sVM_Classification

Data Information

The data is found on Kaggle , uploaded by Deep contractor https://www.kaggle.com/datasets/deepcontractor/smoke-detection-dataset (https://www.kaggle.com/datasets/deepcontractor/smoke-detection-dataset) Classifying to two tpe of smoke detector photoelectric smoke detector and ionization soke detector The classification is determine by fire alarm to result if it is detected or the predictor: * utc * Tempreture * Humidity * target: * Fire.Alarm

#library

```
library(e1071)
```

Read Data

```
df <- read.csv("Data/avocado.csv")
df <- na.omit(df)
df$type <- as.factor(df$type)

str(df)
```

```
## 'data.frame':    14869 obs. of  13 variables:
##  $ Sale.ID         : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ Date            : chr  "3/5/2017 0:00" "2/5/2017 0:00" "3/5/2017 0:00" "2/26/2017 0:00"
...
##  $ AveragePrice    : num  0.44 0.46 0.48 0.49 0.49 0.51 0.51 0.51 0.51 0.51 ...
##  $ Total.Avocados  : int  4973 1750185 4857 4726 1036815 5959 1281938 2272 1269280 1312113
...
##  $ Small.4046      : int  224 1200633 718 253 738315 225 985040 482 1097285 1037699 ...
##  $ Extra.Large.4770: int  0 18325 0 0 11642 0 6314 0 7534 14567 ...
##  $ Large.4225      : int  4749 531227 4139 4473 286858 5734 290584 1790 164461 259847 ...
##  $ Total.Bags      : int  59085 450366 46034 39299 100892 36028 193803 14864 97565 130860 ...
##  $ Small.Bags      : int  639 113752 1385 600 70749 474 62497 123 44647 76814 ...
##  $ Large.Bags      : int  58446 330583 44649 38699 30143 35554 131306 14741 52918 54046 ...
##  $ XLarge.Bags     : int  0 6031 0 0 0 0 0 0 0 0 ...
##  $ type            : Factor w/ 2 levels "conventional",..: 2 1 2 2 1 2 1 2 1 1 ...
##  $ Cities          : chr  "Cincinnati Dayton" "Phoenix Tucson" "Detroit" "Cincinnati Dayton"
...
```

Data Exploration

let’s try to see a classify relation

```
head(df)
```

	Sale.ID <int>	Date <chr>	AveragePrice <dbl>	Total.Avocados <int>	Small.4046 <int>	Extra.Large.4770 <int>	La
1	1	3/5/2017 0:00	0.44	4973	224	0	
2	1	2/5/2017 0:00	0.46	1750185	1200633	18325	
3	1	3/5/2017 0:00	0.48	4857	718	0	
4	1	2/26/2017 0:00	0.49	4726	253	0	
5	1	12/27/2015 0:00	0.49	1036815	738315	11642	
6	1	2/19/2017 0:00	0.51	5959	225	0	

6 rows | 1-8 of 14 columns

```
tail(df)
```

	Sale.ID <int>	Date <chr>	AveragePrice <dbl>	Total.Avocados <int>	Small.4046 <int>	Extra.Large.477 <int
14864	1	10/2/2016 0:00	3.03	2997	297	(
14865	1	8/27/2017 0:00	3.04	5416	419	14
14866	1	3/12/2017 0:00	3.05	1121	1044	(
14867	1	11/6/2016 0:00	3.12	15937	5898	(
14868	1	4/16/2017 0:00	3.17	1338	1256	(
14869	1	10/30/2016 0:00	3.25	13469	2326	(

6 rows | 1-8 of 14 columns

```
dim(df)
```

```
## [1] 14869    13
```

train/test

```
set.seed(1234)
i<- sample(1:nrow(df), 0.8*nrow(df), replace=FALSE)
train <-df[i,]
test <- df[-i,]
```

#run classification

```
glm1 <- glm(train$type~train$AveragePrice+train$Total.Bags,family="binomial",data=train )
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(glm1)
```

```
##
## Call:
## glm(formula = train$type ~ train$AveragePrice + train$Total.Bags,
##     family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.6959  -0.0803   0.0409   0.2670   4.9813
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)        -9.032e-01  2.001e-01  -4.513 6.39e-06 ***
## train$AveragePrice  2.869e+00  1.390e-01  20.645  < 2e-16 ***
## train$Total.Bags   -1.027e-04  2.339e-06 -43.914  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 16489.8  on 11894  degrees of freedom
## Residual deviance:  4158.7  on 11892  degrees of freedom
## AIC: 4164.7
##
## Number of Fisher Scoring iterations: 9
```

try linear

```
train_trim <- train[c(1:500),]
#str(train)
svm1 <- svm(train_trim$type~AveragePrice+ train_trim$Total.Bags, data=train_trim ,kernel="linea
r", cost=10, scale=TRUE)
summary(svm1)
```

```
##
## Call:
## svm(formula = train_trim$type ~ AveragePrice + train_trim$Total.Bags,
##     data = train_trim, kernel = "linear", cost = 10, scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  10
##
## Number of Support Vectors:  105
##
##  ( 52 53 )
##
##
## Number of Classes:  2
##
## Levels:
##  conventional organic
```

evaluate

```
test_trim <- test[c(1:500),]
pred <- predict(svm1,newdata = test_trim)
table(pred,test_trim$type)
```

```
##
## pred           conventional organic
##   conventional          241      31
##   organic               202      26
```

```
mean(pred==test_trim$type)
```

```
## [1] 0.534
```

#diff cost

```
tune1 <- tune(svm,type~AveragePrice+Total.Bags,data=train_trim,kernel="linear",
              ranges=list(cost=c(
0.001
,
0.01
,
0.1
,
1
,
5
,
10
,
100
)))
summary(tune1)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##     5
##
## - best performance: 0.072
##
## - Detailed performance results:
##    cost error dispersion
## 1 1e-03 0.430 0.07788881
## 2 1e-02 0.166 0.05660781
## 3 1e-01 0.114 0.05337498
## 4 1e+00 0.080 0.03126944
## 5 5e+00 0.072 0.02699794
## 6 1e+01 0.072 0.02699794
## 7 1e+02 0.072 0.02699794
```

try polynomial

```
svm2 <- svm(train_trim$type~AveragePrice+ train_trim$Total.Bags,data=train_trim,kernel="polynomi
al", cost=10, scale=TRUE)
summary(svm2)
```

```
##
## Call:
## svm(formula = train_trim$type ~ AveragePrice + train_trim$Total.Bags,
##     data = train_trim, kernel = "polynomial", cost = 10, scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  polynomial
##        cost:  10
##      degree:  3
##      coef.0:  0
##
## Number of Support Vectors:  257
##
##  ( 129 128 )
##
##
## Number of Classes:  2
##
## Levels:
##  conventional organic
```

trying different cost

```
svm2 <- svm(train_trim$type~AveragePrice+ train_trim$Total.Bags,data=train_trim,kernel="polynomi
al", cost=5, scale=TRUE)
summary(svm2)
```

```
##
## Call:
## svm(formula = train_trim$type ~ AveragePrice + train_trim$Total.Bags,
##     data = train_trim, kernel = "polynomial", cost = 5, scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  polynomial
##        cost:  5
##      degree:  3
##      coef.0:  0
##
## Number of Support Vectors:  269
##
##  ( 135 134 )
##
##
## Number of Classes:  2
##
## Levels:
##  conventional organic
```

```
tune1 <- tune(svm,type~AveragePrice+Total.Bags,data=train_trim,kernel="polynomial",
              ranges=list(cost=c(
0.001
,
0.01
,
0.1
,
1
,
5
,
10
,
100
)))
summary(tune1)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##   100
##
## - best performance: 0.074
##
## - Detailed performance results:
##    cost error dispersion
## 1 1e-03 0.450 0.06815016
## 2 1e-02 0.318 0.09016035
## 3 1e-01 0.268 0.08121303
## 4 1e+00 0.244 0.07411702
## 5 5e+00 0.214 0.06328068
## 6 1e+01 0.176 0.05059644
## 7 1e+02 0.074 0.02503331
```

```
pred <-predict(svm2,newdata=test_trim)
mean(pred==test_trim$type)
```

```
## [1] 0.776
```

```
svm3 <- svm(type~AveragePrice+Total.Bags,data=train_trim,cost=10,scale=TRUE)
summary(svm3)
```

```
##
## Call:
## svm(formula = type ~ AveragePrice + Total.Bags, data = train_trim,
##     cost = 10, scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  10
##
## Number of Support Vectors:  120
##
##  ( 64 56 )
##
##
## Number of Classes:  2
##
## Levels:
##  conventional organic
```

```
pred <-predict(svm2,newdata=test_trim)
mean(pred==test_trim$type)
```

```
## [1] 0.776
```

#different costs

```
tune2 <- tune(svm,type~AveragePrice+Total.Bags,data=train_trim,kernel="polynomial",
              ranges=list(cost=c(
0.001
,
0.01
,
0.1
,
1
,
5
,
10
,
100
)))
summary(tune2)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##   100
##
## - best performance: 0.084
##
## - Detailed performance results:
##    cost error dispersion
## 1 1e-03 0.456 0.05872724
## 2 1e-02 0.320 0.08793937
## 3 1e-01 0.268 0.09101892
## 4 1e+00 0.236 0.08262364
## 5 5e+00 0.208 0.08230026
## 6 1e+01 0.180 0.06992059
## 7 1e+02 0.084 0.02270585
```

hyperplane

```
tune_svm3 <- tune(svm,type~AveragePrice+Total.Bags,data=train_trim,kernel=
"radial"
, ranges=list(cost=c(
0.001
,
0.01
,
0.1
,
1
,
5
,
10
,
100
),
gamma=c(
0.5
,
1
,
2
,
3
,
4
)))
summary(tune_svm3)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost gamma
##    10   0.5
##
## - best performance: 0.076
##
## - Detailed performance results:
##     cost gamma error dispersion
## 1  1e-03   0.5 0.482 0.04366539
## 2  1e-02   0.5 0.198 0.06285786
## 3  1e-01   0.5 0.144 0.06310485
## 4  1e+00   0.5 0.096 0.05059644
## 5  5e+00   0.5 0.082 0.05996295
## 6  1e+01   0.5 0.076 0.05561774
## 7  1e+02   0.5 0.078 0.05692100
## 8  1e-03   1.0 0.482 0.04366539
## 9  1e-02   1.0 0.198 0.05996295
## 10 1e-01   1.0 0.122 0.05846176
## 11 1e+00   1.0 0.094 0.04993329
## 12 5e+00   1.0 0.080 0.05734884
## 13 1e+01   1.0 0.082 0.06494442
## 14 1e+02   1.0 0.082 0.06425643
## 15 1e-03   2.0 0.482 0.04366539
## 16 1e-02   2.0 0.218 0.04049691
## 17 1e-01   2.0 0.118 0.05613476
## 18 1e+00   2.0 0.082 0.05028806
## 19 5e+00   2.0 0.086 0.06535374
## 20 1e+01   2.0 0.082 0.06356099
## 21 1e+02   2.0 0.086 0.06257440
## 22 1e-03   3.0 0.482 0.04366539
## 23 1e-02   3.0 0.262 0.06425643
## 24 1e-01   3.0 0.118 0.05202563
## 25 1e+00   3.0 0.082 0.05028806
## 26 5e+00   3.0 0.086 0.05966574
## 27 1e+01   3.0 0.084 0.06168018
## 28 1e+02   3.0 0.084 0.05947922
## 29 1e-03   4.0 0.482 0.04366539
## 30 1e-02   4.0 0.338 0.09953224
## 31 1e-01   4.0 0.124 0.04971027
## 32 1e+00   4.0 0.080 0.05077182
## 33 5e+00   4.0 0.088 0.06051630
## 34 1e+01   4.0 0.088 0.05902918
## 35 1e+02   4.0 0.084 0.06310485
```

analysis/conclusion

The Data shows a good results for all kernals. The best cost for the kernal varies on the choose kernal, such as for polynimial 100 is giving the best model, linear it’s 5 and radial is 10. So the kernal choose is reflecting the dataset plane differently.