R Notebook

```
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------------------------
--- tidyverse 1.2.1 --
```

```
## √ ggplot2 3.1.1       √ purrr   0.3.2
## √ tibble  2.1.1       √ dplyr   0.8.0.1
## √ tidyr   0.8.3       √ stringr 1.4.0
## √ readr   1.3.1       √ forcats 0.4.0
```

```
## -- Conflicts ------------------------------------------------------------ ti
dyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(readr)
library(e1071)
library(MASS)
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```
library(rpart)
library(tree)
library(randomForest)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(gbm)
```

```
## Loaded gbm 2.1.5
```

```
library(ROCR)
```

```
## Loading required package: gplots
```

```
##
## Attaching package: 'gplots'
```

```
## The following object is masked from 'package:stats':
##
##     lowess
```

```
library(stringdist)
library(caret)
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

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

```
location <- read_csv("C:/Users/zhang/Desktop/6240_r/hw4/Location.csv")
```

```
## Parsed with column specification:
## cols(
##   locationID = col_double(),
##   regionID = col_double()
## )
```

```
#pair comparison see if duplicate
itemPairsTest <- read_csv("C:/Users/zhang/Desktop/6240_r/hw4/ItemPairs_test.csv")
```

```
## Parsed with column specification:
## cols(
##   id = col_double(),
##   itemID_1 = col_double(),
##   itemID_2 = col_double()
## )
```

```
itemPairsTrain <- read_csv("C:/Users/zhang/Desktop/6240_r/hw4/ItemPairs_train.csv")
```

```
## Parsed with column specification:
## cols(
##   itemID_1 = col_double(),
##   itemID_2 = col_double(),
##   isDuplicate = col_double(),
##   generationMethod = col_double()
## )
```

```
#info
itemInfoTest <- read_csv("C:/Users/zhang/Desktop/6240_r/hw4/ItemInfo_test.csv")
```

```
## Parsed with column specification:
## cols(
##   itemID = col_double(),
##   categoryID = col_double(),
##   title = col_character(),
##   description = col_character(),
##   images_array = col_character(),
##   attrsJSON = col_character(),
##   price = col_double(),
##   locationID = col_double(),
##   metroID = col_double(),
##   lat = col_double(),
##   lon = col_double()
## )
```

```
itemInfoTrain <- read_csv("C:/Users/zhang/Desktop/6240_r/hw4/ItemInfo_train.csv")
```

```
## Parsed with column specification:
## cols(
##   itemID = col_double(),
##   categoryID = col_double(),
##   title = col_character(),
##   description = col_character(),
##   images_array = col_character(),
##   attrsJSON = col_character(),
##   price = col_double(),
##   locationID = col_double(),
##   metroID = col_double(),
##   lat = col_double(),
##   lon = col_double()
## )
```

```
#maerge location to the item info training and testing datasets
str(itemInfoTrain)
```

```
## Classes 'spec_tbl_df', 'tbl_df', 'tbl' and 'data.frame': 3344613 obs. of  11 vari
ables:
##  $ itemID      : num  1 3 4 7 8 9 12 15 16 19 ...
##  $ categoryID  : num  81 14 84 84 39 39 9 32 27 88 ...
##  $ title       : chr  "Продам Камаз 6520" "Yamaha r6" "iPhone 3gs 8gb" "Xiaomi Mi
4 3г6 RAM + 16г6 ROM белый" ...
##  $ description : chr  "Продам Камаз 6520 20 тонн" "Весь в тюнинге." "Телефон в хо
рошем состоянии, трещин и сколов нет, за все время менялся только аккумулятор(поэтом
у заряд держит "| __truncated__ "Отличный подарок на новый год от \"китайской apple
\"\nНовый в упаковке. Коробку вместе вскроем)\nЭкран: 5 дюймо"| __truncated__ ...
##  $ images_array: chr  "1064094, 5252822, 6645873, 6960145, 9230265" "11919573, 14
412228, 3204180, 6646877" "14384831, 6102021" NA ...
##  $ attrsJSON   : chr  "{\"Вид техники\":\"Грузовики\"}" "{\"Вид техники\":\"Мотоц
иклы\", \"Вид мотоцикла\":\"Спортивные\"}" "{\"Вид телефона\":\"iPhone\"}" "{\"Вид т
елефона\":\"Другие марки\"}" ...
##  $ price       : num  300000 300000 3500 13500 500 7000 445000 1600 1000 5000 ...
##  $ locationID  : num  648140 639040 640650 662210 624360 ...
##  $ metroID     : num  NA NA NA NA NA ...
##  $ lat         : num  64.7 55.7 56.2 55.8 55.8 ...
##  $ lon         : num  30.8 37.3 43.5 37.6 37.6 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   itemID = col_double(),
##   ..   categoryID = col_double(),
##   ..   title = col_character(),
##   ..   description = col_character(),
##   ..   images_array = col_character(),
##   ..   attrsJSON = col_character(),
##   ..   price = col_double(),
##   ..   locationID = col_double(),
##   ..   metroID = col_double(),
##   ..   lat = col_double(),
##   ..   lon = col_double()
##   .. )
```

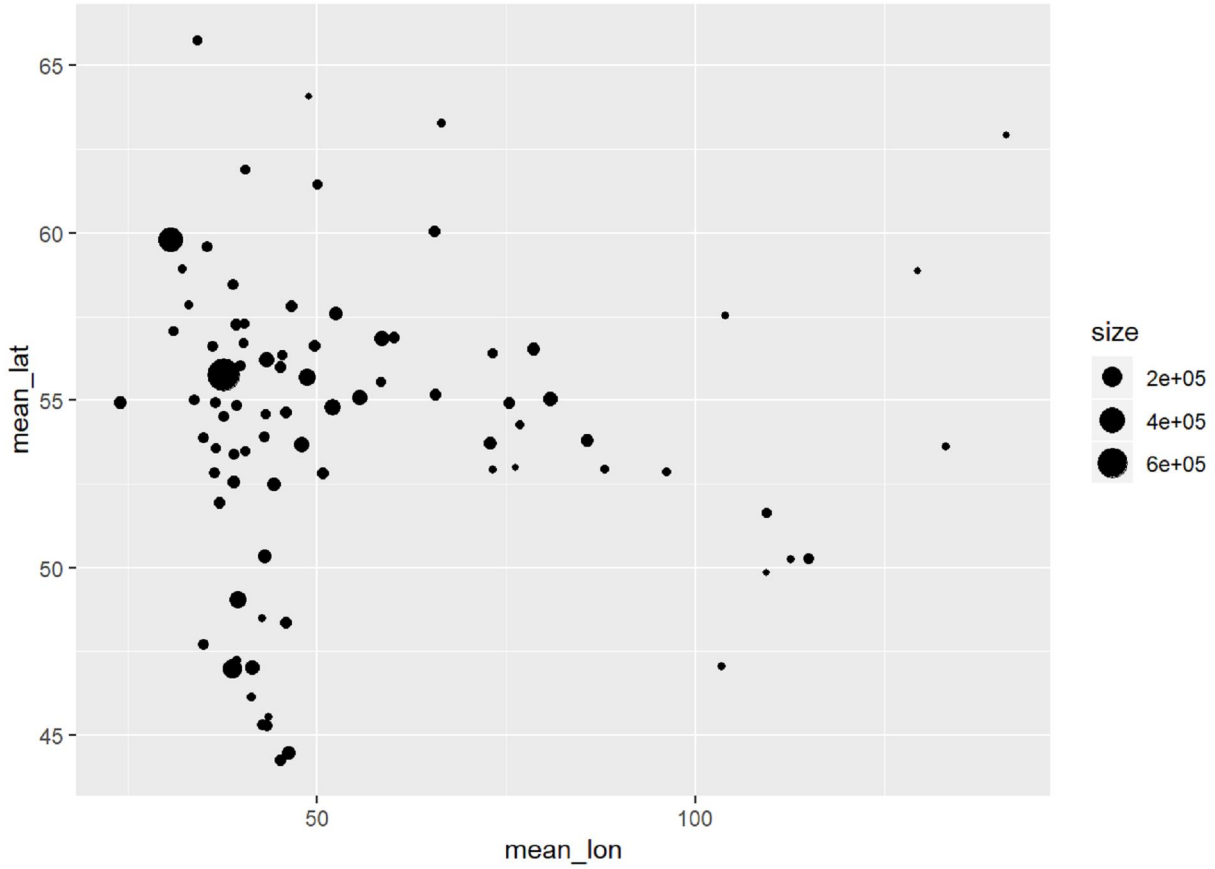
```
itemInfoTrain <- itemInfoTrain %>%
  left_join(location)
```

```
## Joining, by = "locationID"
```

```
itemInfoTest <- itemInfoTest %>%
  left_join(location)
```

```
## Joining, by = "locationID"
```

```
itemInfoTrain %>% group_by(regionID) %>%
  summarise(mean_lat=mean(lat),
            mean_lon=mean(lon),
            size = n()
  ) %>%
  ggplot(aes(y=mean_lat,x=mean_lon,size=size))+
  geom_point()
```



```
##Join training data##

train <- itemPairsTrain %>% left_join(itemInfoTrain,
                                      by = c("itemID_1" = "itemID"))
colnames(train)[5:15] <- paste0(colnames(train)[5:15],"_1")

train <- train %>% left_join(itemInfoTrain,
                             by = c("itemID_2" = "itemID"))
colnames(train)[16:26] <- paste0(colnames(train)[16:26],"_2")


##Join testing data##

test <- itemPairsTest %>% left_join(itemInfoTest,
                                    by = c("itemID_1" = "itemID"))
colnames(test)[4:14] <- paste0(colnames(test)[4:14],"_1")

test <- test %>% left_join(itemInfoTest,
                           by = c("itemID_2" = "itemID"))
colnames(test)[15:25] <- paste0(colnames(test)[15:25],"_2")

##Join testing data##

test <- itemPairsTest %>% left_join(itemInfoTest,
                                    by = c("itemID_1" = "itemID"))
colnames(test)[4:14] <- paste0(colnames(test)[4:14],"_1")

test <- test %>% left_join(itemInfoTest,
                           by = c("itemID_2" = "itemID"))
colnames(test)[15:25] <- paste0(colnames(test)[15:25],"_2")
```

```
#make adjustement to the datasets
quasi_creator_1 <- function(x){
  x %>% mutate(
         distance = sqrt((lat_1-lat_2)^2+(lon_1-lon_2)^2),
         price_min=pmin(log(price_1),log(price_2)),
         price_max=pmax(log(price_1),log(price_2)),
         price.diff=abs(price_1-price_2),
         price.diffpct=1*(abs(price_1-price_2)/pmin(price_1,price_2)<0.20), #if the
difference in price is higher than 15% of the cheaper item, the price is assumed to
be significantly different
         description_same=1*(description_1==description_2),
         location_same=1*(locationID_1==locationID_2),
         region_same=1*(regionID_1==regionID_2),
         category_same=1*(categoryID_1==categoryID_2),
         title_same=1*(title_1==title_2))
}

#change to the datasets using the function created above
train <- train %>% quasi_creator_1
test <- test %>% quasi_creator_1
```

```
#use imputation to fill up the missing spots for train
train$price_min <- ifelse(is.na(train$price_min), mean(train$price_min, na.rm=TRU
E), train$price_min)

train$price_max <- ifelse(is.na(train$price_max), mean(train$price_max, na.rm=TRU
E), train$price_max)

train$distance <- ifelse(is.na(train$distance), mean(train$distance, na.rm=TRUE), tr
ain$distance)

train$price.diffpct <- ifelse(is.na(train$price.diffpct), mean(train$price.diffpct,
na.rm=TRUE), train$price.diffpct)

train$region_same <- ifelse(is.na(train$region_same), mean(train$region_same, na.rm=
TRUE), train$region_same)

train$title_same <- ifelse(is.na(train$title_same), mean(train$title_same, na.rm=TRU
E), train$title_same)

train$description_same <- ifelse(is.na(train$description_same), mean(train$descripti
on_same, na.rm=TRUE), train$description_same)

train$location_same <- ifelse(is.na(train$location_same), mean(train$location_same,
na.rm=TRUE), train$location_same)

train$price_max <- ifelse(is.na(train$price_max), mean(train$price_max, na.rm=TRU
E), train$price_max)

train$price_min <- ifelse(is.na(train$price_min), mean(train$price_min, na.rm=TRU
E), train$price_min)
```

Part A Data Prepartion

1. Randomly subsample some portion of the training dataset for training (anywhere from 3% to 60% of the dataset). I choose 5% of data.

```
set.seed(1)
sampel_id <- sample(nrow(train),
                      size = floor(nrow(train)*.05),replace = FALSE)

sample <- train[sampel_id,]
```

2. Subsample one third of this dataset, this will be your training data.

```
train_id <- sample(seq_len(nrow(sample)), size = nrow(sample)/3)

train <- sample[train_id, ]
rest <- sample[-train_id, ]
```

3. An another one third of the dataset will be your validation data.

```
validate_id <- sample(seq_len(nrow(rest)), size = nrow(rest)/2)

validate <- rest[validate_id, ]
```

4. Final third will be your test dataset.

```
test1 <- rest[-validate_id, ]
```

Part B Model Building

1. Fit 10 different models on the training data

From my HW4, logistic regression generates the highest AUC so I will use logistic model to form 5 models here.

```
# Remove NA Values
train <- na.omit(train)

#logistic
log.fit1 <- glm(isDuplicate ~price_min+price_max+distance+ location_same+description
_same+region_same+category_same+title_same+price.diffpct,data=train,family="binomia
l")

log.fit2 <- glm(isDuplicate ~distance+ location_same+description_same+region_same+ca
tegory_same+title_same+price.diffpct,data=train,family="binomial")

log.fit3 <- glm(isDuplicate ~ location_same+description_same+region_same+category_sa
me+title_same+price.diffpct,data=train,family="binomial")

log.fit4 <- glm(isDuplicate ~ description_same+region_same+category_same+title_same+
price.diffpct,data=train,family="binomial")

log.fit5 <- glm(isDuplicate ~ region_same+category_same+title_same+price.diffpct,dat
a=train,family="binomial")
```

My second best model is a gradient boost machine model.

```
#install.packages("gbm")
library(gbm)

gbm.fit1 <-gbm(isDuplicate~price_min+price_max+distance+location_same+description_sa
me+region_same+title_same+price.diffpct,data=train,
                distribution = "bernoulli",
                n.trees = 200,
                interaction.depth = 2)

gbm.fit2 <-gbm(isDuplicate~price_min+price_max+distance+location_same+description_sa
me+region_same+title_same+price.diffpct,data=train,
                distribution = "bernoulli",
                n.trees = 400,
                interaction.depth = 4)

gbm.fit3 <-gbm(isDuplicate~price_min+price_max+distance+location_same+description_sa
me+region_same+title_same+price.diffpct,data=train,
                distribution = "bernoulli",
                n.trees = 600,
                interaction.depth = 6)

gbm.fit4 <-gbm(isDuplicate~price_min+price_max+distance+location_same+description_sa
me+region_same+title_same+price.diffpct,data=train,
                distribution = "bernoulli",
                n.trees = 800,
                interaction.depth = 8)

gbm.fit5 <-gbm(isDuplicate~price_min+price_max+distance+location_same+description_sa
me+region_same+title_same+price.diffpct,data=train,
                distribution = "bernoulli",
                n.trees = 1000,
                interaction.depth = 10)
```

2. Compute and store the probability predictions for each of these 10 models for both the “validation” and the “test” data?

```
# validation gbm
validate$gbmpred1 <- predict(object = gbm.fit1,
                   newdata = validate,
                   n.trees = 200,
                  type = "response")

validate$gbmpred2 <- predict(object = gbm.fit2,
                   newdata = validate,
                   n.trees = 200,
                  type = "response")

validate$gbmpred3 <- predict(object = gbm.fit3,
                   newdata = validate,
                   n.trees = 200,
                  type = "response")

validate$gbmpred4 <- predict(object = gbm.fit4,
                   newdata = validate,
                   n.trees = 200,
                  type = "response")

validate$gbmpred5 <- predict(object = gbm.fit5,
                   newdata = validate,
                   n.trees = 200,
                  type = "response")

# test gbm
test1$gbmpred1 <- predict(object = gbm.fit1,
                   newdata = test1,
                   n.trees = 200,
                  type = "response")

test1$gbmpred2 <- predict(object = gbm.fit2,
                   newdata = test1,
                   n.trees = 200,
                  type = "response")

test1$gbmpred3 <- predict(object = gbm.fit3,
                   newdata = test1,
                   n.trees = 200,
                  type = "response")

test1$gbmpred4 <- predict(object = gbm.fit4,
                   newdata = test1,
                   n.trees = 200,
                  type = "response")

test1$gbmpred5 <- predict(object = gbm.fit5,
                   newdata = test1,
                   n.trees = 200,
                  type = "response")
```

```
# validation logistic
validate$logpred1<-predict(log.fit1, newdata = validate, type = "response")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading
```

```
validate$logpred2<-predict(log.fit2, newdata = validate, type = "response")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading
```

```
validate$logpred3<-predict(log.fit3, newdata = validate, type = "response")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading
```

```
validate$logpred4<-predict(log.fit4, newdata = validate, type = "response")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading
```

```
validate$logpred5<-predict(log.fit5, newdata = validate, type = "response")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading
```

```
# test logistic
test1$logpred1<-predict(log.fit1, newdata = test1, type = "response")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading
```

```
test1$logpred2<-predict(log.fit2, newdata = test1, type = "response")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading
```

```
test1$logpred3<-predict(log.fit3, newdata = test1, type = "response")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading
```

```
test1$logpred4<-predict(log.fit4, newdata = test1, type = "response")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading
```

```
test1$logpred5<-predict(log.fit5, newdata = test1, type = "response")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading
```

3. Use your favorite classification method to model the responses in the validation data with the new 10 variables, which are the probability estimates. Here I choose logistic regression as my favorite classification method.

```
#newly fitted model
glm.final <- glm(validate$isDuplicate~price_min+price_max+distance+location_same+des
cription_same+region_same+title_same+price.diffpct+gbmpred1+logpred1+gbmpred2+logpre
d2+gbmpred3+logpred3+gbmpred4+logpred4+gbmpred5+logpred5,data = validate, family="bi
nomial")
```

4. Use this last stacking model to obtain classifications for the test data.

```
glm.final.test.predict <-predict(glm.final, newdata = test1, type = "response")
```

5. Compute the AUC score of your final stacked model. The AUC score for the new logistic model is 0.7616529 which is higher than that of the logistic model (0.7455485) I fit last time.

```
final_pred <- glm.final %>%
  predict(test1,type="response") %>%
  prediction(labels=test1$isDuplicate)

performance(final_pred,"auc")@y.values[[1]]
```

```
## [1] 0.7619438
```