Student Name: Sophie Stelmach
Student ID: 400172067

STATS 790
Homework 4

1. Note: I followed this tutorial for creating a model with the lightgbm engine.

Data

The data used was obtained from Kaggle and it contains data on milk quality. The variables include pH, temperature in Celcius, colour (all numeric), taste, odor, fat, turbidity (all binary, 1 if satisfactory, 0 else), and grade (factor with levels ”low”, ”medium” and ”high”). There are 7 predictors in total with grade being the outcome variable. The data contains 1,059 observations and no missing values.

```
milk <- read.csv("milknew.csv") |>
         mutate(Fat = factor(Fat),
                Taste = factor(Taste),
                Turbidity = factor(Turbidity),
                Odor = factor(Odor),
                Grade = factor(Grade,
                               levels= c("low", "medium","high"),
                               ordered = TRUE))|>
          mutate_if(is.character, as.factor)

> summary(milk$Grade)
   low medium   high
   429    374    256
```

Exploratory Data Analysis

The data was investigated using several plots. As we can see, the different grades are difficult to distinguish based solely on their pH, temperature or colour. There is a lot of spread in the low grade and a lot of overlap in the medium and high grades with these variables. There is a relatively even number of observations in each class, but more in the low class than in the others.

From the pairs plot, we can see that more medium grade observations had unsatisfactory turbidity, but both low and high grades had more satisfactory turbidity. Low and medium grades had mostly bad odor, but high grade had good odor. There is not any significant correlation between any of the variables.

```
library(tidyr)
library(dplyr)
library(ggplot2)

ggplot(milk, aes(x = Grade, fill= Grade))+
  geom_bar(alpha =0.5, col = "white")+
  scale_fill_manual(values=c("lightblue", "dodgerblue", "blue4"))+
  theme_bw()

ggplot(milk, aes(x = pH, fill = Grade))+
  geom_histogram(bins = 8, col = "white", alpha = 0.5)+
  xlim(3, 10)+
  scale_fill_manual(values=c("lightblue", "dodgerblue", "blue4"))+
  theme_bw()
```

```
ggplot(milk, aes(x = Temprature, fill = Grade))+
  geom_histogram(bins = 13, col = "white", alpha = 0.5)+
  scale_fill_manual(values=c("lightblue", "dodgerblue", "blue4"))+
  theme_bw()

ggplot(milk, aes(x = Colour, fill = Grade))+
  geom_histogram(bins = 9, col = "white", alpha = 0.5)+
  scale_fill_manual(values=c("lightblue", "dodgerblue", "blue4"))+
  theme_bw()
```

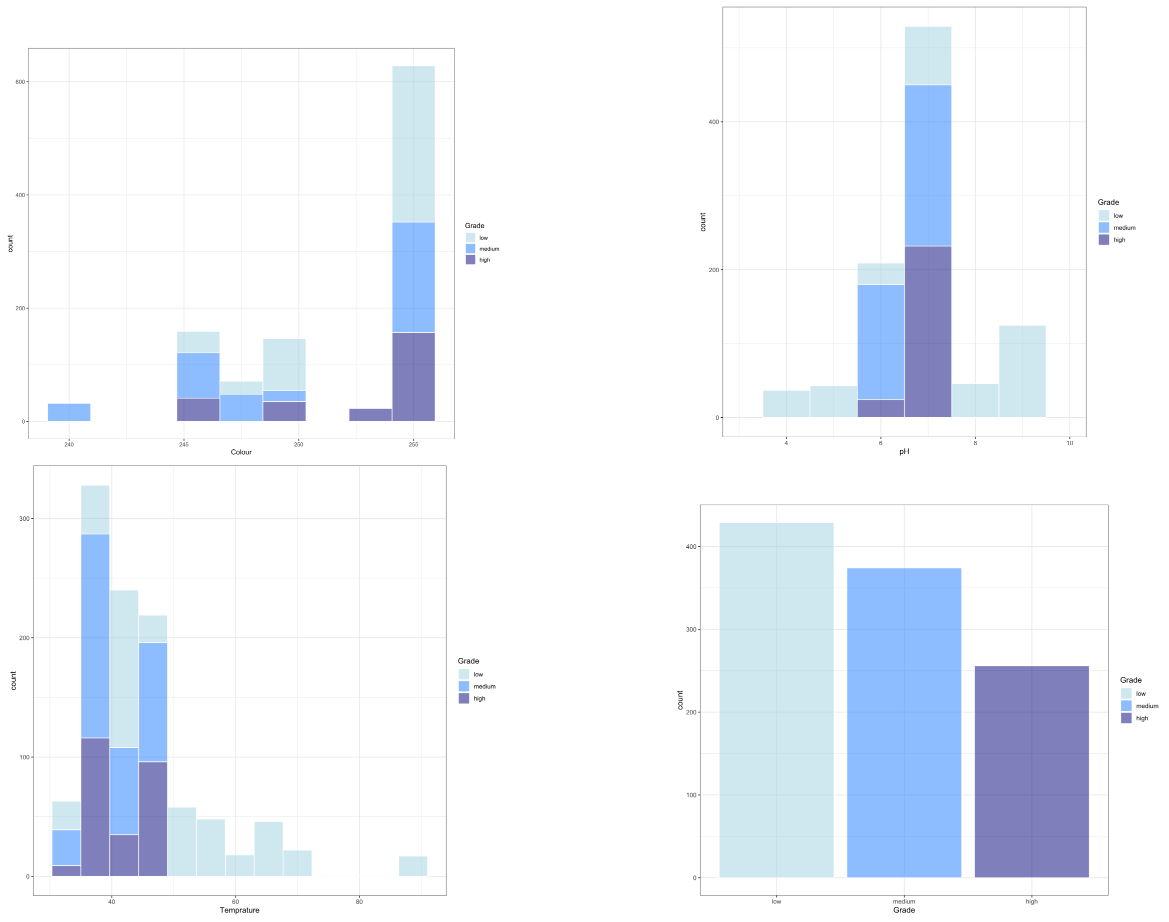

Figure 1: Histograms for colour, pH, temperature, and grade.

```
ggpairs(milk, mapping = aes(col= Grade),
        lower=list(combo=wrap("facethist", binwidth=1.5)))
```

Pre-processing

The data was divided into a 75% split of training and test sets. The split was stratified to ensure that an even number of observations was coming from each grade of milk, since the data was not entirely even. After this, the data was scaled and centered, zero variance variables were removed, and it was ensured that the binary and outcome variables were factors. The lightgbm engine is known to not perform well with dummy variables encoded, so I left that step out in the recipe for that reason (this was also mentioned in the tutorial I linked earlier).

```
set.seed(5)
```

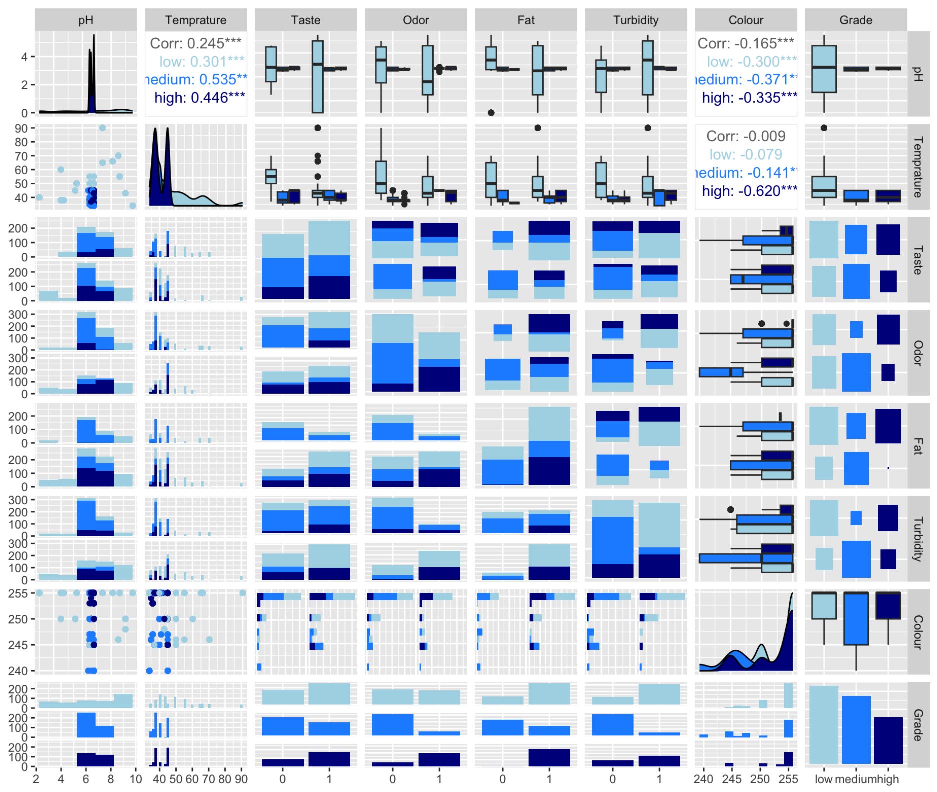

Figure 2: Pairs plot. Light blue = low grade, medium blue = medium grade, dark blue = high grade.

```
# split data
library(sampler)
index <- initial_split(milk, prop= 3/4, strata = Grade)
train <- training(index)
#dim(train)
test <- testing(index)
#dim(test)

library(tidymodels)
library(lightgbm)
library(bonsai)
library(parsnip)
library(yardstick)

milk_recipe <- (
  recipe(Grade ~ ., data = train) |>
  step_normalize(all_numeric()) |>
  step_zv(all_numeric()) |>
#  step_dummy(all_nominal(), -all_outcomes()) |>
  prep()
)
```

Modelling

Next, the model is created. I selected a classification model since the outcome variable is categorical and used the lightgbm engine mostly out of curiosity. I had seen examples of xgboost (including the one done in class) and I'm familiar with random forests, so I wanted to try out something new. What I learned in this process is that this engine is extremely popular in Python, but hard to come across in R. I'm not entirely sure what the reason for this is. Lightgbm uses a leaf-wise algorithm to grow trees, meaning it picks the best leaves and goes with that (reducing loss more than other algorithms). It is also supposedly much faster than other engines.

The model included three hyperparameters to tune: the number of trees, tree depth, and minimum data in each leaf. These all contribute to preventing overfitting, which is common with this engine (source: https://lightgbm.readthedocs.io/en/latest/Parameters-Tuning.html). These hyperparameters were tuned using 10-fold crossvalidation.

```
milk_mod <-
  boost_tree(
    mode = "classification",
    trees = tune(),
    min_n = tune(),
    tree_depth = tune()
  ) |>
  set_engine(engine = "lightgbm")

boost_wflow <-
  workflow() |>
  add_model(milk_mod) |>
  #add_recipe(milk_recipe)
  add_formula(Grade ~ .)
```

```
folds <- bake(milk_recipe, new_data = train) |>
            vfold_cv(v = 10)

grid <- grid_regular(trees(),
                     min_n(),
                     tree_depth(),
                     levels = 10)

fit_resamp <- tune_grid(object = boost_wflow,
                        resamples = folds,
                        grid = grid,
                        control = control_grid(verbose = T))

best_tuned <- fit_resamp |>
  select_best("accuracy")

final_mod <- milk_mod |>
  finalize_model(best_tuned)

> final_mod
Boosted Tree Model Specification (classification)

Main Arguments:
  trees = 445
  min_n = 2
  tree_depth = 5

Computational engine: lightgbm
```

The best model was selected based on accuracy, since one of the main goals of classification is to accurately predict classes. The resulting hyperparameter values were: number of trees = 445, minimum data in each leaf = 2, and tree depth = 5. The model with these parameters was fit to both the training and test data to evaluate performance based on accuracy score (shown after variable importance).

The variable importance was evaluated for the fit based on the training data, and what we see is that pH is the most important variable. In future studies and data collection, it would be important to collect pH of milk samples to determine grade. Temperature is also important, though it's unclear how this was measured, as temperature seems like it could easily be changed depending on the environment, and it's not obvious why different samples would have different temperatures if all other variables are controlled. There is a large jump in importance between temperature and the rest of the variables, so if there is future study regarding milk quality, it might be worth looking into other variables to measure.

```r
fit_lgbm <- fit(final_mod, Grade~ ., data = train)

library(vip)
library(DALEX)
library(DALEXtra)

explainer <-
  explain_tidymodels(
    fit_lgbm,
    data = train,
    y = train$Grade
  )

vip_boost <- model_parts(explainer)
plot(vip_boost)
```

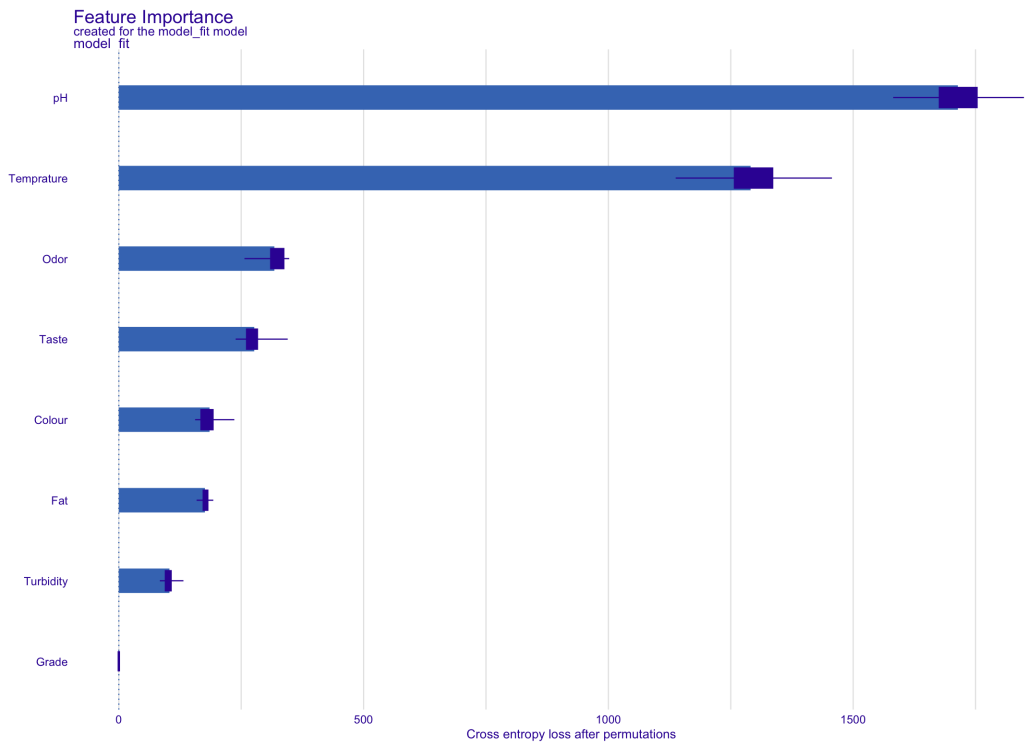

Figure 3: Variable importance plot.

The partial dependence plot for pH doesn't show a clear pattern for milk grade as pH increases.

Thus, even though pH is an important variable, it does not seem reliable in classifying grade of milk in this dataset.

```
library(fastshap)
X <- model.matrix(Grade ~ . - 1, train)
pdp_age <- model_profile(explainer, N = 500,
                         variables = "pH")

plot(pdp_age, geom = "profiles")
```

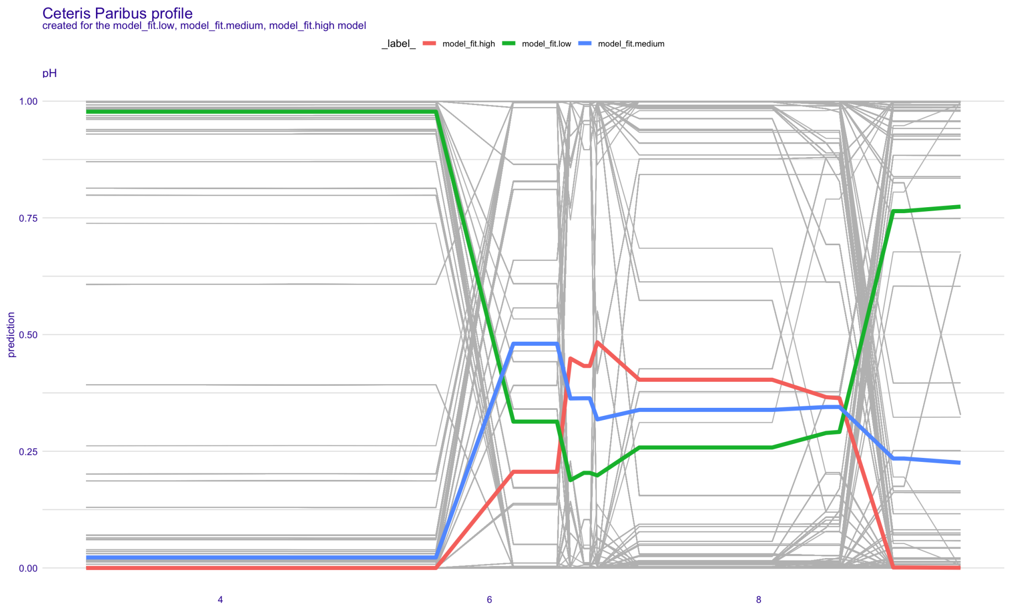

Figure 4: Partial dependence plot for pH.

```
test_proc <- bake(milk_recipe, new_data = test)
test_pred <- fit_lgbm |>
  predict(new_data = test_proc) |>
  cbind(test)

train_proc <- bake(milk_recipe, new_data = train)
train_pred <- fit_lgbm |>
  predict(new_data = train_proc) |>
  cbind(train)
```

Finally, accuracy was calculated for the test and training set model fits.

```
 > train_pred |> metrics(Grade, .pred_class)
# A tibble: 2   3
  .metric  .estimator .estimate
  <chr>    <chr>          <dbl>
1 accuracy multiclass    0.427
2 kap      multiclass    0.0430
> test_pred |> metrics(Grade, .pred_class)
# A tibble: 2   3
  .metric  .estimator .estimate
  <chr>    <chr>          <dbl>
1 accuracy multiclass    0.455
```

```
2 kap      multiclass    0.0874
```

Clearly this model does not perform well, it does not correctly classify even half of the data, even in the training set. It's almost funny how low the accuracy is. The data itself may be partly to blame, since in the exploratory analysis we saw the grades did not have distinct features in terms of the predictor variables, especially the low grade. Low grade data had predictors like pH, temperature, and colour spread out across their range of values, so the model may have had difficulty identifying low grade milk from these features. Similarly, high and medium grade milk had comparable characteristics; they both lingered around a pH level of 6-8 and on the lower end of the temperature range, so they might not have been easily distinguishable. The lightgbm engine might be particular about how categorical variables are specified, though I wasn't able to find many resources for how to most correctly code this in R (most results were for Python). It's also common for lightgbm to over fit data, since it grows leaf-wise, so this seems likely to have occurred.

2. Question 2

   a) **Deriving γ_{jm} for MSE:**

   $L_2 = (y_i - \hat{y})^2$

$$
\begin{aligned}
\gamma_{jm} &= argmin_\gamma \sum L(y_i, f_{m-1}(x_i) + \gamma) \\
&\Rightarrow \frac{\partial}{\partial \gamma} \sum_{i=1}^{N} L(y_i, f_{m-1}(x_i) + \gamma) \\
&= \frac{\partial}{\partial \gamma} \sum_{i=1}^{N} (y_i - f_{m-1}(x_i) + \gamma)^2 \\
&= \frac{\partial}{\partial \gamma} \sum_{i=1}^{N} (y_i^2 - 2y_i f_{m-1}(x_i) + 2y_i\gamma + (f_{m-1}(x_i))^2 - 2f_{m-1}(x_i)\gamma + \gamma^2) \\
&= \sum_{i=1}^{N} (2y_i - 2f_{m-1}(x_i) + 2\gamma) \\
0 &= 2\sum_{i=1}^{N} (y_i - f_{m-1}(x_i) + \gamma) \\
N\gamma &= -2\sum_{i=1}^{N} (y_i - f_{m-1}(x_i)) \\
\gamma &= -\frac{1}{N}\sum_{i=1}^{N} (y_i - f_{m-1}(x_i)) \\
\gamma &= \frac{1}{N}\sum_{i=1}^{N} (f_{m-1}(x_i) - y_i) \\
\Rightarrow \gamma_{jm} &= \frac{1}{N}\sum_{i=1}^{N} (f_{m-1}(x_i) - y_i)
\end{aligned}
$$

   Deriving γ_{jm} for binomial deviance:

   $l(Y, f(x)) = -log(1 + exp(-2Yf(x)) \Rightarrow L(Y, f(x)) = -1 - exp(-2Yf(x))$

$$\begin{aligned}
\gamma_{jm} &= argmin_\gamma \sum L(y_i, f_{m-1}(x_i) + \gamma) \\
&\Rightarrow \frac{\partial}{\partial\gamma} \sum_{i=1}^{N} L(y_i, f_{m-1}(x_i) + \gamma) \\
&= \frac{\partial}{\partial\gamma} \sum_{i=1}^{N} (-1 - exp(-2y_i(f_{m-1}(x_i) + \gamma)) \\
0 &= \sum_{i=1}^{N} 2y_i exp(-2y_i(f_{m-1}(x_i) + \gamma) \\
0 &= 2N \sum_{i=1}^{N} y_i exp(-2y_i(f_{m-1}(x_i) + \gamma) \\
0 &= \sum_{i=1}^{N} y_i exp(-2y_i f_{m-1}(x_i) - 2y_i\gamma) \\
0 &= \sum_{i=1}^{N} y_i exp(-2y_i f_{m-1}(x_i)) exp(-2y_i\gamma) \\
\log 0 &= \log \sum_{i=1}^{N} exp(-2y_i\gamma) \\
1 &= \sum_{i=1}^{N} -2y_i\gamma \\
1 &= -2N\gamma \sum_{i=1}^{N} y_i \\
1/2N\gamma &= \sum_{i=1}^{N} -y_i \\
\Rightarrow \gamma_{jm} &= \frac{1}{\sum_{i=1}^{N} -2y_i}
\end{aligned}$$

I know this is wrong because you can't take $ln(0)$, I just have tried it a million ways and can't get it :/

b) Newton boosting

We use the loss function: $L* = \sum l + 2g\gamma + 1/2h\gamma^2$, where $g = \frac{\partial l}{\partial y_i}$ and $h = \frac{\partial^2 l}{\partial y_i^2}$

Deriving γ_{jm} for MSE:

I'm going to shorten $f_{m-1}(x_i)$ to f for the time being.

$L_2 = (y_i - \hat{y})^2 \Rightarrow g = \frac{\partial l}{\partial y_i} = 2(y_i - f - \gamma), h = \frac{\partial^2 l}{\partial y_i^2} = 2$

$$\begin{aligned} L* &= \sum (y_i - f - \gamma)^2 + 2(y_i - f - \gamma)\gamma + 1/2(2)\gamma^2 \\ L* &= \sum y_i^2 - 2fy_i - 2f\gamma - 2y_i\gamma + f^2 + \gamma^2 + 2y_i\gamma - 2f\gamma - 2\gamma^2 + \gamma^2 \\ L* &= \sum y_i^2 + f^2 - 2y_if - 4f\gamma \end{aligned}$$

$$\begin{aligned} \gamma_{jm} &= argmin_\gamma \sum y_i^2 + f^2 - 2y_if - 4f\gamma \\ &\Rightarrow \frac{\partial}{\partial\gamma} \sum_{i=1}^N y_i^2 + f^2 - 2y_if - 4f\gamma \\ 0 &= \sum_{i=1}^N -4f \end{aligned}$$

We end up with just 0.

Deriving γ_{jm} for binomial deviance:

$-l = Y' \log p + (1 - Y') \log(1 - p) \Rightarrow l = -Y' \log p - (1 - Y') \log(1 - p)$

$= -(\frac{y_i+1}{2}) \log p - (1 - \frac{y_i+1}{2}) \log(1 - p)$.

$\Rightarrow g = \frac{\partial l}{\partial y_i} = -\frac{y_i+1}{2p} + \frac{y_i-1}{2p-2} = \frac{2y-2}{2p(2p-2)} = \frac{y-1}{p(2p-2)} = \frac{y-1}{2p(p-1)}$.

$\Rightarrow h = \frac{\partial^2 l}{\partial y_i^2} = -1/2\frac{(y_i-1)(2p-1)}{(p-1)^2p^2}$.

$$\begin{aligned} L* &= \sum -(\frac{y_i+1}{2}) \log p - (1 - \frac{y_i+1}{2}) \log(1-p) + 2(\frac{y_i-1}{2p(p-1)})\gamma + 1/2(-1/2\frac{(y_i-1)(2p-1)}{(p-1)^2p^2})\gamma^2 \\ &\Rightarrow \frac{\partial}{\partial\gamma} \sum_{i=1}^N -(\frac{y_i+1}{2}) \log p - (1 - \frac{y_i+1}{2}) \log(1-p) + 2(\frac{y_i-1}{2p(p-1)})\gamma + 1/2(-1/2\frac{(y_i-1)(2p-1)}{(p-1)^2p^2})\gamma^2 \\ &= \sum_{i=1}^N (\frac{y_i-1}{p(p-1)}) + (-1/2\frac{(y_i-1)(2p-1)}{(p-1)^2p^2})\gamma \\ 0 &= \sum_{i=1}^N (\frac{y_i-1}{p(p-1)}) + (-1/2\frac{(y_i-1)(2p-1)}{(p-1)^2p^2})\gamma \end{aligned}$$

$$\begin{aligned}
\sum_{i=1}^{N} 1/2\frac{(y_i-1)(2p-1)}{(p-1)^2p^2})\gamma &= \sum_{i=1}^{N}\frac{y_i-1}{p(p-1)} \\
\gamma &= \sum_{i=1}^{N} 2\frac{(y_i-1)(p-1)^2p^2}{p(p-1)(y_i-1)(2p-1)} \\
\gamma &= \sum_{i=1}^{N} 2\frac{(p-1)p}{(2p-1)} \\
\gamma &= \sum_{i=1}^{N} p \\
\gamma_{jm} &= \sum_{i=1}^{N}\frac{1}{1+exp(-2f(x))}
\end{aligned}$$