HW 4 Q1

Ben Howell

3/13/2022

Question 5

```
require(tidyverse)
require(ISLR2)

df <- Default %>%
  dplyr::mutate(default_mode = ifelse(default == "No",
                                      0, 1))
```

```
set.seed(132)
```

```
model1 <- glm(default_mode ~ income + balance,
              data = df, family = "binomial")

summary(model1)
```

```
##
## Call:
## glm(formula = default_mode ~ income + balance, family = "binomial",
##     data = df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4725  -0.1444  -0.0574  -0.0211   3.7245
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.154e+01  4.348e-01 -26.545  < 2e-16 ***
## income       2.081e-05  4.985e-06   4.174 2.99e-05 ***
## balance      5.647e-03  2.274e-04  24.836  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2920.6  on 9999  degrees of freedom
## Residual deviance: 1579.0  on 9997  degrees of freedom
## AIC: 1585
##
## Number of Fisher Scoring iterations: 8
```

Model 1 Approach

```r
s <- c(0.9, 0.8, 0.7, 0.5)

table(df$default)
```

```
##
##   No  Yes
## 9667  333
```

```r
# default rate of ~ 3.33%

for (y in s) {
  split <- sort(sample(nrow(df), nrow(df) * y))

  train <- df[split, ]
  test <- df[-split, ]

  model2 <- glm(default ~ income + balance, data = train,
                family = "binomial")

  pred <- predict.glm(object = model2, newdata = test,
                      type = "response")

  pred <- data.frame(pred) %>%
    # 1 = default
    # 0 = doesn't default
    dplyr::mutate(default_binary = ifelse(pred > 0.5,
                                          1, 0))

  test <- test %>%
    dplyr::mutate(order = row_number()) %>%
    left_join(pred %>%
                dplyr::mutate(order = row_number()),
              by = c("order"))

  def_table <- table(test$default_mode, test$default_binary)

  acc <- scales::percent(sum(diag(def_table) / sum(def_table)),
                         accuracy = 0.1)

  mess <- paste0("For a train/test split of ", y, " and ", 1 - y,
                 " the model has an accuracy of ", acc, "!")
  print(mess)
}
```

```
## [1] "For a train/test split of 0.9 and 0.1 the model has an accuracy of 97.0%!"
## [1] "For a train/test split of 0.8 and 0.2 the model has an accuracy of 97.5%!"
## [1] "For a train/test split of 0.7 and 0.3 the model has an accuracy of 97.5%!"
## [1] "For a train/test split of 0.5 and 0.5 the model has an accuracy of 97.2%!"
```

Dummy Variable Approach

```
s <- c(0.9, 0.8, 0.7, 0.5)

table(df$default)
```

```
##
##   No  Yes
## 9667  333
```

```
# default rate of ~ 3.33%

df <- df %>%
  dplyr::mutate(is_student = ifelse(student == "Yes", 1, 0))

for (y in s) {
  split <- sort(sample(nrow(df), nrow(df) * y))

  train <- df[split, ]
  test <- df[-split, ]

  model2 <- glm(default ~ income + balance + is_student, data = train,
               family = "binomial")

  pred <- predict.glm(object = model2, newdata = test,
                      type = "response")

  pred <- data.frame(pred) %>%
    # 1 = default
    # 0 = doesn't default
    dplyr::mutate(default_binary = ifelse(pred > 0.5,
                                          1, 0))

  test <- test %>%
    dplyr::mutate(order = row_number()) %>%
    left_join(pred %>%
               dplyr::mutate(order = row_number()),
             by = c("order"))

  def_table <- table(test$default_mode, test$default_binary)

  acc <- scales::percent(sum(diag(def_table) / sum(def_table)),
                         accuracy = 0.1)

  mess <- paste0("Train: ", y, "; Test: ", 1 - y,
                " + a student dummy variable, the model has an accuracy of ",
                acc, "!")
  print(mess)
}
```

```
## [1] "Train: 0.9; Test: 0.1 + a student dummy variable, the model has an accuracy of 97.6%!"
## [1] "Train: 0.8; Test: 0.2 + a student dummy variable, the model has an accuracy of 96.7%!"
```

```
## [1] "Train: 0.7; Test: 0.3 + a student dummy variable, the model has an accuracy of 97.2%!"
## [1] "Train: 0.5; Test: 0.5 + a student dummy variable, the model has an accuracy of 97.4%!"
```

It's hard to tell for certain, but it appears that the accuracy of the model slightly improves with the inclusion of the student dummy variable. Using the dummy variable, plus a 90/10 train/test split produced an accuracy rate of 97.6%, the highest of all our iterations. Overall, each of the iterations was around the 97% accuracy rate, with slight variations in each direction.