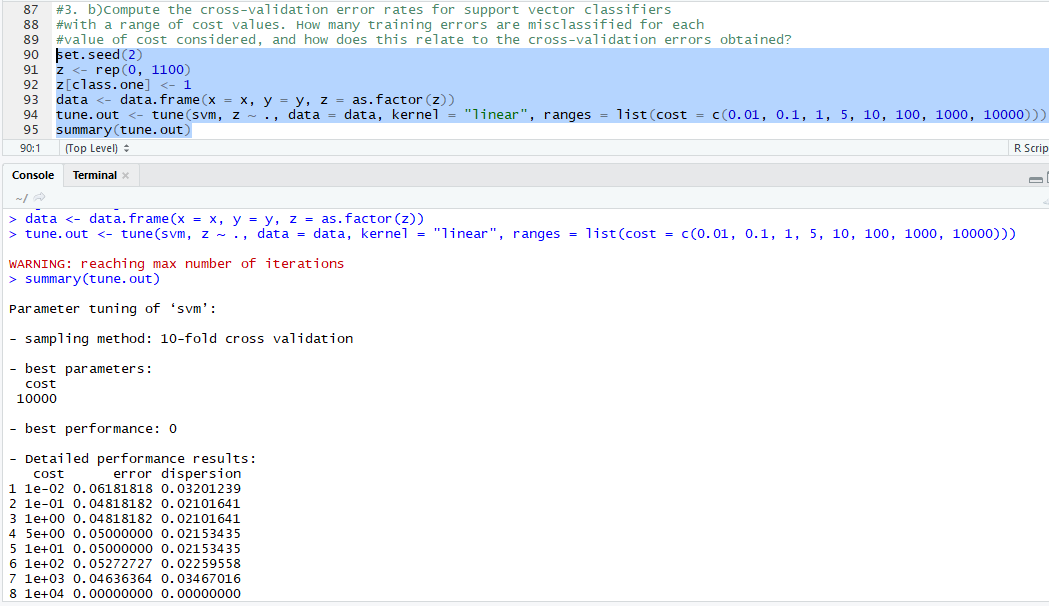
We randomly generate 1000 points and scatter them across line *x=y* with wide margin. We also create noisy points along the line *5x−4y−50=0*. These points make the classes barely separable and also shift the maximum margin classifier.

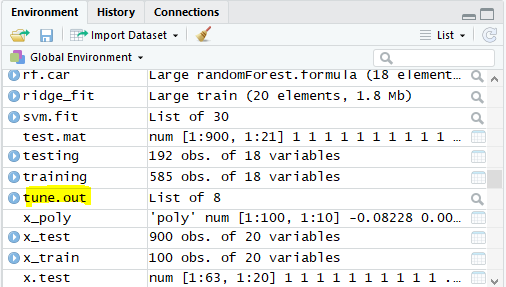


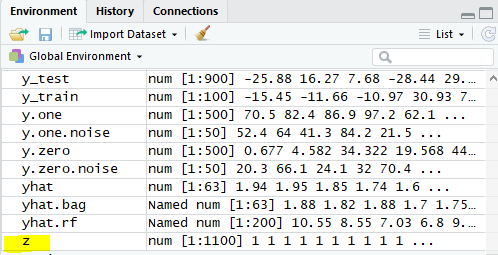




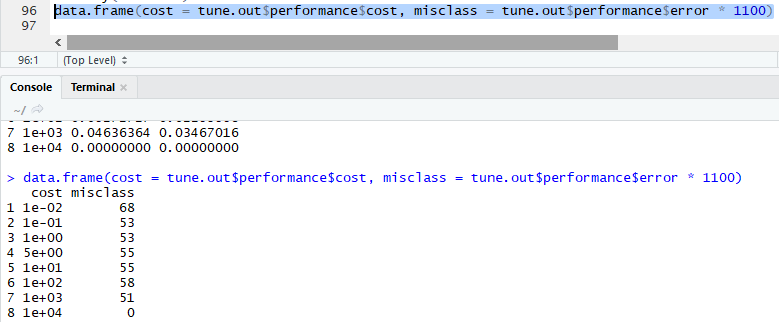
**(b) (5 points) Compute the cross-validation error rates for support vector classifiers with a range of cost values. How many training errors are misclassified for each value of cost considered, and how does this relate to the cross-validation errors obtained?**





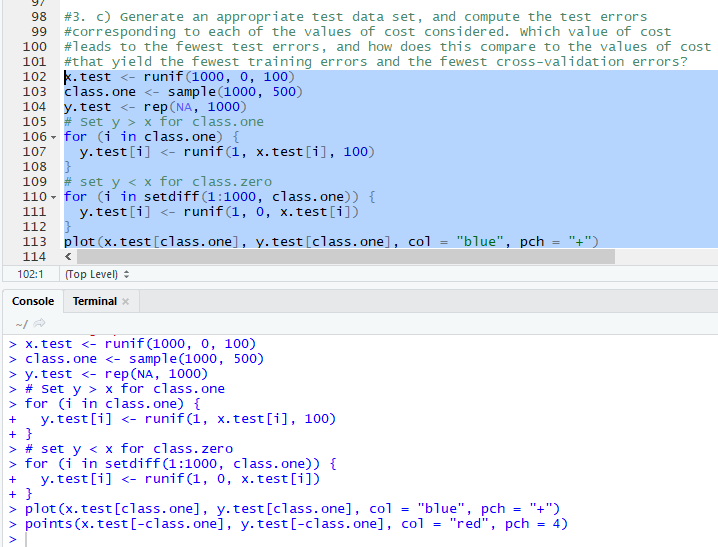


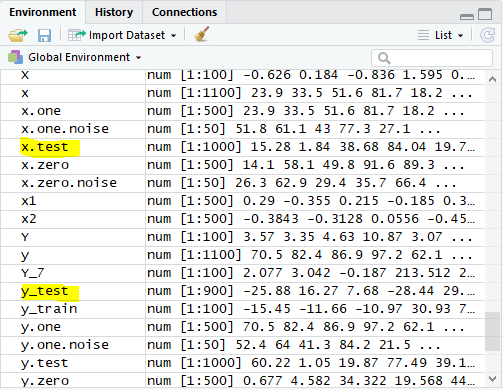
A cost of 10000 seems the best choice of parameter.

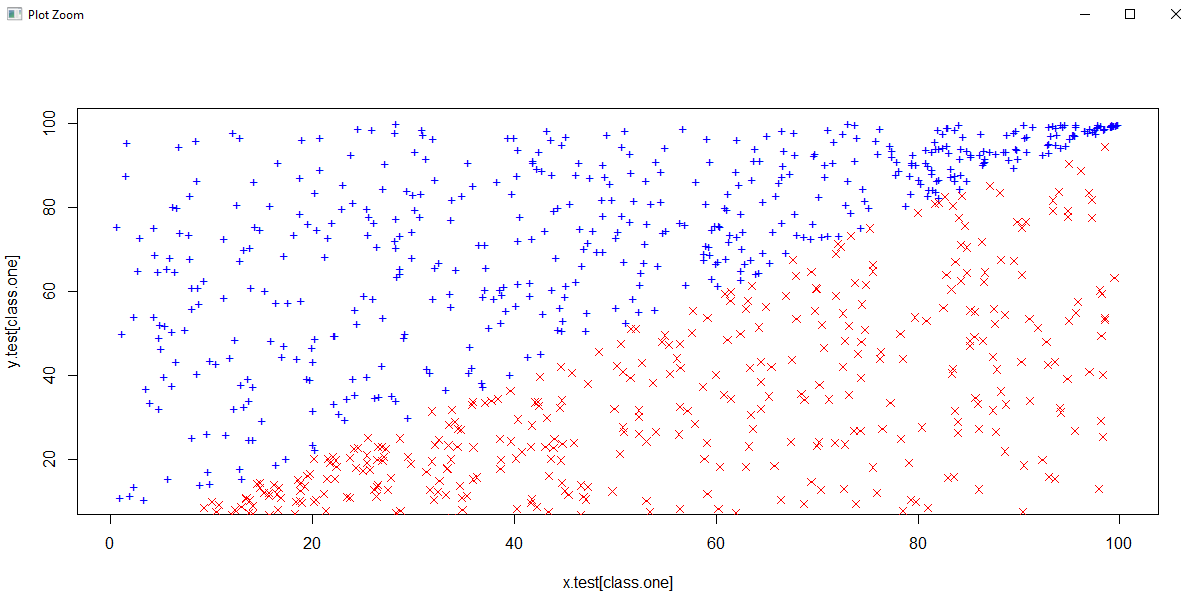


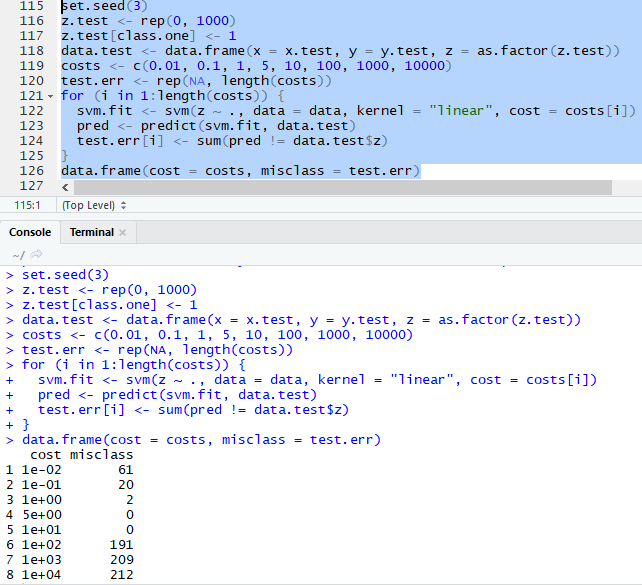
Here a cost of 10000 classify all training points correctly.

**(c) (5 points) Generate an appropriate test data set, and compute the test errors corresponding to each of the values of cost considered. Which value of cost leads to the fewest test errors, and how does this compare to the values of cost that yield the fewest training errors and the fewest cross-validation errors?**









Costs of 1, 5 or 10 seem to perform better on test observations, this is much smaller than the value of 10000 for training observations.

**(d) (5 points) Discuss your results.**

We again see an overfitting phenomenon for linear kernel. A large cost tries to correctly classify noisy-points and hence overfits the train data. A small cost, however, makes a few errors on the noisy test points and performs better on test data.