Sample: Statistics- R Programming

Installing packages

```
library(caret)

## Warning: package 'caret' was built under R version 3.3.3

## Loading required package: lattice

## Warning: package 'lattice' was built under R version 3.3.3

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 3.3.3

library(e1071)

## Warning: package 'e1071' was built under R version 3.3.3

library(kernlab)

##
## Attaching package: 'kernlab'

## The following object is masked from 'package:ggplot2':
##
##     alpha
```

Download data

load the "charity.csv" file

```
# load the data
charity <-read.csv(file.choose())
```

Data Preprocessing

Download and transform data. We split the data into a test, training and validation sample.

```
# predictor transformations

charity$donr <-as.factor(charity$donr)
charity.t <-charity
charity.t$avhv <-log(charity.t$avhv)
# add further transformations if desired
# for example, some statistical methods can struggle when predictors are
highly skewed
```

```
# set up data for analysis

data.train <-charity.t[charity$part=="train",]
x.train <-data.train[,2:21]
c.train <-data.train[,22] # donr
n.train.c <-length(c.train) # 3984
y.train <-data.train[c.train==1,23] # damt for observations with donr=1
n.train.y <-length(y.train) # 1995

data.valid <-charity.t[charity$part=="valid",]
x.valid <-data.valid[,2:21]
c.valid <-data.valid[,22] # donr
n.valid.c <-length(c.valid) # 2018
y.valid <-data.valid[c.valid==1,23] # damt for observations with donr=1
n.valid.y <-length(y.valid) # 999

data.test <-charity.t[charity$part=="test",]
n.test <-dim(data.test)[1] # 2007
x.test <-data.test[,2:21]

x.train.mean <-apply(x.train, 2, mean)
x.train.sd <-apply(x.train, 2, sd)
x.train.std <-t((t(x.train)-x.train.mean)/x.train.sd) # standardize to have
zero mean and unit sd
apply(x.train.std, 2, mean) # check zero mean

##          reg1          reg2          reg3          reg4          home
##  2.151811e-17 -2.526099e-17  3.693258e-17 -6.017778e-17 -9.663428e-18
##          chld          hinc          genf          wrat          avhv
## -2.051129e-17 -1.463197e-17  4.465563e-17 -1.062688e-16 -3.616154e-16
##          incm          inca          plow          npro          tgif
## -1.335981e-16  8.260581e-17  4.848389e-17 -6.685448e-17  2.532023e-17
##          lgif          rgif          tdon          tlag          agif
##  2.669731e-17  5.429393e-18 -1.835154e-16  1.010477e-16 -1.258480e-16

apply(x.train.std, 2, sd) # check unit sd

## reg1 reg2 reg3 reg4 home chld hinc genf wrat avhv incm inca plow npro tgif

##    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1

## lgif rgif tdon tlag agif
##    1    1    1    1    1

data.train.std.c <-data.frame(x.train.std, donr=c.train) # to classify donr
data.train.std.y <-data.frame(x.train.std[c.train==1,], damt=y.train) # to
predict damt when donr=1

x.valid.std <-t((t(x.valid)-x.train.mean)/x.train.sd) # standardize using
```

```
training mean and sd
data.valid.std.c <-data.frame(x.valid.std, donr=c.valid) # to classify donr
data.valid.std.y <-data.frame(x.valid.std[c.valid==1,], damt=y.valid) # to
predict damt when donr=1

x.test.std <-t((t(x.test)-x.train.mean)/x.train.sd) # standardize using
training mean and sd
data.test.std <-data.frame(x.test.std)
```

We build the model of support vectors for predict "donr"

```
fitControl <-trainControl(method ="cv", repeats =10)
model_SVM <-train(donr~., data = data.train.std.c, method ="svmLinear",
trControl = fitControl)
```

Predict svm "donr"

```
predict_train <-predict(model_SVM,  data.valid.std.c[,-21])
confusionMatrix(predict_train, data.valid.std.c$donr)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 825 135
##          1 194 864
##
##                Accuracy : 0.837
##                  95% CI : (0.8201, 0.8528)
##     No Information Rate : 0.505
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6741
##  Mcnemar's Test P-Value : 0.001386
##
##             Sensitivity : 0.8096
##             Specificity : 0.8649
##          Pos Pred Value : 0.8594
##          Neg Pred Value : 0.8166
##              Prevalence : 0.5050
##          Detection Rate : 0.4088
##    Detection Prevalence : 0.4757
##       Balanced Accuracy : 0.8372
##
##        'Positive' Class : 0
##
```

Let us construct another model SVM radial

```
model_SVM_radial <-train(donr~., data = data.train.std.c, method
="svmRadial", trControl = fitControl)
```

```
predict_donr_radial <-predict(model_SVM_radial,  data.valid.std.c[,-21])
confusionMatrix(predict_donr_radial, data.valid.std.c$donr)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 862  90
##          1 157 909
##
##                Accuracy : 0.8776
##                  95% CI : (0.8625, 0.8916)
##     No Information Rate : 0.505
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7553
##  Mcnemar's Test P-Value : 2.675e-05
##
##             Sensitivity : 0.8459
##             Specificity : 0.9099
##          Pos Pred Value : 0.9055
##          Neg Pred Value : 0.8527
##              Prevalence : 0.5050
##          Detection Rate : 0.4272
##    Detection Prevalence : 0.4718
##       Balanced Accuracy : 0.8779
##
##        'Positive' Class : 0
##
```

Let us construct another model SVM Poly

```
model_SVM_poly <-train(donr~., data = data.train.std.c, method ="svmPoly",
trControl = fitControl)
predict_donr_poly <-predict(model_SVM_radial,  data.valid.std.c[,-21])
confusionMatrix(predict_donr_poly, data.valid.std.c$donr)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 862  90
##          1 157 909
##
##                Accuracy : 0.8776
##                  95% CI : (0.8625, 0.8916)
##     No Information Rate : 0.505
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7553
```

```
##  Mcnemar's Test P-Value : 2.675e-05
##
##             Sensitivity : 0.8459
##             Specificity : 0.9099
##          Pos Pred Value : 0.9055
##          Neg Pred Value : 0.8527
##              Prevalence : 0.5050
##          Detection Rate : 0.4272
##    Detection Prevalence : 0.4718
##       Balanced Accuracy : 0.8779
##
##        'Positive' Class : 0
##
```

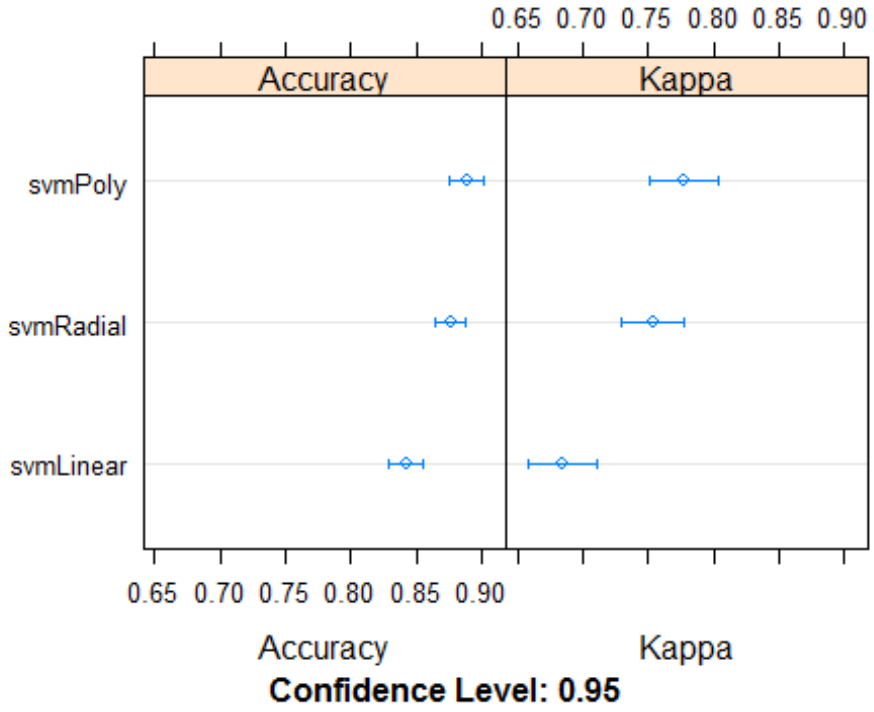
Choose the best model

We see that the most effective model is model_SVM_poly

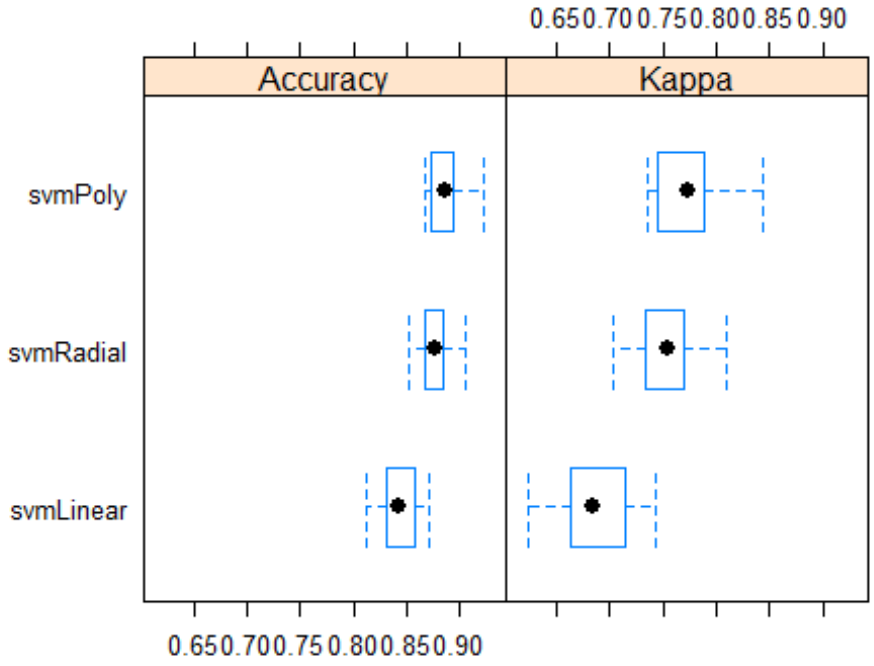
```
results1 =resamples(list(svmLinear = model_SVM, svmRadial = model_SVM_radial,
svmPoly = model_SVM_poly))
summary(results1)
```

```
##
## Call:
## summary.resamples(object = results1)
##
## Models: svmLinear, svmRadial, svmPoly
## Number of resamples: 10
##
## Accuracy
##             Min. 1st Qu. Median   Mean 3rd Qu.   Max. NA's
## svmLinear 0.8116  0.8318 0.8421 0.8424  0.8544 0.8719    0
## svmRadial 0.8518  0.8690 0.8770 0.8768  0.8832 0.9045    0
## svmPoly   0.8672  0.8752 0.8869 0.8888  0.8933 0.9223    0
##
## Kappa
##             Min. 1st Qu. Median   Mean 3rd Qu.   Max. NA's
## svmLinear 0.6231  0.6635 0.6842 0.6847  0.7087 0.7437    0
## svmRadial 0.7035  0.7379 0.7541 0.7535  0.7664 0.8090    0
## svmPoly   0.7343  0.7503 0.7739 0.7776  0.7866 0.8446    0
```

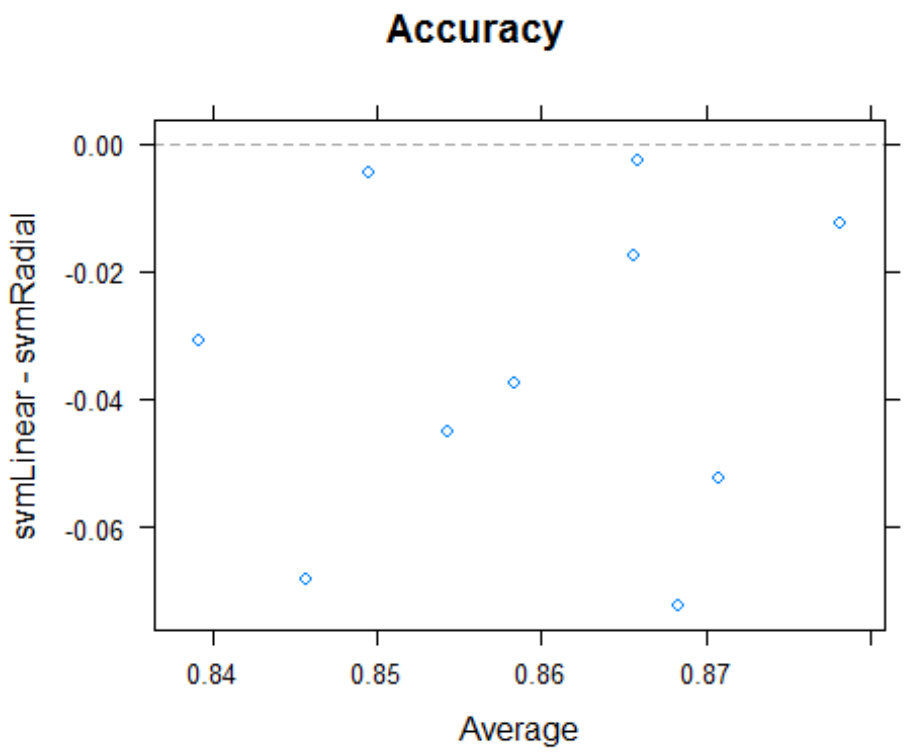
```
dotplot(results1)
```



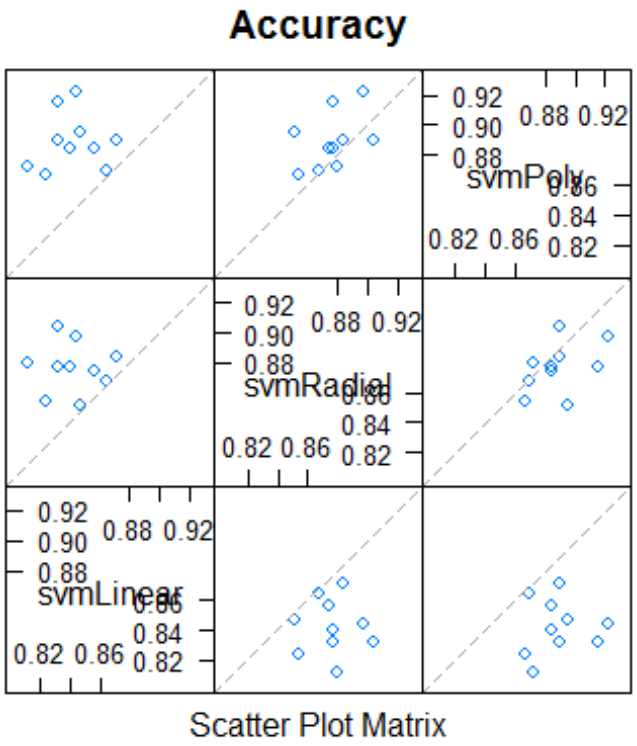

```
bwplot(results1)
```




```
xyplot(results1, what ="BlandAltman")
```




```
splom(results1)
```




```
difValues <-diff(results1)
```

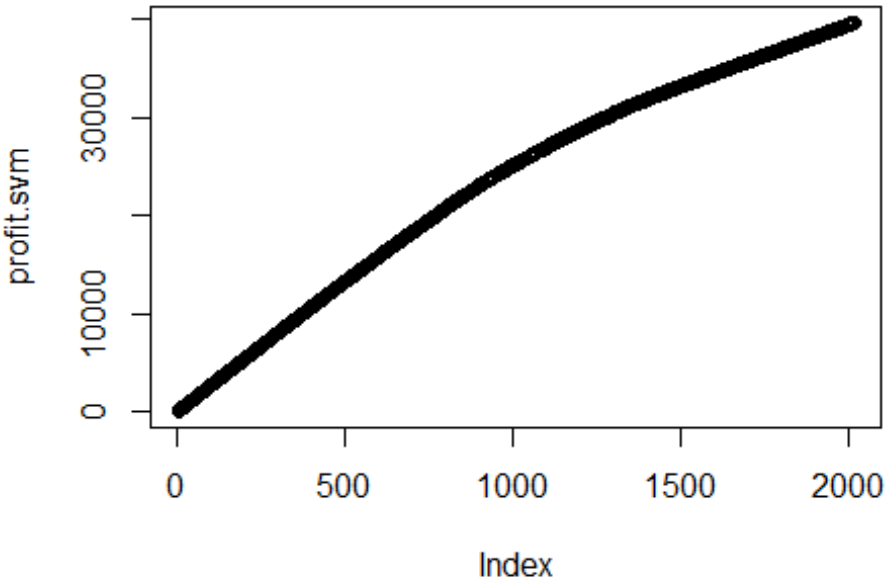
Data

```
data.train.std.c$donr <-as.numeric(data.train.std.c$donr)
c.valid <-as.numeric(c.valid)

model.svm <-svm(donr ~reg1 +reg2 +reg3 +reg4 +home +chld +hinc +I(hinc^2)
+genf +wrat +
avhv +incm +inca +plow +npro +tgif +lgif +rgif +tdon +tlag +agif,
                  data.train.std.c)
post.valid.svm <-predict(model.svm, data.valid.std.c) # n.valid.c post probs
```

calculate ordered profit function using average donation = $14.50 and mailing cost = $2

```
profit.svm <-cumsum(14.5*c.valid[order(post.valid.svm, decreasing=T)]-2)
plot(profit.svm) # see how profits change as more mailings are made
```




```
n.mail.valid <-which.max(profit.svm) # number of mailings that maximizes
profits
c(n.mail.valid, max(profit.svm)) # report number of mailings and maximum
profit

## [1]  2018.0 39710.5
```

We build the model of support vectors for predict "damt

```
cutoff.svm <-sort(post.valid.svm, decreasing=T)[n.mail.valid+1] # set cutoff
based on n.mail.valid
chat.valid.svm <-ifelse(post.valid.svm>cutoff.svm, 1, 0) # mail to everyone
above the cutoff
table(chat.valid.svm, c.valid) # classification table

## < table of extent 0 x 2 >

fitControl <-trainControl(method ="cv", repeats =10)
model_SVM_damt <-train(damt~., data = data.train.std.y, method ="svmLinear",
trControl = fitControl)
summary(model_SVM_damt$bestTune)

##        C
##  Min.   :1
##  1st Qu.:1
##  Median :1
```

```
##  Mean   :1
##  3rd Qu.:1
##  Max.   :1
```

```
print(model_SVM_damt)
```

```
## Support Vector Machines with Linear Kernel
##
## 1995 samples
##   20 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1796, 1796, 1795, 1796, 1796, 1795, ...
## Resampling results:
##
##   RMSE      Rsquared
##   1.291248  0.5627676
##
## Tuning parameter 'C' was held constant at a value of 1
```

Predict svm "damt"

```
predict_train_damt <-predict(model_SVM_damt,  data.valid.std.y)
predict_train_damt <-round(predict_train_damt, digits =0)
```