Part 4 - Interpretation

Ibrahim Yazici

Load packages

This example uses the `tidyverse` suite of packages.

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.2     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.3     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

Read data

The code chunk below reads in the final project data.

```
df <- readr::read_csv("paint_project_train_data.csv", col_names = TRUE)
```

```
## Rows: 835 Columns: 8
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (2): Lightness, Saturation
## dbl (6): R, G, B, Hue, response, outcome
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

The `readr::read_csv()` function displays the data types and column names associated with the data. However, a glimpse is shown below that reveals the number of rows and also shows some of the representative values for the columns.

```
df %>% glimpse()
```

The data consist of continuous and categorical inputs. The `glimpse()` shown above reveals the data type for each variable which state to you whether the input is continuous or categorical. The RGB color model inputs, `R`, `G`, and `B` are continuous (dbl) inputs. The HSL color model inputs consist of 2 categorical inputs, `Lightness` and `Saturation`, and a continuous input, `Hue`. Two outputs are provided. The continuous output, `response`, and the Binary output, `outcome`. However, the data type of the Binary outcome is numeric because the Binary `outcome` is **encoded** as `outcome = 1` for the EVENT and `outcome = 0` for the NON-EVENT.

The code chunk below assembles the data for interpretation of the best regression model. The logit-transformed output is named `y`. The `dfii` dataframe as the original `response` and Binary output, `outcome`, removed. This way we can focus on the variables specific to the regression task.

```
dfii <- df %>%
  mutate(y = boot::logit( (response - 0) / (100 - 0) ) ) %>%
  select(R, G, B,
         Lightness, Saturation, Hue,
         y)
```

The code chunk below assembles the data for interpretation of the best classification model.

```
dfiiiD <- df %>%
  select(-response) %>%
  mutate(outcome = ifelse(outcome == 1, 'event', 'non_event'),
         outcome = factor(outcome, levels = c('event', 'non_event')))
```

By converting `outcome` to a factor, the unique values of the variables are "always known":

```
dfiiiD %>% pull(outcome) %>% levels()
```

```
## [1] "event"     "non_event"
```

However, the value counts are the same as the original encoding.

```
dfiiiD %>% count(outcome)
```

```
## # A tibble: 2 x 2
##   outcome       n
##   <fct>     <int>
## 1 event       191
## 2 non_event   644
```

Interpretation of the Results

Input Importance

i) Regression In `Part_2_Regression` file, by "RMSE" and "Rsquared" metric values, we identified the best model as: "Add Categorical Inputs to Interactions from 3 DOF spline from input R and All Pairwise Interactions of Continuous Inputs G, B, Hue (This is Model 9 in Part iiA)".

Lets call this model as `best_reg_model` and load it below:

```
library(caret)
```

```
## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

```
best_reg_model <- readr::read_rds("best_regression_model.rds")
```

We can see the most important variables associated with our best performing regression model, `best_reg_model`, below.

```
var_imp_reg_best <- varImp(best_reg_model, scale = TRUE)
print(var_imp_reg_best)
```

We can see the full list of 39 variables below, to identify the least important ones as well.

```
top_n_reg_best <- head(var_imp_reg_best, n = 39)
print(top_n_reg_best)
```

We can plot the importance's below.

```
plot(var_imp_reg_best)
```

ii) Classification In `Part_3_Classification` file, by "Accuracy" metric values, we identified the best model as: Gradient boosted tree.

Lets call this model as `best_class_model` and load it below:

```
best_class_model <- readr::read_rds("best_classification_model.rds")
```

We can see the most important variables associated with our best performing classification model, `best_class_model`, below.

```
var_imp_class_best <- varImp(best_class_model, scale = TRUE)
print(var_imp_class_best)
plot(var_imp_class_best)
```

Below we make a surface plot for the hardest to predict Lightness and Saturation combinations in regression:

```
primary_seq <- seq(min(dfii$R), max(dfii$R), length.out = 101)
secondary_seq <- seq(min(dfii$B), max(dfii$B), length.out = 101)

prediction_data <- expand.grid(
  R = primary_seq,
  B = secondary_seq,
  Hue = mean(dfii$Hue),
  G = mean(dfii$G),
  Lightness = "saturated",
  Saturation = "muted"
)

prediction_data$predictions <- predict(best_reg_model, newdata = prediction_data)

ggplot(prediction_data, aes(x = R, y = B, fill = predictions)) +
  geom_raster(interpolate = TRUE) +
  scale_fill_gradientn(colors = terrain.colors(10)) +
  theme_minimal() +
  labs(x = "R", y = "B", fill = "Predicted Value")
```

Below we make a surface plot for the easiest to predict Lightness and Saturation combinations in regression:

```
primary_seq <- seq(min(dfii$R), max(dfii$R), length.out = 101)
secondary_seq <- seq(min(dfii$B), max(dfii$B), length.out = 101)

prediction_data <- expand.grid(
  R = primary_seq,
  B = secondary_seq,
  Hue = mean(dfii$Hue),
  G = mean(dfii$G),
  Lightness = "deep",
  Saturation = "neutral"
)
```

```r
prediction_data$predictions <- predict(best_reg_model, newdata = prediction_data)

ggplot(prediction_data, aes(x = R, y = B, fill = predictions)) +
  geom_raster(interpolate = TRUE) +
  scale_fill_gradientn(colors = terrain.colors(10)) +
  theme_minimal() +
  labs(x = "R", y = "B", fill = "Predicted Value")
```

Below we make a surface plot for the hardest to predict Lightness and Saturation combinations in classification:

```r
primary_seq <- seq(min(dfiiiD$Hue), max(dfiiiD$Hue), length.out = 101)
secondary_seq <- seq(min(dfiiiD$G), max(dfiiiD$G), length.out = 101)

prediction_data <- expand.grid(
  Hue = primary_seq,
  G = secondary_seq,
  B = mean(dfiiiD$B),
  R = mean(dfiiiD$R),
  Lightness = "saturated",
  Saturation = "pure"
)

pred_probs <- predict(best_class_model, newdata = prediction_data, type = "prob")

prediction_data_df <- prediction_data %>% bind_cols(pred_probs)

ggplot(prediction_data_df, aes(x = Hue, y = G, fill = event)) +
  geom_raster(interpolate = TRUE) +
  scale_fill_gradient2(low = 'blue', mid = 'white', high = 'red', midpoint = 0.5, limits = c(0, 1)) +
  labs(fill = "Event Probability") +
  theme_minimal() +
  theme(legend.position = "bottom")
```

Below we make a surface plot for the easiest to predict Lightness and Saturation combinations in classification:

```r
primary_seq <- seq(min(dfiiiD$Hue), max(dfiiiD$Hue), length.out = 101)
secondary_seq <- seq(min(dfiiiD$G), max(dfiiiD$G), length.out = 101)

prediction_data <- expand.grid(
  Hue = primary_seq,
  G = secondary_seq,
  B = mean(dfiiiD$B),
  R = mean(dfiiiD$R),
  Lightness = "pale",
  Saturation = "gray"
)

pred_probs <- predict(best_class_model, newdata = prediction_data, type = "prob")

prediction_data_df <- prediction_data %>% bind_cols(pred_probs)

ggplot(prediction_data_df, aes(x = Hue, y = G, fill = event)) +
  geom_raster(interpolate = TRUE) +
  scale_fill_gradient2(low = 'blue', mid = 'white', high = 'red', midpoint = 0.5, limits = c(0, 1)) +
  labs(fill = "Event Probability") +
```

```r
  theme_minimal() +
  theme(legend.position = "bottom")
```