Homework 5

Alex Ojemann

2023-10-25

Problem 1

To begin I created four new features, Spring, Summer, Fall, and Winter, to represent the season based on the month, using the majority season in cases where a month contains multiple seasons. I did so to capture the effects of seasonal variation without introducing too many dummy variables into my model. I split the data into training and testing sets with 80% in the training set and 20% in the test set. I used backward stepwise feature selection to select features for both models. For logistic regression, backward stepwise selection using BIC resulted in a model with just one feature, Summer, which had an accuracy of 0.559 and an F1 score of 0.380 on the test set. Since the model had so few predictors, I wanted to try to select the features using AIC instead because AIC has less of a penalty for adding features. The model included both the Summer and Fall variables as predictors, but achieved a worse accuracy (0.550) and F1 score (0.306) on the test set than the model selected using BIC. For the support vector machine, there aren't any existing metrics that are used for classification models that penalize a large number of features, so I added a penalty term to the F1 score to use for stepwise feature selection. The resulting model had an accuracy of 0.550 and an F1 score of 0.306. I also tried to use a lasso penalty term to select features for the SVM instead of stepwise selection and the resulting model had an accuracy of 0.550 and an F1 score of 0.306, presumably the same model. So in the end the logistic regression model with just one predictor performed the best.

Problem 2

I read the paper.

Appendix

Logistic Regression with feature selection using BIC:

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
df = read.csv('forestfires.csv')
df = df %>% mutate(fire = ifelse(area>0,1,0))
df$Spring <- ifelse(df$month %in% c('apr', 'may', 'jun'), 1, 0)
df$Summer <- ifelse(df$month %in% c('jul', 'aug', 'sep'), 1, 0)
df$Fall <- ifelse(df$month %in% c('oct', 'nov', 'dec'), 1, 0)
df$Winter <- ifelse(df$month %in% c('jan', 'feb', 'mar'), 1, 0)
df = df[c('FFMC','DMC','DC','ISI','temp','RH','wind','rain','fire','Spring','Summer','Fall','Winter')]

set.seed(323)
sample <- sample(c(TRUE, FALSE), nrow(df), replace=TRUE, prob=c(0.8,0.2))
train  <- df[sample, ]
test   <- df[!sample, ]
df = train

get_bic <- function(myModel){
  n_obs <- nobs(myModel)  # Number of observations
  log_likelihood <- logLik(myModel)  # Log-likelihood
  k <- length(coef(myModel))  # Number of parameters in the model
  bic <- -2 * log_likelihood + k * log(n_obs)
  return(bic)
}

bestModel = glm(fire ~ ., data = df, family = binomial)
end = FALSE
while(end == FALSE){
  num_features = length(bestModel$coefficients) - 1  # Subtract 1 for the intercept
  feature_names = names(coef(bestModel))[-1]  #Exclude the intercept
  bestBic = get_bic(bestModel)
  for(i in 1:num_features){
    features = feature_names[feature_names != feature_names[i]]
    feature_formula <- as.formula(paste("fire ~", paste(features, collapse = "+")))
    model_i = glm(feature_formula, data = df, family = binomial)
    bic = get_bic(model_i)
    if(bic < bestBic){
      bestBic = bic
      bestModel = model_i
    }
  }
  if(length(bestModel$coefficients) - 1 == num_features | length(bestModel$coefficients) - 1 == 1){
    end = TRUE
  }
}

summary(bestModel)

predicted_values <- predict(bestModel, newdata = test, type = "response")
predicted_classes <- ifelse(predicted_values > 0.5, 1, 0)

library(MLmetrics)
```

```
##
## Attaching package: 'MLmetrics'
```

```
## The following object is masked from 'package:base':
##
##     Recall
```

```
actual_values <- test$fire
f1_score <- F1_Score(actual_values, predicted_classes)
accuracy <- Accuracy(y_true = actual_values, y_pred = predicted_classes)
print(f1_score)
print(accuracy)
```

Logistic Regression with feature selection using AIC:

```
library(dplyr)
df = read.csv('forestfires.csv')
df = df %>% mutate(fire = ifelse(area>0,1,0))
df$Spring <- ifelse(df$month %in% c('apr', 'may', 'jun'), 1, 0)
df$Summer <- ifelse(df$month %in% c('jul', 'aug', 'sep'), 1, 0)
df$Fall <- ifelse(df$month %in% c('oct', 'nov', 'dec'), 1, 0)
df$Winter <- ifelse(df$month %in% c('jan', 'feb', 'mar'), 1, 0)
df = df[c('FFMC','DMC','DC','ISI','temp','RH','wind','rain','fire','Spring','Summer','Fall','Winter')]

set.seed(323)
sample <- sample(c(TRUE, FALSE), nrow(df), replace=TRUE, prob=c(0.8,0.2))
train  <- df[sample, ]
test   <- df[!sample, ]
df = train

get_aic <- function(myModel){
  log_likelihood <- logLik(myModel)  # Log-likelihood
  k <- length(coef(myModel))  # Number of parameters in the model
  aic <- -2 * log_likelihood + k * 2
  return(aic)
}

bestModel = glm(fire ~ ., data = df, family = binomial)
end = FALSE
while(end == FALSE){
  num_features = length(bestModel$coefficients) - 1  # Subtract 1 for the intercept
  feature_names = names(coef(bestModel))[-1]  #Exclude the intercept
  bestAic = get_aic(bestModel)
  for(i in 1:num_features){
    features = feature_names[feature_names != feature_names[i]]
    feature_formula <- as.formula(paste("fire ~", paste(features, collapse = "+")))
    model_i = glm(feature_formula, data = df, family = binomial)
    aic = get_aic(model_i)
    if(aic < bestAic){
      bestAic = aic
      bestModel = model_i
    }
  }
  if(length(bestModel$coefficients) - 1 == num_features | length(bestModel$coefficients) - 1 == 1){
    end = TRUE
  }
```

```
}

summary(bestModel)

predicted_values <- predict(bestModel, newdata = test, type = "response")
predicted_classes <- ifelse(predicted_values > 0.5, 1, 0)

library(MLmetrics)
actual_values <- test$fire
f1_score <- F1_Score(actual_values, predicted_classes)
accuracy <- Accuracy(y_true = actual_values, y_pred = predicted_classes)
print(f1_score)
print(accuracy)
```

SVM with feature selection using F1 score:

```
library(dplyr)
library(e1071)
df = read.csv('forestfires.csv')
df = df %>% mutate(fire = ifelse(area>0,1,0))
df$Spring <- ifelse(df$month %in% c('apr', 'may', 'jun'), 1, 0)
df$Summer <- ifelse(df$month %in% c('jul', 'aug', 'sep'), 1, 0)
df$Fall <- ifelse(df$month %in% c('oct', 'nov', 'dec'), 1, 0)
df$Winter <- ifelse(df$month %in% c('jan', 'feb', 'mar'), 1, 0)
df = df[c('FFMC','DMC','DC','ISI','temp','RH','wind','rain','fire','Spring','Summer','Fall','Winter')]

set.seed(323)
sample <- sample(c(TRUE, FALSE), nrow(df), replace=TRUE, prob=c(0.8,0.2))
train  <- df[sample, ]
test   <- df[!sample, ]
df = train

get_f1 <- function(myModel,data){
  predicted_labels <- predict(myModel, newdata = data)

  actual_labels <- train$fire
  tp <- sum(predicted_labels == 1 & actual_labels == 1)
  fp <- sum(predicted_labels == 1 & actual_labels == 0)
  fn <- sum(predicted_labels == 0 & actual_labels == 1)

  precision <- tp / (tp + fp)
  recall <- tp / (tp + fn)
  f1_score <- 2 * precision * recall / (precision + recall) - dim(bestModel@xmatrix[[1]])[2] * 0.01
  return(f1_score)
}

x <- as.matrix(train[-9])  # Extract features
y <- train$fire
bestModel <- ksvm(x, y, type = "C-svc", kernel = "vanilladot", C = 1, prob.model = TRUE)

end = FALSE
while(end == FALSE){
  num_features <- dim(bestModel@xmatrix[[1]])[2]
```

```
  bestF1 = get_f1(bestModel)
  for(i in 1:num_features){
    x_i <- x[, -i, drop = FALSE]
    model_i = ksvm(x_i, y, type = "C-svc", kernel = "vanilladot", C = 1, prob.model = TRUE)
    f1 = get_f1(model_i)
    if(f1 < bestF1){
      bestF1 = f1
      bestModel = model_i
    }
  }
  if(dim(bestModel@xmatrix[[1]])[2] == num_features | dim(bestModel@xmatrix[[1]])[2] == 1){
    end = TRUE
  }
}

summary(bestModel)

predicted_values <- predict(bestModel, newdata = as.matrix(test[-9]), type = "response")
predicted_classes <- ifelse(predicted_values > 0.5, 1, 0)

actual_values <- test$fire
f1_score <- F1_Score(actual_values, predicted_classes)
accuracy <- Accuracy(y_true = actual_values, y_pred = predicted_classes)
print(f1_score)
print(accuracy)
```

SVM with lasso penalty:

```
library(dplyr)
library(kernlab)
df = read.csv('forestfires.csv')
df = df %>% mutate(fire = ifelse(area>0,1,0))
df$Spring <- ifelse(df$month %in% c('apr', 'may', 'jun'), 1, 0)
df$Summer <- ifelse(df$month %in% c('jul', 'aug', 'sep'), 1, 0)
df$Fall <- ifelse(df$month %in% c('oct', 'nov', 'dec'), 1, 0)
df$Winter <- ifelse(df$month %in% c('jan', 'feb', 'mar'), 1, 0)
df = df[c('FFMC','DMC','DC','ISI','temp','RH','wind','rain','fire','Spring','Summer','Fall','Winter')]

set.seed(323)
sample <- sample(c(TRUE, FALSE), nrow(df), replace=TRUE, prob=c(0.8,0.2))
train  <- df[sample, ]
test   <- df[!sample, ]

# Train the SVM with L1 regularization
x_train <- as.matrix(train[-9])  # Extract features
y_train <- train$fire
svm_model <- ksvm(x_train, y_train, type = "C-svc", kernel = "vanilladot", C = 1,
                  kpar = list(), prob.model = TRUE,
                  kcross = 5, penalty = 1, cross = 2)

# Make predictions on the test set
x_test <- as.matrix(test[-9])
predicted_values <- predict(svm_model, x_test)
```

```
predicted_classes <- ifelse(predicted_values > 0.5, 1, 0)
actual_values <- test$fire
f1_score <- F1_Score(actual_values, predicted_classes)
accuracy <- Accuracy(y_true = actual_values, y_pred = predicted_classes)
print(f1_score)
print(accuracy)
```