hw2

Apoorva Srinivasan

10/16/2019

Question 4

```
SAheart = read.table("http://www-stat.stanford.edu/~tibs/ElemStatLearn/datasets/SAheart.data",sep=",",h
```

```
train = SAheart[1:300 ,]
test = SAheart[301:462,]
```

```
log_model = glm(chd ~ ., family = binomial(link = "logit"), data = train)
summary(log_model)
```

```
##
## Call:
## glm(formula = chd ~ ., family = binomial(link = "logit"), data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8176  -0.8282  -0.4348   0.9484   2.4491
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)    -5.013991   1.570094  -3.193  0.00141 **
## sbp            -0.004784   0.007500  -0.638  0.52356
## tobacco         0.069165   0.032321   2.140  0.03236 *
## ldl             0.111462   0.074524   1.496  0.13474
## adiposity       0.046678   0.037295   1.252  0.21072
## famhistPresent  0.826525   0.280107   2.951  0.00317 **
## typea           0.046926   0.015911   2.949  0.00319 **
## obesity        -0.063287   0.053181  -1.190  0.23404
## alcohol         0.005111   0.005927   0.862  0.38849
## age             0.039145   0.014631   2.675  0.00746 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 394.29  on 299  degrees of freedom
## Residual deviance: 317.75  on 290  degrees of freedom
## AIC: 337.75
##
## Number of Fisher Scoring iterations: 4
```

```
fitted.results = predict(log_model, test, type = "response")
fitted.results <- ifelse(fitted.results > 0.5,1,0) #picking 0.5 as the boundry
log_error = mean((test$chd - fitted.results)^2)
log_error
```

```
## [1] 0.2530864
```

```
##same as log_error = mean(fitted.results != test$chd)

std_log_error = sd((test$chd - fitted.results)^2)/ sqrt(nrow(test))
std_log_error
```

```
## [1] 0.03426547
```

LDA

```
lda_model = lda(chd~., data = train)
summary(lda_model)
```

```
##         Length Class  Mode
## prior    2     -none- numeric
## counts   2     -none- numeric
## means   18     -none- numeric
## scaling  9     -none- numeric
## lev      2     -none- character
## svd      1     -none- numeric
## N        1     -none- numeric
## call     3     -none- call
## terms    3     terms  call
## xlevels  1     -none- list
```

```
lda_pred = predict(lda_model, test)$class
lda_error = mean((test$chd - lda_pred)^2)
```

```
## Warning in Ops.factor(test$chd, lda_pred): '-' not meaningful for factors
```

```
lda_error
```

```
## [1] NA
```

```
lda_pred = as.numeric(lda_pred)
test$chd = as.numeric(test$chd)
std_lda_error = sd((test$chd - lda_pred)^2)/ sqrt(nrow(test))
std_lda_error
```

```
## [1] 0.08431974
```

QDA

```
qda_model = qda(chd~., data = train)
qda_model$means
```

```
##        sbp  tobacco      ldl adiposity famhistPresent    typea  obesity
## 0 134.0105 2.629368 4.468421  23.29147      0.2947368 53.04211 25.57868
## 1 138.9091 5.433818 5.451545  27.90682      0.5545455 56.28182 26.76291
##    alcohol      age
## 0 12.41268 38.02632
## 1 18.75418 49.47273
```

```
qda_pred = predict(qda_model, test)$class
qda_pred = as.numeric(qda_pred)
```

```
qda_error = mean((qda_pred - test$chd)^2)
std_qda_error = sd((qda_pred - test$chd)^2)/sqrt(nrow(test))
```

```
log = cbind("Logistic Regression",log_error,std_log_error)
lda = cbind("LDA",lda_error, std_lda_error)
qda = cbind("QDA",qda_error, std_qda_error)

summary_table = rbind(log, lda, qda)
colnames(summary_table) = c("Model", "Test Error", "Std Error")
```

Given that the test error and the standard errors are similar for all three models, I'd pick logistic since it's the easiest to interpret.

Question 3

```
set.seed(20)
x = runif(50, min = 0, max = 1)
```

Generating 100 training sets

```
gen_train = list()
for (i in 1:100) {
set.seed(i)
y = sin(2*pi*x^3)^3 + rnorm(50, mean = 0, sd = 1)
gen_train[[i]] = cbind(x,y)
}
```

OLS with linear model

```
ols_linear_pred = list()
for (i in 1:100) {
fit = lm(y~x, data = as.data.frame(gen_train[[i]]))
pred = predict(fit)
ols_linear_pred[[i]] = pred
}

ols_linear_pred = bind_cols(ols_linear_pred)
```

OLS with cubic polynomial model

```
ols_cub_pred = list()
for (i in 1:100) {
fit = lm(y~poly(x,3), data = as.data.frame(gen_train[[i]]))
pred = predict(fit)
ols_cub_pred[[i]] = pred
}
ols_cub_pred = bind_cols(ols_cub_pred)
```

Cubic spline (or B-spline) with 2 knots at 0.33 and 0.66.

```
cub_spline_pred = list()
for (i in 1:100) {
  fit = lm(y ~bs(x, 0.33, 0.66), data = as.data.frame(gen_train[i]))
  pred = predict(fit)
  cub_spline_pred[[i]] = pred
}
cub_spline_pred = bind_cols(cub_spline_pred)
```

Fit natural cubic spline with 5 knots at 0.1, 0.3, 0.5, 0.7 and 0.9 in each training set and get the vector of fitted value

```
ncub_spline_pred = list()
for (i in 1:100) {
fit = lm(y~ns(x,knots=c(0.1,0.3,0.5,0.7,0.9)), data = as.data.frame(gen_train[[i]]))
pred = predict(fit)
ncub_spline_pred[[i]] = pred
}
ncub_spline_pred = bind_cols(ncub_spline_pred)
```

Fit smoothing spline with tuning parameter chosen by GCV in each training set and get the vector of fitted value

```
smooth_spline_pred = list()
for (i in 1:100) {
fit = smooth.spline(x=gen_train[[i]][,1], y =gen_train[[i]][,2], cv=FALSE)
pred = predict(fit)$y
smooth_spline_pred[[i]] = pred
}
smooth_spline_pred = bind_cols(smooth_spline_pred)
```

Calculating pointwise variance

```
ols_linear_var = apply(ols_linear_pred, 1, var)
ols_cub_var = apply(ols_cub_pred, 1, var)
cub_spline_var = apply(cub_spline_pred, 1, var)
ncub_spline_var = apply(ncub_spline_pred, 1,var)
smooth_spline_var = apply(smooth_spline_pred, 1,var)
var_df = data_frame(x,ols_linear_var,ols_cub_var,cub_spline_var,ncub_spline_var,smooth_spline_var)
```

```
## Warning: `data_frame()` is deprecated, use `tibble()`.
## This warning is displayed once per session.
```

##Plotting poitwise variance

```
ggplot(var_df) +
  geom_line(aes(x = x, y = ols_linear_var, color = "OLS Linear Spline")) +
  geom_point(aes(x = x, y = ols_linear_var, color = "OLS Linear Spline")) +
  geom_line(aes(x = x, y = ols_cub_var, color = "OLS Cubic Spline")) +
geom_point(aes(x = x, y = ols_cub_var, color = "OLS Cubic Spline")) +
geom_line(aes(x = x, y = cub_spline_var, color = "Cubic Spline")) +
```

```
geom_point(aes(x = x, y = cub_spline_var, color = "Cubic Spline")) +
geom_line(aes(x = x, y = ncub_spline_var, color = "Natural Cubic Spline")) +
geom_point(aes(x = x, y = ncub_spline_var, color = "Natural Cubic Spline")) +
geom_line(aes(x = x, y = smooth_spline_var, color = "Smoothing Spline")) +
geom_point(aes(x = x, y = smooth_spline_var, color = "Smoothing Spline")) +
theme_bw()+
labs(title = "Simulation results of pointwise variance between 5 models",
x = "X",
y = "Pointwise variance")
```