STATS 790
Assignment #3

Dean Hansen - 400027416

05 April, 2023

Contents

Question #1

The inspiration for the following workflow comes from Julia Silge's blog post found here, and the prostate cancer dataset comes from ESL (introduced on page 3).

The dataset predictors are described below (table borrowed from here). This is a regression problem as our output is continuous.

Variable	Description
lcavol	(log) Cancer Volume
lweight	(log) Weight
age	Patient age
lbph	(log) Vening Prostatic Hyperplasia
svi	Seminal Vesicle Invasion
lcp	(log) Capsular Penetration
gleason	Gleason score (how easily glands are identified in tissue)
pgg45	Percent of Gleason score 4 or 5
lpsa (response*)	(log) Prostate Specific Antigen

```
## get the data and clean it
url <- "https://hastie.su.domains/ElemStatLearn/datasets/prostate.data"
df_prostate <- read.table(file = url,
                          sep = "\t",
                          header = TRUE) %>%
  select(-X, -train) %>%
  as_tibble()

df_prostate_scaled <- df_prostate
df_prostate_scaled[1:8] <- scale(df_prostate_scaled[1:8], TRUE, TRUE)
```

Create some summary plots of the data.

```
## pairs plot using GGally
GGally::ggpairs(df_prostate_scaled,
        lower = list(continuous = wrap("points", alpha = 0.1, size = 0.1),
                                combo ="facethist",
                                discrete = "facetbar",
                                na =  "na"))
```

Plot age against the lpsa.

```
ggplot(df_prostate, aes(x = age, y = lpsa, color = svi)) +
  geom_point()
```

Below we setup the train and test data splits.

```
## the data comes with a train/test column
## we will use our own splits
## based on data info in ESL - scale predictors before fitting

data_split <- initial_split(df_prostate, prop = 0.7)
train_data <- training(data_split)
testing_data <- testing(data_split)
```

Setup our recipe for BART using dbarts engine.

```
bart_recipe  <- (
  recipe(lpsa ~ ., data = train_data)
  |> step_center(all_numeric())
  |> step_scale(all_numeric())
  |> prep()
)

bart_mod <- (
  bart(mode = "regression",
```

```
              trees = tune())
  |> set_engine(engine = "dbarts")
)
print(bart_mod)

bart_wflow <- (
  workflow()
  |> add_model(bart_mod)
  |> add_recipe(bart_recipe)
)
```

Tune the BART regression model recipe.

```
tt <- tune_grid(object = bart_wflow,
                grid = 2,
                resamples = vfold_cv(train_data),
                metrics   = metric_set(rmse),
                control = control_grid(save_pred = TRUE)
                )

cc <- collect_metrics(tt)
cp <- collect_predictions(tt)

show_best(tt)
ss <- select_best(tt)

bm <- finalize_model(bart_mod, select_best(tt))
bm_fit <- fit(bm, lpsa ~ ., data = train_data)
```

Create an explainer object for the BART model.

```
explainer_bart <-
  explain_tidymodels(
    bm_fit,
    data = train_data,
    y = train_data$lpsa)


set.seed(101)
shap_boost <- predict_parts(explainer = explainer_bart,
                            new_observation = train_data[1,],
                            type = "shap",
                            B = 1)
```

Plot the shap_boost object to get variable importance and vip_boost.

```
plot(shap_boost)
vip_bart <- model_parts(explainer_bart)
plot(vip_bart)
```

Question #2

a) First we derive the optimal weight for each tree, γ_{jm}, for the MSE (note: we assume there are n_{jm} elements in the R_{jm} terminal region). As shown below, the optimal weight is the average residual in each terminal node.

$$\frac{d}{d\gamma_{jm}} L(y_i, f_{m-1} + \gamma_{jm}) = \frac{d}{d\gamma_{jm}} [\frac{1}{2} \sum_{x_i \in R_{jm}} (y_i - (f_{m-1}(x_i) + \gamma_{jm}))^2] \tag{1}$$

$$0 = \sum_{x_i \in R_{jm}} (y_i - (f_{m-1}(x_i) + \gamma_{jm})) \tag{2}$$

$$0 = \sum_{x_i \in R_{jm}} (y_i - f_{m-1}(x_i) - \gamma_{jm}) \tag{3}$$

$$0 = \sum_{x_i \in R_{jm}} (y_i - f_{m-1}(x_i)) - n_{jm}\gamma_{jm} \tag{4}$$

$$n_{jm}\gamma_{jm} = \sum_{x_i \in R_{jm}} (y_i - f_{m-1}(x_i)) \tag{5}$$

$$\gamma_{jm} = \frac{\sum_{x_i \in R_{jm}} (y_i - f_{m-1}(x_i))}{n_{jm}} \tag{6}$$

As per piazza, instead of the deviance we can use the log-likelihood as a loss function. For binary classification, we only need to define the probability of success, which comes from fitting the tree model and applying the logit transformation. We define $z = f_{m-1}(x_i) + \gamma_{jm}$ to simplify notation and $p = \frac{1}{1+e^{-z}} = \frac{e^z}{1+e^z}$ as our probability of success. Below we derive the optimal weight under the log-likelihood loss function.

$$\frac{d}{d\gamma_{jm}} L(y_i, f_{m-1}(x_i) + \gamma_{jm}) = \frac{d}{d\gamma_{jm}} [- \sum_{x_i \in R_{jm}} y_i \log(p) + (1 - y_i)\log(1 - p)] \tag{7}$$

$$= \frac{d}{d\gamma_{jm}} [\sum_{x_i \in R_{jm}} -y_i \log(p) - (1 - y_i)\log(1 - p)] \tag{8}$$

$$= \frac{d}{d\gamma_{jm}} [\sum_{x_i \in R_{jm}} -y_i \log(p) + y_i \log(1 - p) - \log(1 - p)] \tag{9}$$

$$= \frac{d}{d\gamma_{jm}} [\sum_{x_i \in R_{jm}} -y_i (\log(p) - \log(1 - p)) - \log(1 - p)] \tag{10}$$

$$= \frac{d}{d\gamma_{jm}} [\sum_{x_i \in R_{jm}} -y_i \log(\frac{p}{1 - p}) - \log(1 - p)] \tag{11}$$

We have the following identities:

$$\log(1 - p) = \log(1 - \frac{e^z}{1 + e^z}) = \log(\frac{1}{1 + e^z}) = -\log(1 + e^{f_{m-1}(x_i) + \gamma_{jm}}) \tag{12}$$

$$\log(\frac{p}{1 - p}) = \log(p) - \log(1 - p) = \log(e^z) = f_{m-1}(x_i) + \gamma_{jm} \tag{13}$$

Plugging this back into the last line, we can take the derivative to find the optimal weight as follows:

$$0 = \frac{d}{d\gamma_{jm}} [\sum_{x_i \in R_{jm}} -y_i (f_{m-1}(x_i) + \gamma_{jm}) + \log(1 + e^{f_{m-1}(x_i) + \gamma_{jm}})] \tag{14}$$

$$0 = \sum_{x_i \in R_{jm}} (-y_i + \frac{e^{f_{m-1}(x_i) + \gamma_{jm}}}{1 + e^{f_{m-1}(x_i) + \gamma_{jm}}}) \tag{15}$$

$$0 = \sum_{x_i \in R_{jm}} (-y_i + \frac{1}{1 + e^{-(f_{m-1}(x_i) + \gamma_{jm})}}) \tag{16}$$

$$0 = \sum_{x_i \in R_{jm}} (-y_i + \frac{1}{1 + e^{-(f_{m-1}(x_i) + \gamma_{jm})}}) \tag{17}$$

$$\tag{18}$$

$\sum_{x_i \in R_{jm}} (-y_i + \frac{1}{1 + e^{-(f_{m-1}(x_i) + \gamma_{jm})}})$

b) For the MSE, since the 1st derivative is a constant, the 2nd derivative will be zero, thus the

optimal weights will again be $\gamma_{jm} = \frac{\sum_{x_i \in R_{jm}} (y_i - f_{m-1}(x_i))}{n_{jm}}$.

For the binomial log-likelihood, we first take the 2nd derivative below.