PAC Report

Peng Yan

2024-11-29

During the PAC competition, I tried two kinds of models: linear regression and xgboost, to get the best result. According to my work, the xgboost model outperforms the linear regression model. However, the processes for improving these two models are completely different and enhance my understanding of both.As a result, I choose the xgboost model to get my final submission. On Canvas, you can see a R Script file. That includes the code I use to get my final Kaggle submission result. You can use it to chech my work.

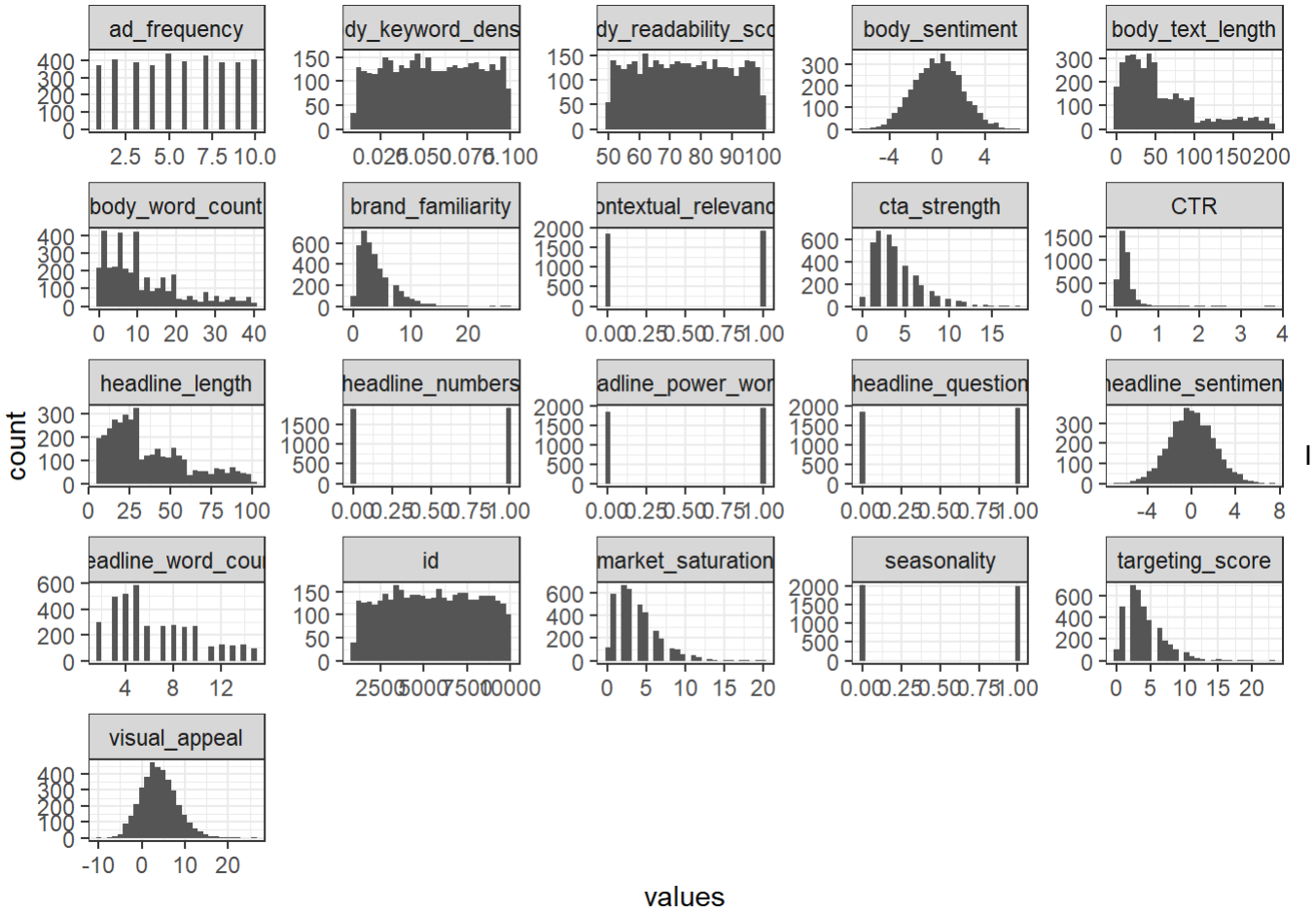
The first thing to do is to load the datasets.nzd is the train set withour data with zero CTR.

```
#read trainset and testset
setwd("~/PAC File")         #This code is used for my PC settings. Change it on yours
data = read.csv('analysis_data.csv')         #train
score = read.csv('scoring_data.csv')         #test
data$z = ifelse(data$CTR == 0, 1, 0)

#create nonzero CTR trainset for xgboost model2
nzd = data[data$z == 0,]
```

Data Exploration

In the data exploration, I first plot the distribution of the numeric predictors from the training data. We can see that the distribution of the predictors is complex, and there are many options to choose the combinations of predictors. The CTR plot is quite interesting. There are many 0 values in the CTR, which resembles a binary value instead of a numeric value. This insight helps me in my final model building. I use the lasso method to select the data I need for the linear regression model. Here is the code I used. Using a limited number of predictors improves my linear regression model. However, the XGBoost model has its own method for selecting predictors, so I do not need to limit the predictors and data for it. Including all of the data and predictors in the model is the best way to achieve the results. On the other hand, the CTR distribution is interesting. There are many 0 values in the CTR, which resembles a binary value instead of a numeric value. This insight helps me in my final model building.

```
library(ggplot2)
library(dplyr); library(tidyr)
data |>
  select_if(is.numeric)|>
  pivot_longer(cols = 1:21,names_to = 'numeric_predictor', values_to = 'values'  )|>
  ggplot(aes(x = values))+
  geom_histogram()+
  facet_wrap(numeric_predictor~., scales = 'free')+
  theme_bw()
```



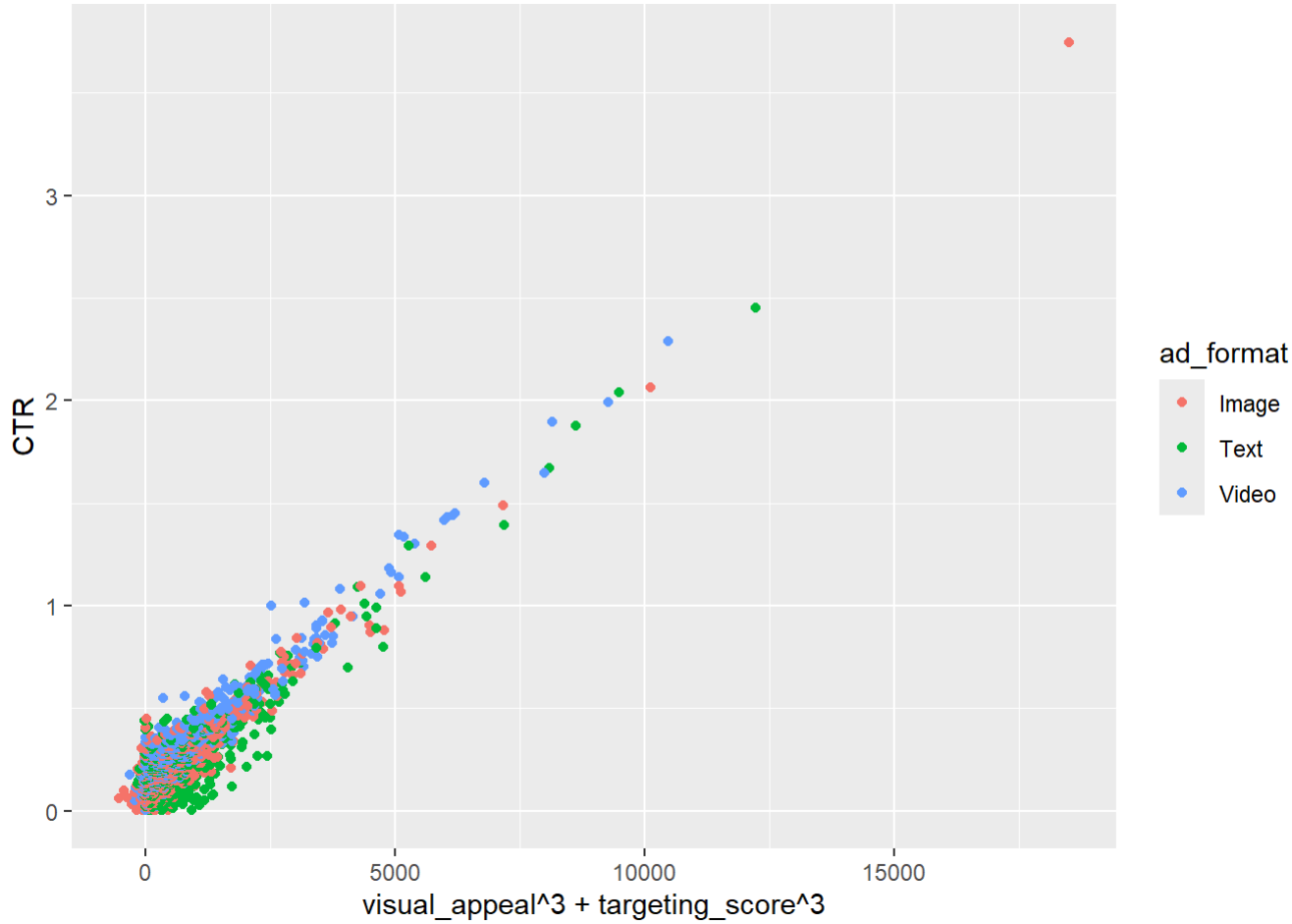
use the lasso method to select the data I need for the linear regression model. Using a limited number of predictors improves my linear regression model. However, the XGBoost model has its own method for selecting predictors, so I do not need to limit the predictors and data for it. Including all of the data and predictors in the model is the best way to achieve the results. On the other hand, the CTR distribution is interesting. There are many 0 values in the CTR, which resembles a binary value instead of a numeric value. This insight helps me in my final model building.

```
library(glmnet)

clean_data <- na.omit(data)
x = model.matrix(CTR~.-1,data=clean_data)
y = clean_data$CTR

set.seed(617)
cv_lasso = cv.glmnet(x = x,
                     y = y,
                     alpha = 1,
                     type.measure = 'mse')
coef(cv_lasso, s = cv_lasso$lambda.1se) |>
  round(6)
```

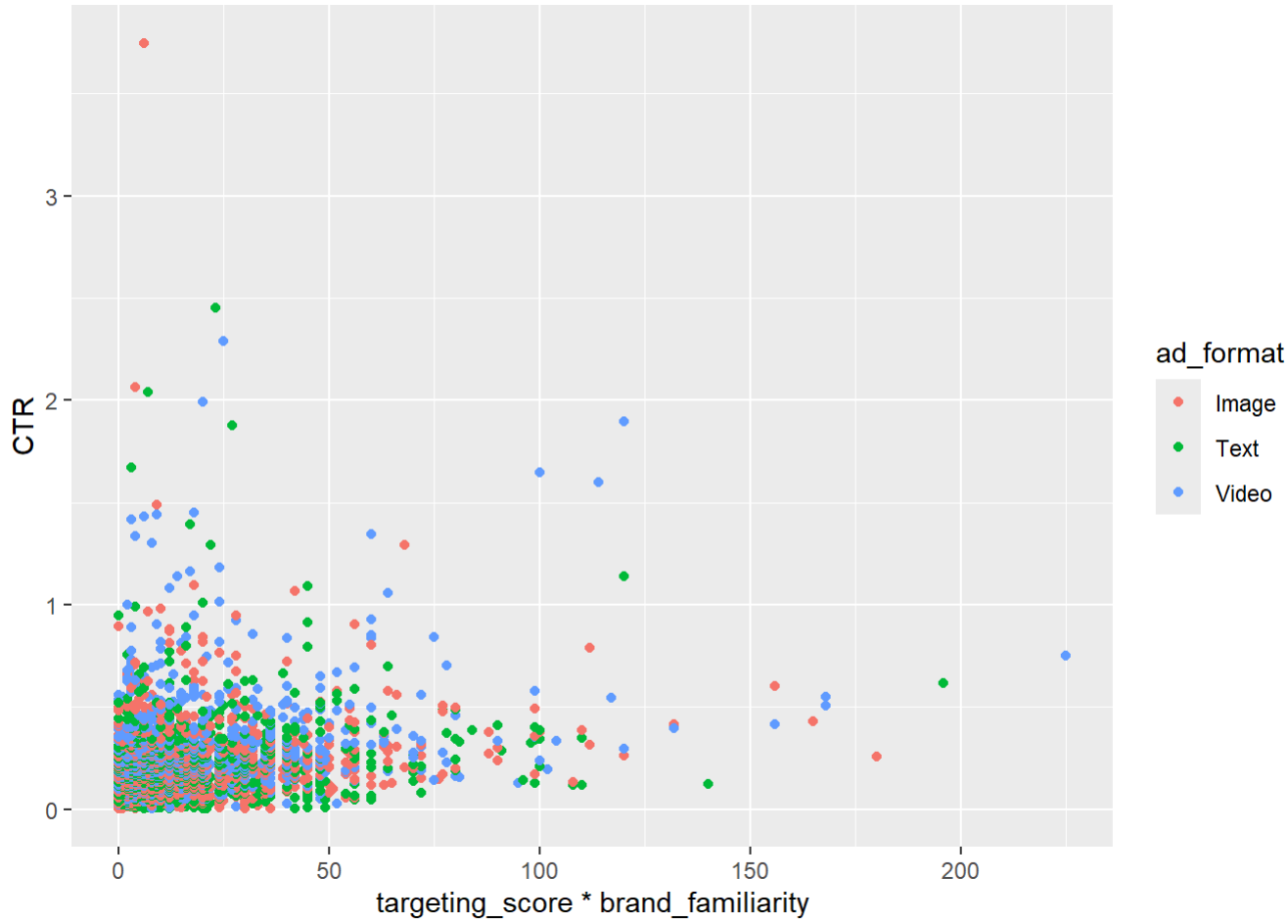
Feature Expansion

Between improving the two models, I tried using feature expansion to create new predictors to improve my model performance. However, this method failed in my models. I think this method is a good idea, but my understanding of the predictors is not deep enough to support it. Here is the CTR plot of some new predictors I created. The plot is good, but the prediction with them are bad.

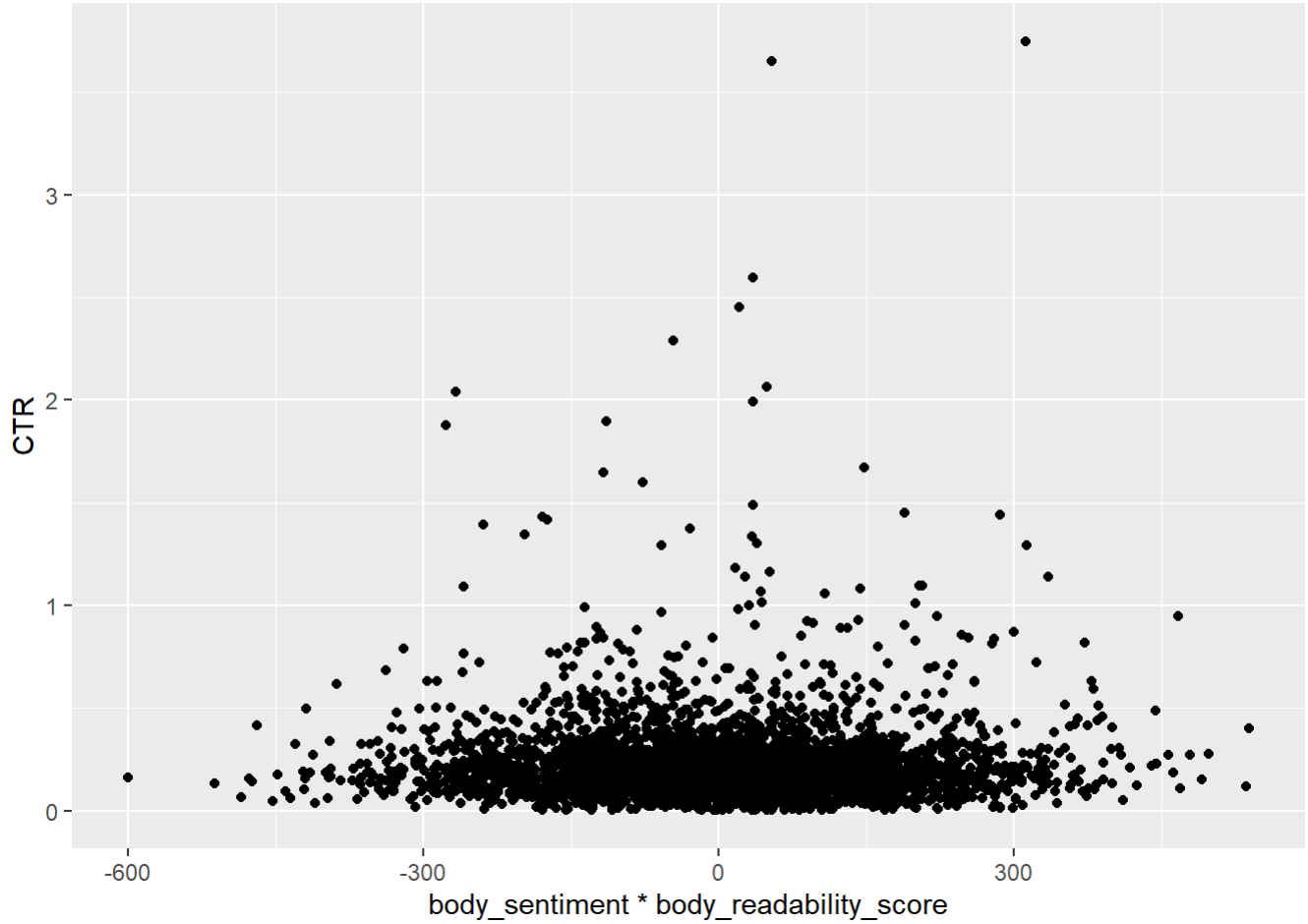
```
ggplot(nzd,aes(x=visual_appeal^3+targeting_score^3,y=CTR,color = ad_format))+
  geom_point()
```




```
ggplot(nzd,aes(x=targeting_score*brand_familiarity,y=CTR,color = ad_format))+
  geom_point()
```




```
ggplot(nzd,aes(x=body_sentiment*body_readability_score,y=CTR))+
  geom_point()
```



Data Preparation

Linear regression

The most significant aspect of data preparation for linear regression is handling NA values. For all numeric values, I replace NA values with the median of their respective columns. For categorical values, I employ two methods to address NA values: using the mode value and creating a new category called ‘Other.’ The results indicate that the second option, introducing the new category, enhances the outcomes. This improvement may be due to the special meanings associated with the NA values data.

Xgboost

The data treatment in XGBoost is different from linear regression. XGBoost does not accept categorical input, so I use the method I learned in our class to convert all categorical variables to dummy values. Additionally, the CTR, which is the output, should be separated from the predictors. The code I use is below. On the other hand, XGBoost can handle NA values automatically. In general, I find that no changes in the dataset can improve the XGBoost model’s work better. This may only be applicable in this competition. Here is the code I use to prepare data for Xgboost.

```
#data treatment for xgboost

library(vtreat)

trt = designTreatmentsZ(dframe = data,
                        varlist = names(data)[2:28])
newvars = trt$scoreFrame[trt$scoreFrame$code%in% c('clean','lev'),'varName']

train_input = prepare(treatmentplan = trt,
                      dframe = data,
                      varRestriction = newvars)     #for xgboost model 1 to decide whether CT
R is zero

test_input = prepare(treatmentplan = trt,
                     dframe = score,
                     varRestriction = newvars)


train_input2 = prepare(treatmentplan = trt,
                       dframe = nzd,
                       varRestriction = newvars)   #for xgboost model2
```

Model

Linear Regression

The linear regression is a simple but direct model to make predictions. I talked about the feature selection in the data exploration part. There are some things more I find in my model testing process. GAM can be used to increase the flexibility of numeric predictors but, it cannot change that linear regression has quite bad flexibility in category predictors, so I build multiple models using linear regression based on ad_format. The prediction of this method is better than using single model.

Xgboost

Zero or Not

The xgboost is the way I choose to use in the final submission. It is more powerful than linear regression. As I mentioned before, I use two xgboost models to get my final result. The two models are similar. In both of them, I use cross-validation to get the best round. The first one is to test whether the CTR is zero. This model output is binary, and I need to set a binary value to decide whether the value is zero. Since this model's accuracy is controlled by the hyperparameter. I choose the hyperparameters briefly and use 0.275 as the binary control value. The result shows that the accuracy is not bad. Only 1 out of 4000 predictions is wrong, and the error is false negative, which I think is acceptable.

```
library(xgboost)
set.seed(123)
params <- list(
  objective = "binary:logistic",
  eta = 0.1,
  max_depth = 10
)

# Cross Validation
cv_results <- xgb.cv(
  data=as.matrix(train_input),
  label = data$z,
  params = params,
  nrounds = 10000,
  nfold = 10,
  verbose = 1,
  early_stopping_rounds = 100,
  metrics = "rmse",                # Metric for evaluation
  maximize = FALSE
)
best_nrounds <- cv_results$best_iteration
#xgboost
xgboost1 = xgboost(
  data = as.matrix(train_input),
  params = params,
  label = data$z,
  nrounds = best_nrounds,
  verbose = 0
)
pred_train = predict(xgboost1,
                     newdata=as.matrix(train_input))
#determine 0 and test it
o = ifelse(pred_train>0.275, 1, 0)
data[o != data$z,'CTR']
pred_train[o != data$z]
```

Numeric

Then I can use all data with nonzero CTR to build my second model. I need to choose the parameters carefully for this one. Using grid search is a good idea, but it can not control the max_delta_step. I need to run the code several times to find my answer. Following is my grid search code.

```
grid <- expand.grid(
  nrounds = 3000,  # Number of boosting iterations
  max_depth = 1,  # Tree depth
  gamma = 0,
  min_child_weight = 1,
  eta = 0.1,  # Learning rate
  colsample_bytree = c(0.1,0.3,0.4,0.5,0.6,0.7,0.8,0.9),  # Subsample ratio of columns
  subsample = c(0.4,0.5,0.6,0.7,0.8,0.9)  # Subsample ratio of the training instance
)

set.seed(123)
train_control <- trainControl(
  method = "cv",
  number = 5,                    # 3-fold cross-validation
  verboseIter = TRUE,
  allowParallel = TRUE           # if you have multiple cores
)

xgb_model <- train(
  train_input2, nzd$CTR,
  method = "xgbTree",
  trControl = train_control,
  max_delta_step = 0.6,
  tuneGrid = grid,
  metric = "rsme"             # change to other metrics if needed
)
```

After I find the best parameters, I can run the code to get the result. In the two XGboost models, I use cross-validation to get my best rounds. Also, I tried some small changes in the best rounds to reduce the overfit. I find that minus 20 rounds can make the result better.

Final Code

This is my final code. Running this can get my Kaggle submission. I also include the R script file in my submission. You can also run that.

```
#read trainset and testset
data = read.csv('analysis_data.csv')         #train
score = read.csv('scoring_data.csv')         #test
library(dplyr); library(tidyr)
#insert a new column to show whether the CTR os 0

data$z = ifelse(data$CTR == 0, 1, 0)

#create nonzero CTR trainset for xgboost model2
nzd = data[data$z == 0,]

#data treatment for xgboost

library(vtreat)

trt = designTreatmentsZ(dframe = data,
                        varlist = names(data)[2:28])
newvars = trt$scoreFrame[trt$scoreFrame$code%in% c('clean','lev'),'varName']

train_input = prepare(treatmentplan = trt,
                      dframe = data,
                      varRestriction = newvars)     #for xgboost model 1 to decide whether CT
R is zero

test_input = prepare(treatmentplan = trt,
                     dframe = score,
                     varRestriction = newvars)


train_input2 = prepare(treatmentplan = trt,
                       dframe = nzd,
                       varRestriction = newvars)   #for xgboost model2
#model1 decide whether the CTR is zero
library(xgboost)
set.seed(123)
params <- list(
  objective = "binary:logistic",
  eta = 0.1,
  max_depth = 10
)

## Cross Validation
cv_results <- xgb.cv(
  data=as.matrix(train_input),
  label = data$z,
  params = params,
  nrounds = 10000,
  nfold = 10,
  verbose = 1,
  early_stopping_rounds = 100,
  metrics = "rmse",
  maximize = FALSE
)
best_nrounds <- cv_results$best_iteration
## build model1
```

```
xgboost1 = xgboost(
  data = as.matrix(train_input),
  params = params,
  label = data$z,
  nrounds = best_nrounds,
  verbose = 0
)
pred_train = predict(xgboost1,
                     newdata=as.matrix(train_input))
#test the result using trainset(these tests result shows that the error is small)
o = ifelse(pred_train>0.275, 1, 0)  #I choose 0.275 as binary value
data[o != data$z,'CTR']
pred_train[o != data$z]
sum(o != data$z)
#prediction of model1
pred = predict(xgboost1,
               newdata=as.matrix(test_input))
table(pred)
o1 = ifelse(pred>0.275, 1, 0)
table(o1)

#model2 numeric output
set.seed(123)
params <- list(
  objective = "reg:squarederror",
  eta = 0.1,
  max_depth = 1,
  subsample = 0.9,
  colsample_bytree = 0.6,
  max_delta_step = 0.6

)

#Cross Validation
cv_results <- xgb.cv(
  data=as.matrix(train_input2),
  label = nzd$CTR,
  params = params,
  nrounds = 10000,
  nfold = 5,
  verbose = 1,
  early_stopping_rounds = 100,
  metrics = "rmse",                # Metric for evaluation
  maximize = FALSE
)
best_nrounds <- cv_results$best_iteration-20
#Build model2
xgboost2 = xgboost(
  data = as.matrix(train_input2),
  params = params,
  label =  nzd$CTR,
  nrounds = best_nrounds,
  verbose = 0
)
#numeric prediction
pred1 = predict(xgboost2,newdata=as.matrix(test_input))
```

```
submission_file = data.frame(id = score$id, CTR = pred1)
#overwrite the 0 CTR with our first prediction
submission_file[o1 == 1,]$CTR=0
#The negative value should not exist. Use zero instead.
submission_file[submission_file$CTR < 0,]$CTR = 0
write.csv(submission_file, 'sample_submission.csv',row.names = F)
#Show the final output
table(submission_file$CTR)
```

Limitation

First, I failed to use feature expansion. I think this method can increase my accuracy a lot. If I got more time to understand those predictors better, I would utilize this successfully in my model. Second, I did not reduce the influence of the seed in my model, which means my model can be good or bad by luck. There are some ways to reduce the influence of seed. In my next Kaggle competition, I will use them to get a better model. Finally, I find reducing the round a little bit can improve my prediction, but I do not find a way to optimize it. I think I can improve it in the future.