**Лабораторная работа №6**

**“Метод опорных векторов”**

по дисциплине “Машинное обучение”

|  |  |
| --- | --- |
| Выполнил  студент гр. 33504/2 | Лелюхин Д.О. |
| Руководитель | Селин И.А. |

**Оглавление**

[**Первое задание:** 3](#_Toc511340528)

[**Второе задание:** 4](#_Toc511340529)

[**Третье задание:** 9](#_Toc511340530)

[**Четвертое задание:** 12](#_Toc511340531)

[**Пятое задание:** 14](#_Toc511340532)

[**Шестое задание:** 17](#_Toc511340533)

# **Первое задание:**

Постройте алгоритм метода опорных векторов типа "C-classification" с параметром C = 1, используя ядро "linear". Визуализируйте разбиение пространства признаков на области с помощью полученной модели. Выведите количество полученных опорных векторов, а также ошибки классификации на обучающей и тестовой выборках.

**Код программы:**

#Unit 1

library(e1071)

A\_tran=read.table("svmdata1.txt",header = TRUE, sep="\t")

A\_test=read.table("svmdata1test.txt",header = TRUE, sep="\t")

symbols.pallete = c("Blue", "Red")

area.pallete = function(n = 2)

{

cols = rainbow(n)

cols[1:2] = c("PaleGreen", "Pink")

return(cols)

}

plot(X1 ~ X2, A\_tran, col = Color)

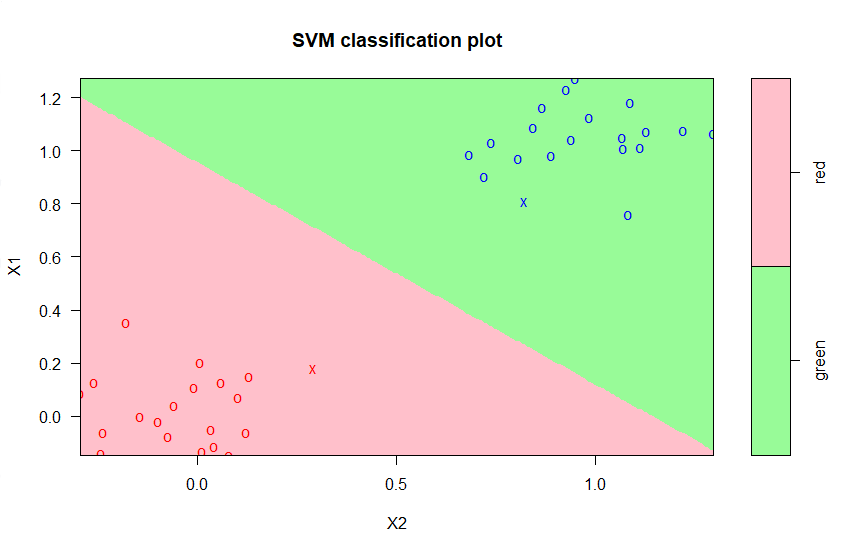
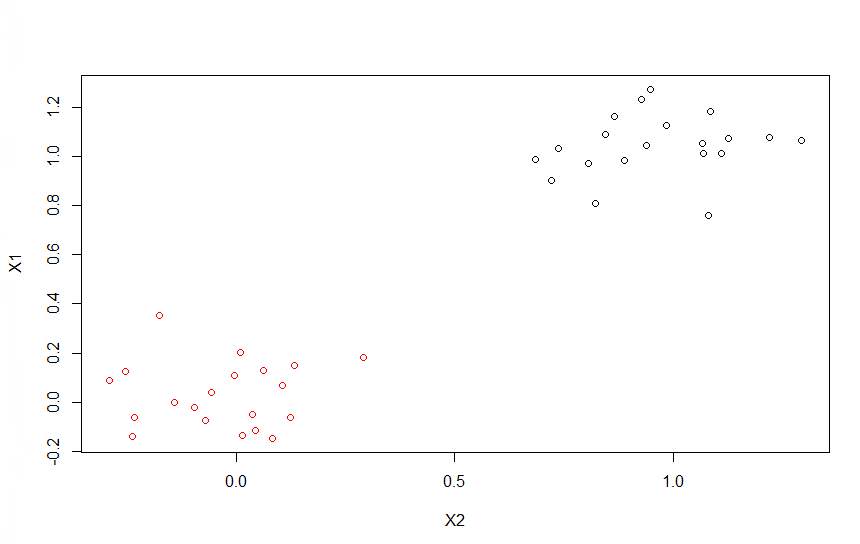
svmModelLinear = svm(Color ~ ., data = A\_tran, type = "C-classification", cost = 1, kernel = "linear")

plot(svmModelLinear, A\_tran, grid = 250, symbolPalette = symbols.pallete, color.palette = area.pallete)

predictionsTrain = predict(svmModelLinear, A\_tran)

table(A\_tran$"Color", predictionsTrain)

**Результаты:**



Предсказания на выборке *A\_test:*

predictionsTrain

green red

green 20 0

red 0 20

Ошибок нет.

# **Второе задание:**

Используя алгоритм метода опорных векторов типа "C-classification" с линейным ядром, добейтесь нулевой ошибки сначала на обучающей выборке, а затем на тестовой, путем изменения параметра C. Выберите оптимальное значение данного параметра и объясните свой выбор. Всегда ли нужно добиваться минимизации ошибки на обучающей выборке?

**Код программы:**

#Unit 2

B\_train=read.table("svmdata2.txt",header = TRUE, sep="\t")

B\_test=read.table("svmdata2test.txt",header = TRUE, sep="\t")

plot(X1 ~ X2, B\_train, col = Colors)

for (i in 1:200)

{

svmModelLinear = svm(Colors ~ ., data = B\_train, type = "C-classification", cost = i, kernel = "linear")

predictionsTrain = predict(svmModelLinear, B\_train)

print(i)

print(table(B\_train$"Colors", predictionsTrain))

}

area.pallete = function(n = 2)

{

cols = rainbow(n)

cols[1:2] = c("PaleGreen", "Pink")

return(cols)

}

plot(X1 ~ X2, B\_train, col = Colors)

c = 1

svmModelLinear = svm(Colors ~ ., data = B\_train, type = "C-classification", cost = c, kernel = "linear")

plot(svmModelLinear, B\_train, grid = 250, symbolPalette = symbols.pallete, color.palette = area.pallete)

predictionsTrain = predict(svmModelLinear, B\_train)

table(B\_tran$"Colors", predictionsTrain)

**Результаты:**

**Тренировочная выборка:**

[1] 182

predictionsTrain

green red

green 25 0

red 1 24

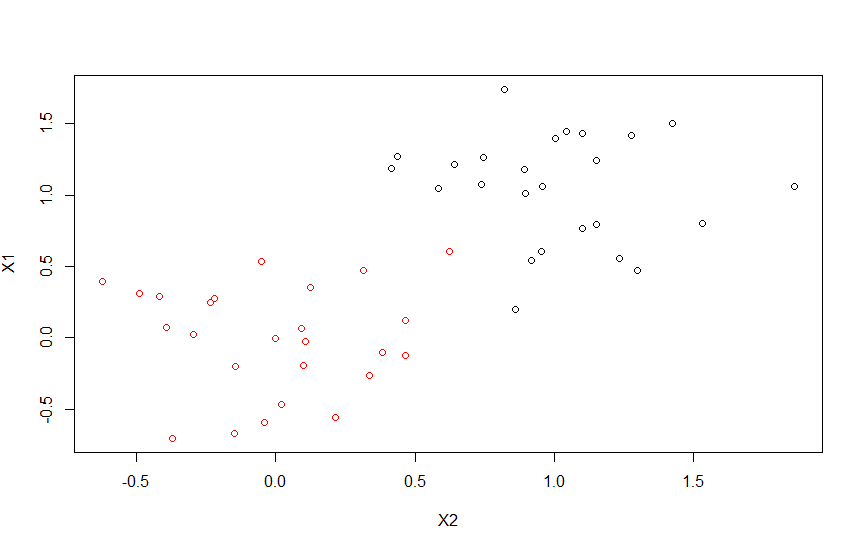
[1] 183

predictionsTrain

green red

green 25 0

red 0 25

****

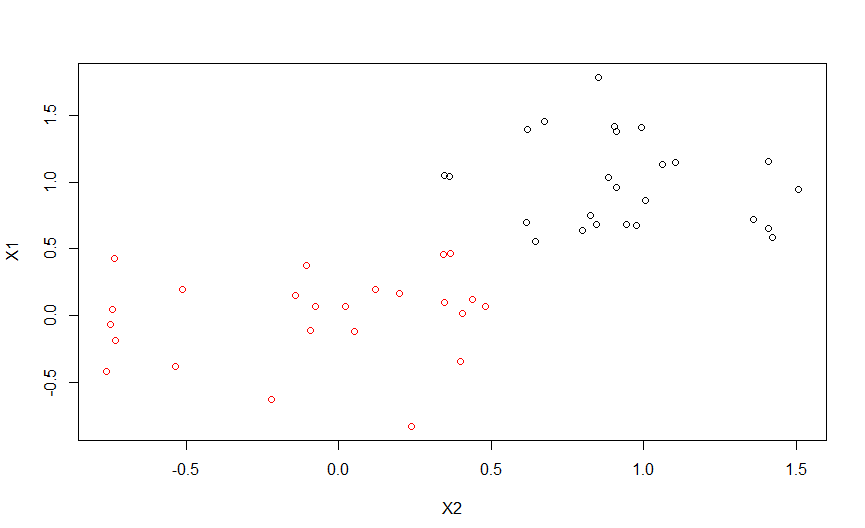
**Тестовая выборка:**

predictionsTest

green red

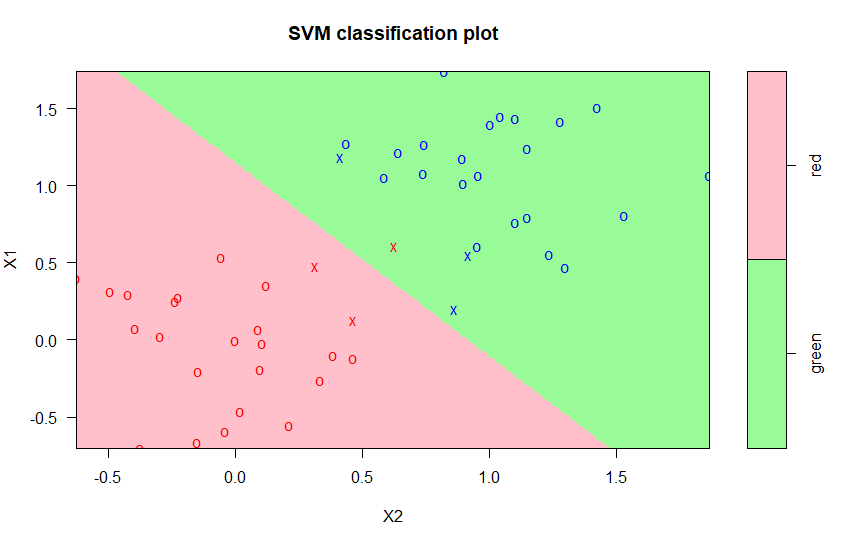
green 20 0

red 0 20

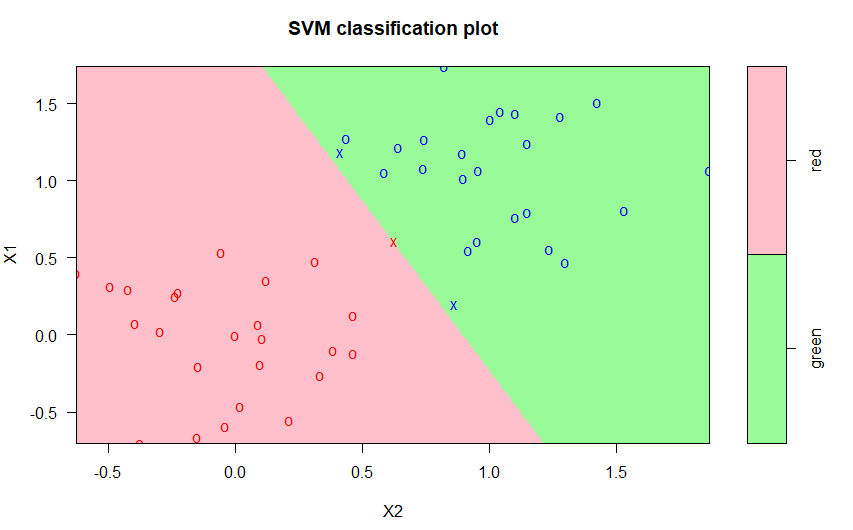
****

**Тренировочная выборка:**

**C=1**

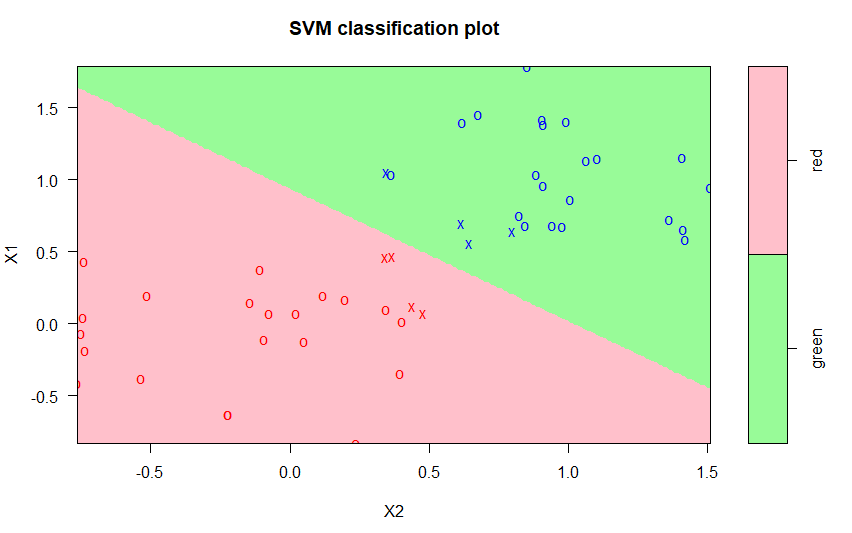


**C = 183**

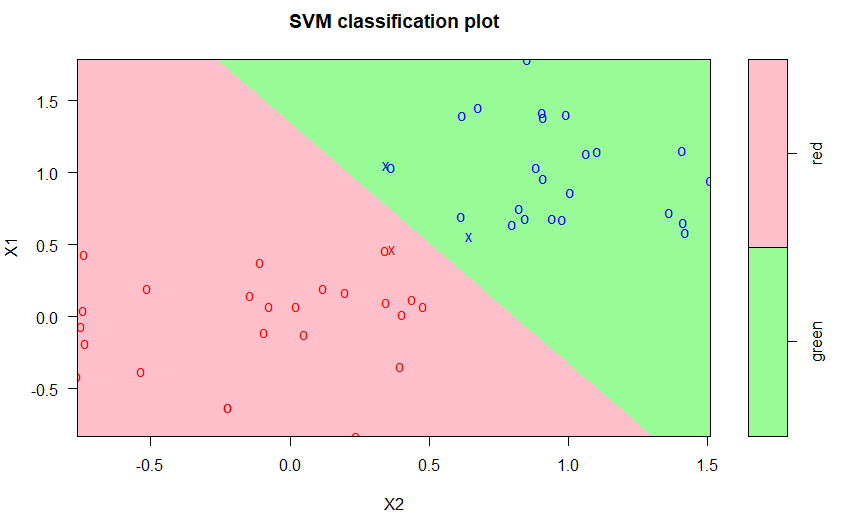


**Тестовая выборка:**

**C=1**



**С=183**



Для тестовой выборки при изменении параметра C кардинальных изменений замечено не было, для тренировочной выборки при изменении параметра C переобучение уменьшилось. Следовательно, для тренировочной выборки нужно добиваться минимальной ошибки, для тестовой нет.

# **Третье задание:**

Среди ядер "polynomial", "radial" и "sigmoid" выберите оптимальное в плане количества ошибок на тестовой выборке. Попробуйте различные значения параметра degree для полиномиального ядра.

**Код программы:**

#Unit 3

C\_test=read.table("svmdata3test.txt",header = TRUE, sep="\t")

area.pallete = function(n = 2)

{

cols = rainbow(n)

cols[1:2] = c("PaleGreen", "Pink")

return(cols)

}

svmModelLinear = svm(Colors ~ ., data = C\_test, type = "C-classification", cost = 1, kernel = "polynomial", degree = 20)

plot(svmModelLinear, C\_test, grid = 250, symbolPalette = symbols.pallete, color.palette = area.pallete)

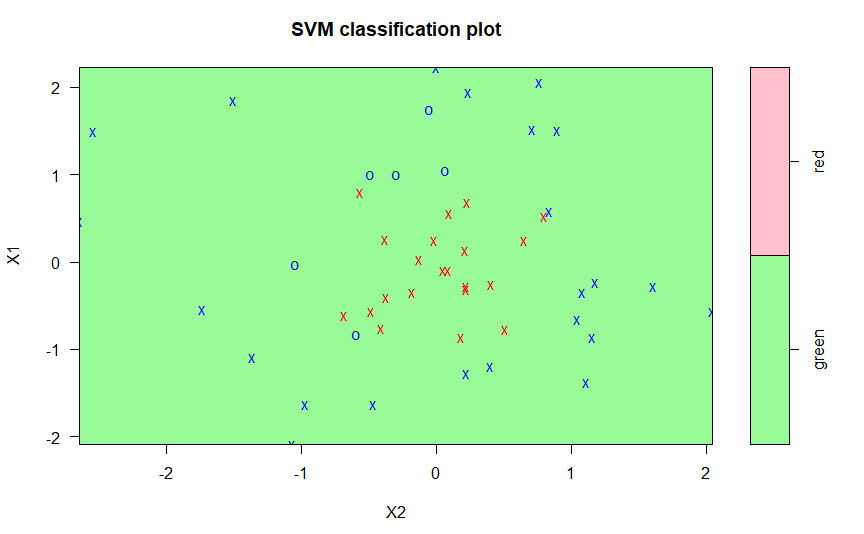
predictionsTest = predict(svmModelLinear, C\_test)

table(C\_test$"Color", predictionsTest)

**Результаты:**

**Polynomial:**

**Degree=1:**



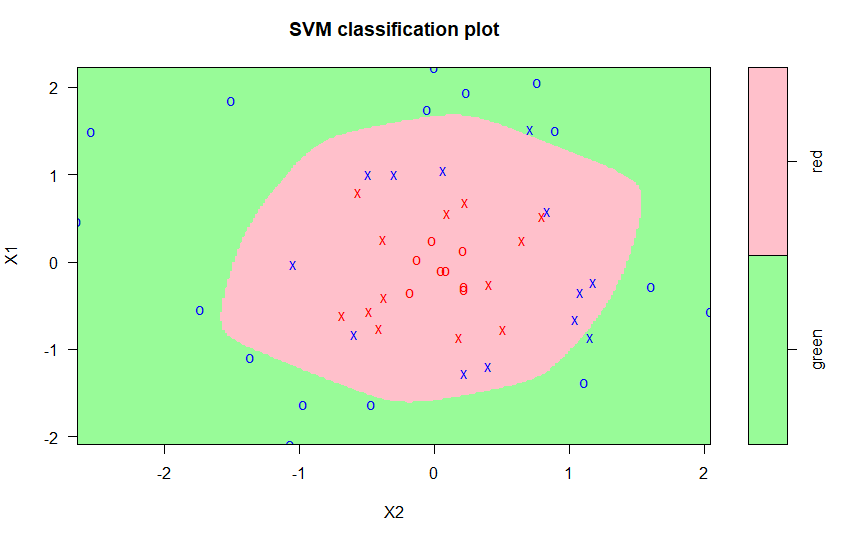
predictionsTest

green red

green 29 0

red 21 0

**Degree=20:**



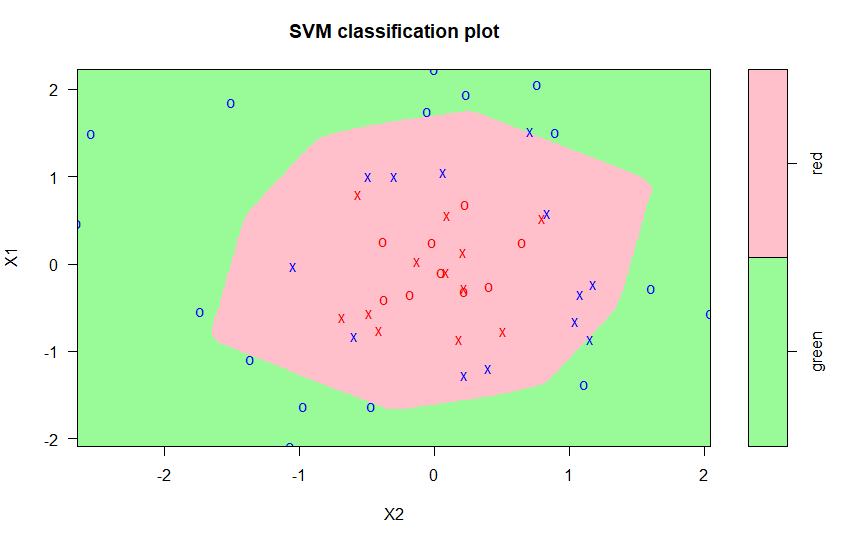
predictionsTest

green red

green 18 11

red 0 21

**Degree=100:**



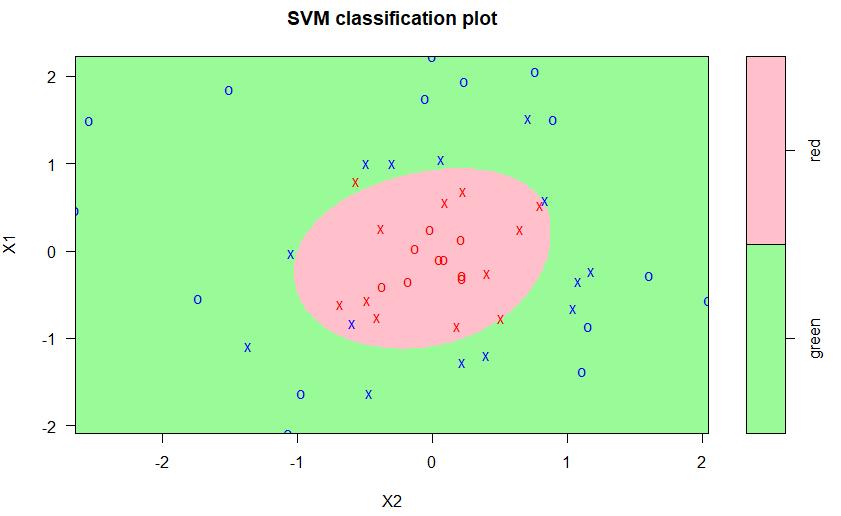
predictionsTest

green red

green 18 11

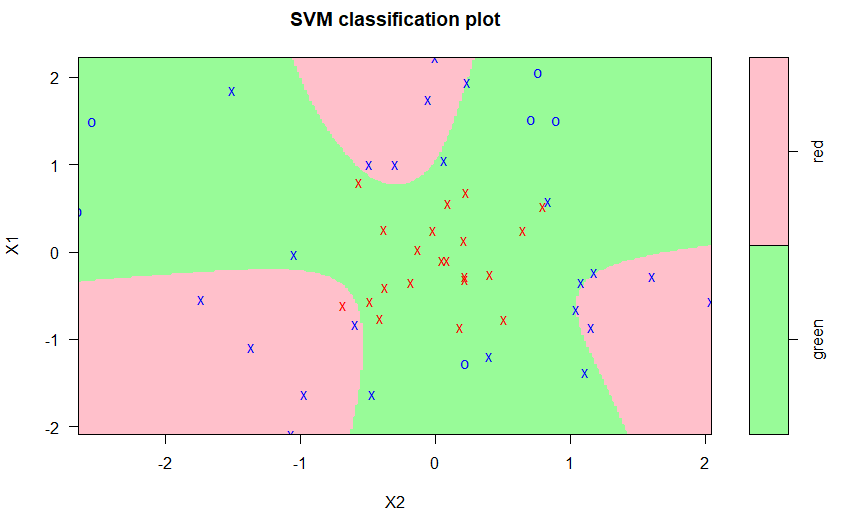
red 0 21

**Radial:**



|  |
| --- |
| predictionsTest  green red  green 28 1  red 1 20 |
|  |
| |  | | --- | |  | |

**Sigmoid:**



predictionsTest

green red

green 15 14

red 20 1

Оптимальное в плане количества ошибок на тестовой выборке ядро – radial.

# **Четвертое задание:**

Среди ядер "polynomial", "radial" и "sigmoid" выберите оптимальное в плане количества ошибок на тестовой выборке.

**Код программы:**

#Unit 4

D\_test=read.table("svmdata4test.txt",header = TRUE, sep="\t")

area.pallete = function(n = 2)

{

cols = rainbow(n)

cols[1:2] = c("PaleGreen", "Pink")

return(cols)

}

svmModelLinear = svm(Colors ~ ., data = D\_test, type = "C-classification", cost = 1, kernel = "polynomial")

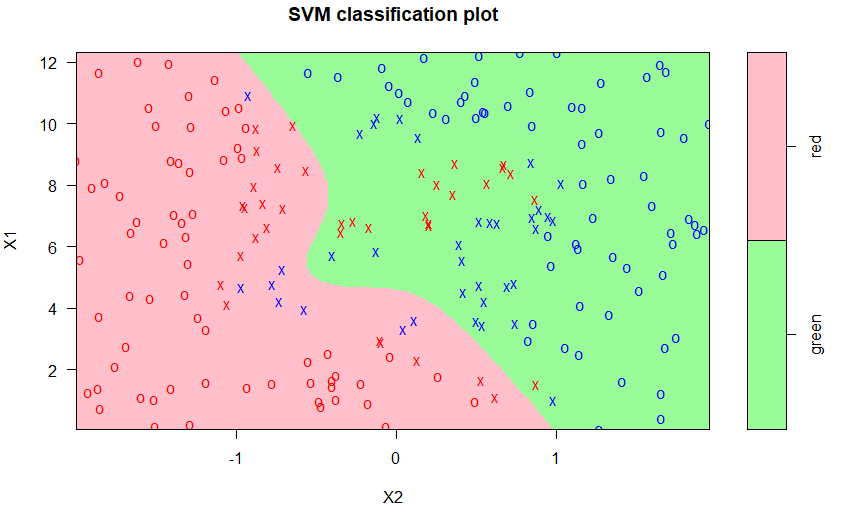
plot(svmModelLinear, D\_test, grid = 250, symbolPalette = symbols.pallete, color.palette = area.pallete)

predictionsTest = predict(svmModelLinear, D\_test)

table(D\_test$"Color", predictionsTest)

**Результаты:**

**Polynomial:**



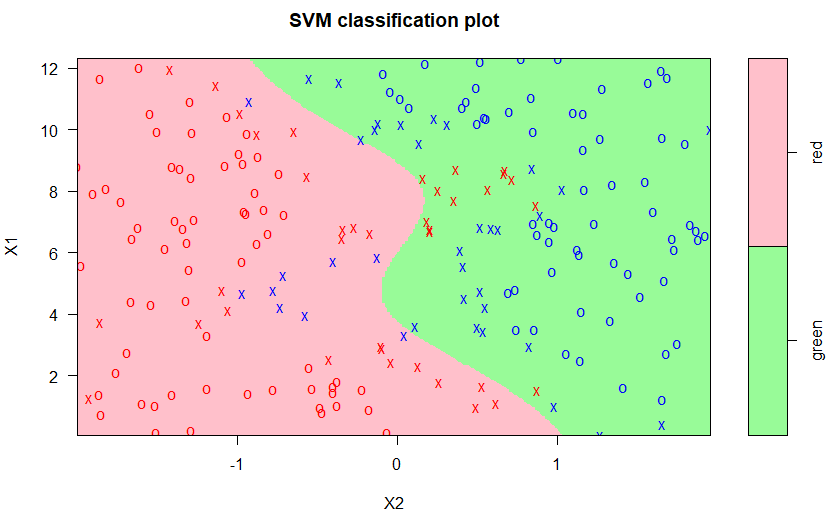
predictionsTest

green red

green 91 8

red 17 84

**Radial:**



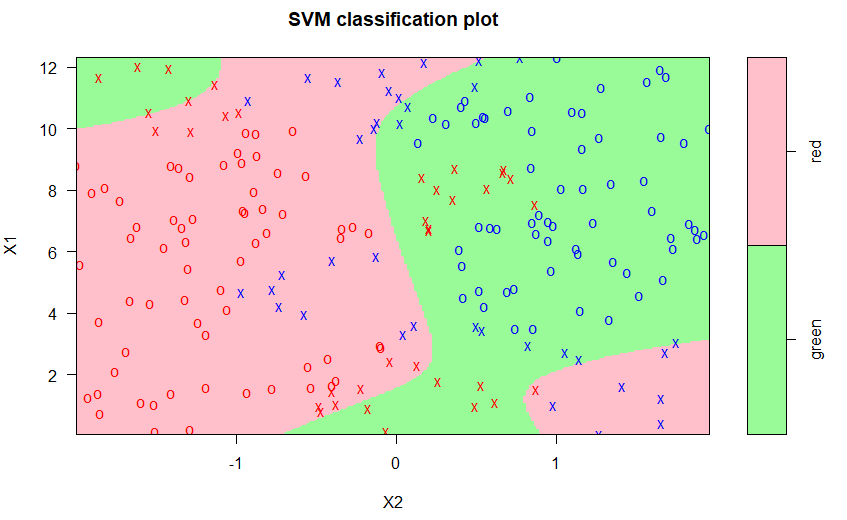
predictionsTest

green red

green 90 9

red 13 88

**Sigmoid:**



predictionsTest

green red

green 73 26

red 22 79

Оптимальное – radial.

# **Пятое задание:**

Среди ядер "polynomial", "radial" и "sigmoid" выберите оптимальное в плане количества ошибок на тестовой выборке. Изменяя значение параметра gamma, продемонстрируйте эффект переобучения, выполните при этом визуализацию разбиения пространства признаков на области.

**Код программы:**

#Unit 5

E\_test=read.table("svmdata4test.txt",header = TRUE, sep="\t")

area.pallete = function(n = 2)

{

cols = rainbow(n)

cols[1:2] = c("PaleGreen", "Pink")

return(cols)

}

svmModelLinear = svm(Colors ~ ., data = E\_test, type = "C-classification", cost = 1, kernel = "sigmoid")

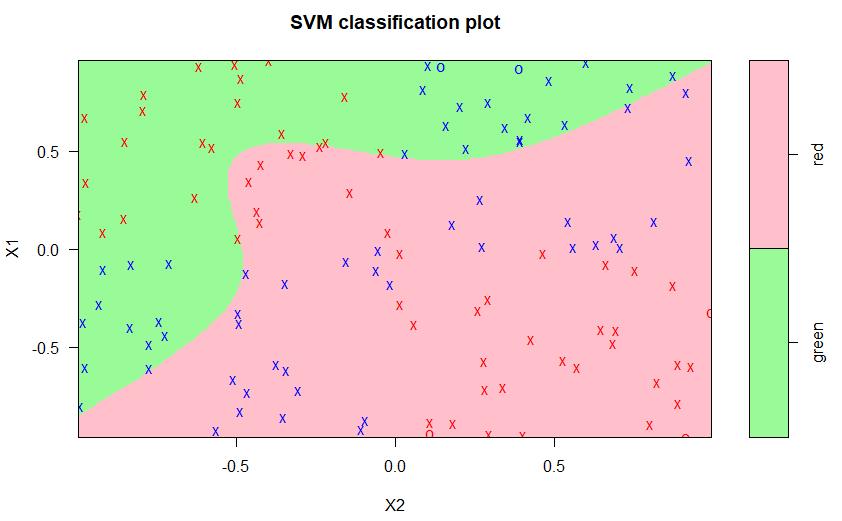
plot(svmModelLinear, E\_test, grid = 250, symbolPalette = symbols.pallete, color.palette = area.pallete)

predictionsTest = predict(svmModelLinear, E\_test)

table(E\_test$"Color", predictionsTest)

**Результаты:**

**Polynomial:**



predictionsTest

green red

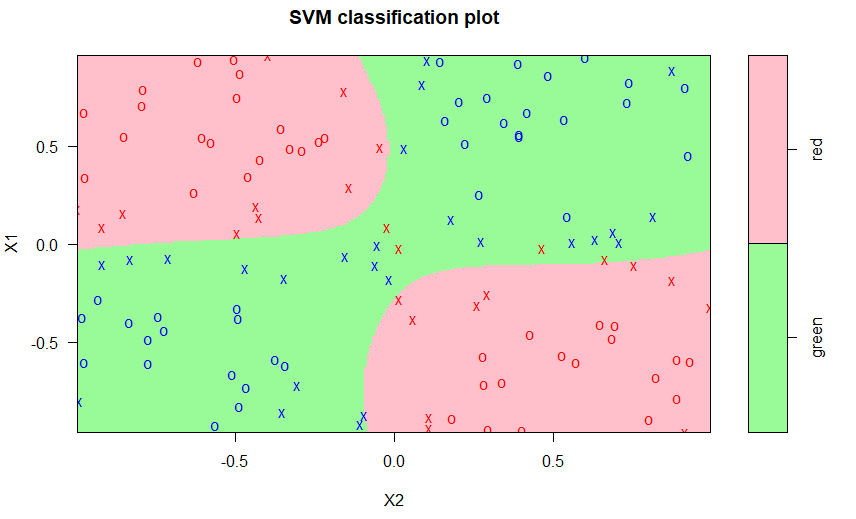
green 30 30

red 21 39

**Radial:**

***Переобучение***

**Gamma = 1**



predictionsTest

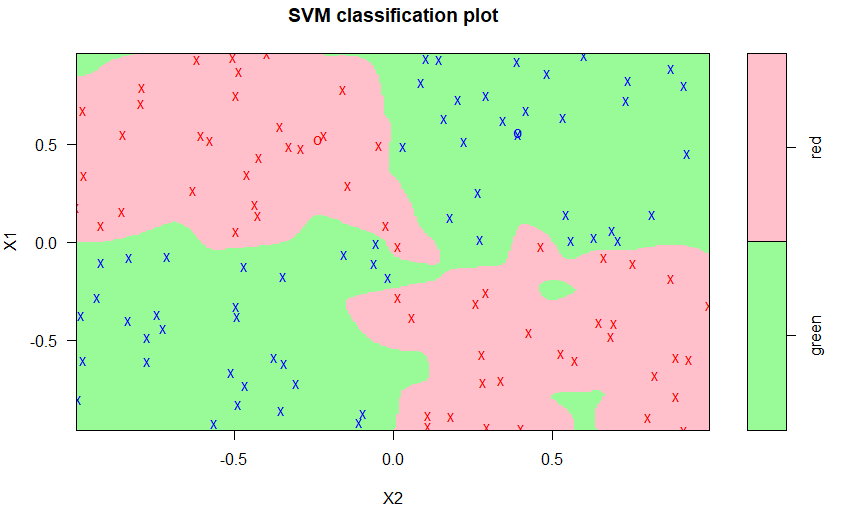
green red

green 59 1

red 4 56

***Переобучение:***

**Gamma = 60**



predictionsTest

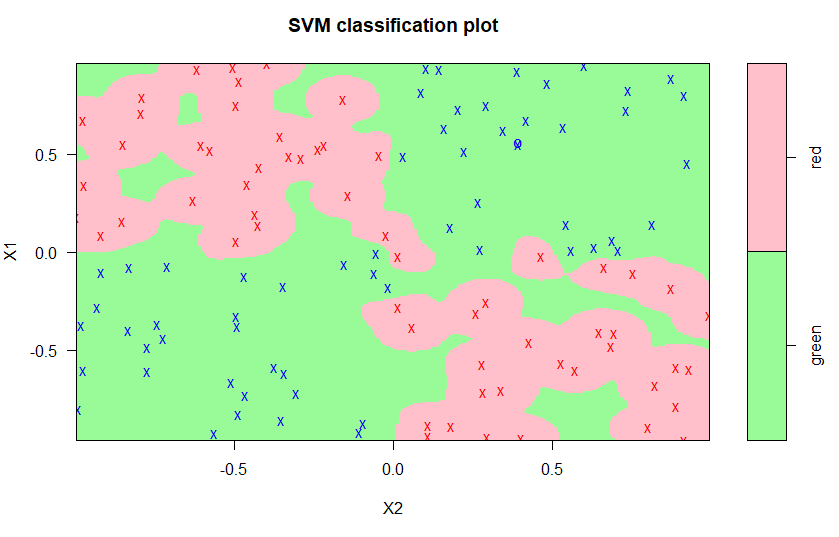
green red

green 60 0

red 0 60

***Переобучение:***

**Gamma = 100**



predictionsTest

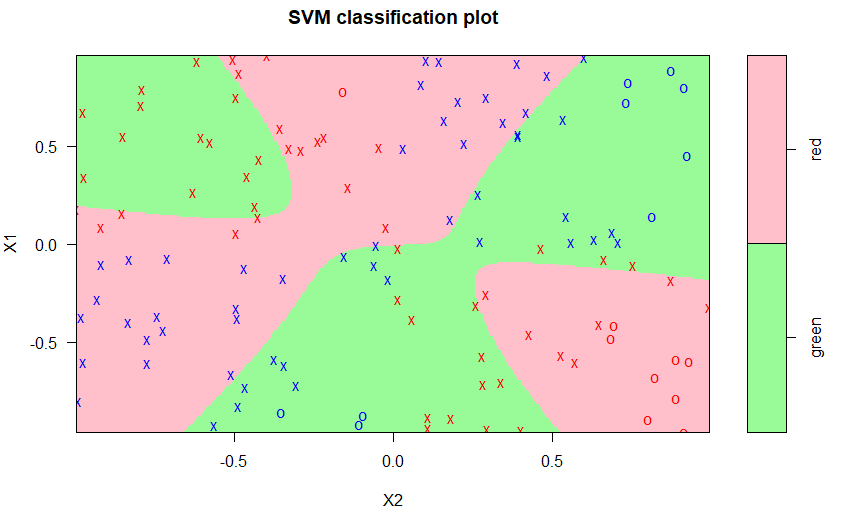
green red

green 60 0

red 0 60

**Sigmoid:**

**:**



predictionsTest

green red

green 28 32

red 29 31

Оптимальное – Radial.

# **Шестое задание:**

Постройте алгоритм метода опорных векторов типа "eps-regression" с параметром C = 1, используя ядро "radial". Отобразите на графике зависимость среднеквадратичной ошибки на обучающей выборке от значения параметра ε. Прокомментируйте полученный результат.

**Код программы:**

#Unit 6

F <-read.table("svmdata6.txt",header = TRUE, sep="\t")

set.seed(0)

plot(F$X, F$Y)

svmModel = svm(F$X, F$Y,type = "eps-regression", cost = 1, kernel = "radial", epsilon = 0.5)

points(F$X[svmModel$index], F$Y[svmModel$index], col = "red")

predctions = predict(svmModel,F$X)

lines(F$X, predctions, col = "dodgerblue", lwd = 2)

lines(F$X, predctions + svmModel$epsilon, col = "cyan")

lines(F$X, predctions - svmModel$epsilon, col = "cyan")

msquared = c()

for(i in seq(0.05, 1.5, by = 0.05)){

svmModel = svm(F$X, F$Y, type = "eps-regression", cost = 1, kernel = "radial", epsilon = i,cross = 1)

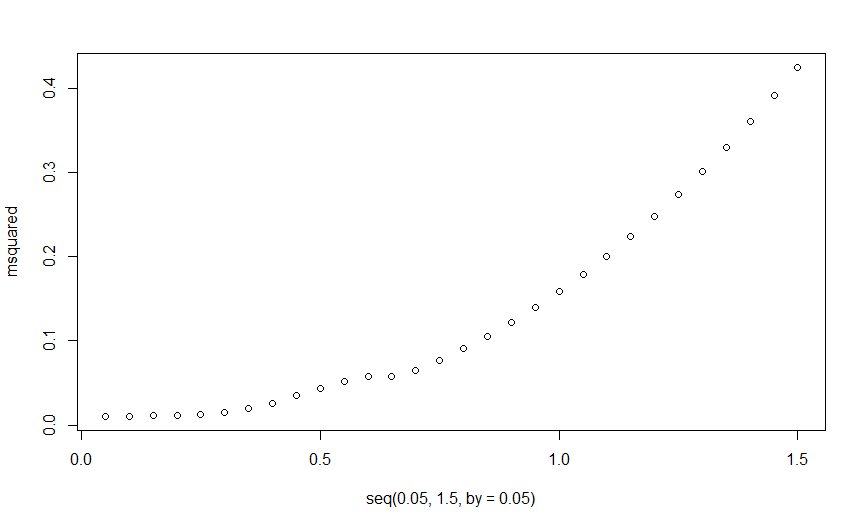
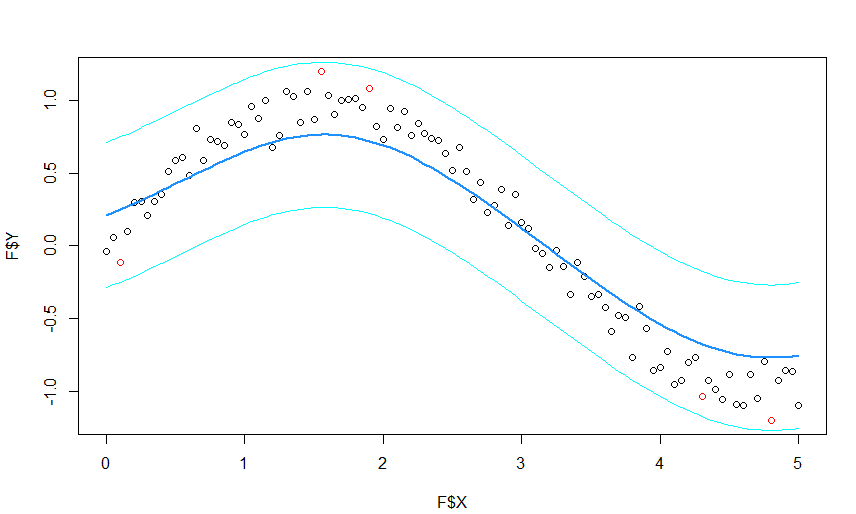
predctions = predict(svmModel, F$X)

msquared =c(msquared,sum((predctions - F$Y) ^ 2) / length(predctions))

}

plot(msquared, x = seq(0.05, 1.5, by = 0.05))

**Результаты:**



С увеличением ε перестаем считать за ошибки важные отклонения, из-за этого ошибка растет.