Assignment 5 - Classification

Saha Debanshee Gopal

November 29, 2016

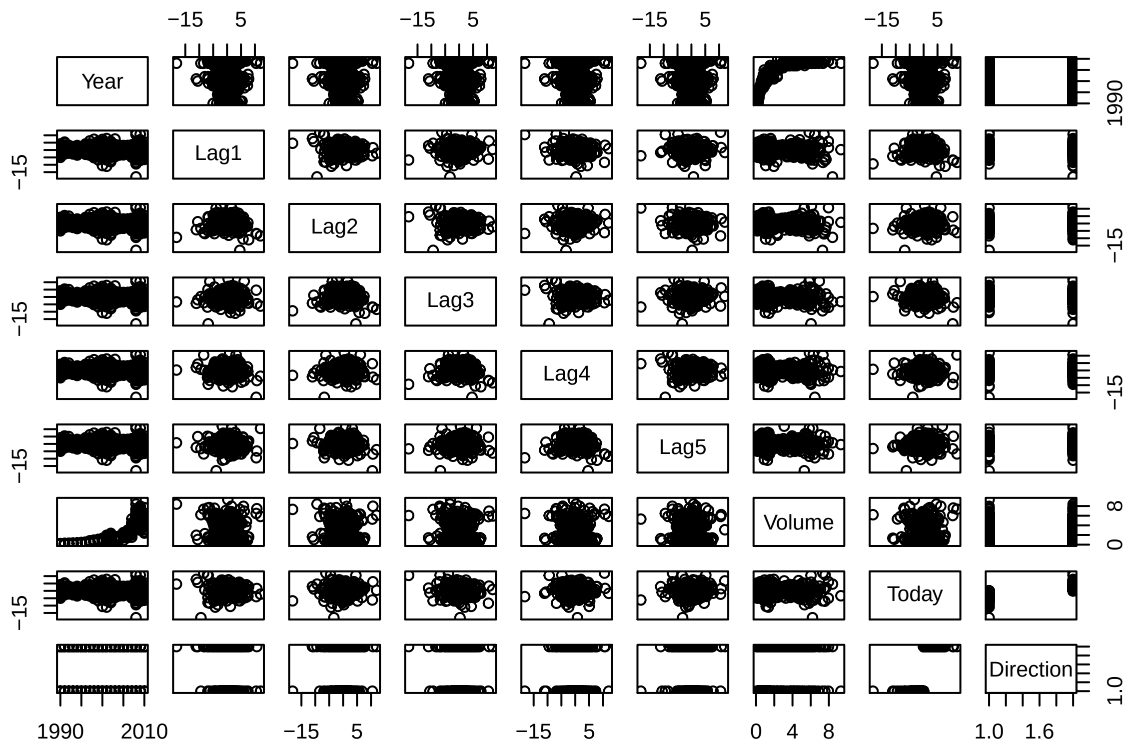
Question 1.

(a) Produce some numerical and graphical summaries of the Weekly data. Do there appear to be any patterns?

```
library(ISLR)
summary(Weekly)
```

```
##       Year           Lag1               Lag2               Lag3
##  Min.   :1990   Min.   :-18.1950   Min.   :-18.1950   Min.   :-18.1950
##  1st Qu.:1995   1st Qu.: -1.1540   1st Qu.: -1.1540   1st Qu.: -1.1580
##  Median :2000   Median :  0.2410   Median :  0.2410   Median :  0.2410
##  Mean   :2000   Mean   :  0.1506   Mean   :  0.1511   Mean   :  0.1472
##  3rd Qu.:2005   3rd Qu.:  1.4050   3rd Qu.:  1.4090   3rd Qu.:  1.4090
##  Max.   :2010   Max.   : 12.0260   Max.   : 12.0260   Max.   : 12.0260
##       Lag4               Lag5              Volume
##  Min.   :-18.1950   Min.   :-18.1950   Min.   :0.08747
##  1st Qu.: -1.1580   1st Qu.: -1.1660   1st Qu.:0.33202
##  Median :  0.2380   Median :  0.2340   Median :1.00268
##  Mean   :  0.1458   Mean   :  0.1399   Mean   :1.57462
##  3rd Qu.:  1.4090   3rd Qu.:  1.4050   3rd Qu.:2.05373
##  Max.   : 12.0260   Max.   : 12.0260   Max.   :9.32821
##      Today          Direction
##  Min.   :-18.1950   Down:484
##  1st Qu.: -1.1540   Up  :605
##  Median :  0.2410
##  Mean   :  0.1499
##  3rd Qu.:  1.4050
##  Max.   : 12.0260
```

```
pairs(Weekly)
```




```r
cor(Weekly[,-9])
```

```
##              Year         Lag1        Lag2        Lag3         Lag4
## Year    1.00000000 -0.032289274 -0.03339001 -0.03000649 -0.031127923
## Lag1   -0.03228927  1.000000000 -0.07485305  0.05863568 -0.071273876
## Lag2   -0.03339001 -0.074853051  1.00000000 -0.07572091  0.058381535
## Lag3   -0.03000649  0.058635682 -0.07572091  1.00000000 -0.075395865
## Lag4   -0.03112792 -0.071273876  0.05838153 -0.07539587  1.000000000
## Lag5   -0.03051910 -0.008183096 -0.07249948  0.06065717 -0.075675027
## Volume  0.84194162 -0.064951313 -0.08551314 -0.06928771 -0.061074617
## Today  -0.03245989 -0.075031842  0.05916672 -0.07124364 -0.007825873
##                Lag5      Volume        Today
## Year   -0.030519101  0.84194162 -0.032459894
## Lag1   -0.008183096 -0.06495131 -0.075031842
## Lag2   -0.072499482 -0.08551314  0.059166717
## Lag3    0.060657175 -0.06928771 -0.071243639
## Lag4   -0.075675027 -0.06107462 -0.007825873
## Lag5    1.000000000 -0.05851741  0.011012698
## Volume -0.058517414  1.00000000 -0.033077783
## Today   0.011012698 -0.03307778  1.000000000
```

Year and Volume appear to have a relationship. No other patterns are discernible.

(b) Use the full data set to perform a logistic regression with Direction as the response and the five lag variables plus Volume as predictors. Use the summary function to print the results. Do any of the predictors appear to be statistically significant? If so, which ones?

```
attach(Weekly)
glm.fit <-  glm(Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume,
                data = Weekly,
                family = binomial)
summary(glm.fit)
```

```
##
## Call:
## glm(formula = Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 +
##     Volume, family = binomial, data = Weekly)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6949  -1.2565   0.9913   1.0849   1.4579
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.26686    0.08593   3.106   0.0019 **
## Lag1        -0.04127    0.02641  -1.563   0.1181
## Lag2         0.05844    0.02686   2.175   0.0296 *
## Lag3        -0.01606    0.02666  -0.602   0.5469
## Lag4        -0.02779    0.02646  -1.050   0.2937
## Lag5        -0.01447    0.02638  -0.549   0.5833
## Volume      -0.02274    0.03690  -0.616   0.5377
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1496.2  on 1088  degrees of freedom
## Residual deviance: 1486.4  on 1082  degrees of freedom
## AIC: 1500.4
##
## Number of Fisher Scoring iterations: 4
```

Lag 2 appears to have some statistical significance with a Pr(>|z|) = 3%.

(c) Compute the confusion matrix and overall fraction of correct predictions. Explain what the confusion matrix is telling you about the types of mistakes made by logistic regression.

```
glm.probs <-  predict(glm.fit, type="response")
glm.pred <-  rep("Down", length(glm.probs))
```

```r
glm.pred[glm.probs>.5] <-  "Up"
table(glm.pred, Direction)
```

```
##         Direction
## glm.pred Down  Up
##     Down   54  48
##     Up    430 557
```

Percentage of currect predictions: (54+557)/(54+557+48+430) = 56.1%. Weeks the market goes up the logistic regression is right most of the time, 557/(557+48) = 92.1%. Weeks the market goes up the logistic regression is wrong most of the time 54/(430+54) = 11.2%.

(d) Now fit the logistic regression model using a training data period from 1990 to 2008, with Lag2 as the only predictor. Compute the confusion matrix and the overall fraction of correct predictions for the held out data (that is, the data from 2009 and 2010).

```r
train <-  (Year < 2009)
Weekly.0910 <-  Weekly[!train,]
glm.fit <-  glm(Direction ~ Lag2, data = Weekly, family = binomial, subset = train)
glm.probs <-  predict(glm.fit, Weekly.0910, type="response")
glm.pred <-  rep("Down", length(glm.probs))
glm.pred[glm.probs>.5] <-  "Up"
Direction.0910 <-  Direction[!train]
table(glm.pred, Direction.0910)
```

```
##         Direction.0910
## glm.pred Down Up
##     Down    9  5
##     Up     34 56
```

```r
mean(glm.pred == Direction.0910)
```

```
## [1] 0.625
```

(e) Repeat (d) using LDA.

```r
library(MASS)
lda.fit <-  lda(Direction ~ Lag2, data = Weekly, subset = train)
lda.pred <-  predict(lda.fit, Weekly.0910)
table(lda.pred$class, Direction.0910)
```

```
##       Direction.0910
##        Down Up
##   Down    9  5
##   Up     34 56
```

```r
mean(lda.pred$class == Direction.0910)
```

```
## [1] 0.625
```

(f) Repeat (d) using QDA.

```
qda.fit <-  qda(Direction ~ Lag2, data = Weekly, subset = train)
qda.class <-  predict(qda.fit, Weekly.0910)$class
table(qda.class, Direction.0910)
```

```
##          Direction.0910
## qda.class Down Up
##      Down    0  0
##      Up     43 61
```

```
mean(qda.class == Direction.0910)
```

```
## [1] 0.5865385
```

A correctness of 58.7% even though it picked Up the whole time.

(g) Repeat (d) using KNN with K = 1.

```
library(class)
train.X <-  as.matrix(Lag2[train])
test.X <-  as.matrix(Lag2[!train])
train.Direction <-  Direction[train]
set.seed(1)
knn.pred <-  knn(train.X, test.X, train.Direction, k = 1)
table(knn.pred, Direction.0910)
```

```
##         Direction.0910
## knn.pred Down Up
##     Down   21 30
##     Up     22 31
```

```
mean(knn.pred == Direction.0910)
```

```
## [1] 0.5
```

(h) Which of these methods appears to provide the best results on this data?

Logistic regression and LDA methods provide similar test error rates.

(i) Experiment with different combinations of predictors, including possible transformations and interactions, for each of the methods. Report the variables, method, and associated confusion matrix that appears to provide the best results on the held out data. Note that you should also experiment with values for K in the KNN classifier.

Logistic regression with Lag2:Lag1

```
glm.fit <-  glm(Direction~Lag2:Lag1, data = Weekly, family = binomial, subset = train)
glm.probs <-  predict(glm.fit, Weekly.0910, type = "response")
glm.pred <-  rep("Down", length(glm.probs))
glm.pred[glm.probs>.5] <-  "Up"
```

```r
Direction.0910 <-  Direction[!train]
table(glm.pred, Direction.0910)
```

```
##         Direction.0910
## glm.pred Down Up
##     Down    1  1
##     Up     42 60
```

```r
mean(glm.pred == Direction.0910)
```

```
## [1] 0.5865385
```

LDA with Lag2 interaction with Lag1

```r
lda.fit <-  lda(Direction ~ Lag2:Lag1, data = Weekly, subset = train)
lda.pred <-  predict(lda.fit, Weekly.0910)
mean(lda.pred$class == Direction.0910)
```

```
## [1] 0.5769231
```

QDA with sqrt(abs(Lag2))

```r
qda.fit <-  qda(Direction ~ Lag2 + sqrt(abs(Lag2)), data = Weekly, subset = train)
qda.class <-  predict(qda.fit, Weekly.0910)$class
table(qda.class, Direction.0910)
```

```
##          Direction.0910
## qda.class Down Up
##      Down   12 13
##      Up     31 48
```

```r
mean(qda.class == Direction.0910)
```

```
## [1] 0.5769231
```

KNN k =10

```r
knn.pred <-  knn(train.X, test.X, train.Direction, k = 10)
table(knn.pred, Direction.0910)
```

```
##         Direction.0910
## knn.pred Down Up
##     Down   17 18
##     Up     26 43
```

```r
mean(knn.pred == Direction.0910)
```

```
## [1] 0.5769231
```

KNN k = 100

```
knn.pred <-  knn(train.X, test.X, train.Direction, k = 100)
table(knn.pred, Direction.0910)
```

```
##          Direction.0910
## knn.pred Down Up
##     Down    9 12
##     Up     34 49
```

```
mean(knn.pred == Direction.0910)
```

```
## [1] 0.5576923
```

Out of these permutations, the original LDA and logistic regression have better performance in terms of test error rate.

Question 2.

a. Remove the observations for whom the salary information is unknown, and then log-transform the salaries.

b. Create a training set consisting of the first 200 observations, and a test set consisting of the remaining observations.

c. Perform boosting on the training set with 1000 trees for a range of values of the shrinkage parameter ??. Produce a plot with different shrinkage values on the x-axis and the corresponding training set MSE on the y-axis.

d. Produce a plot with different shrinkage values on the x-axis and the corresponding test set MSE on the y-axis.

Ans a-d: We prepare the data and apply boosting on a range of shrinkage parameters, plotting both the train and test MSE values resulting from the proposed shrinkage. The resulting plot is displayed

```
library(ISLR)
full.hit <- Hitters[!is.na(Hitters$Salary),]
full.hit$Salary <- log(full.hit$Salary)

tr <- sample(1:nrow(full.hit), 200)
hit.tr <- full.hit[tr,]
hit.te <- full.hit[-tr,]

shrinkage <- seq(0,0.03,0.00005)
tr.mse <- array(NA,length(shrinkage))
te.mse <- array(NA,length(shrinkage))

library(gbm)
```

```
## Loading required package: survival

## Loading required package: lattice

## Loading required package: splines

## Loading required package: parallel

## Loaded gbm 2.1.1
```

```
for (i in 1:length(shrinkage)) {
  hit.boost <- gbm(Salary ~., data=hit.tr, distribution='gaussian',
        n.trees=1000, shrinkage=shrinkage[i], verbose=F)
    tr.mse[i] <- mean((predict(hit.boost, hit.tr, n.trees=1000) - hit.tr$Salary)^2)
    te.mse[i] <- mean((predict(hit.boost, hit.te, n.trees=1000) - hit.te$Salary)^2)
}
```

e. Compare the test MSE of boosting to the test MSE that results from applying two of the regression approaches.

Ans: We compare the boosting's minimum test error of 0.19 with the test error with that of a simple linear model, and also lasso regression. Linear regression obtains a slightly better MSE than lasso, which is unsurprising as we have a much larger number of samples than covariates so we don't really need to enforce sparsity. Nevertheless the MSE of linear regression, 0.39, is vastly greater than boosting's 0.19. Boosting wins.

```
min(te.mse)
```

```
## [1] 0.1595733
```

Linear Regression

```
hit.lm <- lm(Salary ~., hit.tr)
mean((predict(hit.lm, hit.te) - hit.te$Salary)^2)
```

```
## [1] 0.4146355
```

```
library(glmnet)
```

```
## Loading required package: Matrix

## Loading required package: foreach

## Loaded glmnet 2.0-5
```

```
hit.las.cv <- cv.glmnet(as.matrix(hit.tr[,-c(19,20,14,15)]),
                        as.matrix(hit.tr[,19]), alpha=1)

hit.las <- glmnet(as.matrix(hit.tr[,-c(19,20,14,15)]),
                  as.matrix(hit.tr[,19]), alpha=1, lambda=hit.las.cv$lambda.min)
mean((predict(hit.las,
              as.matrix(hit.te[,-c(19,20,14,15)])) - hit.te$Salary)^2)
```

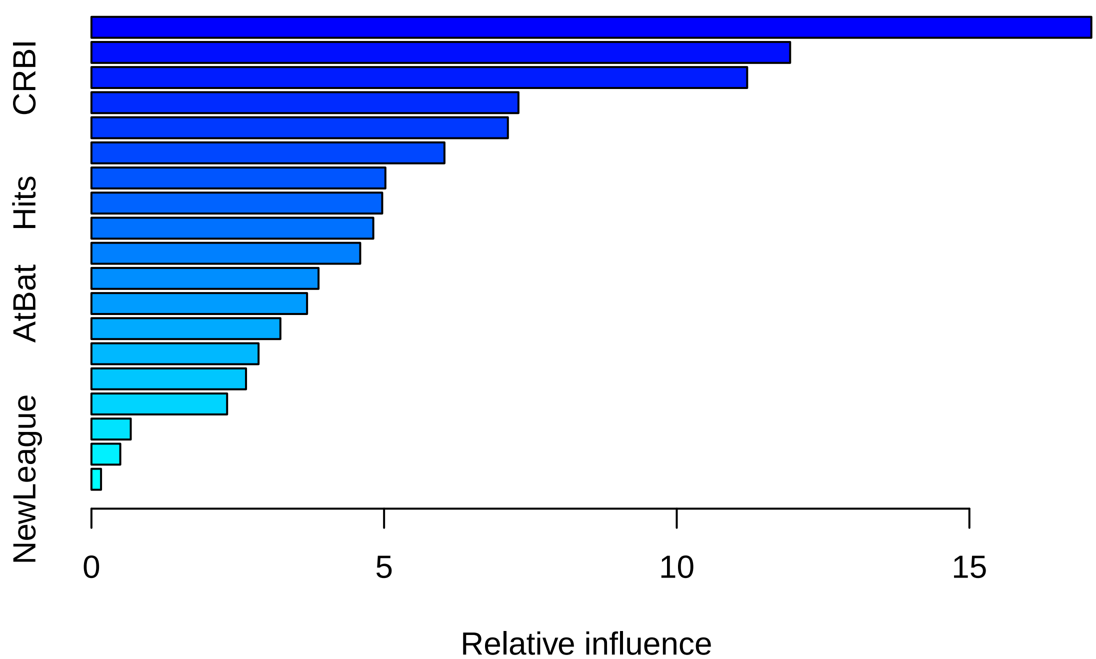
```
## [1] 0.4153775
```

f. Which variables appear to be the most important predictors in the boosted model?

g. Now apply bagging to the training set. What is the test set MSE for this approach?

Ans f,g: We investigate the variable importance by plotting our boosting model, using the summary() command.

```
summary(hit.boost)
```




```
##                 var    rel.inf
## CAtBat       CAtBat 17.0837641
## CRuns         CRuns 11.9372686
## CRBI           CRBI 11.2033430
## PutOuts     PutOuts  7.2957328
## CHits         CHits  7.1152143
## RBI             RBI  6.0312491
## CHmRun       CHmRun  5.0225992
## Hits           Hits  4.9678668
## HmRun         HmRun  4.8159081
## Years         Years  4.5920290
## Walks         Walks  3.8798353
## AtBat         AtBat  3.6838350
## CWalks       CWalks  3.2285088
```

```
## Errors       Errors  2.8551563
## Assists     Assists  2.6410109
## Runs           Runs  2.3180210
## Division   Division  0.6715143
## League       League  0.4931718
## NewLeague NewLeague  0.1639714
```

We see the three most influential variables are CRuns, CAtBat, CRBI. These are all career statistics; so how many runs the player has made, career bats, runs batted in, in respective order, the most influential information pertaining to his salary.

We apply bagging to the training set, using random forests.

```r
library(randomForest)
```

```
## randomForest 4.6-12
```

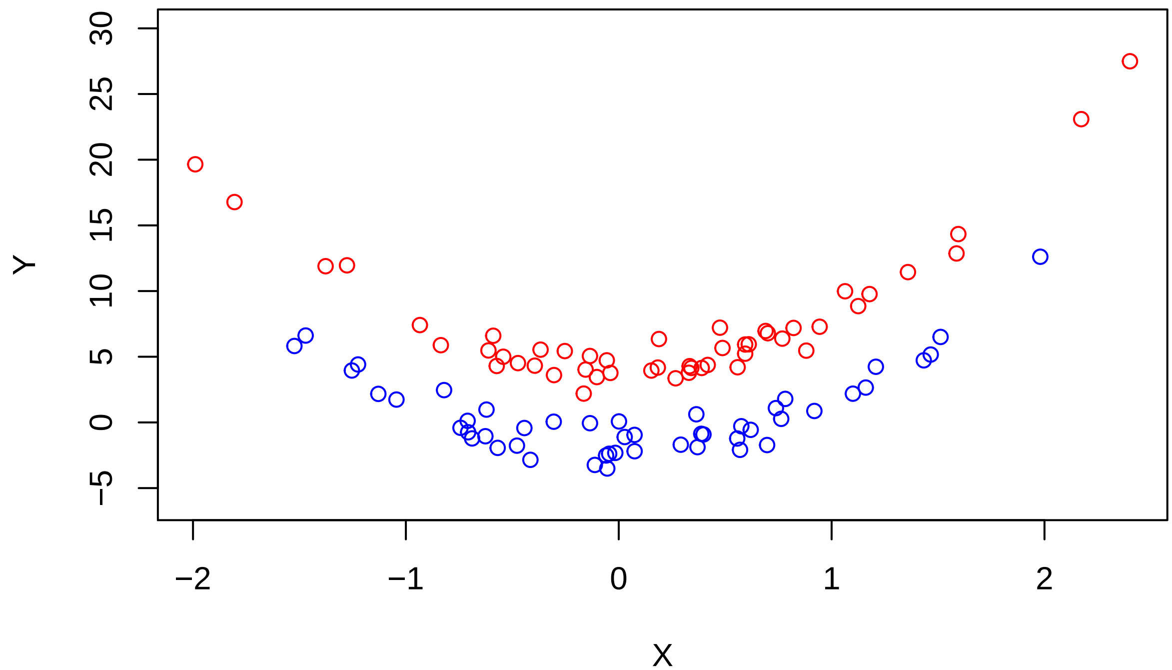
```
## Type rfNews() to see new features/changes/bug fixes.
```

```r
hit.rf <- randomForest(Salary ~., hit.tr, mtry=(ncol(hit.tr)-1), importance=T)
mean((predict(hit.rf, hit.te) - hit.te$Salary)^2)
```

```
## [1] 0.1533271
```

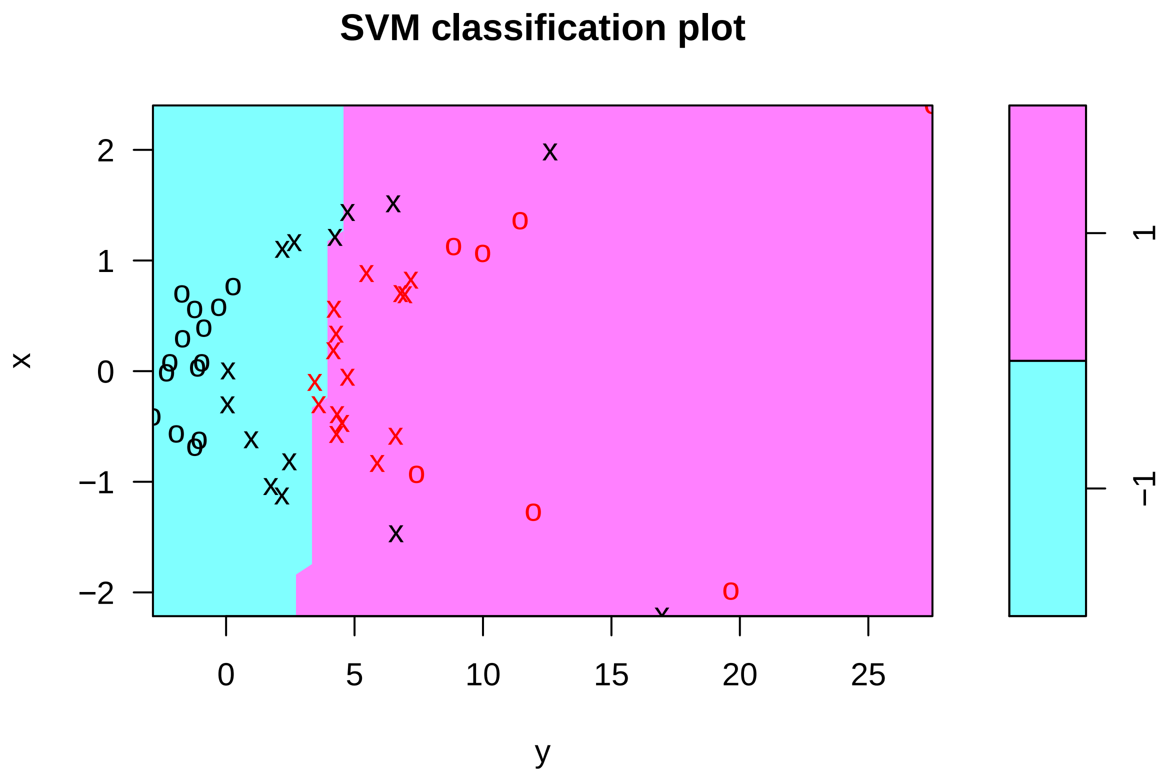
Question 3.

```r
library(e1071)
set.seed(1)
x <- rnorm(100)
y <- 4 * x^2 + 1 + rnorm(100)
class <- sample(100, 50)
y[class] <- y[class] + 3
y[-class] <- y[-class] - 3
plot(x[class], y[class], col = "red", xlab = "X", ylab = "Y", ylim = c(-6, 30))
points(x[-class], y[-class], col = "blue")
```



We fit a support vector classifier on the training data

```
z <- rep(-1, 100)
z[class] <- 1
data <- data.frame(x = x, y = y, z = as.factor(z))
train <- sample(100, 50)
data.train <- data[train, ]
data.test <- data[-train, ]
svm.linear <- svm(z ~ ., data = data.train, kernel = "linear", cost = 10)
plot(svm.linear, data.train)
```

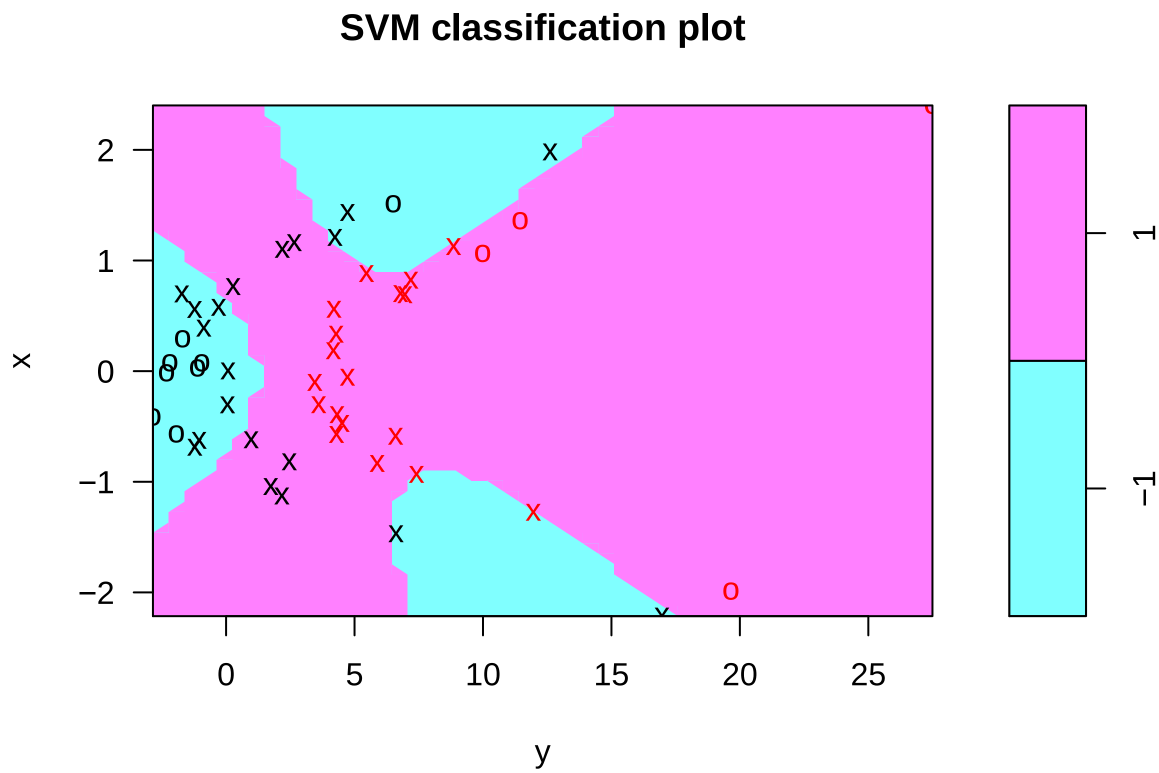



```r
table(predict = predict(svm.linear, data.train), truth = data.train$z)
```

```
##        truth
## predict -1  1
##      -1 22  0
##      1   6 22
```

The support vector classifier makes 6 errors on the training data. Next, we fit a support vector machine with a polynomial kernel.

```r
svm.poly <- svm(z ~ ., data = data.train, kernel = "polynomial", cost = 10)
plot(svm.poly, data.train)
```

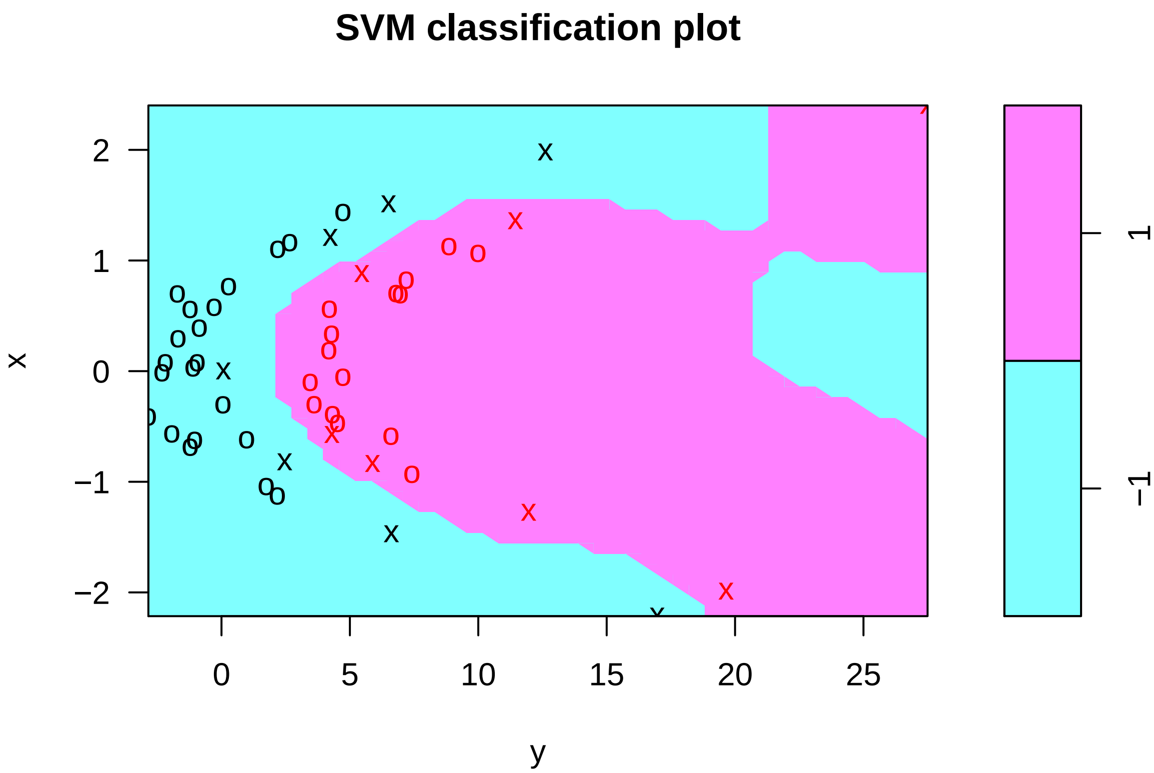



```
table(predict = predict(svm.poly, data.train), truth = data.train$z)
```

```
##         truth
## predict -1  1
##      -1 19  0
##      1   9 22
```

The support vector machine with a polynomial kernel of degree 3 makes 9 errors on the training data. Finally, we fit a support vector machine with a radial kernel and a gamma of 1.

```
svm.radial <- svm(z ~ ., data = data.train, kernel = "radial", gamma = 1, cost = 10)
plot(svm.radial, data.train)
```

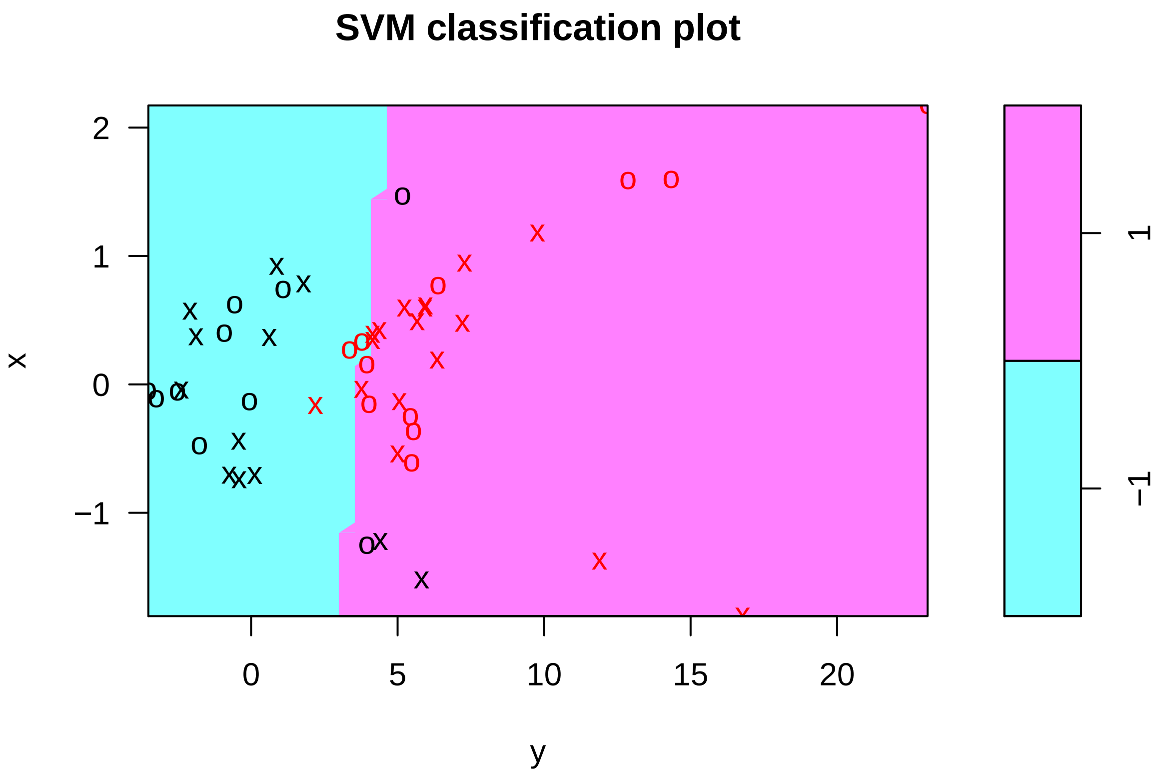



```
table(predict = predict(svm.radial, data.train), truth = data.train$z)
```

```
##        truth
## predict -1  1
##      -1 28  0
##      1   0 22
```

The support vector machine with a radial kernel makes 0 error on the training data.Now, we check how these models fare when applied to the test data.

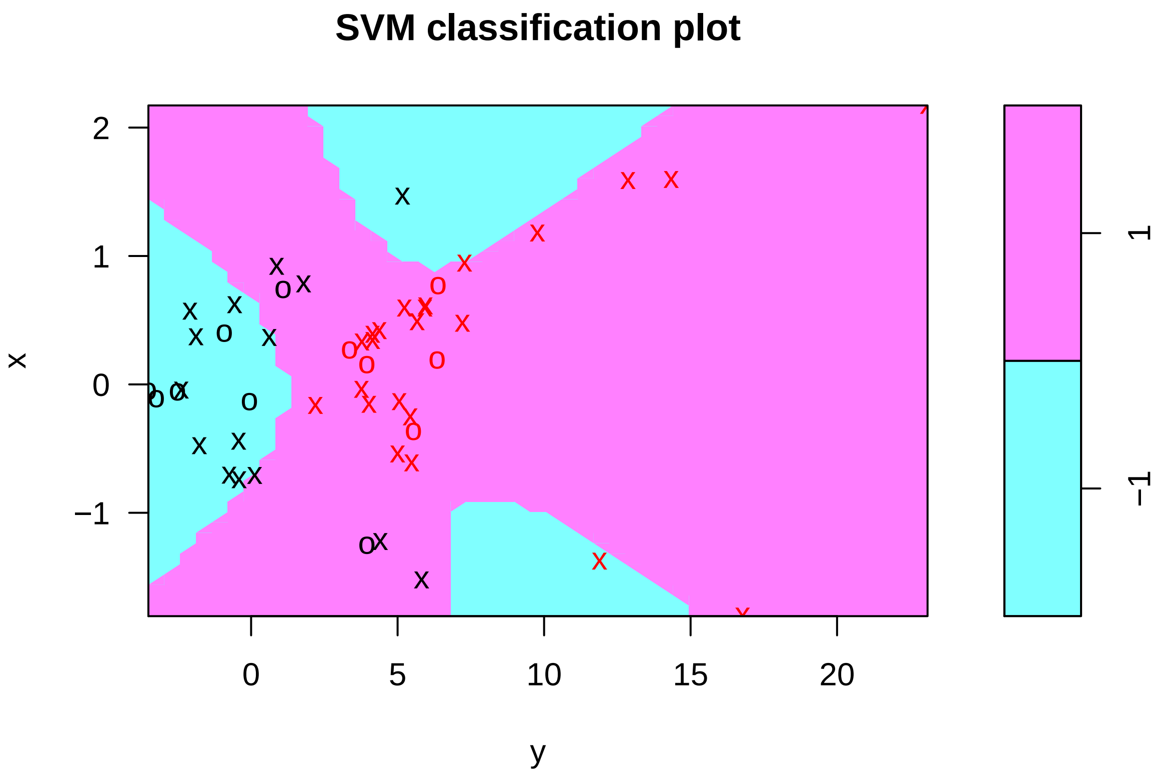
```
plot(svm.linear, data.test)
```




```
table(predict = predict(svm.linear, data.test), truth = data.test$z)
```

```
##         truth
## predict -1  1
##      -1 18  2
##      1   4 26
```

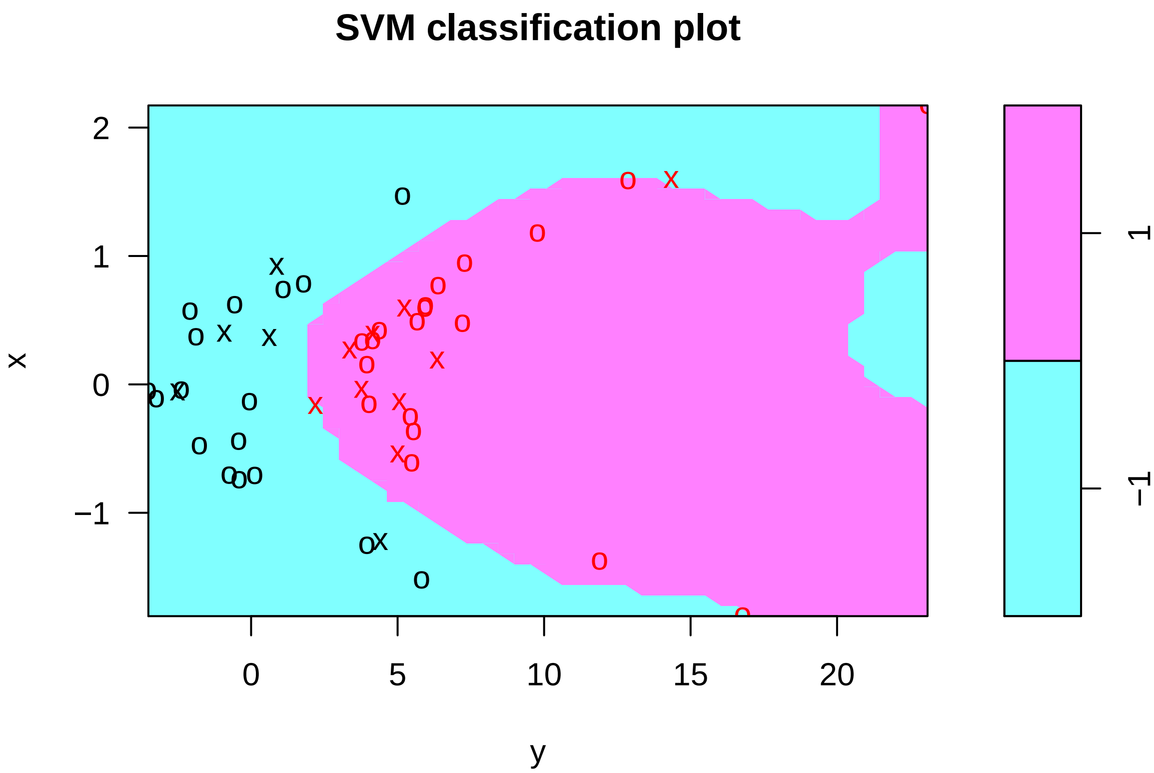
```
plot(svm.poly, data.test)
```




```
table(predict = predict(svm.poly, data.test), truth = data.test$z)
```

```
##         truth
## predict -1  1
##      -1 14  1
##      1   8 27
```

```
plot(svm.radial, data.test)
```




```
table(predict = predict(svm.radial, data.test), truth = data.test$z)
```

```
##        truth
## predict -1  1
##      -1 22  1
##      1   0 27
```

We may see that the linear, polynomial and radial support vector machines classify respectively 9, 6 and 1 observations incorrectly. So, radial kernel is the best model in this setting.