ST310 Course Project

Chris Chia, Mun Fai Chan, Zhen Yen Chan

21/04/21

ST310 Course Project

Exploratory Data Analysis

We have obtained data from the Kaggle March Tabular Playground competition. The data consists of *anonymised features*, which correspond to an outcome variable. Although the data is anonymised, we are told that it relates to a problem in insurance.

```
library(tidyverse)
library(ggplot2)
library(tidymodels)
library(xgboost)
library(catboost)
```

```
df <- read.csv(file = "../data/train.csv")
cat(c("The dimensions of the array are :", dim(df)[1], ", ", dim(df)[2]))
```

```
## The dimensions of the array are : 300000 ,  32
```

The dataset consists of 30 features, 19 *categorical variables* and 11 numerical variables. We will need to process these categorical variables in some manner, which we shall consider in the subequent section.

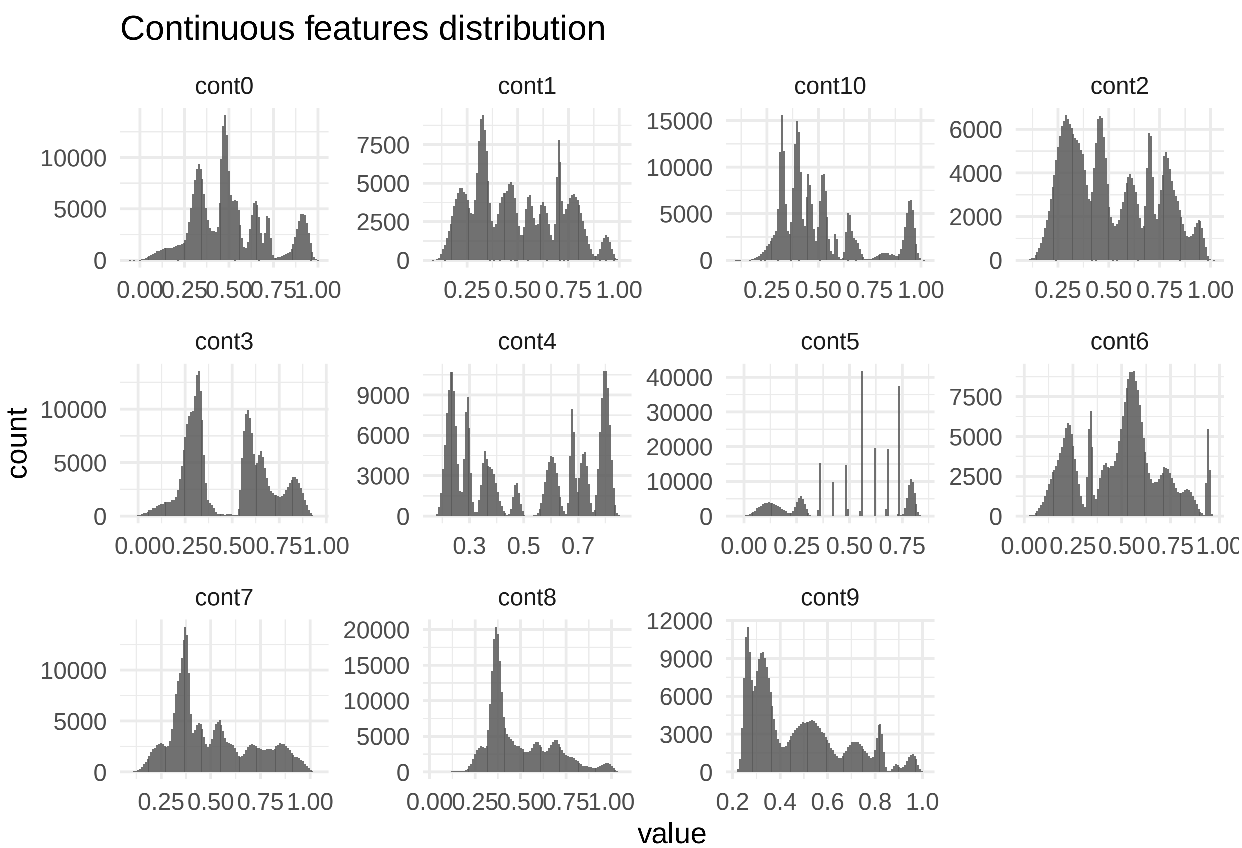
With 30 variables, and no *specific* knowledge about which features may be potentially useful.

Univariate

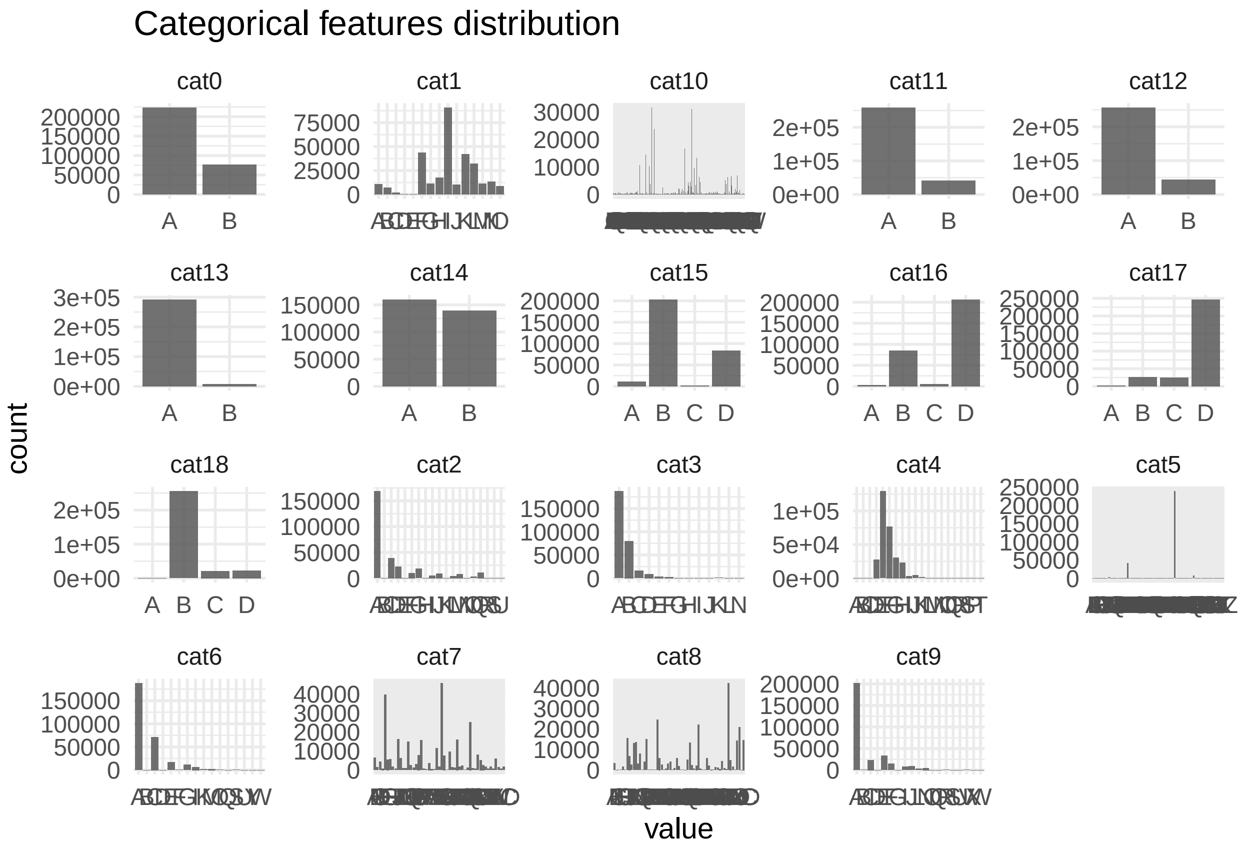
We first partition the data into our training and testing proportions

```
rownames(df) = df$id
set.seed(1)
sam = sample(1:nrow(df), 4000)
df_sample = df[sam,-1]

cat_features = 1:19

set.seed(1)
df_split <- initial_split(df_sample, prop = 3/4)
df_train <- training(df_split)
df_test <- testing(df_split)
```

```
df %>% pivot_longer(cols = starts_with("cont"), names_to  = "cont") %>%
   ggplot(aes(x = value))+
   geom_histogram(bins = 100, alpha = 0.85)+
   ggtitle("Continuous features distribution")+
   facet_wrap(cont~.,scales = "free")+
   theme_minimal()
```



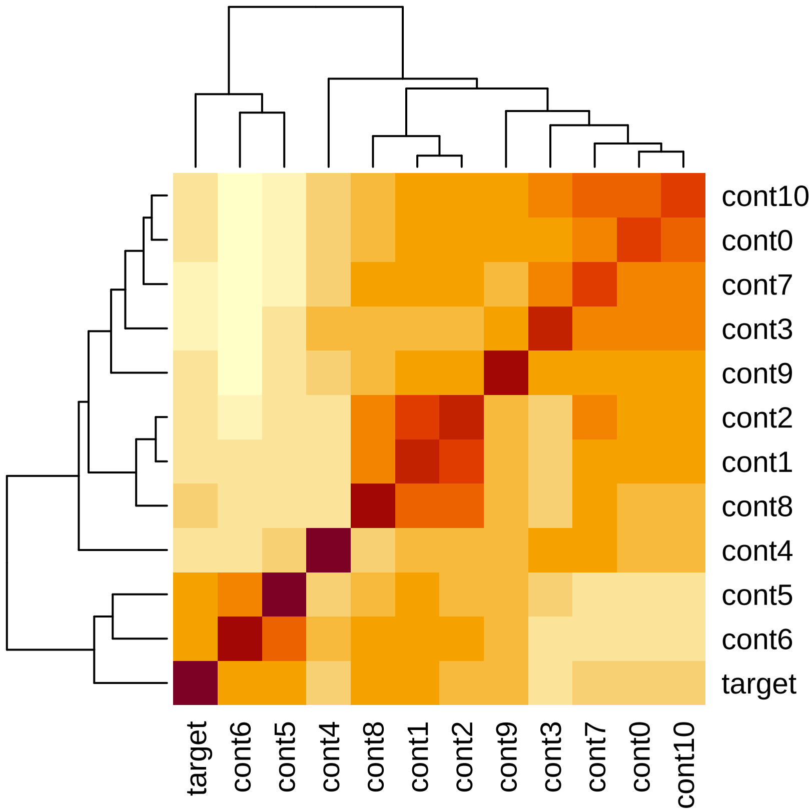

```r
df %>% pivot_longer(cols = contains(c("cat")), names_to  = "cat") %>%
   ggplot(aes(x = value))+
   geom_bar(alpha = 0.85)+
   ggtitle("Categorical features distribution")+
   facet_wrap(cat~.,scales = "free")+
   theme_minimal()
```



Bivariate

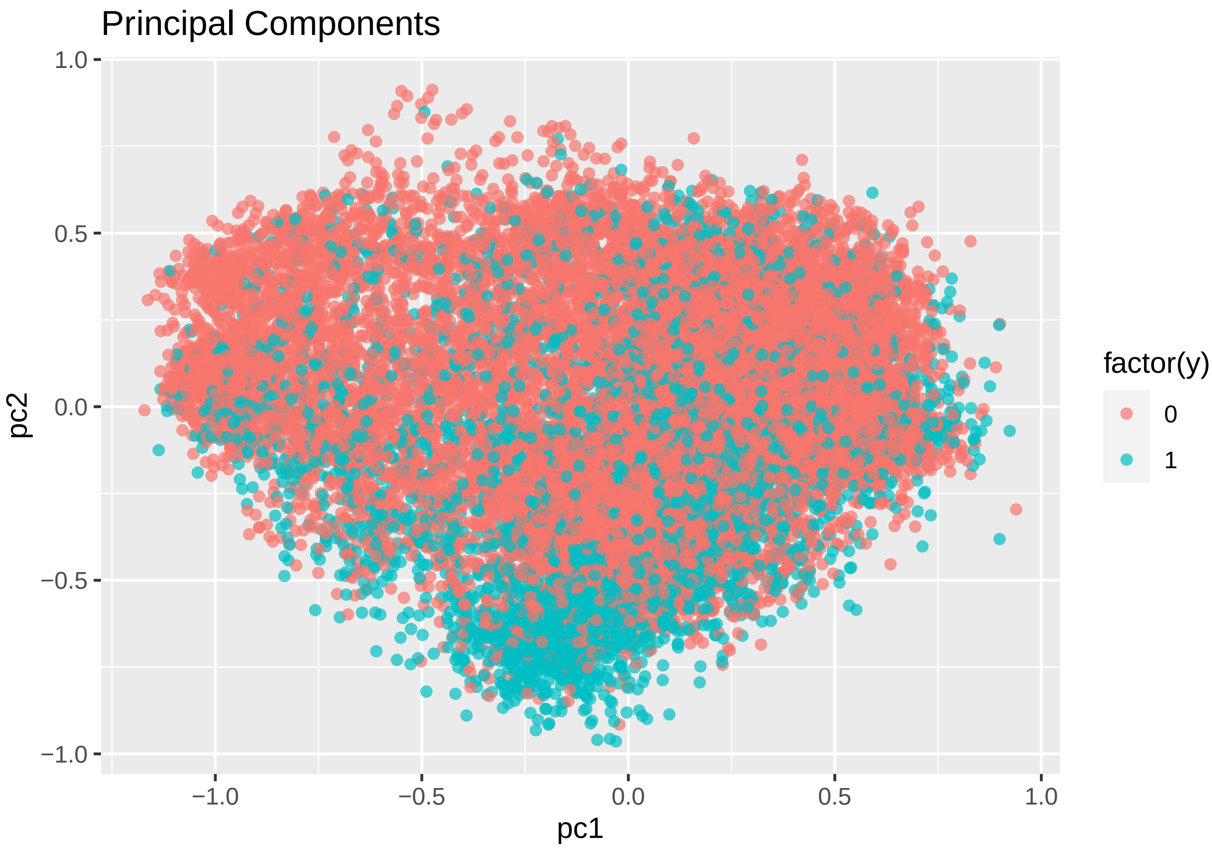
We inspect the correlation matrix for our continuous variables, which could potentially indicate problems with multicollinearity if we are use to use (general) linear models. By eye, there do not appear to be significant clusters of correlation.

```
df_num <- select_if(df[,-1], is.numeric)
cor_matrix <- cor(df_num)
heatmap(cor_matrix)
```



We use Principal Components Analysis (PCA) as a means to visualise the data in low-dimension, to determine if there are any explicitly discernible trends. By eye, there does not appear to be any significant differenc in the PCA representations for each class.

```
pcs <- prcomp(df_num[,-length(df_num)])
set.seed(2021)
ind <- sample(1:dim(df)[1], 20000)
sample <- data.frame(pcs$x[ind,1], pcs$x[ind,2], df[ind, "target"])
names(sample) = c("pc1", "pc2", "y")
ggplot(sample) + geom_jitter(aes(x = pc1, y = pc2, colour = factor(y)),alpha=0.7) + ggtitle('Principal (
```



Preprocessing

Modelling

We demonstrate a 'from scratch' Stochastic Gradient Descent routine for this classification problem

The logistic loss with a $\mathcal{L}^2$ penalty is given by:

$$l(\boldsymbol{\beta}) = -\sum_{i=1}^{N} y_i log(p(x_i;\boldsymbol{\beta})) + (1-y_i) log(1-p(x_i;\boldsymbol{\beta})) = \sum_{i=1}^{N}\left[y_i log\left(\frac{p(x_i;\boldsymbol{\beta})}{1-p(x_i;\boldsymbol{\beta})}\right) + log(1-p(x_i;\boldsymbol{\beta}))\right]$$

$$l(\boldsymbol{\beta}) = -\sum_{i=1}^{N}\left[y_i \boldsymbol{\beta}^T x_i - log(1+exp(\boldsymbol{\beta}^T x_i))\right]$$

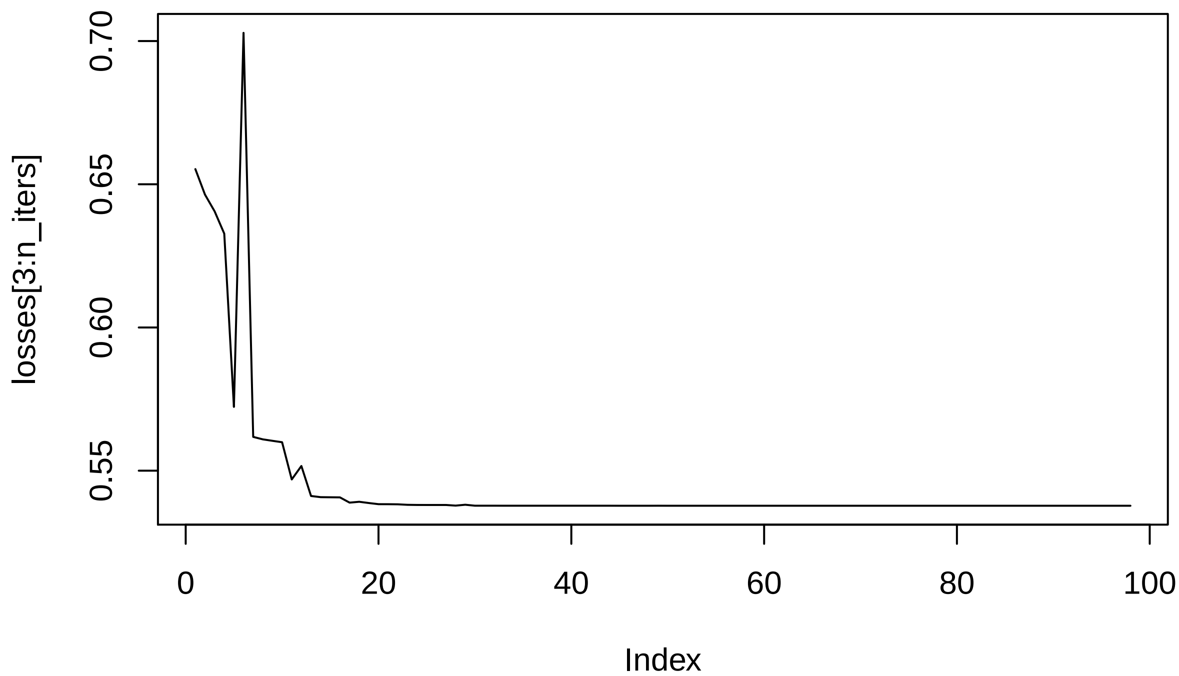
Then with the inclusion of the reularisation term:

$$l(\boldsymbol{\beta}) = -\sum_{i=1}^{N}\left[y_i \boldsymbol{\beta}^T x_i - log(1+exp(\boldsymbol{\beta}^T x_i))\right] - \lambda \boldsymbol{\beta}^T \boldsymbol{\beta}$$

The gradient is given by:

$$\nabla(\boldsymbol{\beta}) = -\sum_{i=1}^{N}\left[y_i x_i - \frac{x_i exp(\boldsymbol{\beta}^T x_i)}{1+exp(\boldsymbol{\beta}^T x_i)}\right] - \lambda 2\boldsymbol{\beta}$$

```r
set.seed(2021)
df_split <- initial_split(df, prop = 3/4)
df_train <- training(df_split)
df_test <- testing(df_split)

# binary crossentropy / log-loss
log_loss <- function(x, y, betas, lambda){
  logits <- x %*% betas
  - (t(y) %*% logits - sum(log(1 + exp(logits))) + lambda * t(betas) %*% betas) / dim(x)[1]
}

# logistic regression gradients
gradients <- function(x, y, betas, lambda){
  logits <- x %*% betas
  - (t(x) %*% (y - exp(logits)/(1 + exp(logits)))) - lambda *2 * betas / dim(x)[1]
}

# train-test
X_train = as.matrix(df_train[,grepl("cont", colnames(df_train))])
y_train = as.matrix(df_train$target)
X_test = as.matrix(df_test[,grepl("cont", colnames(df_train))])
y_test = as.matrix(df_test$target)


p = dim(X_train)[2]

lambda = 0
n_iters <- 100
init_step_size <- 1e-6

set.seed(2021)
beta_init <- matrix(rnorm(p),nrow=p)
beta_path <- matrix(rep(0, n_iters * p), nrow = n_iters, ncol=p)
beta_path[1,] = beta_init

last_grad <- grad <- gradients(X_train, y_train, beta_path[1,], lambda)
beta_path[2,] = beta_init - init_step_size * grad
grad <- gradients(X_train, y_train, beta_path[2,], lambda)

losses <- rep(0, n_iters)

for (i in 3:n_iters){
    step_size <- as.numeric(t(beta_path[i - 1,] - beta_path[i - 2,]) %*% (grad - last_grad) /
                (t(grad - last_grad) %*% (grad - last_grad)))
    beta_path[i,] <- beta_path[i - 1,] - step_size * grad
    last_grad <- grad
    grad <- gradients(X_train, y_train, beta_path[i, ], lambda)
    losses[i] <- log_loss(X_train, y_train, beta_path[i,], lambda)
}
plot(losses[3:n_iters], type="l")
```




```r
pred_train <- 1 / (1 + exp(-X_train %*% beta_path[100,]))
pred_test <- 1 / (1 + exp(-X_test %*% beta_path[100,]))
c(max(1 - mean(y_train), mean(y_train)),
  max(1 - mean(y_test), mean(y_test)),
  mean(((pred_train > 0.5) * 1) == y_train),
  mean(((pred_test > 0.5) * 1) == y_test))
```

```
## [1] 0.7349289 0.7357333 0.7559867 0.7586133
```

Linear Fit (Logistic Regression)

Nonlinear Fit (Tree-based)

```r
bst <- xgboost(data = X_train, y_train, max_depth = 2, nround = 10,
               verbose=0,
               objective='binary:logistic')
pred_test <- (predict(bst, X_test)) > 0.5 * 1
pred_train <- (predict(bst, X_train) > 0.5) * 1
mean(pred_train == y_train)
```

XGBoost

```
## [1] 0.7723689
```

```r
mean(pred_test == y_test)
```

```
## [1] 0.7726267
```

```r
# devtools::install_url('https://github.com/catboost/catboost/releases/download/v0.25.1/catboost-R-Darw

X_train = select(df_train, -c("target"))
y_train = df_train$target
X_test = select(df_test, -c("target"))
y_test = df_test$target

cat_features = 0:18
pool <- catboost.load_pool(X_train, y_train, cat_features = cat_features)
model <- catboost.train(pool, params=list(depth = 8, iterations = 10, loss_function='Logloss', verbose=0

train_preds <- catboost.predict(model, catboost.load_pool(X_train), prediction_type = 'Probability')
test_preds <- catboost.predict(model, catboost.load_pool(X_test), prediction_type = 'Probability')


c(mean((train_preds > 0.5) * 1 == y_train), mean((test_preds > 0.5) * 1 == y_test))
```

CatBoost

```
## [1] 0.8462133 0.8434133
```

```r
# df_train <- recipe( ~ ., data = df_train)
# df_train <- df_train %>% mutate(original = cat0)
# ref_cell <-
#   df_train %>%
#   step_dummy(cat0) %>% df_train
#
# df_train %>% mutate(across(starts_with("cat"), as.factor))
#
#
# dummies <- rec %>%
#   step_dummy(cat0) %>%
#   prep(training = df_train)
#
# lr_mod <-
#   logistic_reg() %>%
#   set_engine("glmnet")
#
# df_train$target = as.factor(df_train$target)
#
# rec <- recipe(target ~., data = df_train)
#
# flights_wflow <-
#   workflow() %>%
#   add_model(lr_mod) %>%
#   add_recipe(rec)
#
# flights_fit <-
#   flights_wflow %>%
#   fit(data = df_train)
```

Interpretation

Bibliography

- Preprocessing - encode categorical variables+
- Use small fold;
- Implement SGD from scratch
- Run GLM for 1 Var
- Run GLM / Logistic Regression - all
- Run GLM Ridge
- Run Trees - small fold;
-

https://www.kaggle.com/frankmollard/h2o-ml-ensemble

Hastie, T. and Tibshirani, R. and Friedman, J.H. 2009. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer.

James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani. 2017. "An Introduction To Statistical Learning." Springer.