Solutions Online Test 3

Instructions: The test is worth up to 5 points. Each correct response is worth 1 point. There is a bonus question. Download the files "WorkspaceOnlineTest3part1.RData", "WorkspaceOnlineTest3part2.RData", "DataAnalyticsFunctions.R" and "OnlineTest3.R" to your working directory for this assignment. Open and execute the code "OnlineTest3.R" to answer the following questions.

The following three questions pertain to the churn problem discussed in class.

1. Consider using a logistic regression model to estimate the probability a customer will churn and the following splitting of the data into a train subset and a test subset:

```
foldid <- rep(1:nfold,each=ceiling(n/nfold))[sample(1:n)]
train <- which(foldid==1)
test  <- which(foldid==2)
model.logistic <-glm(Churn~., data=churndata, subset=train, family="binomial")
```

Your intern provides you with several different ways to evaluate the performance of the model. Given that you will present this in a business setting, which of options below is more suitable to communicate the performance of the model:

a)
```
pred.logistic<- predict(model.logistic, newdata=churndata[train,], type="response")
resp <-R2(y=churndata$Churn[train], pred=pred.logistic, family="binomial")
paste("We propose to use a logistic regression model. Its ", expression(R^2),"is", reps)
```

b)
```
pred.logistic<- predict(model.logistic, newdata=churndata[test,], type="response")
resp <-R2(y=churndata$Churn[test], pred=pred.logistic, family="binomial")
paste("We propose to use a logistic regression model. Its OOS", expression(R^2),"is", resp)
```

c)
```
pred.logistic<- predict(model.logistic, newdata=churndata[train,], type="response")
values <- FPR_TPR( (pred.logistic >= .3) , churndata$Churn[train]=="Yes" )
resp <- values$ACC
paste("We propose to use a logistic regression model. Its in-sample accuracy is", resp)
```

d) ✓
```
pred.logistic<- predict(model.logistic, newdata=churndata[test,], type="response")
values <- FPR_TPR( (pred.logistic >= .3) , churndata$Churn[test]=="Yes" )
resp <- values$ACC
paste("We propose to use a logistic regression model. Its OOS accuracy is", resp)
```

The initial code

```
> nfold <- 2
> n <- nrow(churndata)
> set.seed(1)
> foldid <- rep(1:nfold,each=ceiling(n/nfold))[sample(1:n)]
> train <- which(foldid==1)
> test  <- which(foldid==2)
```

splits the data into two parts, "train" and "test" (defines a foldid for each observation to be either 1 or 2 randomly and uses that id to split the sample of n observatinos). Then a logistic regression using the train data is performed

```
> model.logistic <-glm(Churn ~ . , data=churndata, subset=train, family
="binomial")
```

Note the use of `family="binomial"` *to tell R that it is a logistic regression, and the use of* `subset=train` *to tell R that it should use the subsample (train) in the regression.*

Option a) runs the regression but it reports in-sample R^2.

```
> pred.logistic<- predict(model.logistic, newdata=churndata[train,], ty
pe="response")
> resp <-R2(y=churndata$Churn[train], pred=pred.logistic, family="binom
ial")
```

(Note the use of `churndata[train,]` *that tells R to predict the training set which was used to fit the data. R^2 is computed via the function* `R2(…)`*.) This is not effective as it is not clear if you are overfitting the data by providing an "in-sample" performance.*

Option b) provides Out of Sample R^2.

```
> pred.logistic <- predict(model.logistic, newdata=churndata[test,], ty
pe="response")
> resp <-R2(y=churndata$Churn[test], pred=pred.logistic, family="binomi
al")
```

(Note the use of `churndata[test,]` *that tells R to predict the training set which was used to fit the data. R^2 is computed via the function* `R2(…)`*.) Although it is important to report and out of sample performance, this metric is not very interpretable for applications (especially in the logistic regression case). It tells you the proportion of deviations explained by the model relative to the model without variables (null model). However, deviations are measures via the likelihood which makes it harder to interpret than linear regression.*

Reporting accuracy in this binary setting provides a clear notion of performance.

Option c) offer an "in-sample" ACC which is not satisfactory as it could have been achieved via overfitting the data.

```
> pred.logistic<- predict(model.logistic, newdata=churndata[train,], ty
pe="response")
> values<-FPR_TPR((pred.logistic >=.3) ,churndata$Churn[train]=="Yes" )
> resp <- values$ACC
```

(Note the use of `churndata[train,]` *that tells R to predict the training set which was used to fit the data. Accuracy is computed via the function* `FPR_TPR(…)`*.)*

Option d) provides the out of the sample accuracy which is interpretable and likely to be the performance of the model when predicting.

```
> pred.logistic<-predict(model.logistic,newdata=churndata[test,], type=
"response")
> values<-FPR_TPR( (pred.logistic>=.3), churndata$Churn[test]=="Yes" )
> resp <- values$ACC
```

(Note the use of `churndata[test,]` *that tells R to predict the training set which was used to fit the data. Accuracy is computed via the function* `FPR_TPR(…)`*.) This is the correct option.*

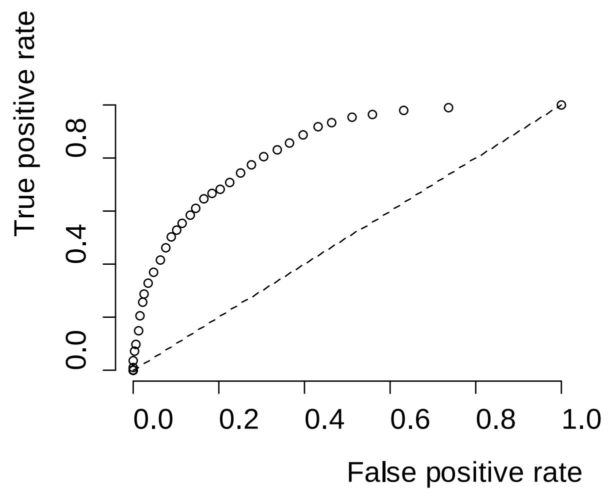
2. During a discussion of how to evaluate the quality of data mining, you were asked about principled ways to avoid overfitting when developing models. Which of the following options makes a better use of data?

a) ✓ performance measures from a k-fold cross validation
b) performance measures from an OOS experiment based on a hold-out sample
c) in-sample performance measures
d) comparing prediction of the model with the training data

Option a) is the correct answer as it suggests to use performance measures obtained from a k-fold cross validation. This is better than OOS experiment based on hold-out sample (option b) since in the cross validation we use every observation to fit a model and to evaluate the performance measure. Also, cross validation allows us to observe the fluctuations across folds which gives us an idea of fluctuation.

In-sample measures typically overfit the data and it is interesting to avoid. (This critique applies to option c) and c). Note that one can consider regularization approach but this was not mentioned in the question.)

3. In order to use make binary predictions with a logistic regression model you typically need to pick a threshold t for the predicted probability so that you predict Y=1 if the predicted probability is larger than t. The code provided in the file considers many choices of t and evaluate the associated models using false positive rate and true positive rate via 10-fold cross validation.



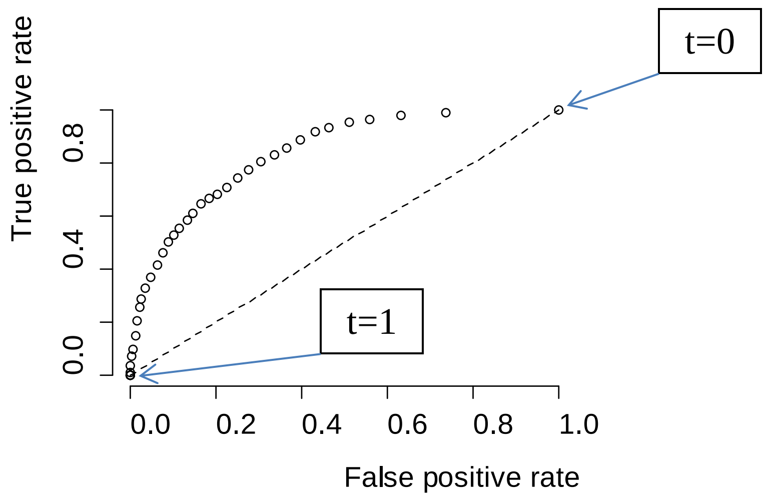
Among the values below what is the largest value of t such that we obtain TPR of .8 in this 10-fold cross validation?

a) 0.1
b) ✓0.25
c) 0.75
d) 0.9

The values of t are stored in

```
> val
 [1] 0.000 0.025 0.050 0.075 0.100 0.125 0.150 0.175 0.200 0.225 0.250
[12] 0.275 0.300 0.325 0.350 0.375 0.400 0.425 0.450 0.475 0.500 0.525
[23] 0.550 0.575 0.600 0.625 0.650 0.675 0.700 0.725 0.750 0.775 0.800
[34] 0.825 0.850 0.875 0.900 0.925 0.950 0.975 1.000
```

The vector of values of t starts from 0 with increments of 0.025 until 1. Keep in mind that as we increase t (from zero to one), in order to predict $\hat{Y} = 1$ via the rule $P(Y = 1) \geq t$, we will decrease the TPR and increase the FPR. So as we increase t, we move from the right to the left (and from high to low).



As shown in the code, the corresponding values of TPR is in

```
> rowMeans(CV.logistic.TPR)
 [1] 1.000000000 0.991596200 0.978855877 0.961949200 0.952770250
 [6] 0.934680823 0.920612568 0.894500674 0.851201017 0.830518700
[11] 0.802796127 0.773784550 0.739741967 0.711798293 0.692547135
[16] 0.671106350 0.644092332 0.607007036 0.585732556 0.549833683
[21] 0.527630677 0.490557990 0.451144828 0.418816204 0.369300594
[26] 0.331129627 0.297659936 0.254917986 0.201130533 0.153573088
[31] 0.104578415 0.069906068 0.035716701 0.015280229 0.004782581
[36] 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000
[41] 0.000000000
```

Since we want TPR at least 0.8, we have that the 11th entry of the vector is the lowest we can go.

```
> rowMeans(CV.logistic.TPR)[11]
[1] 0.8027961
```

This corresponds to t equal to 0.25

```
> val[11]
[1] 0.25
```

This is option b).

The following <u>three</u> questions pertain to the German credit data discussed in class (see "GermanCreditDescription.pdf" for details; note that "Default" is part of the data).

4. Similarity measures motivate various predictive methods. A marketing consultant wants to apply k-nearest neighbor as a predictive model. The consultant creates a matrix with several features of past customers

x <- model.matrix(Default~., data=germancredit)[,-1]

The consultant also has a code to create “train” and “test” subsets. Which of the following options of reasoning/code is more appropriate to evaluate the k-nn’s predictive accuracy?

a) Since k-nn is based on similarity there is no need to have an out of sample measure of performance

```
knn15 <- knn(train=x[train,], test=x[train,], cl=y[train], k=15, prob=TRUE)
prob.winning.class <-attr(knn15, "prob")
prob.default <-ifelse(knn15=="1", prob.winning.class, 1-prob.winning.class)
BinaryAccuracy( prob.default>.5, germancredit$Default[train])
```

b) Since k-nn is based on similarity we need to use a zero threshold on the probability

```
knn15 <- knn(train=x[train,], test=x[train,], cl=y[train], k=15, prob=TRUE)
prob.winning.class <-attr(knn15, "prob")
prob.default <-ifelse(knn15=="1", prob.winning.class, 1-prob.winning.class)
BinaryAccuracy( prob.default>0, germancredit$Default[train])
```

c) Since k-nn is based on similarity we should train in the test set to construct the predictions

```
knn15 <- knn(train=x[test,], test=x[test,], cl=y[train], k=15, prob=TRUE)
prob.winning.class <-attr(knn15, "prob")
prob.default <-ifelse(knn15=="1", prob.winning.class, 1-prob.winning.class)
BinaryAccuracy( prob.default>0, germancredit$Default[train])
```

d) ✓**k-nn is subject to overfitting as any data mining tool. We should provide some out of sample accuracy measure**

```
knn15 <- knn(train=x[train,], test=x[test,], cl=y[train], k=15, prob=TRUE)
prob.winning.class <-attr(knn15, "prob")
prob.default <-ifelse(knn15=="1", prob.winning.class, 1-prob.winning.class)
BinaryAccuracy( prob.default>.5, germancredit$Default[test])
```

Let me emphasize that because a method is based on similarity, it does not makes it less subject to overfitting.

Option a) is inappropriate as the use of any prediction method will be out of sample. Thus measuring the performance using an in-sample metric is likely to lead to an overly optimistic assess of the performance.

Option b) suggest to use the rule of prob.default > 0 to create predictions. This seems to predict that nearly all customers will default (very conservative and not related to the similarity measure). Also, by using the train set to fit the model and to evaluate the model, we are prone to get a overly optimistic assess of the performance of the method.

Option c) suggest to train the model in the test which cannot work.

Option d) is the only that trains in the train set and measure the performance in the test set. This assessment of performance is likely much less biased than in-sample measures.

R note: The code explanation is as follows. The command knn

```
> knn15 <-knn(train=x[train,],test=x[test,],cl=y[train],k=15,prob=TRUE)
```

runs the k-means with k=15 (uses the 15 nearest neighbors to define the estimate of the prediction). The "prob=TRUE" requests a probability estimate (instead of simply 0 or 1). Regarding the first, second and third argument of knn (highlighted below)

knn(train=x[train,], test=x[test,], cl=y[train],…)

we have that the first corresponds to the features (X) of training set. The second, test=x[test,], is the features (X) of the test set. Finally, the third is the target (Y) of the training set.

The command

```
> prob.winning.class <-attr(knn15, "prob")
```

Stores in `prob.winning.class` the largest between the probability of Y = 1 or Y=0 (the "winning" classification) for each point in the test set. This is a bit strange but it is how it was defined. To get the probability of default for each observation we need to convert some of the probabilities using

```
>prob.default<-ifelse(knn15=="1",prob.winning.class,1-prob.winning.clas
s)
```

It is a "if" command. If we predict class 1 (default), `knn15=="1"`, we keep the values of `prob.winning.class`. Otherwise, if we predict 0 (no default) we need to adjust the probability as `1-prob.winning.class`.

To compute the accuracy we used the .5 threshold to turn the probability into a rule to assign default (1) or no default (0) using `prob.default>.5`

```
> BinaryAccuracy( prob.default>.5, germancredit$Default[test])
```

A business insight is that overpromising might be helpful in the short term but it has bad consequences in the long term. Similar issues occur with being overly optimistic about the performance of the method. It is bound to disappoint you and not meet expectations in the deployment.

5. A marketing consultant wants to segment the market of customers that apply to loans. The consultant proposes to segment the market into three groups. Which of the following commands is more suitable for that?

a) Create numeric covariates based on the variables and apply the k-means algorithm with 3 centers. The consultant also summarizes how the clusters perform with respect to default.

```
x <- model.matrix(Default~., data=germancredit)[,-1]
three.clusters <- kmeans(x, 3, nstart=10)
tapply(germancredit$Default, three.clusters$cluster, table )
```

b) ✓**Create numeric covariates based on the variables, standardize the columns, and apply the k-means algorithm with 3 centers. The consultant also summarizes how the clusters perform with respect to default.**
x <- model.matrix(Default~., data=germancredit)[,-1]
x.scaled <- scale(x)
three.clusters <- kmeans(x.scaled, 3, nstart=10)
tapply(germancredit$Default, three.clusters$cluster, table)

c) Create numeric covariates based on the variables, append the information of default, and apply the k-means algorithm with 3 centers. The consultant also summarizes how the clusters perform with respect to default.
x <- model.matrix(Default~., data=germancredit)[,-1]
three.clusters <- kmeans(cbind(germancredit$Default,x), 3, nstart=10)
tapply(germancredit$Default, three.clusters$cluster, table)

d) Create numeric covariates based on the variables, append the information of default, standardize the columns, and apply the k-means algorithm with 3 centers. The consultant also summarizes how the clusters perform with respect to default.
x <- model.matrix(Default~., data=germancredit)[,-1]
x.scaled <- scale(cbind(germancredit$Default,x))
three.clusters <- kmeans(x.scaled, 3, nstart=10)
tapply(germancredit$Default, three.clusters$cluster, table)

Option a) does not standardize the variables which make the units used in the data matter. This is an issue that needs to be avoided.

Option b) standardize the columns (so they are unit free) and apply the k-means with 3 centers (i.e. k=3). This is an unsupervised method that leads to the creation of segments. It is important not to use default into the calculation of segments. These are intended to be how customers "organize" not how they behave with respect to default. (This typically allows us to interpret these segments.) However, it is interesting to compute the default rates on each segment as suggested (but default did not drive the creation of the segments).

Options c) and d) treat the default as another column. This has the issue of using the target as part of the unsupervised learning. This typically should be avoided in unsupervised methods (note that when a customer comes, you would like to identify its segment but there is no "default" information.)

6. The use of 3 clusters suggested by the consultant in Question 3 is motivated by simplicity. In order to select the number of clusters in a data-driven, you intend to use information criteria (a regularization technique). More specifically, BIC (Bayesian Information Criteria) due to its performance to avoid overfitting.

```
x <- model.matrix(Default~., data=germancredit)[,-1]
kfit <- lapply(1:200, function(k) kmeans(scale(x),k))
source("kIC.R")
kbic  <- sapply(kfit,kIC,"B")
plot(kbic, xlab="# of clusters (k)", ylab="Information Criterion", ylim=range(c(kbic)), type="l", lwd=2)
abline(v=which.min(kbic),col=4)
text(100,mean(kbic),"BIC")
kmin <- which.min(kbic)
k3   <- 3
1 - sum(kfit[[kmin]]$tot.withinss)/kfit[[kmin]]$totss
1 - sum(kfit[[k3]]$tot.withinss)/kfit[[k3]]$totss
```

Although there is no "perfect" choice for the number of clusters, which of the following statements better characterize the choice of the number of clusters?

a) ✓**It seems that 3 clusters are "too few" as they explain only 8.6% of the variation while BIC (which is guarding from overfitting) generates a model with 37.7%. The consultant should consider segmenting further the market (from the current 3 segments) to obtain a better balance. For example six clusters yields**

```
1 - sum(kfit[[6]]$tot.withinss)/kfit[[6]]$totss
```

b) It seems that 3 clusters is more than enough as the BIC selects less than 10 clusters.

```
kmin
```

c) It is important to use AIC instead of BIC because it is important to obtain more clusters to further segment.

```
kaic  <- sapply(kfit,kIC, "A")
which.min(kaic)
plot(kaic, xlab="# of clusters (k)", ylab="Information Criterion", ylim=range(c(kaic)), type="l", lwd=2)
```

d) The difference between the number of clusters selected based on BIC and AIC is substantial which indicates we cannot use k-means to segment.

```
which.min(kaic)
1 - sum(kfit[[which.min(kaic)]]$tot.withinss)/kfit[[which.min(kaic)]]$totss
which.min(kbic)
1-sum(kfit[[which.min(kbic)]]$tot.withinss)/kfit[[which.min(kbic)]]$totss
```

Option b) is incorrect as BIC selects well over 10 clusters (see kmin value).

Although AIC yields more clusters than BIC, option c) is incorrect as the goal is not to use more clusters. It is easy to use more clusters but that will lead to overfitting (and it is hard to interpret the meaning of too many clusters).
The difference option d) refers to is common. (AIC selects more clusters.) The main issue here would be trying to interpret that many clusters. This is why option a) is favorable. It

understands that 3 clusters might be too simplistic (and explain very little) and advocate for more clusters. Sometimes the data-driven ways suggests too many clusters that can be also hard to interpret. A compromise is probably a good option but more than 3 segments is desirable.