Hw3 Part 2 Review

CS100

Hyperparameters

- These are parameters that are used to specify a learning model (eg. the k in k -NN, the depth of a decision tree)

- Naive Bayes does not have any hyperparameters, beyond the choice of variables.

- To control these, we call *rpart.* For instance:

```
nb2 <- NaiveBayes(Category ~ Serving.Size + Cholesterol, data = nutrition_train)
dt2 <- rpart(Category ~ Serving.Size + Calories, data = nutrition_train)
```

Here we're calling r part and using 2 variables from the dataset

Task: model building

For each section (i.e., number of features), you are tasked with building three models:

- k-NN
- decision trees
- naive Bayes

Checking accuracy: use the *predict* to compare your efficiency with the benchmarks set

Best 1-feature model

Variable used here: Serving.Size

```
nb1 <- NaiveBayes(Category ~ Serving.Size, data = nutrition_train)
dt1 <- rpart(Category ~ Serving.Size, data = nutrition_train)
knn1 <- knn(dplyr::select(nutrition_train, Serving.Size), dplyr::select(nutrition_test,
Serving.Size), nutrition_train$Category, k = 8)
```

Printing out results:

```
print(acc(predict(nb1, nutrition_test)))
print(acc(predict(dt1, nutrition_test, type="class")))
print(acc(knn1))
```

Best 2-feature model

Variable used here: Serving.Size, Calories

```
nb2 <- NaiveBayes(Category ~ Serving.Size + Cholesterol, data = nutrition_train)
dt2 <- rpart(Category ~ Serving.Size + Calories, data = nutrition_train)
knn2 <- knn(dplyr::select(nutrition_train, Serving.Size, Calories), dplyr::select(nutrition_test,
Serving.Size, Calories), nutrition_train$Category, k = 8)
```

Printing out results:

```
print(acc(predict(nb2, nutrition_test)))
print(acc(predict(dt2, nutrition_test, type="class")))
print(acc(knn2))
```

Best 3-feature model

Variable used here: Serving.Size, Calories, Cholesterol

```
nb3 <- NaiveBayes(Category ~ Cholesterol + Serving.Size + Vitamin.A....Daily.Value., data =
nutrition_train)
dt3 <- rpart(Category ~ Serving.Size + Calories + Cholesterol, data = nutrition_train)
knn3 <- knn(dplyr::select(nutrition_train, Serving.Size, Calories, Cholesterol),
dplyr::select(nutrition_test, Serving.Size, Calories, Cholesterol), nutrition_train$Category, k =
8)
```

Printing out results:

```
print(acc(predict(nb3, nutrition_test)))
print(acc(predict(dt3, nutrition_test, type="class")))
print(acc(knn3))
```

4-feature model

Variable used here: Serving.Size, Calories, Cholesterol, Sodium

```
# # Sugars, Cholesterol, Sodium give the best results
nb4 <- NaiveBayes(Category ~ Vitamin.A....Daily.Value. + Cholesterol + Serving.Size + Calories,
data = nutrition_train)
dt4 <- rpart(Category ~ Serving.Size + Calories + Cholesterol + Sodium, data = nutrition_train)
knn4 <- knn(dplyr::select(nutrition_train, Serving.Size, Calories, Cholesterol, Sodium),
dplyr::select(nutrition_test, Serving.Size, Calories, Cholesterol, Sodium),
nutrition_train$Category, k = 8)
```

Printing out results:

```
print(acc(predict(nb4, nutrition_test)))
print(acc(predict(dt4, nutrition_test, type="class")))
print(acc(knn4))
```

You get the idea. Now explore combinations and come up with the best model you can!

Cross-validation

Firstly, merge the nutrition training and test sets into a larger data frame:

```
nutrition_cv <- merge(nutrition_train, nutrition_test, all = TRUE)
folds <- createFolds(nutrition$Category, k = 3)
```

Now, use the `caret` library to create three folds, just as you did in studio.
Re-evaluate your models: i.e., train on folds 1 and 2, test on fold 3, and so on...
Observe: does the accuracy of either of your models change when you use three folds instead?

```
print("NB test on fold 1")
nb <- NaiveBayes(Category ~ Serving.Size + Calories + Cholesterol, data =
nutrition_cv[c(folds[[2]], folds[[3]]),])
print(mean(predict(nb, nutrition_cv[folds[[1]],]) == nutrition_cv[folds[[1]],]$Category))

print("NB test on fold 2")
nb <- NaiveBayes(Category ~ Serving.Size + Calories + Cholesterol, data =
nutrition_train[c(folds[[1]], folds[[3]]),])
print(mean(predict(nb, nutrition_cv[folds[[2]],]) == nutrition_cv[folds[[2]],]$Category))

print("NB test on fold 3")
nb <- NaiveBayes(Category ~ Serving.Size + Calories + Cholesterol, data =
nutrition_train[c(folds[[1]], folds[[2]]),])
print(mean(predict(nb, nutrition_cv[folds[[3]],]) == nutrition_cv[folds[[3]],]$Category))
```