Arbeitsblatt 6

Aufgabe 1: Orangensaft

Wir wollen vorhersagen, welches Produkt ein Kunde kauft, basierend auf einigen Eigenschaften des Kunden und des Produkts. Dazu verwenden wir einen Datensatz, der 1070 Käufe enthält, bei denen der Kunde entweder Citrus Hill (CH) oder Minute Maid (MM) Orangensaft gekauft hat. Die Klassifizierungsaufgabe besteht darin, vorherzusagen, ob ein Kunde Orangensaft der Marke CH oder MM kauft (Zielvariable `Purchase`). Der Datensatz OJ ist im ISLR-Paket enthalten oder auf Moodle zu finden. (Die Aufgabe stammt aus dem Buch ISLR.)

a) Verwenden Sie die OJ-Daten und erstellen Sie ein Trainingsset mit einer Zufallsstichprobe von 800 Beobachtungen und ein Testset mit den restlichen Beobachtungen.

```
library(ISLR)
library(caret)
data(OJ)
dim(OJ)
```

```
[1] 1070   18
```

```
set.seed(1)
train <- createDataPartition(y=OJ$Purchase, p=0.8, list=F)
OJ.train <- OJ[train, ]
OJ.test <- OJ[-train, ]
```

b) Passen Sie einen Support-Vektor-Klassifikator an die Trainingsdaten an, indem Sie in der `train`-Funktion `method='svmLinear2'` und Kosten `cost = 0.01` verwenden. Zielvariable `Purchase` mit allen anderen Variablen als Prädiktoren. Verwenden Sie `..$finalModel`, um das endgültige Modell zu sehen.

```
x <- trainControl(method="none")
model_svm <- train(Purchase ~ ., data = OJ.train, method='svmLinear2', tuneGrid=data.frame(cost=c(0.
model_svm$finalModel
```

```
Call:
svm.default(x = as.matrix(x), y = y, kernel = "linear", cost = param$cost,
    probability = classProbs)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  0.01

Number of Support Vectors:  469
```

```
# Support vector classifier creates 469 support vectors
# out of 800 training points.
```

c) Was ist die Genauigkeit von Training- und Testdaten?

```
train.pred <- predict(model_svm, OJ.train)
confusionMatrix(data=train.pred, reference = OJ.train$Purchase)
```

```
Confusion Matrix and Statistics

          Reference
Prediction  CH  MM
        CH 467  83
        MM  56 251

               Accuracy : 0.8378
                 95% CI : (0.8114, 0.8619)
    No Information Rate : 0.6103
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.654

 Mcnemar's Test P-Value : 0.02743

            Sensitivity : 0.8929
            Specificity : 0.7515
         Pos Pred Value : 0.8491
         Neg Pred Value : 0.8176
             Prevalence : 0.6103
         Detection Rate : 0.5449
   Detection Prevalence : 0.6418
      Balanced Accuracy : 0.8222

       'Positive' Class : CH
```

```
test.pred <- predict(model_svm, OJ.test)
confusionMatrix(data=test.pred, reference = OJ.test$Purchase)
```

```
Confusion Matrix and Statistics

          Reference
Prediction  CH  MM
        CH 114  23
        MM  16  60

               Accuracy : 0.8169
                 95% CI : (0.7583, 0.8664)
    No Information Rate : 0.6103
    P-Value [Acc > NIR] : 6.226e-11

                  Kappa : 0.6091

 Mcnemar's Test P-Value : 0.3367

            Sensitivity : 0.8769
            Specificity : 0.7229
```

```
         Pos Pred Value : 0.8321
         Neg Pred Value : 0.7895
             Prevalence : 0.6103
         Detection Rate : 0.5352
   Detection Prevalence : 0.6432
      Balanced Accuracy : 0.7999

       'Positive' Class : CH
```

Der Fehler in den Testdaten ist nur leicht grösser als der Fehler in den Trainingsdaten.

d) Optimieren Sie den Cost-Parameter. Berücksichtigen Sie für das Argument `tuneGrid` Werte im Bereich von 0.01 bis 10.

```
repeats <- 3
numbers <- 10
x <- trainControl(method="repeatedcv",
                  number=numbers,
                  repeats=repeats)
model_svm2 <- train(Purchase ~ ., data = OJ.train, method="svmLinear2",
                    tuneGrid=data.frame(cost=10^seq(-2, 1, by = 0.25)),
                    metric= "Accuracy", trControl=x)
model_svm2
```

```
Support Vector Machines with Linear Kernel

857 samples
 17 predictor
  2 classes: 'CH', 'MM'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 771, 770, 770, 772, 772, 772, ...
Resampling results across tuning parameters:

  cost         Accuracy   Kappa
   0.01000000  0.8319788  0.6419700
   0.01778279  0.8300317  0.6388814
   0.03162278  0.8315775  0.6419509
   0.05623413  0.8319605  0.6427261
   0.10000000  0.8300044  0.6387797
   0.17782794  0.8303965  0.6396165
   0.31622777  0.8292337  0.6371563
   0.56234133  0.8300090  0.6387677
   1.00000000  0.8292381  0.6371399
   1.77827941  0.8288687  0.6357838
   3.16227766  0.8292655  0.6364224
   5.62341325  0.8280981  0.6340244
  10.00000000  0.8284857  0.6348504

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was cost = 0.01.
```

e) Was ist die Genauigkeit von Training- und Testdaten mit dem optimierten Cost-Parameter?

```
train.pred <- predict(model_svm2, OJ.train)
confusionMatrix(data=train.pred, reference = OJ.train$Purchase)
```

```
Confusion Matrix and Statistics

          Reference
Prediction  CH  MM
        CH 467  83
        MM  56 251

               Accuracy : 0.8378
                 95% CI : (0.8114, 0.8619)
    No Information Rate : 0.6103
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.654

 Mcnemar's Test P-Value : 0.02743

            Sensitivity : 0.8929
            Specificity : 0.7515
         Pos Pred Value : 0.8491
         Neg Pred Value : 0.8176
             Prevalence : 0.6103
         Detection Rate : 0.5449
   Detection Prevalence : 0.6418
      Balanced Accuracy : 0.8222

       'Positive' Class : CH
```

```
test.pred <- predict(model_svm2, OJ.test)
confusionMatrix(data=test.pred, reference = OJ.test$Purchase)
```

```
Confusion Matrix and Statistics

          Reference
Prediction  CH  MM
        CH 114  23
        MM  16  60

               Accuracy : 0.8169
                 95% CI : (0.7583, 0.8664)
    No Information Rate : 0.6103
    P-Value [Acc > NIR] : 6.226e-11

                  Kappa : 0.6091

 Mcnemar's Test P-Value : 0.3367

            Sensitivity : 0.8769
```

```
            Specificity : 0.7229
         Pos Pred Value : 0.8321
         Neg Pred Value : 0.7895
             Prevalence : 0.6103
         Detection Rate : 0.5352
   Detection Prevalence : 0.6432
      Balanced Accuracy : 0.7999

       'Positive' Class : CH
```

f) Zeichnen Sie die ROC-Kurve für den optimierten Klassifizierer auf den Testdaten. Damit sie in der `predict` den `typ='prob'` verwenden können, müssen Sie hier in `trainControl` `classProbs = TRUE` setzen, da SVM standardmässig keine Wahrscheinlichkeitsvorhersagen liefert.

```
x <- trainControl(method="repeatedcv",
                  number=numbers,
                  repeats=repeats,
                  classProbs = TRUE)
model_svm3 <- train(Purchase ~ ., data = OJ.train, method="svmLinear2",
                    tuneGrid=data.frame(cost=c(0.01)),  trControl=x)
pred_prob <- predict(model_svm3, OJ.test, type='prob')
library(pROC)
res.roc <- roc(OJ.test$Purchase, pred_prob$'CH')
```

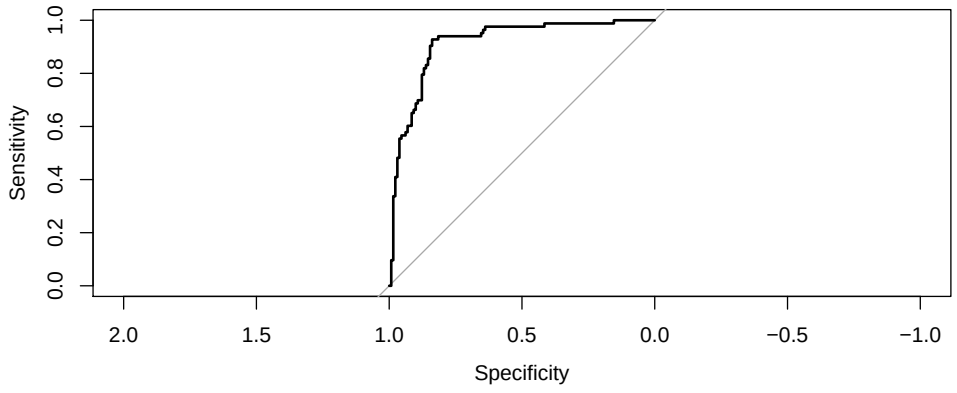
```
Setting levels: control = CH, case = MM
```

```
Setting direction: controls > cases
```

```
res.roc
```

```
Call:
roc.default(response = OJ.test$Purchase, predictor = pred_prob$CH)

Data: pred_prob$CH in 130 controls (OJ.test$Purchase CH) > 83 cases (OJ.test$Purchase MM).
Area under the curve: 0.9133
```

```
plot(res.roc)
```



Exercise 2: Nicht-lineare Kernel

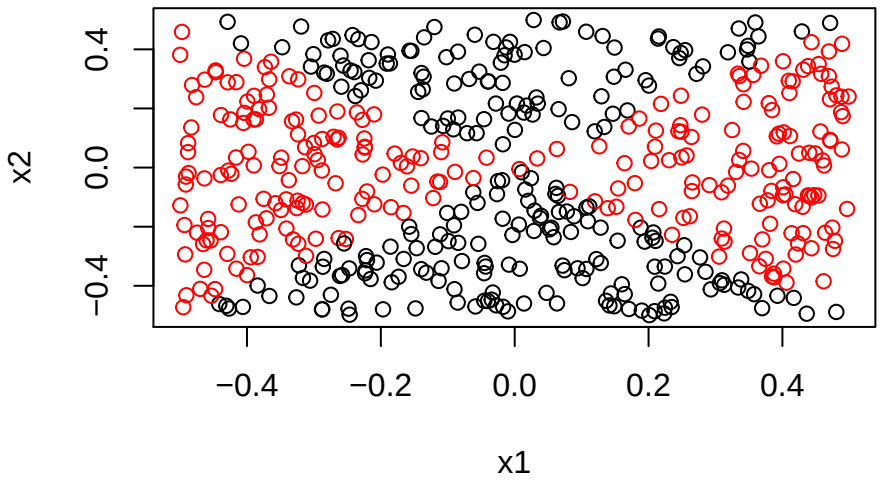
Eine SVM kann auch mit einem nichtlinearen Kernel ausgestattet werden, um eine Klassifizierung anhand einer nichtlinearen Entscheidungsgrenze durchzuführen. (Idee aus ISLR.)

a) Erzeugen Sie einen Datensatz mit n = 500 und p = 2, so dass die Beobachtungen zu zwei Klassen mit einer quadratischen Entscheidungsgrenze gehören. Sie können dies zum Beispiel wie folgt tun:

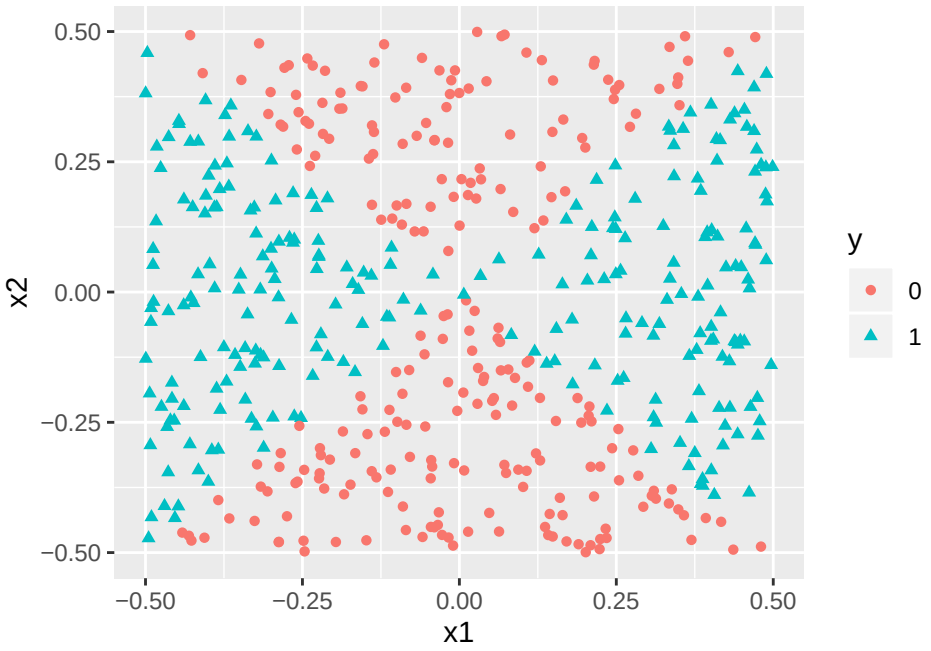
```
set.seed(4)
x1 <- runif(500) - 0.5
x2 <- runif(500) - 0.5
y <- as.factor(1 * (x1^2 - x2^2 > 0))
```

b) Zeichnen Sie die Beobachtungen in einen Scatterplot und färben Sie die Beobachtungen entsprechend ihrer Klassenbezeichnungen ein.

```
plot(x=x1, y=x2, col=y)
```




```
library(ggplot2)
dat <- data.frame(x1 = x1, x2 = x2, y = y)
ggplot(dat, aes(x = x1, y = x2, col = y, shape = y)) + geom_point()
```



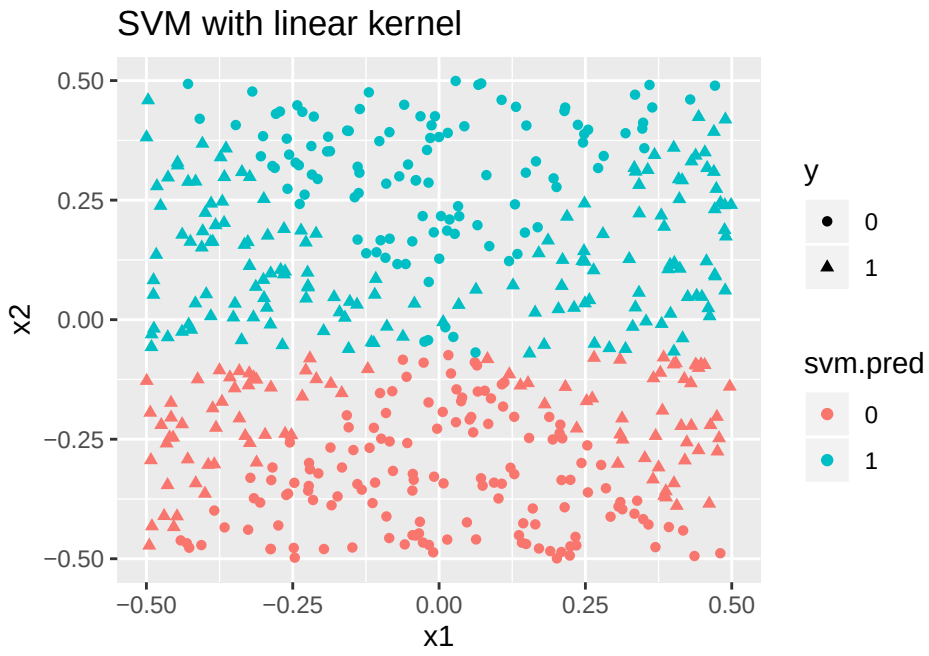
c) Passen Sie einen linearen Support-Vektor-Klassifikator an die Daten an (`method='svmLinear2'` und `c =0.1`). Berechnen Sie mit `predict` die Klassenvorhersage für die Trainingsdaten. Zeichnen Sie die Beobachtungen mit den vorhergesagten Klassen (Farbe) und den tatsächlichen Klasse (Symbol z.B. `pch` in `plot` oder `shape` in `ggplot`) auf.

```
x <- trainControl(method="none")
model_svm <- train(y ~ x1 + x2, data=dat, method='svmLinear2',
                   tuneGrid=data.frame(cost=c(0.1)),
                   trControl=x)
svm.pred <- predict(model_svm, dat)
confusionMatrix(data=svm.pred, reference=y)
```

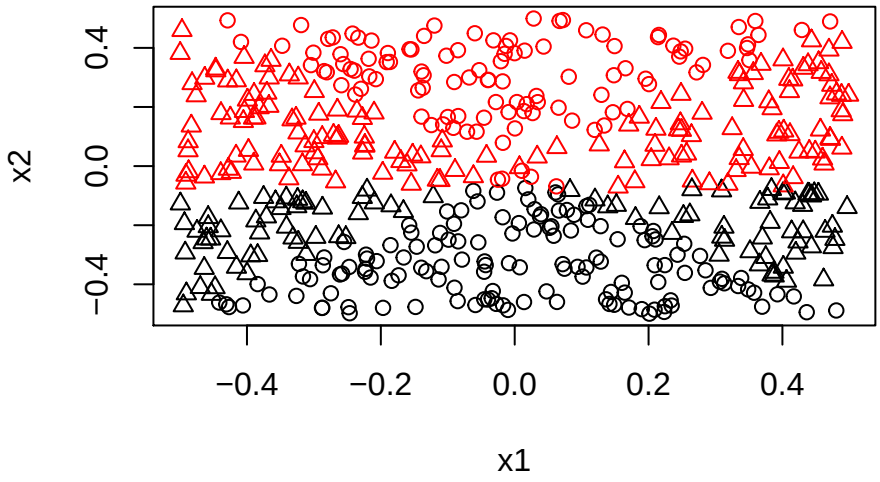
```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 140  96
         1 110 154

               Accuracy : 0.588
                 95% CI : (0.5434, 0.6315)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : 4.809e-05

                  Kappa : 0.176

 Mcnemar's Test P-Value : 0.3651

            Sensitivity : 0.5600
            Specificity : 0.6160
```

```
          Pos Pred Value : 0.5932
          Neg Pred Value : 0.5833
              Prevalence : 0.5000
          Detection Rate : 0.2800
    Detection Prevalence : 0.4720
       Balanced Accuracy : 0.5880

        'Positive' Class : 0
```

```
svm.pred <- as.factor(svm.pred)
ggplot(dat, aes(x = x1, y = x2, col = svm.pred, shape = y)) + geom_point() +
  ggtitle("SVM with linear kernel")
```




```
plot(x=x1, y=x2, col=svm.pred, pch=as.numeric(y))
```



Die Entscheidungsgrenze ist, wie vorgegeben, linear. Die Vorhersagen stimmen nicht mit der wahren Klassenbezeichnung überein. Ein linearer Kernel, sogar mit kleinem Kostparameter, kann keine nichtlineare Entscheidungsgrenze gefunden werden.

d) Passen Sie eine SVM an die Daten an, indem Sie einen radialen Kernel (`method='svmRadial'`) verwenden. Passt das Modell besser? Wenn Sie Lust haben, können Sie auch noch einen polynomialen Kernel ausprobieren (`method='svmPoly'`), Sie können sich dabei auf Polynome zweiten Grads beschränken (`degree=2`).

```
x <- trainControl(method="repeatedcv",
                  number=10,
                  repeats=3)
modelLookup("svmRadial")
```

```
      model parameter label forReg forClass probModel
1 svmRadial     sigma Sigma   TRUE     TRUE      TRUE
2 svmRadial         C  Cost   TRUE     TRUE      TRUE
```

```
svm_radial <- train(y~., data=dat, method='svmRadial',
                    tuneGrid=expand.grid(sigma=seq(0.5,5,0.5), C=c(0.01, 0.1, 0.5, 1, 1.5)),
                    trControl=x)
svm_radial
```

```
Support Vector Machines with Radial Basis Function Kernel

500 samples
  2 predictor
  2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 3 times)
```

```
Summary of sample sizes: 450, 450, 450, 450, 450, 450, ...
Resampling results across tuning parameters:
```

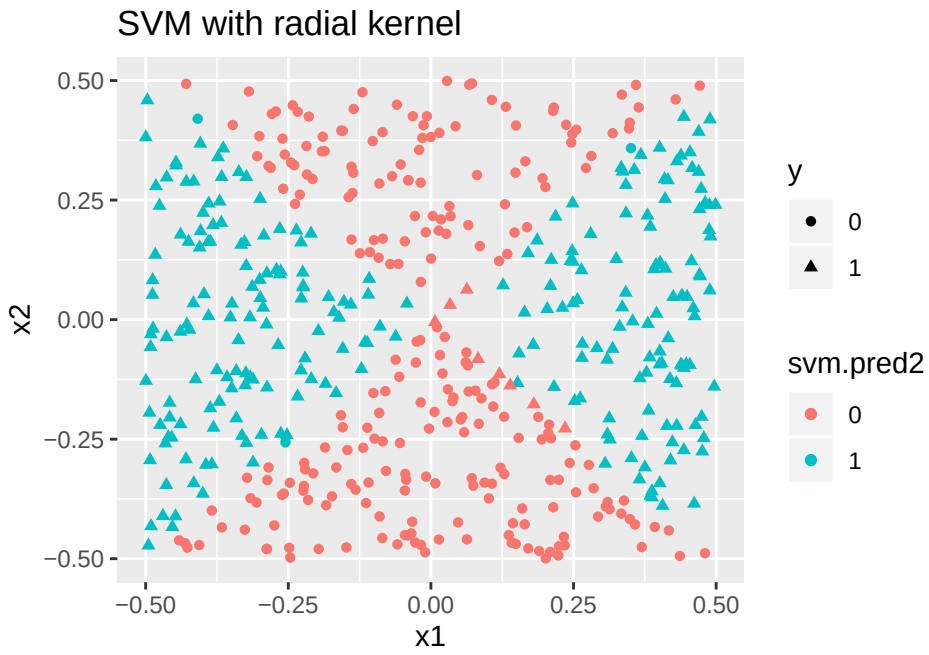
sigma	C	Accuracy	Kappa
0.5	0.01	0.8833333	0.7666667
0.5	0.10	0.9246667	0.8493333
0.5	0.50	0.9446667	0.8893333
0.5	1.00	0.9420000	0.8840000
0.5	1.50	0.9573333	0.9146667
1.0	0.01	0.9193333	0.8386667
1.0	0.10	0.9340000	0.8680000
1.0	0.50	0.9433333	0.8866667
1.0	1.00	0.9506667	0.9013333
1.0	1.50	0.9553333	0.9106667
1.5	0.01	0.9280000	0.8560000
1.5	0.10	0.9406667	0.8813333
1.5	0.50	0.9466667	0.8933333
1.5	1.00	0.9513333	0.9026667
1.5	1.50	0.9560000	0.9120000
2.0	0.01	0.9380000	0.8760000
2.0	0.10	0.9453333	0.8906667
2.0	0.50	0.9500000	0.9000000
2.0	1.00	0.9506667	0.9013333
2.0	1.50	0.9586667	0.9173333
2.5	0.01	0.9420000	0.8840000
2.5	0.10	0.9466667	0.8933333
2.5	0.50	0.9486667	0.8973333
2.5	1.00	0.9526667	0.9053333
2.5	1.50	0.9580000	0.9160000
3.0	0.01	0.9426667	0.8853333
3.0	0.10	0.9480000	0.8960000
3.0	0.50	0.9480000	0.8960000
3.0	1.00	0.9573333	0.9146667
3.0	1.50	0.9613333	0.9226667
3.5	0.01	0.9433333	0.8866667
3.5	0.10	0.9493333	0.8986667
3.5	0.50	0.9480000	0.8960000
3.5	1.00	0.9573333	0.9146667
3.5	1.50	0.9600000	0.9200000
4.0	0.01	0.9460000	0.8920000
4.0	0.10	0.9500000	0.9000000
4.0	0.50	0.9513333	0.9026667
4.0	1.00	0.9580000	0.9160000
4.0	1.50	0.9580000	0.9160000
4.5	0.01	0.9466667	0.8933333
4.5	0.10	0.9506667	0.9013333
4.5	0.50	0.9520000	0.9040000
4.5	1.00	0.9593333	0.9186667
4.5	1.50	0.9573333	0.9146667
5.0	0.01	0.9493333	0.8986667
5.0	0.10	0.9520000	0.9040000

```
  5.0    0.50  0.9526667  0.9053333
  5.0    1.00  0.9593333  0.9186667
  5.0    1.50  0.9586667  0.9173333

Accuracy was used to select the optimal model using the largest value.
The final values used for the model were sigma = 3 and C = 1.5.
```

```
svm.pred2 <- predict(svm_radial, dat)
confusionMatrix(data=svm.pred2, reference=y)
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 247   8
         1   3 242

               Accuracy : 0.978
                 95% CI : (0.961, 0.989)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.956

 Mcnemar's Test P-Value : 0.2278

            Sensitivity : 0.9880
            Specificity : 0.9680
         Pos Pred Value : 0.9686
         Neg Pred Value : 0.9878
             Prevalence : 0.5000
         Detection Rate : 0.4940
   Detection Prevalence : 0.5100
      Balanced Accuracy : 0.9780

       'Positive' Class : 0
```

```
svm.pred2 <- as.factor(svm.pred2)
ggplot(dat, aes(x = x1, y = x2, col = svm.pred2, shape = y)) +
  geom_point() +
  ggtitle("SVM with radial kernel")
```




```
## Polynomial Kernel
modelLookup("svmPoly")
```

```
    model parameter             label forReg forClass probModel
1 svmPoly    degree Polynomial Degree   TRUE     TRUE      TRUE
2 svmPoly     scale             Scale   TRUE     TRUE      TRUE
3 svmPoly         C              Cost   TRUE     TRUE      TRUE
```

```
svm_poly <- train(y~., data=dat, method="svmPoly", tuneGrid=
                    expand.grid(degree=2, scale=c(0.1,  1,  2,  5, 10),
                                C=c(0.1,  1, 5, 8, 10,15)), trControl=x)
svm_poly
```

```
Support Vector Machines with Polynomial Kernel

500 samples
  2 predictor
  2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 450, 450, 450, 450, 450, 450, ...
Resampling results across tuning parameters:

  scale  C     Accuracy   Kappa
   0.1    0.1  0.7040000  0.4080000
   0.1    1.0  0.9313333  0.8626667
   0.1    5.0  0.9493333  0.8986667
   0.1    8.0  0.9506667  0.9013333
   0.1   10.0  0.9560000  0.9120000
```

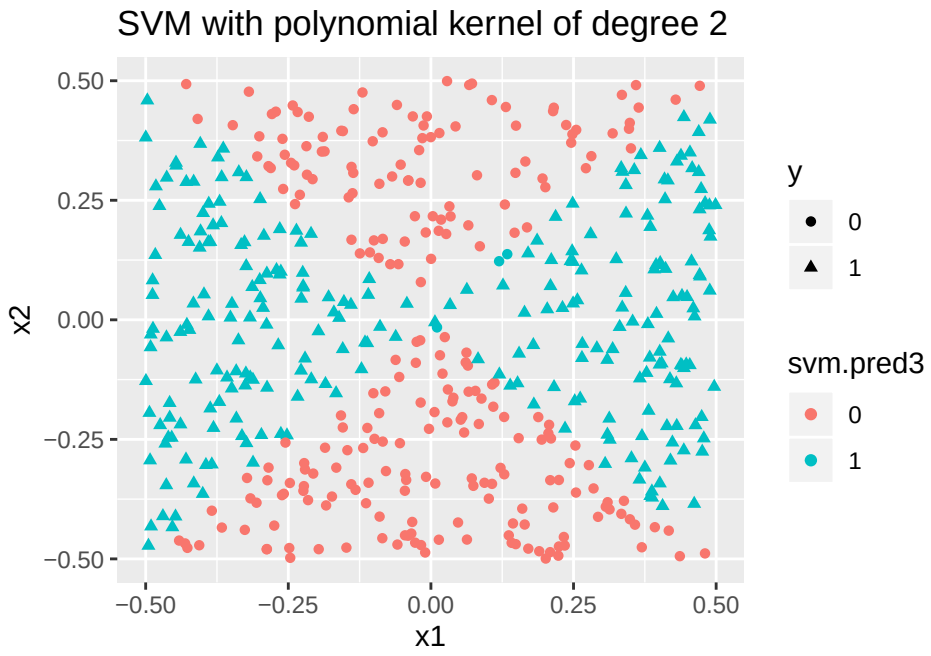
```
   0.1   15.0  0.9626667  0.9253333
   1.0    0.1  0.9560000  0.9120000
   1.0    1.0  0.9706667  0.9413333
   1.0    5.0  0.9806667  0.9613333
   1.0    8.0  0.9813333  0.9626667
   1.0   10.0  0.9786667  0.9573333
   1.0   15.0  0.9780000  0.9560000
   2.0    0.1  0.9660000  0.9320000
   2.0    1.0  0.9813333  0.9626667
   2.0    5.0  0.9806667  0.9613333
   2.0    8.0  0.9860000  0.9720000
   2.0   10.0  0.9866667  0.9733333
   2.0   15.0  0.9906667  0.9813333
   5.0    0.1  0.9760000  0.9520000
   5.0    1.0  0.9833333  0.9666667
   5.0    5.0  0.9886667  0.9773333
   5.0    8.0  0.9893333  0.9786667
   5.0   10.0  0.9900000  0.9800000
   5.0   15.0  0.9860000  0.9720000
  10.0    0.1  0.9786667  0.9573333
  10.0    1.0  0.9913333  0.9826667
  10.0    5.0  0.9866667  0.9733333
  10.0    8.0  0.9833333  0.9666667
  10.0   10.0  0.9853333  0.9706667
  10.0   15.0  0.9873333  0.9746667

Tuning parameter 'degree' was held constant at a value of 2
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were degree = 2, scale = 10 and C = 1.
```

```
svm.pred3 <- predict(svm_poly, dat)
confusionMatrix(data=svm.pred3, reference=y)
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 247   0
         1   3 250

               Accuracy : 0.994
                 95% CI : (0.9826, 0.9988)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.988

 Mcnemar's Test P-Value : 0.2482

            Sensitivity : 0.9880
            Specificity : 1.0000
         Pos Pred Value : 1.0000
```

```
         Neg Pred Value : 0.9881
             Prevalence : 0.5000
         Detection Rate : 0.4940
   Detection Prevalence : 0.4940
      Balanced Accuracy : 0.9940

       'Positive' Class : 0
```

```
svm.pred3 <- as.factor(svm.pred3)
ggplot(dat, aes(x = x1, y = x2, col = svm.pred3, shape = y)) +
  geom_point() +
  ggtitle("SVM with polynomial kernel of degree 2")
```



Die Entscheidungsgrenze ist, wie für einen radialen Kernel erwartet, nicht linear. Die vorhergesagten Klassen kommen den wahren Klassen sehr nahe. Beim polynomialen Kernel ist die Vorhersage ähnlich gut.