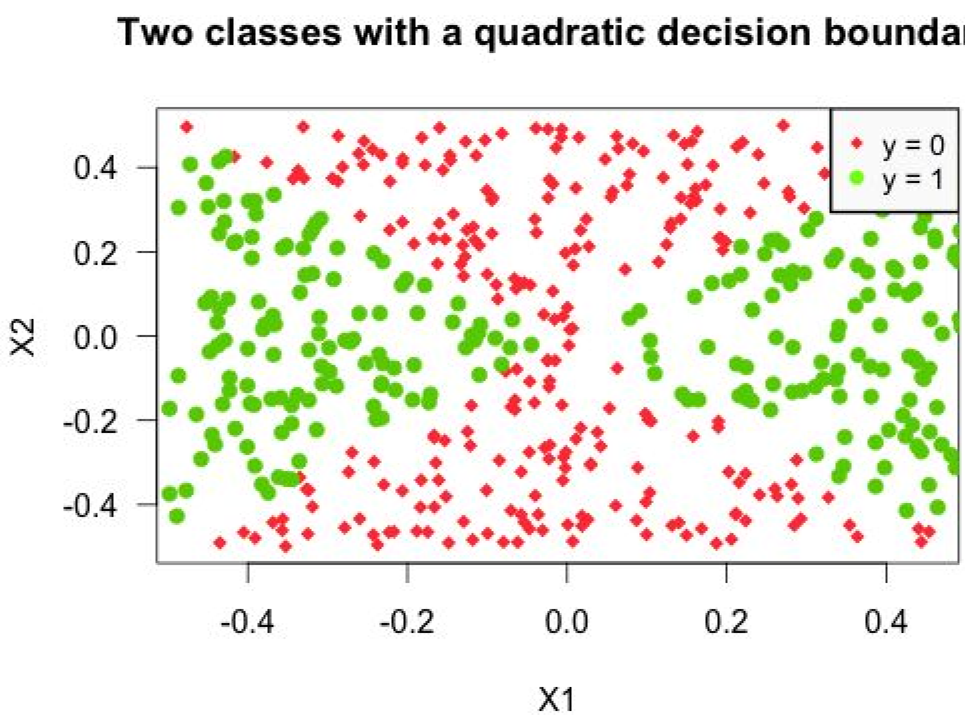
ISLR Chapter 9 Ex. 5
Tatiana Romanchishina

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

(a) 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*( x1 ^2 - x2 ^2 > 0)
```

(b) Plot the observations, colored according to their class labels. Your plot should display X 1 on the x-axis, and X 2 on the y-axis.



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

Call:
glm(formula = y ~ x1 + x2, family = "binomial")

Deviance Residuals:

Min	1Q	Median	3Q	Max
-1.179	-1.139	-1.112	1.206	1.257

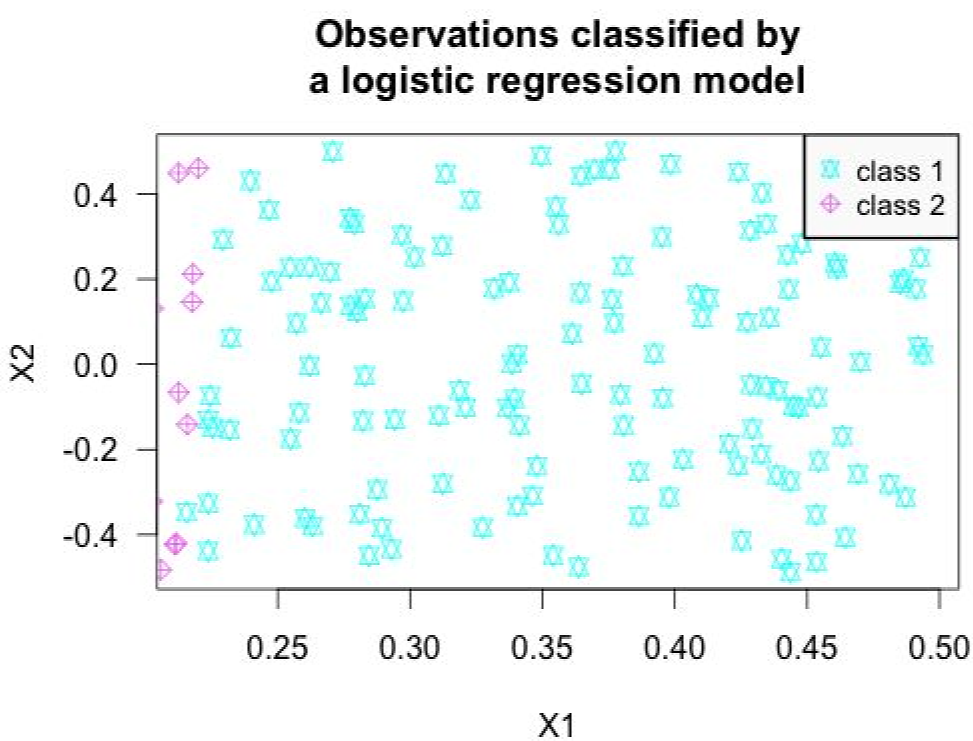
Coefficients:

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	-0.087260	0.089579	-0.974	0.330
x1	0.196199	0.316864	0.619	0.536
x2	-0.002854	0.305712	-0.009	0.993

(Dispersion parameter for binomial family taken to be 1)
Null deviance: 692.18 on 499 degrees of freedom
Residual deviance: 691.79 on 497 degrees of freedom
AIC: 697.79
Number of Fisher Scoring iterations: 3

The variables do not appear to have a significant relationship with the response variable.

(d) 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.



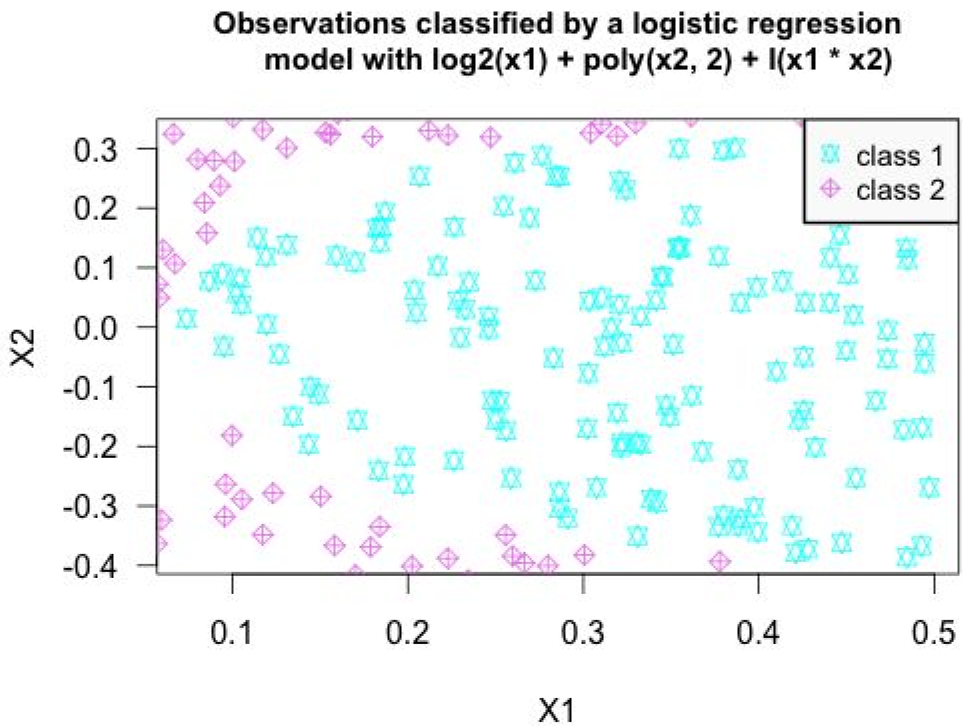
I used the probability threshold of .489 to produce two classes. The decision boundary seems linear and clear.

(e) Now fit a logistic regression model to the data using nonlinear functions of X1 and X2 as predictors (e.g. $X1^2$, X1 × X2 , log(X2), and so forth).

```
Call:
glm(formula = y ~ log2(x1) + poly(x2, 2) + I(x1 * x2), family = "binomial")

Deviance Residuals:
     Min        1Q      Median       3Q       Max
-1.98542  -0.06837   0.00753   0.09873   2.37857

Coefficients:
                 Estimate   Std. Error z value Pr(>|z|)
(Intercept)       10.1540     1.9023   5.338   9.41e-08 ***
log2(x1)           4.8614     0.9048   5.373   7.75e-08 ***
poly(x2, 2)1     -41.0661    24.2062  -1.697   0.0898 .
poly(x2, 2)2    -104.6153    18.6898  -5.597   2.18e-08 ***
I(x1 * x2)        18.8225    10.5556   1.783   0.0746 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)
 Null deviance: 318.830  on 229  degrees of freedom
Residual deviance:  65.559  on 225  degrees of freedom
  (270 observations deleted due to missingness)
AIC: 75.559
Number of Fisher Scoring iterations: 8
```

I used log2(x1) + poly(x2, 2) + I(x1 * x2) as independent variables, and all of the relationships appear very significant.

(f) 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 non-linear. 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 boundary is nonlinear, but does not resemble the original decision boundary.

(g) 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.

Parameter tuning of ‘svm’:

- sampling method: 10-fold cross validation

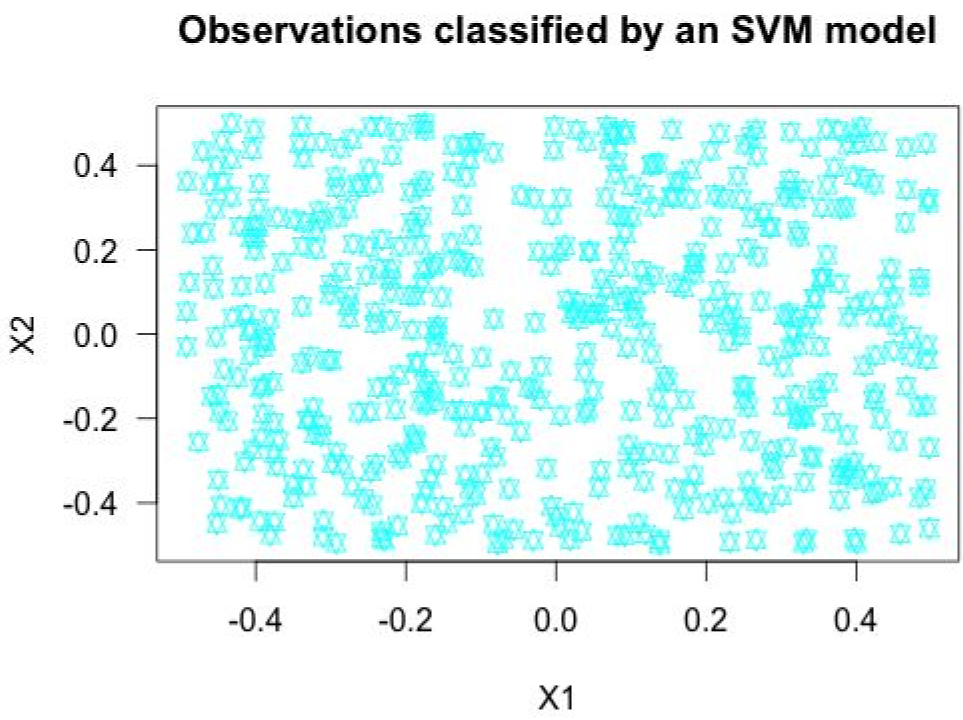
- best parameters:
 cost
  0.1

- best performance: 0.478

- Detailed performance results:
   cost error dispersion
1 1e-05 0.484 0.04599517
2 1e-04 0.484 0.04599517
3 1e-03 0.484 0.04599517
4 1e-02 0.484 0.04599517
5 1e-01 0.478 0.05996295

6 1e+00 0.492 0.03794733
7 1e+01 0.492 0.03794733
8 5e+01 0.492 0.03794733
9 1e+02 0.492 0.03794733

Best performance is close to other results and is a pretty high error rate. All the observations are classified into one and the same class.



(h) 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.

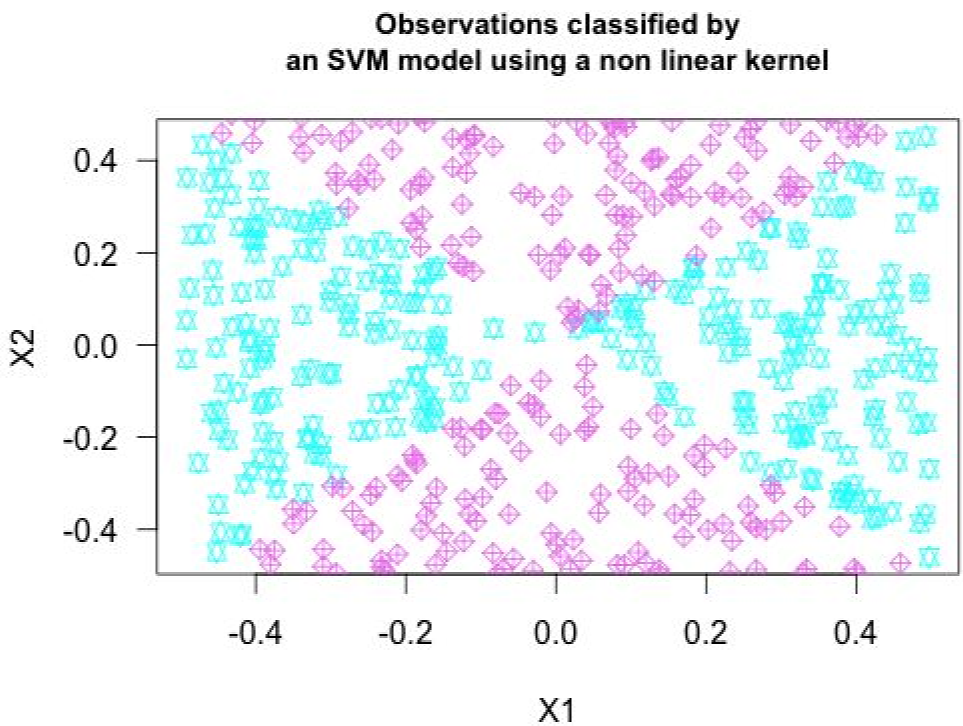
Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best parameters:
 cost
   50

- best performance: 0.022

- Detailed performance results:
cost error dispersion
1 1e-05 0.456 0.15770577
2 1e-04 0.456 0.15770577
3 1e-03 0.456 0.15770577
4 1e-02 0.456 0.15770577
5 1e-01 0.084 0.03502380
6 1e+00 0.056 0.02270585
7 1e+01 0.026 0.02674987
8 5e+01 0.022 0.01988858
9 1e+02 0.024 0.02065591



(i) Comment on your results.

The SVM model with non-linear kernel produced the best result. With the help of the tune() function, I was able to quickly find out the best tuning parameter from a range of parameters.