KNN

Vazgen Tadevosyan

April 18, 2019

You are required to submit both Markdown and HTML files. Data set (Spam E-mail Data) relevant for this assignment can be found in the R package "DAAG".

Problem 1 (10 pt.)

- a. Suppose the sysadmin wants to understand and predict the process of recognizing the emails as spam for new e-mails which make up 20% of your initial data. Compare the performance of the logit and k-NN for classification on your data. Which one is better? Why? (Use the ROC curve to find the best cutoff value and cross-validation for choosing the value of k. Show both results graphically).
- b. What are the differences **(at least 3)** of these algorithms (in general)?

```
library(ROCR)
library(ggplot2)
library(gridExtra)
library(ggcorrplot)
library(DAAG)
library(caret)
library(pROC)
library(class)
library(BBmisc)
data("spam7")
addmargins(table(spam7$yesno))
```

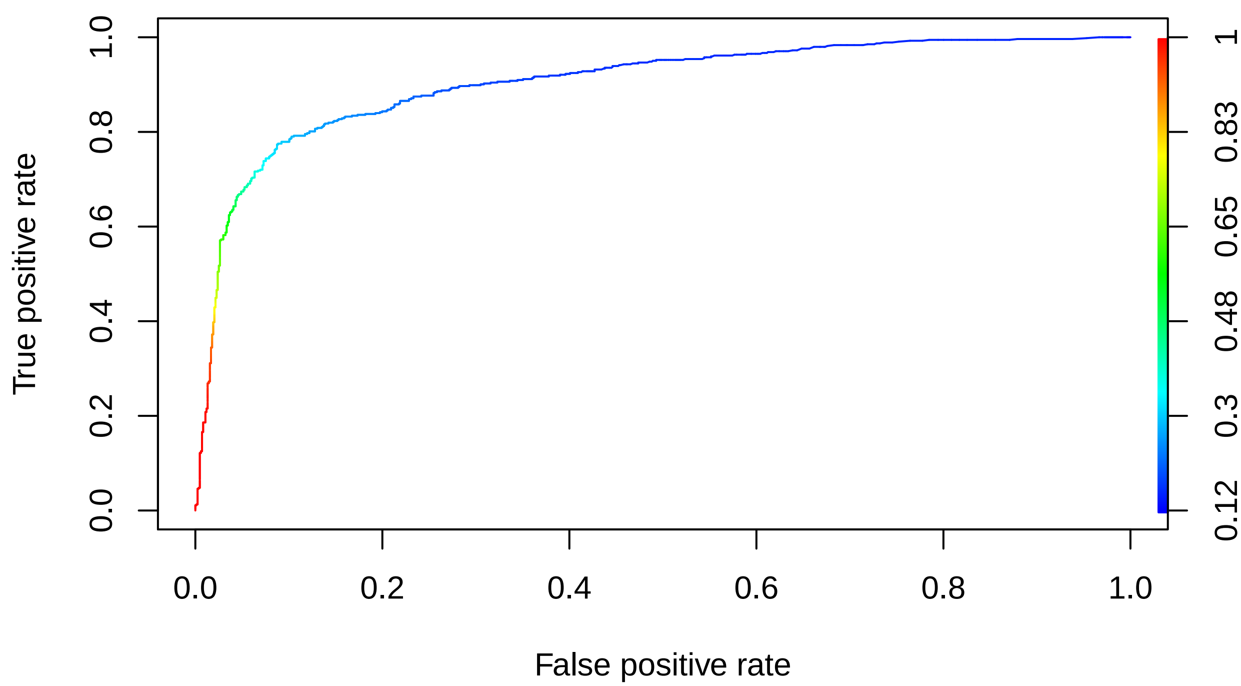
```
##
##    n    y  Sum
## 2788 1813 4601
```

```
set.seed(1)
train_index<-createDataPartition(spam7$yesno,p = 0.7,list = F)
train<-spam7[train_index,]
test<-spam7[-train_index,]
##Logit
logit<-glm(yesno~.,data = train,family = 'binomial')
predicted<-predict(logit,test,type = "response")
predicted1<-ifelse(predicted>0.5,"y","n")
confusionMatrix(factor(predicted1),test$yesno,positive = "y")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   n   y
##          n 801 194
##          y  35 349
##
##                Accuracy : 0.8339
##                  95% CI : (0.8132, 0.8532)
##     No Information Rate : 0.6062
##     P-Value [Acc > NIR] : < 2.2e-16
```

```
##
##                  Kappa : 0.6334
##  Mcnemar's Test P-Value : < 2.2e-16
##
##            Sensitivity : 0.6427
##            Specificity : 0.9581
##         Pos Pred Value : 0.9089
##         Neg Pred Value : 0.8050
##             Prevalence : 0.3938
##         Detection Rate : 0.2531
##   Detection Prevalence : 0.2785
##      Balanced Accuracy : 0.8004
##
##       'Positive' Class : y
##
```

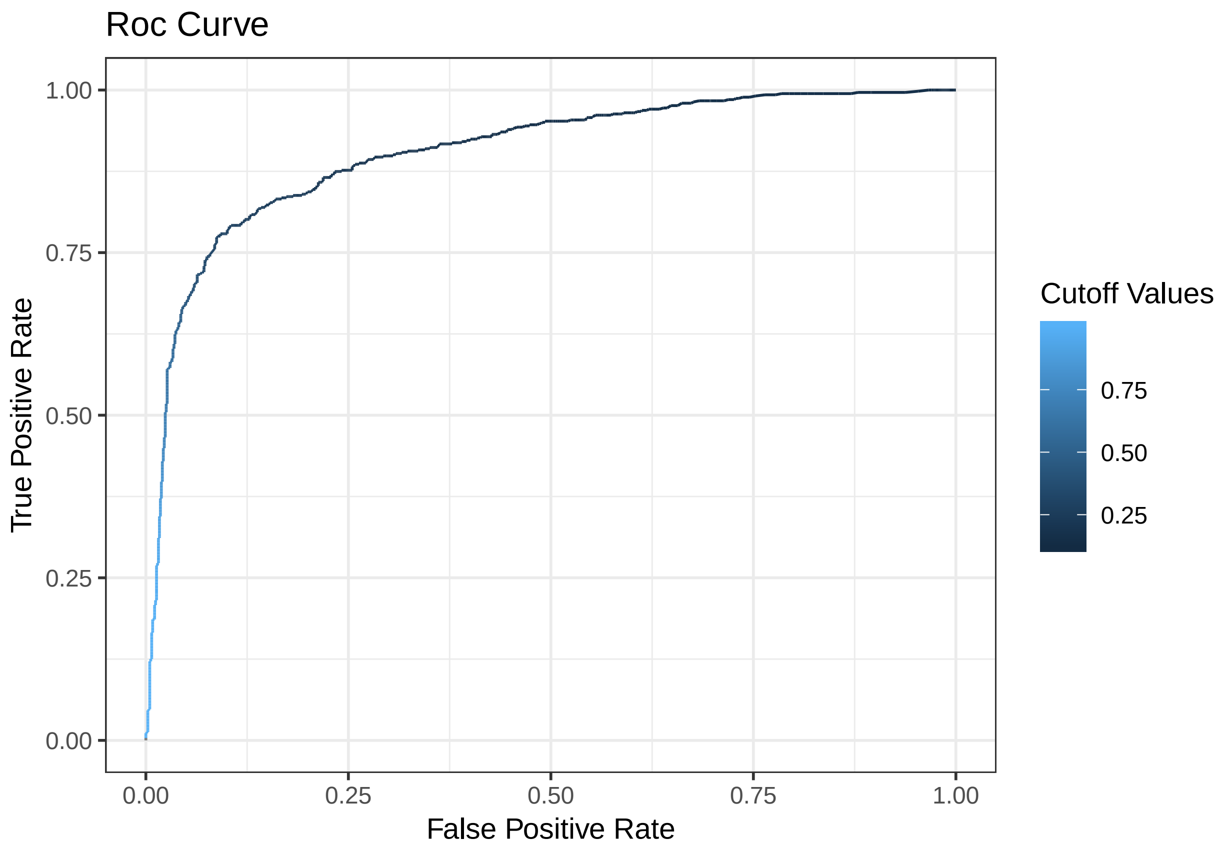
```
R1<-ifelse(predicted1=="y",1,0)
R2<-ifelse(test$yesno=="y",1,0)
rrr1<-roc(R2,R1)
g1<-ggroc(rrr1,alpha = 0.5, colour = "green", linetype = 1, size = 1)+ggtitle("ROC curve for Model Logi

P_test<-ROCR::prediction(predicted,test$yesno)
perf<-performance(P_test,'tpr','fpr')
plot(perf,colorize=T)
```



```r
performance(P_test,'auc')@y.values
```

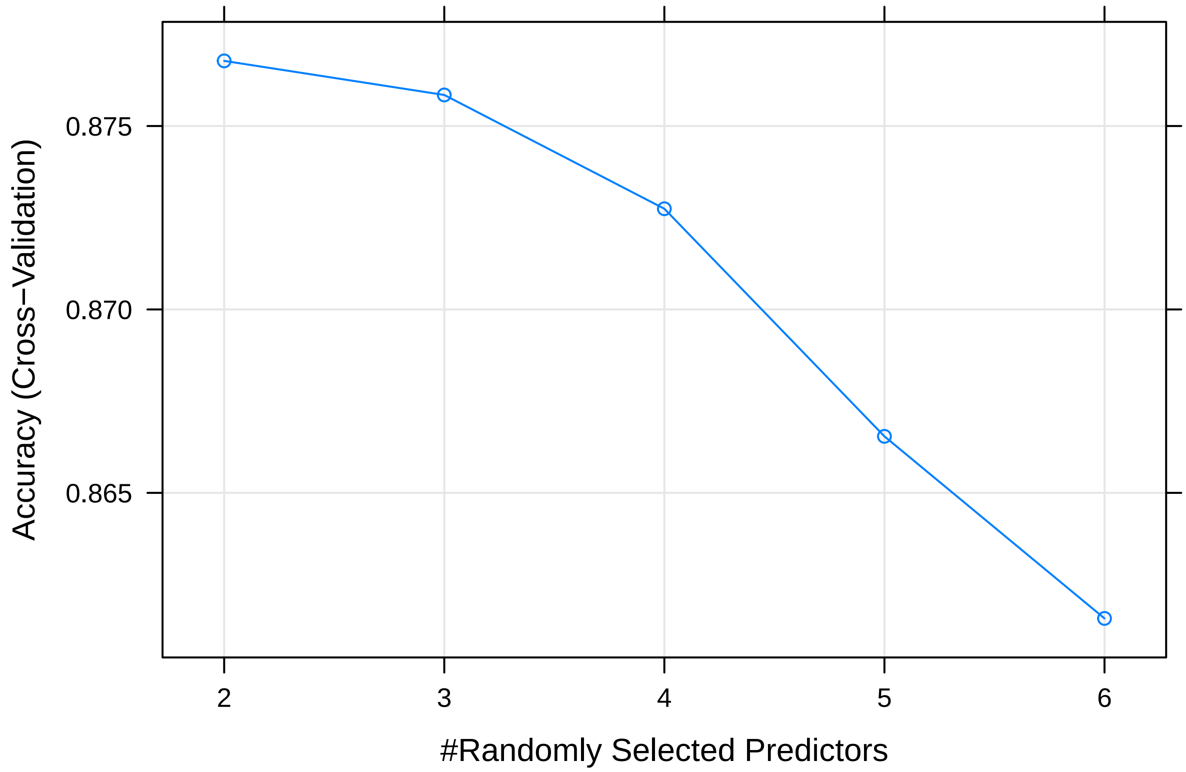
```
## [[1]]
## [1] 0.9043899
```

```r
FPR<-unlist(perf@x.values)
TPR<-unlist(perf@y.values)
alpha=unlist(perf@alpha.values)
df<-data.frame(FPR,TPR,alpha)
ggplot(df,aes(x=FPR,y=TPR,color=alpha)) + geom_line()+
  theme_bw()+ labs(x="False Positive Rate",y="True Positive Rate",title="Roc Curve",color="Cutoff Value
```

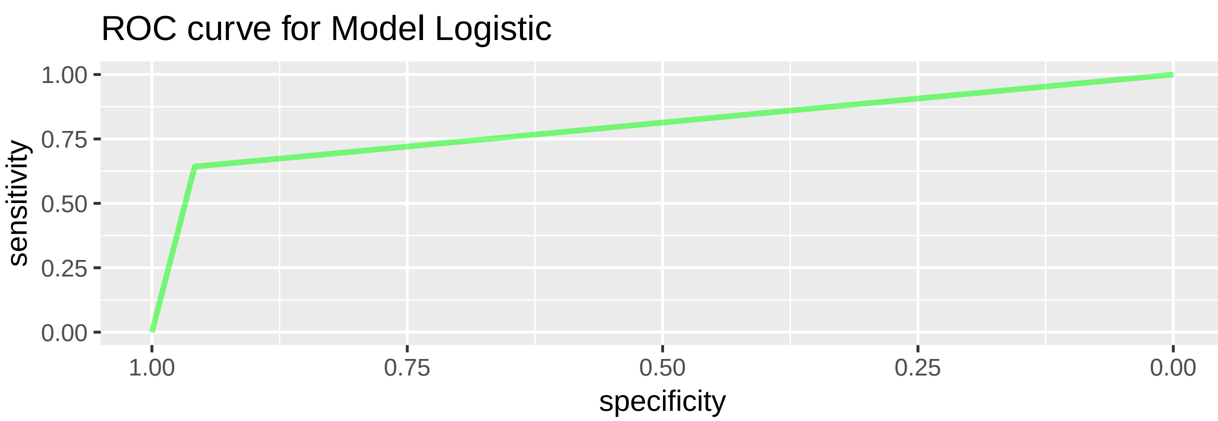


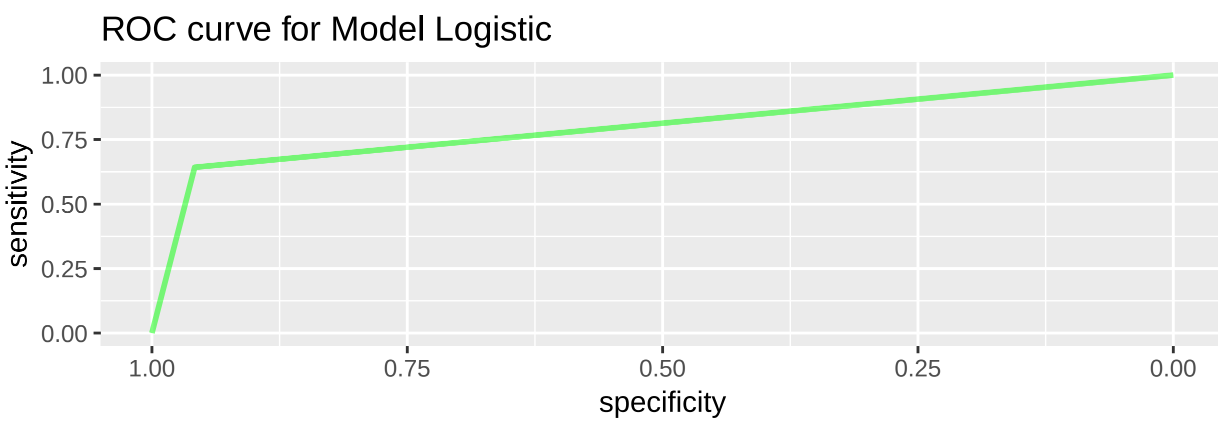

```r
##KNN
data("spam7")

spam7$crl.tot<-scale(spam7$crl.tot)
set.seed(1)
train_index<-createDataPartition(spam7$yesno,p = 0.7,list = F)
train<-spam7[train_index,]
test<-spam7[-train_index,]
crtl<-trainControl(method = "cv",number = 5)
```

```r
knn_c<-train(yesno~.,data=train,trControl=crtl,tuneLength=5)
plot(knn_c)
```




```
knn1<-knn(train = train[,-7],test = test[,-7],cl = train$yesno,k=3)
conf<-confusionMatrix(knn1,test$yesno,positive = "y")
R11<-ifelse(knn1=="y",1,0)
R22<-ifelse(test$yesno=="y",1,0)
rrr2<-roc(R22,R11)
g2<-ggroc(rrr2,alpha = 0.5, colour = "green", linetype = 1, size = 1)+ggtitle("ROC curve for Model KNN")


grid.arrange(g1,g1)
```





Knn is better because Sensitivity and Accuracy scores are better than scores provided by Logit model.

KNN is a non-parametric model, where LR is a parametric model. KNN is comparatively slower than Logistic Regression. KNN supports non-linear solutions where LR supports only linear solutions. LR can derive confidence level (about its prediction), whereas KNN can only output the labels.

Problem 2 (10 pt.)

- a. Suppose the sysadmin wants to predict the total length of words in capitals (based on their content and type) for new e-mails which make up 20% of your initial data. Compare the result of the linear regression and k-NN regression by solving the task. Which one is better? (Use RMSE, R squared to solve the task.)
- b. When will regression outperform the k-NN?

```
library(Metrics)
data("spam7")
set.seed(42)
spam7$yesno<-ifelse(spam7$yesno=="y",1,0)
index<-sample(nrow(spam7),nrow(spam7)*.8,replace = F)
train<-spam7[index,]
test<-spam7[-index,]
OLS<-lm(crl.tot~.,data = train)
summary(OLS)# Rsquared 0.08774
```

```
##
## Call:
```

```
## lm(formula = crl.tot ~ ., data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1670.0  -162.8  -116.5     0.5  8714.1
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   153.60      10.90  14.086  < 2e-16 ***
## dollar        201.03      38.64   5.203 2.07e-07 ***
## bang          -21.60      10.07  -2.145 0.032004 *
## money          98.86      26.43   3.740 0.000187 ***
## n000          121.02      27.05   4.473 7.94e-06 ***
## make           89.11      28.76   3.098 0.001962 **
## yesno         219.10      19.68  11.134  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 505.4 on 3673 degrees of freedom
## Multiple R-squared:  0.09878,    Adjusted R-squared:  0.0973
## F-statistic:  67.1 on 6 and 3673 DF,  p-value: < 2.2e-16
```

```r
OLS_PRED<-predict(OLS,test)
rmse(actual = test$crl.tot,predicted = OLS_PRED)#531.2555
```

```
## [1] 817.3943
```

```r
head(train)
```

```
##      crl.tot dollar  bang money n000 make yesno
## 4210      72  0.213 0.284     0    0 0.00     0
## 4311     311  0.000 0.059     0    0 0.00     0
## 1316     341  0.093 2.001     0    0 0.27     1
## 3819      33  0.000 0.000     0    0 0.00     0
## 2951      10  0.000 0.000     0    0 0.00     0
## 2386      53  0.000 0.000     0    0 0.00     0
```

```r
grid<-expand.grid(k=5:10)
set.seed(42)
model<-train(
 crl.tot~.,data=train,method='knn',
 trControl=trainControl("cv",number = 4),
 preProcess=c("center","scale"),
 tuneGrid=grid
)
model
```

```
## k-Nearest Neighbors
##
## 3680 samples
##    6 predictor
##
## Pre-processing: centered (6), scaled (6)
## Resampling: Cross-Validated (4 fold)
## Summary of sample sizes: 2761, 2760, 2759, 2760
## Resampling results across tuning parameters:
```

```
##
##   k   RMSE      Rsquared   MAE
##    5  430.2968  0.3951279  174.7312
##    6  436.3251  0.3791691  176.4263
##    7  475.8166  0.2780594  182.9299
##    8  477.2299  0.2720076  184.8276
##    9  478.1256  0.2689025  184.6216
##   10  479.6103  0.2623987  185.4297
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was k = 5.
```

I choose k=5 KNN regression provided better model Rmse is less that ols Rmse, regarding knn R_squared (0.3951279) is bigger than ols' one(0.08)

When will regression outperform the k-NN? When assumptions of "LINE" is met OLS can outperform knn regression.

Bonus 1 (1 pt.)

- Calculate the Cohen's kappa value without additional functions.
- Explain the meaning of using kappa statistics.

```
conf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   n   y
##          n 767 114
##          y  69 429
##
##                Accuracy : 0.8673
##                  95% CI : (0.8482, 0.8848)
##     No Information Rate : 0.6062
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7179
##  Mcnemar's Test P-Value : 0.001144
##
##             Sensitivity : 0.7901
##             Specificity : 0.9175
##          Pos Pred Value : 0.8614
##          Neg Pred Value : 0.8706
##              Prevalence : 0.3938
##          Detection Rate : 0.3111
##    Detection Prevalence : 0.3611
##       Balanced Accuracy : 0.8538
##
##        'Positive' Class : y
##
```

```r
Po<-(767+429)/(767+114+69+429)
Pyes<-((429+69 )/(767+114+69+429))*((429+114)/(767+114+69+429))
Pno<-((767+114)/(767+114+69+429))*((767+ 69)/(767+114+69+429))
Pe<-Pyes+Pno
kappa<-(Po-Pe)/(1-Pe)
kappa
```

```
## [1] 0.7179458
```

Cohen's Kappa is like classification accuracy, except that it is normalized at the baseline of random chance on your dataset. It is a more useful measure to use on problems that have an imbalance in the classes (e.g. 70-30 split for classes 0 and 1 and you can achieve 70% accuracy by predicting all instances are for class 0) The Kappa will tell you how much better, or worse, your classifier is than what would be expected by random chance. If you were to randomly assign cases to classes (i.e. a kind of terribly uninformed classifier), you'd get some correct simply by chance. Therefore, you will always find that the Kappa value is lower than the overall accuracy.

Bonus 2 (2 pt.)

Suppose we have one explanatory variable with equal values.

- How can we use 1-NN to predict the response value of new observation with the same value of an explanatory variable? Is there any problem?
- Do not use R to solve this task, or use it just as a supplementary tool.

The problem is that when new point comes distances will be the same with all existing points so it cannot chose the one closest neighbor as all are the same, so we should not have variables in our dataset that have variance equal to 0.