Applied Predictive Modeling 1/15/2018

Chapter 4: Overfitting and Model Tuning

Name: Kamaru-Deen Lawal

Email: NA

1

(a) Cross Validation: The data set is large enough where the each fold will result in a set where the number of samples is greater than the number of predictors for fairly large values of k. Problematic because of the unequal distribution of classes, but better than repeatedly generating sets of $12,495$ elements.

(b)

```
# 10 - Fold Cross Validation
set.seed(777)
trainrows <- createDataPartition(classes, p = .8, list = FALSE)
trainclasses <- classes[trainrows]
cvsplits <- createFolds(trainclasses, k = 10, list = TRUE, returnTrain = TRUE)
```

2

(a) Bootstrapping: $n = 1,107; p = 165$. The number of samples relative to the number of predictors makes it so that kf CV will result in problems for even small values of k, and having a single train/test split seems unreasonable given the skew in the data.

(b)

```
# Bootstrapping
resamples <- createResample(trainclasses, times = 10, list = TRUE)
```

3

(a) Four components yields the optimal R^2 value. Because the standard error is .0308 we are looking for the simplest model in $[.545 - .0308, .545 + .0308] = [.5142, .5755]$. The simplest model in this range has 3 components.

(b)

```
# % Tolerance Calculation
resampler2 <- c(.444, .5, .533, .545, .542, .537, .534, .534, .52, .507)
for (i in 1:10) {
  pertol <- ((.545 - resampler2[i]) / resampler2[i]) * 100
  print(pertol)
}
```

If a 10% loss in R^2 is acceptable, then the optimal number of components is two.

(c)

```
# % Tolerance Calculation
resampler2 <- c(.69, .67, .58, .54, .51, .45, .51)
for (i in 1:7) {
  pertol <- ((.69 - resampler2[i]) / resampler2[i]) * 100
  print(pertol)
}
```

(d) Given the model's prediction time, model complexity, and R^2 estimates I would select the SVM model.

4

(a)

```r
data(oil)
str(oilType)
# Original Table Distribution
(table(oilType) / 96) * 100
barplot((table(oilType) / 96) * 100, main = "Frequency Distr.", xlab = "Class", ylab =
    "Percent", ylim = c(0, 100), col = "green")

# Sample Distrbiutions
set.seed(777)
(table(sample(oilType, 60)) / 60) * 100
```

Original Distribution

A	B	C	D	E	F	G
38.5	27.0	3.1	7.2	11.4	10.4	2.0

Sample Distributions

A	B	C	D	E	F	G
40.0	23.3	3.3	8.3	11.6	11.6	1.6
35.0	31.6	1.6	10.0	10.0	8.3	3.3
35.0	28.3	1.6	8.3	13.3	10.0	3.3
45.0	20.0	3.3	5.0	10.0	15.0	1.6
43.3	30.0	5.0	3.3	8.3	8.3	1.6

Overall, the sample distributions seem to be inaccurate compared to the original distributions. This difference was even more noticeable in the classes that made up a small percentage of the distribution.

(b)

```r
# Stratified Random Sample w/ 80% of data
trainrows <- createDataPartition(oilType, p = .8, list = FALSE, groups = min(7,
    length(oilType)))
trainclasses <- oilType[trainrows]
(table(trainclasses) / 79) * 100
```

Stratified Sample Distribution

A	B	C	D	E	F	G
37.9	26.5	3.7	7.6	11.4	10.1	2.5

This resulting distribution was much closer to the original distributions compared to the randomly sampled sets.

(c) LOOCV ($k = n$) allows us to make the most of the scarce data. Having an entire test set seems unreasonable given the combination of severe class imbalance, and small number of samples. k-Fold cross validation for small values of k also could also work, but there is a high chance that one fold could contain all of the representatives for a given class, which means the other $k - 1$ folds would not be able to train on any examples with that class.

(d)

```
# CI for Prob of Sucess in Binomial Test
arr20 <- c(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20)
for (i in 1: 20) {
  print(i)
  print(binom.test(arr20[i], 20))
}
arr10 <- c(1,2,3,4,5,6,7,8,9,10)
for (i in 1: 10) {
  print(i)
  print(binom.test(arr10[i], 10))
}
# Plot Resulting Widths
y20 <- c(.24,.3,.34,.38,.41,.43,.44,.44,.45,.45,.45,.44,.44,.43,.41,.38,.34,.30,.24,.17)
y10 <- c(.44,.53,.59,.61,.63,.61,.59,.53,.44,.31)
plot(arr20, y20, col = "blue", type = "b", xlab = "Success", ylab = "Proportion", ylim =
    c(0, 1.0))
lines(arr10, y10, col = "red", type = "b")
```

i	j	95% CI Width
1	20	.24
2	20	.30
3	20	.34
4	20	.38
5	20	.41
6	20	.43
7	20	.44
8	20	.44
9	20	.45
10	20	.45
11	20	.45
12	20	.44
13	20	.44
14	20	.43
15	20	.41
16	20	.38
17	20	.34
18	20	.30
19	20	.24
20	20	.17
1	10	.44
2	10	.53
3	10	.59
4	10	.61
5	10	.63
6	10	.61
7	10	.59
8	10	.53
9	10	.44
10	10	.31

The confidence intervals are narrower for larger values of n.