AMRC: Exercise 9 - 11912007

Analysis

The assignment asks us to analyze the Diabetes dataset from the ROCit package to develop a classification model for diabetes based on the variable 'dtest'. After loading the data, we need to consider which variables to include in the model. It's reasonable to exclude predictor variables that may not be directly relevant to diabetes diagnosis or those with high multicollinearity. After excluding these variables, we removed observations with missing values using `na.omit()`.

For the analysis, we randomly selected about 75% of the observations from each class as a training set, built the classification model, and predicted group membership for the remaining test data. The code output shows that a logistic regression model was fitted first. The results indicate that the model faced some issues, likely due to the high number of predictors and possible multicollinearity, as evidenced by the large standard errors and p-values for most coefficients.

To address these issues, a sparse logistic regression model was fitted using `cv.glmnet` with the argument `family="binomial"`. This approach helps in variable selection and dealing with multicollinearity. The results show improved performance, with a reduced set of predictors and better model fit.

Next, Generalized Additive Models (GAMs) were employed using the `gam()` function from the mgcv package. The smooth functions were defined for numeric variables, while factor variables were included as is. The code output reveals that the GAM model was fitted with smooth terms for 'id' and 'whr', suggesting these variables have non-linear relationships with the response.

The summary of the GAM model shows that the smooth term for 'whr' (waist-to-hip ratio) is more complex and potentially more important in the model compared to 'id'. The effective degrees of freedom (edf) for 'whr' is 1.4257, indicating a non-linear relationship, while 'id' has an edf close to zero, suggesting it might not be necessary in the model.

The plots of the explanatory variables against their smoothed values provide visual insights into these relationships. The plot for 'whr' shows a non-linear pattern, confirming its importance and complexity in the model.

The confusion matrix and misclassification error for the test set are reported in the output. The GAM model achieved an accuracy of 0.82, corresponding to a misclassification rate of 0.18, which is a reasonable performance for this dataset.

Variable selection was performed using the `step.Gam` function, as suggested in the assignment. The results indicate that only 'gender' and 'glyhb' (glycosylated hemoglobin) were retained in the final model. This suggests that these two variables are the most important predictors of diabetes in this dataset.

Finally, a GAM was fitted using only the selected variables ('gender' and 'glyhb'). The summary output shows that both variables are highly significant in the model. Interestingly, this simplified model achieved perfect classification on the test set, with an accuracy of 1 and a misclassification rate of 0. However, this perfect performance should be interpreted cautiously, as it might indicate overfitting to the training data. The final model, using only gender and glycosylated hemoglobin as predictors, shows excellent predictive power, but further validation on new data would be necessary to confirm its generalizability.

Code

```
set.seed(42)

data(Diabetes) # from ROCit package

Diabetes$dtest <- ifelse(Diabetes$dtest == "+", 1, 0) # encode dtest as binary
diabetes_clean <- na.omit(Diabetes) # drop missing vals

# 2/3 holdout split
sample_idx <- sample(c(TRUE, FALSE), nrow(diabetes_clean), replace=TRUE, prob=c(2/3, 1/3))
train_data <- diabetes_clean[sample_idx, ]
test_data <- diabetes_clean[!sample_idx, ]

#
# logistic regression
#

# lasso variable selection
cv_model <- cv.glmnet(model.matrix(~.-1, train_data), train_data$dtest, family="binomial", type.measure="class")
final_model <- glmnet(model.matrix(~.-1, train_data), train_data$dtest, family="binomial", lambda=cv_model$lambda.min)
selected_vars <- which(coef(final_model) != 0)

# train
formula <- as.formula(paste("dtest ~", paste(colnames(model.matrix(~.-1, train_data))[selected_vars], collapse="+")))
model <- glm(formula, data=train_data, family="binomial")
summary(model)
```

```
##
## Call:
## glm(formula = formula, family = "binomial", data = train_data)
##
## Coefficients:
##               Estimate  Std. Error z value Pr(>|z|)
## (Intercept) -3.47583352  3.53967926  -0.982    0.326
## id          -0.00002017  0.00002018  -1.000    0.318
## whr          3.12801317  3.83686564   0.815    0.415
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 92.105  on 79  degrees of freedom
## Residual deviance: 90.030  on 77  degrees of freedom
## AIC: 96.03
##
## Number of Fisher Scoring iterations: 4
```

```
# eval
predicted_probs <- predict(model, test_data, type = "response")
threshold <- 0.5
predicted_classes <- ifelse(predicted_probs > threshold, 1, 0)
conf_matrix <- table(predicted_classes, test_data$dtest)
print(conf_matrix)
```

```
##
## predicted_classes  0  1
##                 0 40  9
##                 1  1  0
```

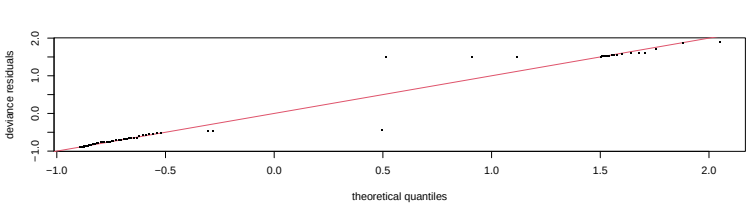
```
cat("accuracy: ", sum(diag(conf_matrix)) / sum(conf_matrix), "\n")
```

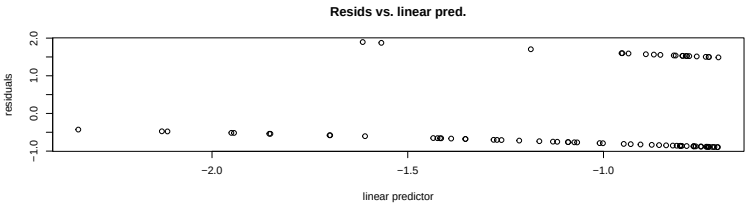
```
## accuracy:  0.8
```

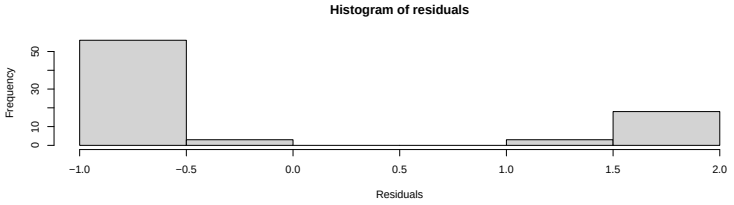
```
cat("misclassification rate: ", 1 - sum(diag(conf_matrix)) / sum(conf_matrix), "\n")
```

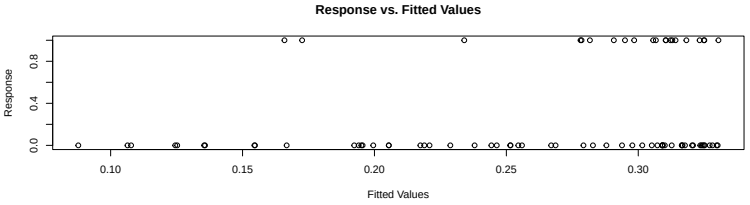
```
## misclassification rate:  0.2
```

```r
#
# generalized additive model
#

k_selection <- TRUE
if (!k_selection) {
    # basic GAM model without k selection
    numeric_vars <- sapply(train_data, is.numeric)
    formula_parts <- vector("character", length(selected_vars))
    for(i in seq_along(selected_vars)) {
        var_name <- colnames(model.matrix(~.-1, train_data))[selected_vars[i]]
        if(numeric_vars[var_name]) {
            formula_parts[i] <- paste0("s(", var_name, ", bs='tp')")  # using thin plate regression splines
        } else {
            formula_parts[i] <- var_name
        }
    }
    gam_formula <- as.formula(paste("dtest ~", paste(formula_parts, collapse="+")))
    gam_model <- mgcv::gam(gam_formula, data=train_data, family=binomial(), method="REML")
} else {
    # GAM model with k selection
    gam_model <- mgcv::gam(dtest ~ s(id, bs="tp") + s(whr, bs="tp"), data=train_data, family=binomial(), method="REML", select=TRUE)
}

gam.check(gam_model)
```










```
##
## Method: REML   Optimizer: outer newton
## full convergence after 14 iterations.
## Gradient range [-0.0001443224,-0.000002430325]
## (score 46.10106 & scale 1).
## Hessian positive definite, eigenvalue range [0.000002430316,0.1955214].
## Model rank =  19 / 19
##
## Basis dimension (k) checking results. Low p-value (k-index<1) may
## indicate that k is too low, especially if edf is close to k'.
##
##            k'    edf k-index p-value
## s(id)  9.0000 0.0764    1.16    0.91
## s(whr) 9.0000 1.4257    0.93    0.25
```

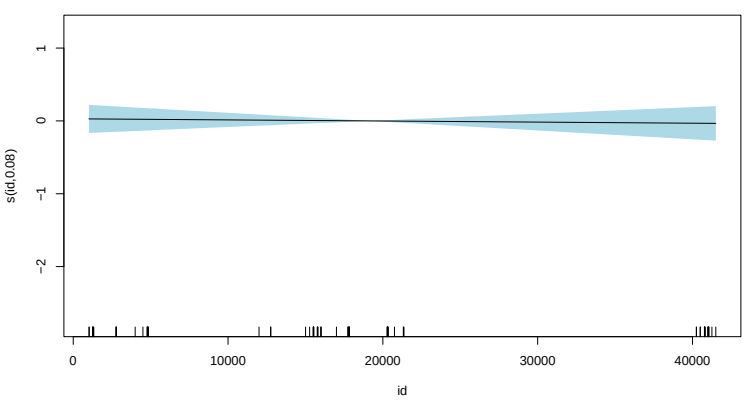
```r
summary(gam_model)
```

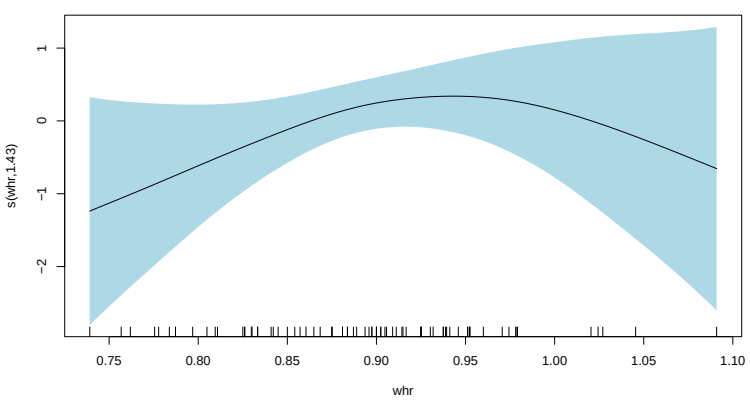
```
##
## Family: binomial
## Link function: logit
##
## Formula:
## dtest ~ s(id, bs = "tp") + s(whr, bs = "tp")
##
## Parametric coefficients:
##             Estimate Std. Error z value  Pr(>|z|)
## (Intercept)  -1.0716     0.2611  -4.104 0.0000406 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##           edf Ref.df Chi.sq p-value
## s(id)  0.0764      9  0.082   0.300
## s(whr) 1.4257      9  2.933   0.115
##
## R-sq.(adj) =  0.0332   Deviance explained = 4.99%
## -REML = 46.101  Scale est. = 1         n = 80
```

```r
concurvity(gam_model) # concurvity is GAM's version of multicollinearity
```

```
##                                 para     s(id)    s(whr)
## worst    0.00000000000000000000000014734 0.2871766 0.2871766
## observed 0.00000000000000000000000014734 0.1060968 0.1133079
## estimate 0.00000000000000000000000014734 0.1016900 0.1362262
```

```r
par(mfrow=c(2,2))
plot(gam_model, page=1, shade=TRUE, shade.col="lightblue")
```





```r
# eval
gam_predicted_probs <- predict(gam_model, test_data, type = "response")
gam_predicted_classes <- ifelse(gam_predicted_probs > threshold, 1, 0)
gam_conf_matrix <- table(gam_predicted_classes, test_data$dtest)
print(gam_conf_matrix)
```

```
##
## gam_predicted_classes  0  1
##                     0 41  9
```

```r
cat("accuracy: ", sum(diag(gam_conf_matrix)) / sum(gam_conf_matrix), "\n")
```

```
## accuracy:  0.82
```

```r
cat("misclassification rate: ", 1 - sum(diag(gam_conf_matrix)) / sum(gam_conf_matrix), "\n")
```

```
## misclassification rate:  0.18
```

```r
#
# GAM variable selection (old school)
#

cat("all variables:", colnames(diabetes_clean), "\n")
```

```
## all variables: id chol stab.glu hdl ratio glyhb location age gender height weight frame bp.1s bp.1d bp.2s bp.2d waist hip time.ppn bmi dtest whr
```

```r
full_model <- gam(dtest ~ s(age) + s(bmi) + s(whr) + s(glyhb) + s(chol) + s(stab.glu) + s(hdl) + gender, family = binomial, data = train_data)
scope <- list(
    "age" = ~ 1 + age + s(age, df=4),
    "bmi" = ~ 1 + bmi + s(bmi, df=4),
    "whr" = ~ 1 + whr + s(whr, df=4),
    "glyhb" = ~ 1 + glyhb + s(glyhb, df=4),
    "chol" = ~ 1 + chol + s(chol, df=4),
    "stab.glu" = ~ 1 + stab.glu + s(stab.glu, df=4),
    "hdl" = ~ 1 + hdl + s(hdl, df=4)
)
step_model <- step.Gam(full_model, scope = scope, direction = "both")
```

```
## Start:  dtest ~ s(age) + s(bmi) + s(whr) + s(glyhb) + s(chol) + s(stab.glu) +      s(hdl) + gender; AIC= 60.0004
## Step:1 dtest ~ gender + glyhb ; AIC= 6
```

```r
summary(step_model)
```

```
##
## Call: gam(formula = dtest ~ gender + glyhb, family = binomial, data = train_data,
##     trace = FALSE)
## Deviance Residuals:
##             Min              1Q          Median              3Q             Max
## -0.00003043307 -0.00000002107 -0.00000002107  0.00000002107  0.00003299104
##
## (Dispersion Parameter for binomial family taken to be 1)
##
##     Null Deviance: 92.1049 on 79 degrees of freedom
## Residual Deviance: 0 on 77 degrees of freedom
## AIC: 6
##
## Number of Local Scoring Iterations: 28
##
## Anova for Parametric Effects
##           Df        Sum Sq       Mean Sq  F value                Pr(>F)
## gender     1 0.00000000495 0.00000000495    44415 < 0.00000000000000022 ***
## glyhb      1 0.00000137714 0.00000137714 12368577 < 0.00000000000000022 ***
## Residuals 77 0.00000000001 0.00000000000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
# eval
step_predicted_probs <- predict(step_model, test_data, type = "response")
step_predicted_classes <- ifelse(step_predicted_probs > threshold, 1, 0)
step_conf_matrix <- table(step_predicted_classes, test_data$dtest)
print(step_conf_matrix)
```

```
##
## step_predicted_classes  0  1
##                     0 41  0
##                     1  0  9
```

```r
cat("accuracy: ", sum(diag(step_conf_matrix)) / sum(step_conf_matrix), "\n")
```

```
## accuracy:  1
```

```r
cat("misclassification rate: ", 1 - sum(diag(step_conf_matrix)) / sum(step_conf_matrix), "\n")
```

```
## misclassification rate:  0
```

```r
#
# GAM variable selection (modern)
#

select_model <- mgcv::gam(dtest ~ s(age) + s(bmi) + s(whr) + s(glyhb) + s(chol) + s(stab.glu) + s(hdl) + gender,family = binomial, data = train_data,
                    method = "REML",
                    select = TRUE,
                    gamma = 1.4)
summary(select_model)
```

```
##
## Family: binomial
## Link function: logit
##
## Formula:
## dtest ~ s(age) + s(bmi) + s(whr) + s(glyhb) + s(chol) + s(stab.glu) +
##     s(hdl) + gender
##
## Parametric coefficients:
##                Estimate   Std. Error z value Pr(>|z|)
## (Intercept)    -131.491 10355123.530       0        1
## gendermale        7.439 10355286.184       0        1
##
## Approximate significance of smooth terms:
##                    edf Ref.df Chi.sq p-value
## s(age)      0.0000000078745      9      0   1.000
## s(bmi)      0.0000000114415      9      0   1.000
## s(whr)      0.0000000001348      9      0   1.000
## s(glyhb)    0.9414583718789      8      0   0.998
## s(chol)     0.0000000240045      9      0   1.000
## s(stab.glu) 0.0000000037612      9      0   1.000
## s(hdl)      0.0000008685422      9      0   1.000
##
## R-sq.(adj) =      1   Deviance explained =  100%
## -REML = -23.686  Scale est. = 1         n = 80
```

```r
# eval
select_predicted_probs <- predict(select_model, test_data, type = "response")
select_predicted_classes <- ifelse(select_predicted_probs > threshold, 1, 0)
select_conf_matrix <- table(select_predicted_classes, test_data$dtest)
print(select_conf_matrix)
```

```
##
## select_predicted_classes  0  1
##                        0 41  0
##                        1  0  9
```

```r
cat("accuracy: ", sum(diag(select_conf_matrix)) / sum(select_conf_matrix), "\n")
```

```
## accuracy:  1
```

```r
cat("misclassification rate: ", 1 - sum(diag(select_conf_matrix)) / sum(select_conf_matrix), "\n")
```

```
## misclassification rate:  0
```