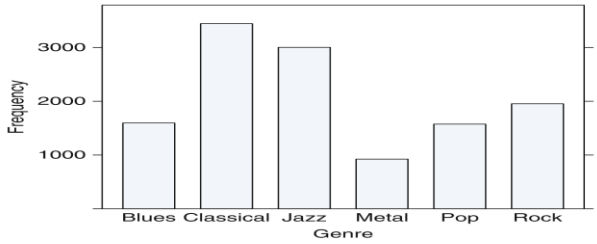
Over-Fitting and Model Tuning

Ibrahim Odumas Odufowora

2017-10-06

Question 1: Music Genre Dataset:



The frequency distribution of genres in the music data.

- The dimension of the dataset is 12495(samples) by 191(predictors).

Q1(a) Data splitting method(s) for these data?:

Using the information above, the number of samples (12495) is largely greater than the number of predictors (195). Hence, it is visible to split the dataset into training and test set, this will enable the evaluation of model performance and tuning parameter selection.

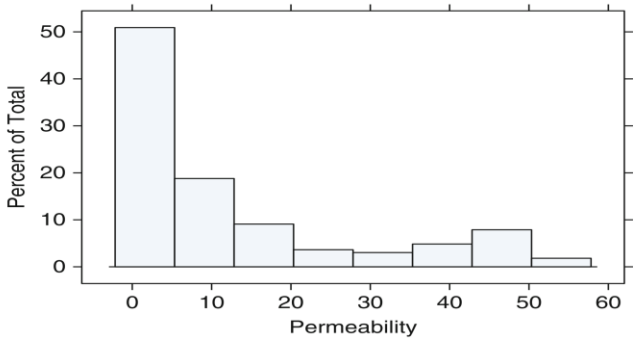
Given the imbalance in the distribution of classes in the response variable, classical has the highest percentage and metal with the lowest(from the figure above), it might be suggested to use stratified random sampling method to split the dataset.

Q1(b) Code for implementing the approach(es):

The createDataPartition function in the caret package would be used to split the dataset into training and test datasets using the classes:

- trainingRows = createDataPartition(classes, p = .80, list= FALSE)
  (apportioning 80% as training set)

Question 2: Permeability Dataset:



The frequency distribution of permeability value in the dataset.

- The dimension of the dataset is 165(samples) by 1107(predictors).

Q2(a) Data splitting method(s) for these data?:

From the figure above, the distribution of permeability value is skewed. Also, the number of samples (165) is largely less than the number of predictors (1107). Because the sample size is small, it is not advised to split the dataset into training and test set, splitting the dataset into testing and training sets might affect the ability to get a good linkage between the predictors and the response variables. In this case, resampling techniques should be used to select tuning parameters and estimate performance.

Q2(b) Code for implementing the approach(es):

The createDataPartition function in the caret package would be used to create stratified sampling such that the class distribution is sustainably maintained. The following code would be used to create multiple iterations of 4 folds cross-validation:

- multi_folds = createMultiFolds(permeability, k = 4, times = 20)

Hence, at each time 75% of the data will be used as the training and 25% as testing.

Seed should be set for reproducibility.

Question 3: Partial Least Square (PLS):

Q3(a): Calculate the PLS components provides the most parsimonious model:

	Resampled R^2	
Components	Mean	Std. Error
1	0.444	0.0272
2	0.500	0.0298
3	0.533	0.0302
4	0.545	0.0308
5	0.542	0.0322
6	0.537	0.0327
7	0.534	0.0333
8	0.534	0.0330
9	0.520	0.0326
10	0.507	0.0324

From the table above, the best setting is at 4 PLS, at one standard error it has the following boundaries:

- lower boundary 0.545 - 0.0308 = 0.5142

- upper boundary 0.545 + 0.0308 = 0.5758

Setting with 3 PLS (0.534) has R^2 that is better than the lower boundary (0.5142), hence, a model with 3 PLS components is the most parsimonious model (simpler).

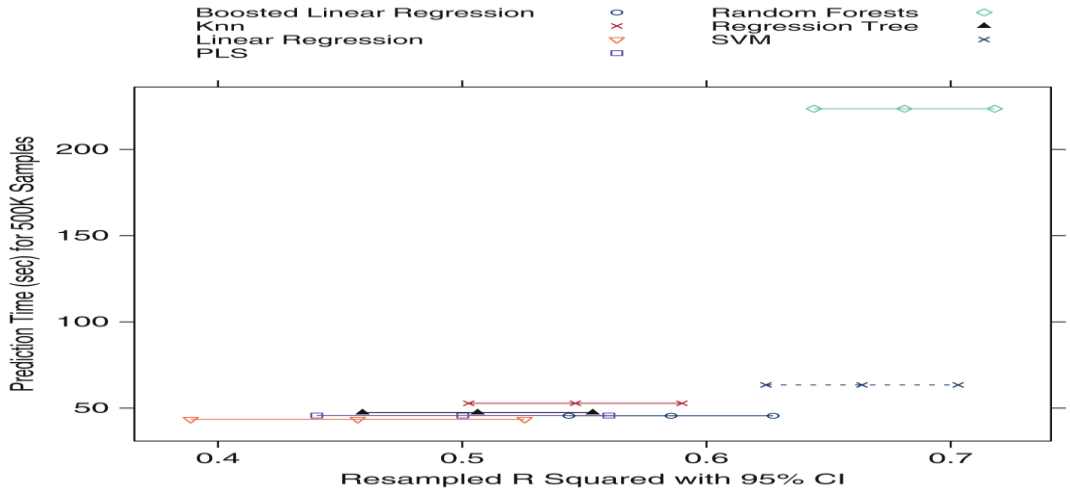
Q3(b): Compute tolerance values and estimate the optimal PLS component at 10% loss of R^2:

Resampled R2			
Components	Mean	Std. Error	Tolerance
1	0.4440	0.0272	-0.1853
2	0.5000	0.0298	-0.0826
3	0.5330	0.0302	-0.0220
4	0.5450	0.0308	0.0000
5	0.5420	0.0322	-0.0055
6	0.5370	0.0327	-0.0147
7	0.5340	0.0333	-0.0202
8	0.5340	0.0330	-0.0202
9	0.5200	0.0326	-0.0459
10	0.5070	0.0326	-0.0697
Numerical Optimal value = 0.545			

Computed tolerance values

Given that a 10% loss is accepted, then the best optimal number of PLS components is at 2 PLS components with R^2 of 0.500.

Q3(c): Select the model(s) that optimizes R^2:



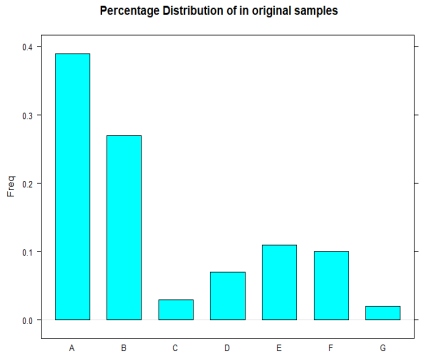
Computed tolerance values

From the figure above the random forest has the highest value of R^2, albeit, the R^2 value for the SVM is relatively close to that of the random forest, with some overlap. Thus, the best models in terms of optimal R^2 values are random forest and support vector machine.

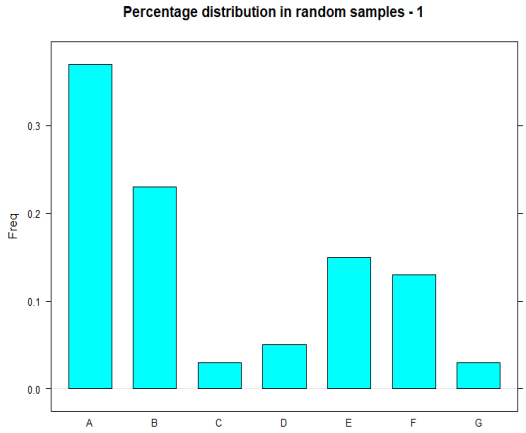
Q3(d): Select model(s) based on prediction time, model complexity, and R2 estimates:

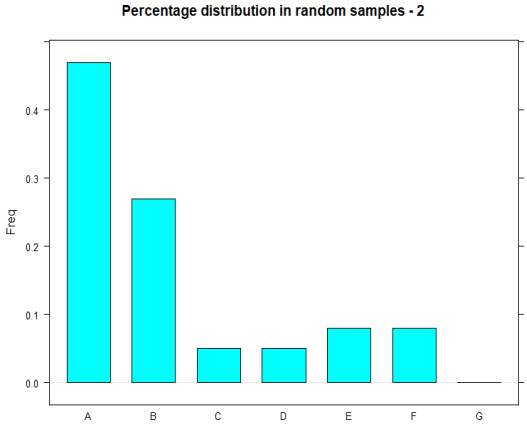
Given each model's prediction time, model complexity, and R^2 estimates **the SVM should be chosen since it is fast and its R^2 is relatively close to the best R^2.**

Question 4: Oil:



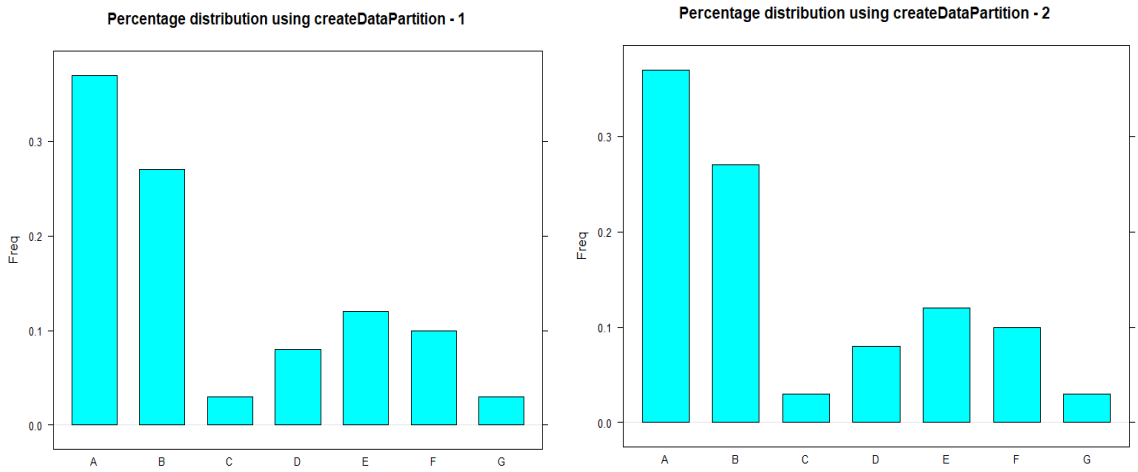
Q4(a) Sampling using random sample:





Frequencies in the random sample differ from that of the original samples. 30 different random samplings of 60 samples each were further looked, yet there frequencies distribution differ from the original sample. In some instance the frequency of 'G' is zero (as seen above), hence, the training set will not capture all the classes. This might be ineffective for modeling.

Q4(b) Sampling using the caret package function createDataPartition to create stratified random sample:



The createDataPartition function generates random samples that are significantly closer to the original sample in terms of the frequency distribution. It tends to relatively maintain the frequencies distribution in the original dataset. When compared with the use of random sampling, it produces a better result in terms of keeping the frequencies distribution of the original dataset. Also, this tends to include all the classes in the random sample selection.

Thus, this is preferred over the approach used in Q(4a).

Q4(c) Determining the performance of a model with small sample size:

In any case, where there is small sample size, it might be inefficient to partition the dataset into train and test datasets. This is because the train set might not be sufficient to capture all aspects of the predictors.

Hence, LOOCV would be a reasonable option to determine the performance of the model.

Q4(d) Understanding the uncertainty of a test set using binomial test:

Table of width using different sample size and accuracy

sample_size	accuracy	lower_bound	upper_bound	width
10	0.90	0.555	0.997	0.442
15	0.90	0.681	0.998	0.317
20	0.90	0.683	0.988	0.305
25	0.90	0.688	0.975	0.287
30	0.90	0.735	0.979	0.244
20	0.75	0.509	0.913	0.404
20	0.80	0.563	0.943	0.380
20	0.85	0.621	0.968	0.347
20	0.90	0.683	0.988	0.305
20	0.95	0.751	0.999	0.248

From the table above, the width of the confidence interval for reduces as the sample size increases. Likewise, the width reduces as the accuracy increases. Hence, if accuracy cannot be increased, then increased sample size can aid a better model. Also, if sample size cannot be increased, then increased accuracy would result in a better model.

Appendix:

##Q3(b)

```{r}
error = c(0.0272,  0.0298,  0.0302,  0.0308, 0.0322,  0.0327,  0.0333,  0.0330, 0.0326, 0.0324)

mean = c(0.444,  0.500, 0.533, 0.545,  0.542, 0.537, 0.534, 0.534, 0.520, 0.507)

toler = round((mean - 0.545) / 0.545, 4)
```

#Question 4: Oil:

```{r}
data(oil)

#str(oilType)

tb = round(table(oilType) / 96, 2)

barchart(tb, horizontal = F, main = 'Percentage Distribution of in original samples')

dist = as.data.frame(round(table(oilType) / 96, 2))

#kable(dist, caption = 'Percentage Distribution of in original samples')
```

##Q4(a) Sampling using random sample:

```{r}
sampNum = 60

set.seed(23123)

list_table = vector(mode = "list", length = 30)


#tb = round(table(oilType) / 96, 2)

#barchart(tb, horizontal = F)


for(i in 1:length(list_table))

 list_table[[i]] = round(table(sample(oilType, size = sampNum)) / 60, 2)

 #list_table[[i]] = sample(oilType, size = sampNum))
```

barchart(list_table[[1]], horizontal = F, main = 'Percentage distribution in random samples - 1')

barchart(list_table[[2]], horizontal = F, main = 'Percentage distribution in random samples - 2')

#cat("Percentage distribution in random samples")

#head(list_table, 5)

```

##Q4(b) Sampling using the caret package function createDataPartition to create stratified random sample:

```{r}

seed.set = 234901

list_caret = createDataPartition(oilType, p = 0.59, times = 30)

perc_caret = lapply(list_caret, function(x, y) round(table(y[x])/60, 2), y = oilType)

barchart(perc_caret[[1]], horizontal = F, main = 'Percentage distribution using createDataPartition - 1')

barchart(perc_caret[[2]], horizontal = F, main = 'Percentage distribution using createDataPartition - 2')

#cat("Percentage distribution using createDataPartition")

#head(perc_caret, 5)

```

##Q4(d) Understanding the uncertainty of a test set using binomial test:

```{r}

sample_size = c(10, 15, 20, 25, 30, 20, 20, 20, 20, 20)

accuracy = c(0.9, 0.9, 0.9, 0.9, 0.9, 0.75, 0.80, 0.85, 0.9, 0.95)

bin1 = binom.test(round(accuracy[1]*sample_size[1]), sample_size[1])

dt = t(as.data.frame(round(bin1$conf.int, 3)))


for(i in 2:10)

{

  bin = binom.test(round(accuracy[i]*sample_size[i]), sample_size[i])

```
  new_tb = t(as.data.frame(round(bin$conf.int, 3)))
  dt = rbind(dt, new_tb)
}

rownames(dt) = NULL
colnames(dt) = c('lower_bound', 'upper_bound')

dt1 = data.frame(sample_size, accuracy)
dt2 = cbind(dt1, dt)
dt2$width = dt2$upper_bound - dt2$lower_bound
kable(dt2, caption = "Table of width using diffrent sample size and accuracy")
```