project2.R

Weifeng She

11/21/2016

1. Use linear regression to predict profit based on all available numeric variables. Graph the train and test MSE as a function of the train set size (averaged over 10 random data partitions as described above)?

```r
library(GGally)
library(reshape)
library(ggplot2)
library(tm)
```

```
## Loading required package: NLP
##
## Attaching package: 'NLP'
##
## The following object is masked from 'package:ggplot2':
##
##     annotate
```

```r
library(caret)
```

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

```r
#library(qdap)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
##
## The following object is masked from 'package:reshape':
##
##     rename
##
## The following object is masked from 'package:GGally':
##
##     nasa
##
## The following objects are masked from 'package:stats':
##
##     filter, lag
##
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(topicmodels)
library(tidyr)
```

```
##
## Attaching package: 'tidyr'
##
## The following object is masked from 'package:reshape':
##
##     expand
```

```r
library(reshape2)
```

```
##
## Attaching package: 'reshape2'
##
## The following objects are masked from 'package:reshape':
##
##     colsplit, melt, recast
```

```r
library(Matrix)
```

```
##
## Attaching package: 'Matrix'
##
## The following object is masked from 'package:tidyr':
##
##     expand
##
## The following object is masked from 'package:reshape':
##
##     expand
##
## The following objects are masked from 'package:base':
##
##     crossprod, tcrossprod
```

In this dataset numeric variables include: Runtime, Metascore, imdbRating imdbVotes , tomatoMeter, tomatoRating, tomatoReviews,tomatoFresh, tomatoRotten, tomatoUserMeter, tomatoUserRating, tomatoUserReviews. Since tomatoMeter = tomatoFresh/(tomatoFresh + tomatoRotten) and tomatoReviews = tomatoFresh + tomatoRotten, both are not used for model building.

The lm model based on all the numeric variables could get mse at 1.02e+16 for test data.

```r
load("movies_merged")
movies_merged = subset(movies_merged, Type == "movie")
dim(movies_merged) # after subset, there are only 40000 movies left
```

```
## [1] 40000    39
```

```r
# remove all the NA rows for Gross and Budget
movies_merged = movies_merged[complete.cases(movies_merged$Gross),]
movies_merged = movies_merged[complete.cases(movies_merged$Budget),]
# create Profit column
movies_merged$Profit <- movies_merged$Gross - movies_merged$Budget
# drop Gross and Budget column
```

```r
movies_merged$Gross <- NULL
movies_merged$Budget <- NULL # 4558 X 38
movies_merged = movies_merged[movies_merged$Year >= 2000, ] # 3332 X 38
dim(movies_merged) # only 3332 rows left
```

```
## [1] 3332   38
```

```r
# print out the column names
colnames(movies_merged)
```

```
##  [1] "Title"             "Year"              "Rated"
##  [4] "Released"          "Runtime"           "Genre"
##  [7] "Director"          "Writer"            "Actors"
## [10] "Plot"              "Language"          "Country"
## [13] "Awards"            "Poster"            "Metascore"
## [16] "imdbRating"        "imdbVotes"         "imdbID"
## [19] "Type"              "tomatoMeter"       "tomatoImage"
## [22] "tomatoRating"      "tomatoReviews"     "tomatoFresh"
## [25] "tomatoRotten"      "tomatoConsensus"   "tomatoUserMeter"
## [28] "tomatoUserRating"  "tomatoUserReviews" "tomatoURL"
## [31] "DVD"               "BoxOffice"         "Production"
## [34] "Website"           "Response"          "Domestic_Gross"
## [37] "Date"              "Profit"
```

```r
# convert factor variable to numeric
movies_merged$Metascore <- as.numeric(as.character(movies_merged$Metascore))
```

```
## Warning: NAs introduced by coercion
```

```r
# check how many NAs in each column
sapply(movies_merged, function(x) sum(is.na(x)))
```

```
##             Title              Year             Rated          Released
##                 0                 0                 0                41
##           Runtime             Genre          Director            Writer
##                 0                 0                 0                 0
##            Actors              Plot          Language           Country
##                 0                 0                 0                 0
##            Awards            Poster         Metascore        imdbRating
##                 0                 0               420                43
##         imdbVotes            imdbID              Type       tomatoMeter
##                43                 0                 0               396
##       tomatoImage      tomatoRating     tomatoReviews       tomatoFresh
##                 0               396               395               395
##      tomatoRotten   tomatoConsensus   tomatoUserMeter  tomatoUserRating
##               395                 0               195               193
## tomatoUserReviews         tomatoURL               DVD         BoxOffice
##                89                 0               276                 0
##        Production           Website          Response    Domestic_Gross
##                 0                 0                 0                 0
##              Date            Profit
##                 0                 0
```

```r
# we can see there are no missing value in both Budget and Gross columns
# select the numeric columns
movies_numeric <- movies_merged[, c(5, 15, 16, 17, 22, 24, 25, 27, 28, 29, 38)]
dim(movies_numeric) # 3332 X 11
```

```
## [1] 3332   11
```

```r
head(movies_numeric)
```

```
##    Runtime Metascore imdbRating imdbVotes tomatoRating tomatoFresh
## 13  96 min         9        2.3     37613          1.7           1
## 21  95 min        27        4.6     22611          3.3          13
## 27 118 min        37        5.7     59443          4.6          50
## 28 105 min        53        5.5     13197          5.3          43
## 29 106 min        54        6.4      8427          5.8          64
## 38  89 min        32        4.1     25931          3.4          11
##    tomatoRotten tomatoUserMeter tomatoUserRating tomatoUserReviews
## 13          116              10              2.0             57878
## 21          103              45              3.1            401851
## 27          119              42              2.8            212460
## 28           72              62              3.3             65086
## 29           55              63              3.3             42035
## 38           76              21              2.5            261558
##       Profit
## 13 -11821431
## 21  78114471
## 27  19944017
## 28  13351350
## 29  18508485
## 38  47192859
```

```r
# convert Runtime to numeric value
head(sort(unique(movies_numeric$Runtime)))
```

```
## [1] "1 min"   "10 min"  "100 min" "101 min" "102 min" "103 min"
```

```r
tail(sort(unique(movies_numeric$Runtime)))
```

```
## [1] "95 min" "96 min" "97 min" "98 min" "99 min" "N/A"
```

```r
# Runtime_num function modified from project 1
Runtime_num <- c()

for(i in seq_along(movies_numeric$Runtime)){

        x <- strsplit(movies_numeric$Runtime[i], ' ')[[1]]
        if (length(x) ==  2){
                if(x[2] == "min") #3) "xx min"
                {Runtime_num <- c(Runtime_num, suppressWarnings(as.numeric(x[1])))}
                # 4)"xx h"
```

```
                else{Runtime_num <- c(Runtime_num, suppressWarnings(as.numeric(x[1])) * 60)}
        }
        # 2) "XX h xx min"
        else if( length((x) == 4)) {
                Runtime_num <- c(Runtime_num, suppressWarnings(as.numeric(x[1])) * 60  + suppressWarning
        }
        # 1)"N/A"
        else Runtime_num <- c(Runtime_num, NA)
}

movies_numeric$Runtime <- Runtime_num

head(movies_numeric)
```

```
##    Runtime Metascore imdbRating imdbVotes tomatoRating tomatoFresh
## 13      96         9        2.3     37613          1.7           1
## 21      95        27        4.6     22611          3.3          13
## 27     118        37        5.7     59443          4.6          50
## 28     105        53        5.5     13197          5.3          43
## 29     106        54        6.4      8427          5.8          64
## 38      89        32        4.1     25931          3.4          11
##    tomatoRotten tomatoUserMeter tomatoUserRating tomatoUserReviews
## 13          116              10              2.0             57878
## 21          103              45              3.1            401851
## 27          119              42              2.8            212460
## 28           72              62              3.3             65086
## 29           55              63              3.3             42035
## 38           76              21              2.5            261558
##       Profit
## 13 -11821431
## 21  78114471
## 27  19944017
## 28  13351350
## 29  18508485
## 38  47192859
```

```
# check missing value for each row
sapply(movies_numeric, function(x) sum(is.na(x)))
```

```
##           Runtime         Metascore        imdbRating         imdbVotes
##                37               420                43                43
##      tomatoRating       tomatoFresh      tomatoRotten   tomatoUserMeter
##               396               395               395               195
##  tomatoUserRating tomatoUserReviews            Profit
##               193                89                 0
```

```
# remove all the NA rows
# convert missing value to the median of each column
for( i in 1:10){
movies_numeric[, i][is.na(movies_numeric[,i])] <-
  median(movies_numeric[,i], na.rm = T)
}
```

```r
# write funtion to calculate mse for train and test dataset
calculate_MSE <- function(dataset, percent){

  # splict data to train and test
  sample_size <- floor(percent * nrow(dataset))
  train_index <- sample(seq_len(nrow(dataset)), size = sample_size)
  train <- dataset[train_index,]
  test <- dataset[-train_index,]

  # train model
  lm_model <- lm(Profit~., data = train)
  #summary(lm_model)
  # predict train accuracy
  profit_pred_test <- predict(lm_model, newdate = test)
  train_mse <- mean((train$Profit - predict(lm_model, train)) ^ 2)
  test_mse <- mean((test$Profit - predict(lm_model, test)) ^ 2)

  # combine train_mse and test_mse and return it
  c(train_mse,test_mse)
}

# write function to do the run calculate_MSE iter times and calculate the mean
calculate_MSE_mean <- function(dataset, iter, percent){
    each_mse <- c(0, 0)
    for(j in 1:iter) {
      mse <- calculate_MSE(dataset, percent)
      each_mse <- each_mse + mse
    }
    each_mse / iter
}

# calculate the mse with different percent of train and test data range from 0.05 to 0.95
final_mse<- vector(, 3)
for(i in 0:18){

  percent <- 0.05 + i * 0.05
  each_mse <- calculate_MSE_mean(movies_numeric, iter = 100, percent = percent)

    each_mse <- c(percent, each_mse)

  final_mse <- rbind(final_mse, each_mse)

}

# remove the first placeholder row
final_mse <- final_mse[-1, ]
colnames(final_mse) <- c("train_percent", "train", "test")
#head(final_accuracy)
rownames(final_mse) <- NULL

final_mse <- as.data.frame(final_mse)
print(paste("the best mse for train set with only numeric variable is:", min(final_mse$train), sep = "
```

```
## [1] "the best mse for train set with only numeric variable is: 8854841074823548"
```

```r
print(paste("the best mse for test set with only numeric variable is:", min(final_mse$test), sep = " "))
```

```
## [1] "the best mse for test set with only numeric variable is: 10147885749491402"
```

```r
final_mse$train_percent <- factor(as.character(final_mse$train_percent))
head(final_mse)
```

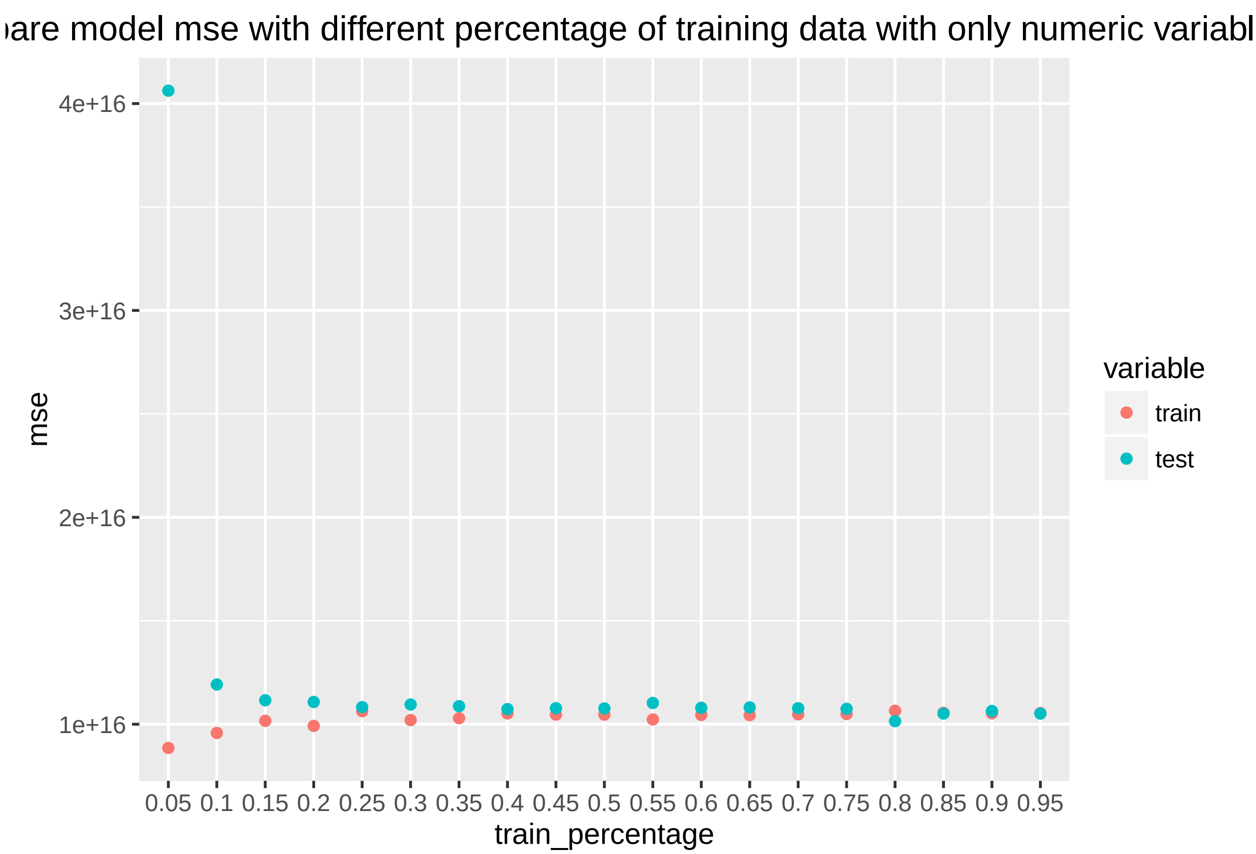
```
##   train_percent        train         test
## 1          0.05 8.854841e+15 4.062174e+16
## 2           0.1 9.578617e+15 1.192184e+16
## 3          0.15 1.016340e+16 1.115761e+16
## 4           0.2 9.919299e+15 1.107632e+16
## 5          0.25 1.063093e+16 1.082671e+16
## 6           0.3 1.020243e+16 1.095256e+16
```

```r
melt_mse <- melt(final_mse, id = "train_percent")

colnames(melt_mse) <- c("train_percentage", "variable", "mse")

ggplot(melt_mse,
       aes(x = train_percentage, y=mse, color = variable)) +
  geom_point() +

  ggtitle("Compare model mse with different percentage of training data with only numeric variables")
```



2. Try to improve the prediction quality in (1) as much as possible by adding feature transformations of the numeric variables. Explore both numeric transformations such as power transforms and non-numeric transformations of the numeric variables like binning (e.g.,is_budget_greater_than_3M). Explain which transformations you used and why you chose them. Graph the train and test MSE as a function of the train set size (averaged over 10 random data partitions as described above)?

The strategy is to create new varibles by taking log, square, cube of each numeric variables, run lm model with all created variables, then only select the variables which contribute significantly to the final model.

The lm model based on numeric and transformed numeric variables could get mse at 9.2e+15 for test data.

```r
log_val <- function(x) log10(x)
sqr_val <- function(x) x^2
cub_val <- function(x) x^3
func_list <- c(log_val, sqr_val, cub_val)

func_name <- c("log_", "sqr_", "cub_")
col_name <- colnames(movies_numeric)

# avoid the Profit column
col_name <- col_name[-length(col_name)]
# calculate the mse for lm model with any tranformation of the numeric varibles
# mse_original <- calculate_MSE_mean(movies_numeric, iter = 10, percent = 0.7)
# only select the test mse
# mse_original_test <- mse_original[2]
keeped_columns <- data.frame(matrix(NA, nrow = dim(movies_numeric)[1], ncol= 0))
keeped_column_names <- c()
for (i in seq_along(func_list)){
  # iterate through each transformation function
    func <- func_list[i]
   # iterate through each numeric column
     for (j in seq_along(col_name) ) {

       created_col_name <- paste(func_name[i], col_name[j], sep = "")
       created_col <- func_list[[i]](movies_numeric[, col_name[j]] + 0.01)
       # print(created_col_name)
       # add this created column to movies_numeric dataframe
       #movies_numeric[,created_col_name]<- func_list[[i]](movies_numeric[, col_name[j]] + 0.01)
        #print(head(movies_numeric))
       #mse <- calculate_MSE_mean(movies_numeric, iter = 10, percent = 0.7)
       #print(mse)
       #mse_test <- mse[2]
          #if((mse_original_test - mse_test)/mse_original_test > 0.05)
           #{ print(created_col_name)
        keeped_columns <- cbind(keeped_columns, created_col)
        keeped_column_names <- c(keeped_column_names, created_col_name)
          #   }
             # remove this newly created column before next iteration
             #movies_numeric <- movies_numeric[, 1:11]
          }
}

colnames(keeped_columns) <- keeped_column_names
#head(keeped_columns)
```

```r
# because we cubed each column and some variable will be extreme large, it is necessary to normalize th

# then we normalize the data

preObj <- preProcess(keeped_columns, method=c("center", "scale"))
keeped_columns<- predict(preObj, keeped_columns)
#keeped_columns_1 <- cbind(keeped_columns, Profit = movies_merged$Profit)
#lm_model_keep <- lm(Profit~., data = keeped_columns_1)
#summary(lm_model_keep)

# then we combine all these created columns with numeric data
movies_combined1 <-  cbind(movies_numeric, keeped_columns)
lm_model_1 <- lm(Profit~., data = movies_combined1)
summary(lm_model_1)
```

```
##
## Call:
## lm(formula = Profit ~ ., data = movies_combined1)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -444082085  -36046040   -4427347   22718331 1592472350
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)            9.862e+09  2.955e+09   3.337 0.000857 ***
## Runtime               -5.782e+06  2.489e+06  -2.323 0.020247 *
## Metascore              1.857e+06  5.728e+06   0.324 0.745780
## imdbRating            -5.816e+08  4.138e+08  -1.405 0.159989
## imdbVotes              5.554e+02  6.997e+01   7.938 2.79e-15 ***
## tomatoRating          -7.569e+08  3.004e+08  -2.520 0.011786 *
## tomatoFresh            1.121e+06  3.409e+05   3.288 0.001019 **
## tomatoRotten          -5.718e+05  4.850e+05  -1.179 0.238456
## tomatoUserMeter       -4.047e+06  2.199e+06  -1.840 0.065839 .
## tomatoUserRating      -3.992e+08  2.332e+08  -1.712 0.087037 .
## tomatoUserReviews      1.262e+02  1.076e+01  11.726  < 2e-16 ***
## log_Runtime            2.475e+07  9.686e+06   2.555 0.010670 *
## log_Metascore         -3.623e+06  1.911e+07  -0.190 0.849675
## log_imdbRating         1.770e+08  1.248e+08   1.419 0.156073
## log_imdbVotes         -3.755e+06  3.699e+06  -1.015 0.310059
## log_tomatoRating       2.243e+08  1.114e+08   2.014 0.044116 *
## log_tomatoFresh        6.991e+06  4.773e+06   1.465 0.143094
## log_tomatoRotten      -1.492e+07  4.327e+06  -3.447 0.000574 ***
## log_tomatoUserMeter    7.572e+06  4.722e+06   1.603 0.108962
## log_tomatoUserRating   1.304e+07  1.023e+07   1.274 0.202669
## log_tomatoUserReviews  5.567e+06  3.156e+06   1.764 0.077828 .
## sqr_Runtime            1.039e+08  7.366e+07   1.410 0.158534
## sqr_Metascore         -8.529e+07  1.540e+08  -0.554 0.579650
## sqr_imdbRating         8.286e+08  5.589e+08   1.483 0.138266
## sqr_imdbVotes          5.544e+06  1.465e+07   0.378 0.705128
## sqr_tomatoRating       1.528e+09  5.134e+08   2.976 0.002941 **
## sqr_tomatoFresh       -9.436e+07  3.230e+07  -2.921 0.003512 **
## sqr_tomatoRotten       1.368e+07  2.562e+07   0.534 0.593458
```

```
## sqr_tomatoUserMeter    1.588e+08  8.523e+07   1.863 0.062547 .
## sqr_tomatoUserRating   2.954e+08  2.077e+08   1.423 0.154940
## sqr_tomatoUserReviews -1.241e+09  9.751e+07 -12.722  < 2e-16 ***
## cub_Runtime           -1.548e+07  3.587e+07  -0.431 0.666202
## cub_Metascore          7.068e+07  7.844e+07   0.901 0.367608
## cub_imdbRating        -3.918e+08  2.310e+08  -1.696 0.090004 .
## cub_imdbVotes         -1.373e+07  9.170e+06  -1.497 0.134464
## cub_tomatoRating      -7.804e+08  2.187e+08  -3.568 0.000365 ***
## cub_tomatoFresh        7.838e+07  1.762e+07   4.447 8.98e-06 ***
## cub_tomatoRotten       1.323e+07  1.342e+07   0.985 0.324496
## cub_tomatoUserMeter   -1.281e+08  4.818e+07  -2.658 0.007888 **
## cub_tomatoUserRating  -5.896e+07  1.064e+08  -0.554 0.579636
## cub_tomatoUserReviews  8.474e+08  7.381e+07  11.480  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 95160000 on 3291 degrees of freedom
## Multiple R-squared:  0.5702, Adjusted R-squared:  0.5649
## F-statistic: 109.1 on 40 and 3291 DF,  p-value: < 2.2e-16
```

```r
# from the summary of the model we only keep the most significat columns

created_numeric <- keeped_columns[,c("log_tomatoRotten","sqr_tomatoRating", "sqr_tomatoFresh", "sqr_toma
movies_combined2 <-  cbind(movies_numeric, created_numeric)


final_mse<- vector(, 3)
for(i in 0:18){

  percent <- 0.05 + i * 0.05
  each_mse <- calculate_MSE_mean(movies_combined2, iter = 100, percent = percent)

  each_mse <- c(percent, each_mse)

  final_mse <- rbind(final_mse, each_mse)

}

# remove the first placeholder row
final_mse <- final_mse[-1, ]
colnames(final_mse) <- c("train_percent", "train", "test")
#head(final_accuracy)
rownames(final_mse) <- NULL

final_mse <- as.data.frame(final_mse)

print(paste("the best mse for train set with only numeric variable is:", min(final_mse$train), sep = "
```

```
## [1] "the best mse for train set with only numeric variable is: 7703903222041098"
```

```r
print(paste("the best mse for test set with only numeric variable is:", min(final_mse$test), sep = " "))
```

```
## [1] "the best mse for test set with only numeric variable is: 9165599030183038"
```

```r
final_mse$train_percent <- factor(as.character(final_mse$train_percent))
head(final_mse)
```

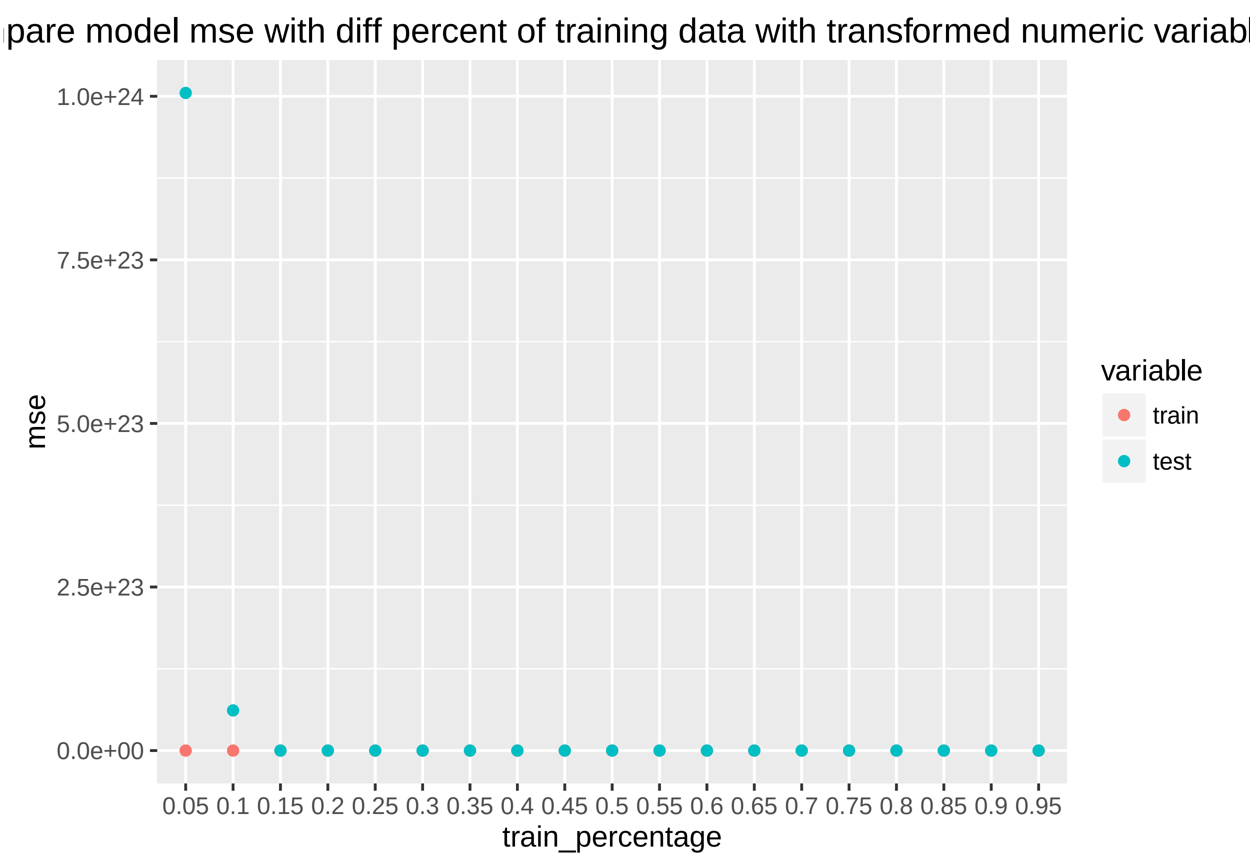
```
##   train_percent        train         test
## 1          0.05 7.703903e+15 1.005110e+24
## 2           0.1 7.991587e+15 6.136566e+22
## 3          0.15 8.345994e+15 2.219761e+16
## 4           0.2 9.268905e+15 1.077984e+16
## 5          0.25 9.154711e+15 1.021068e+16
## 6           0.3 9.496776e+15 9.934684e+15
```

```r
melt_mse <- melt(final_mse, id = "train_percent")

colnames(melt_mse) <- c("train_percentage", "variable", "mse")

ggplot(melt_mse,
       aes(x = train_percentage, y=mse, color = variable)) +
  geom_point() +

  ggtitle("Compare model mse with diff percent of training data with transformed numeric variables")
```



3. Write code that featurizes genre (can use code from Part-I), actors, directors, and other categorical variables. Explain how you encoded the variables into features.

for categorical variables genre, actors, rate, director, writer, language, country, awards, and production, Website, only rate and production are single variable at each row and we can directly convert them to dummy varibles.

for categorical variables genre, actors, director, writer, language, country, they all contains multiple varibles for each row. Therefore we could not use one-hot encoding of dummy varibles to convert them into numerical varible. The strategy is to find the terms for each row by converting to document term matrix, find the most abundance terms and only choose these terms to create dummy varible.

for the Plot column, basically it is just free text. I use topic modeling to automatically classify sets of documents into themes. The algorithm that I used is Latent Dirichlet Allocation(LDA). The basic assumption behind LDA is that each of the documents in a collection consist of a mixture of collection-wide topics. However, in reality we observe only documents and words, not topics – the latter are part of the hidden (or latent) structure of documents. The aim is to infer the latent topic structure given the words and document. LDA does this by recreating the documents in the corpus by adjusting the relative importance of topics in documents and words in topics iteratively.

```
##            n.a     luc_besson     john_logan david_s._goyer      ethan_coen
##             68             20             15             12              12
##      joel_coen  marlon_wayans    woody_allen      adam_mckay  alex_kurtzman
##             12             12             12             11              11
```

```
## [1] 3332    8
```

```
##     drama    comedy    action   romance adventure     crime  thriller
##      1627      1262       719       577       569       568       525
##    horror   mystery   fantasy
##       353       282       242
```

```
## [1] 3332    7
```

```
##      robert_de_niro       mark_wahlberg         owen_wilson
##                  29                  25                  23
##   samuel_l._jackson        adam_sandler         ben_stiller
##                  23                  22                  22
##         johnny_depp          matt_damon        bruce_willis
##                  22                  22                  21
##      george_clooney       gerard_butler          jack_black
##                  20                  20                  20
##       jason_statham matthew_mcconaughey        nicolas_cage
##                  20                  20                  20
```

```
## [1] 3332    8
```

```
## steven_soderbergh    clint_eastwood      ridley_scott       woody_allen
##                15                13                12                12
##               n.a        shawn_levy  steven_spielberg        ethan_coen
##                11                11                11                10
##         joel_coen  robert_rodriguez         ron_howard    peter_farrelly
##                10                10                10                 9
##     adam_shankman     antoine_fuqua    bobby_farrelly
##                 8                 8                 8
```

```
## [1] 3332    7
```

```
##    english    spanish     french     german    russian    italian 
##       3232        323        290        158        120        117 
##   japanese   mandarin     arabic      hindi  cantonese      latin 
##         78         67         65         48         39         34 
##  ukrainian portuguese     hebrew 
##         34         31         29
```

```
## [1] 3332    9
```

```
##       usa   germany    canada    france australia     spain     india 
##      2909       335       274       257        94        67        55 
##     italy     japan     china   ireland     south      hong      kong 
##        54        50        40        39        34        32        32 
##       new 
##        24
```

```
## [1] 3332    6
```

```
##   0%  25%  50%  75% 100% 
##    0    1    5   16  548
```

```
## [1] 3332    4
```

```
## 
##         G       N/A     NC-17 NOT RATED        PG     PG-13         R 
##        58       246         3        74       413      1132      1371 
##     TV-14      TV-G     TV-PG   UNRATED 
##         1         3         3        28
```

```
##    rate_PG rate_PG_13     rate_R 
##        413       1132       1371
```

```
## [1] 3332    3
```

```
## [1] 562
```

```
## 
##                    N/A  Warner Bros. Pictures     Universal Pictures 
##                    245                    213                    202 
##       20th Century Fox     Paramount Pictures          Sony Pictures 
##                    199                    139                    119 
## Sony Pictures Classics        New Line Cinema   Walt Disney Pictures 
##                     85                     78                     64 
##          Miramax Films     Columbia Pictures         Focus Features 
##                     59                     57                     57 
##        Lionsgate Films  The Weinstein Company           Warner Bros. 
##                     50                     45                     45
```

```
##    pro_warner pro_universal  pro_20th_cen pro_paramount      pro_sony 
##           213           202           199           139           119
```

```
## [1] 3332    5

## [1]  3332 13752

## [1] 34

## [1] "The size of the vocabulary is :  13752"

## [1] "The top frequencied words: "

##   find   life    one    get    new   year   will friend famili   live
##    862    825    740    665    663    644    603    592    579    549

## [1] 3332    1

##      Topic 1  Topic 2 Topic 3 Topic 4 Topic 5
## [1,] "world"  "two"   "get"   "new"   "friend"
## [2,] "must"   "man"   "life"  "love"  "famili"
## [3,] "find"   "take"  "will"  "stori" "year"
## [4,] "set"    "one"   "can"   "young" "find"
## [5,] "group"  "team"  "one"   "life"  "father"
## [6,] "secret" "kill"  "time"  "becom" "old"

## [1] 3332    5
```

4. Use linear regression to predict profit based on all available non-numeric variables (using the transformations in (3). Graph the train and test MSE as a function of the train set size (averaged over 10 random data partitions as described above)?

First I build a lm model based on all the created categorical variables from question 3. Then select the significant variables. The lm model based on selecyted transformed categorical variables could get mse at 1.3e+16 for test data.

```r
# combine all the variables created in question 3
movies_categorical <- cbind(sub_writer_df, sub_genre_df, sub_actors_df, sub_director_df, sub_language_d

dim(movies_categorical) # 3332 X 63
```

```
## [1] 3332   63
```

```r
lm_model_2 <- lm(Profit~., data = movies_categorical)
summary(lm_model_2)
```

```
##
## Call:
## lm(formula = Profit ~ ., data = movies_categorical)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -370917646  -54638932   -6830492   31375190 2043801003
```

```
##
## Coefficients: (4 not defined because of singularities)
##                            Estimate Std. Error t value Pr(>|t|)    
## (Intercept)               118684486   33968802   3.494 0.000482 ***
## writer_n.a                -17022360   16437933  -1.036 0.300487    
## writer_luc_besson          39295793   23371992   1.681 0.092796 .  
## writer_john_logan          14721935   24761166   0.595 0.552180    
## writer_david_s._goyer     102633037   28029107   3.662 0.000255 ***
## writer_ethan_coen         -16393233   34344996  -0.477 0.633173    
## writer_joel_coen                 NA         NA      NA       NA    
## writer_marlon_wayans       38648477   27763306   1.392 0.163996    
## writer_woody_allen          6165704   34465560   0.179 0.858031    
## drama                     -40717474    4885648  -8.334  < 2e-16 ***
## comedy                     -5511601    5070136  -1.087 0.277085    
## action                     24042773    6123764   3.926 8.81e-05 ***
## romance                     1534035    5906624   0.260 0.795100    
## adventure                  68496060    6788016  10.091  < 2e-16 ***
## crime                      -5411301    6319268  -0.856 0.391885    
## thriller                    9598797    6380240   1.504 0.132560    
## actor_robert_de_niro       23994210   22142962   1.084 0.278620    
## actor_mark_wahlberg        56174721   23875008   2.353 0.018688 *  
## actor_owen_wilson         -23189372   25607385  -0.906 0.365228    
## actor_samuel_l._jackson    18597783   24776099   0.751 0.452927    
## actor_adam_sandler         21588192   25825409   0.836 0.403256    
## actor_ben_stiller          67835554   26436963   2.566 0.010334 *  
## actor_johnny_depp          89459792   25278927   3.539 0.000407 ***
## actor_matt_damon          -16372757   25497585  -0.642 0.520834    
## directo_steven_soderbergh  32831640   30834215   1.065 0.287054    
## directo_clint_eastwood     -1490747   33076057  -0.045 0.964054    
## directo_ridley_scott       37496284   34454063   1.088 0.276544    
## directo_woody_allen              NA         NA      NA       NA    
## directo_n.a                97256170   38260751   2.542 0.011070 *  
## directo_shawn_levy         30606752   36792166   0.832 0.405536    
## directo_steven_spielberg   60980253   35990783   1.694 0.090298 .  
## english                    15990290   12835062   1.246 0.212916    
## spanish                     9894979    7121917   1.389 0.164815    
## french                     17151220    7817260   2.194 0.028304 *  
## german                     -7932228   10077900  -0.787 0.431285    
## russian                     4292305   11304547   0.380 0.704195    
## italian                    16194774   11585933   1.398 0.162269    
## japanese                    3054709   13511087   0.226 0.821146    
## mandarin                  -21760269   14944809  -1.456 0.145477    
## arabic                      7060430   15180581   0.465 0.641894    
## usa                        36429596    7323563   4.974 6.89e-07 ***
## germany                   -16404864    7034195  -2.332 0.019753 *  
## canada                    -13241070    7586039  -1.745 0.081000 .  
## france                    -14010904    8440279  -1.660 0.097009 .  
## australia                 -40406135   12512973  -3.229 0.001254 ** 
## spain                     -32923172   15153973  -2.173 0.029884 *  
## award_l_1                -144375106    6126556 -23.565  < 2e-16 ***
## award_l_2                -128216447    6154874 -20.832  < 2e-16 ***
## award_l_3                -100156811    6022918 -16.629  < 2e-16 ***
## award_l_4                        NA         NA      NA       NA    
## rate_PG                     6173040    9032283   0.683 0.494376    
```

```
## rate_PG_13                   12062110    7779533   1.550 0.121120
## rate_R                      -22546510    7430920  -3.034 0.002431 **
## pro_warner                   28028746    8738876   3.207 0.001352 **
## pro_universal                29838782    8842308   3.375 0.000748 ***
## pro_20th_cen                 32196888    9010177   3.573 0.000357 ***
## pro_paramount                -2071921   10575174  -0.196 0.844683
## pro_sony                     18733962   11386706   1.645 0.100015
## top_1                       105064500   49194204   2.136 0.032777 *
## top_2                       -55050524   52764283  -1.043 0.296873
## top_3                        12027916   51125391   0.235 0.814019
## top_4                      -107990521   50137047  -2.154 0.031320 *
## top_5                              NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 116800000 on 3273 degrees of freedom
## Multiple R-squared:  0.3561, Adjusted R-squared:  0.3447
## F-statistic: 31.21 on 58 and 3273 DF,  p-value: < 2.2e-16
```

```r
# then I selected all the columns which are significant on the lm model.
created_categorical <- movies_categorical[,c("writer_david_s._goyer","drama", "action", "adventure", "a

final_mse<- vector(, 3)
for(i in 0:18){

  percent <- 0.05 + i * 0.05
  each_mse <- calculate_MSE_mean(created_categorical, iter = 100, percent = percent)

  each_mse <- c(percent, each_mse)

  final_mse <- rbind(final_mse, each_mse)

}

# remove the first placeholder row
final_mse <- final_mse[-1, ]
colnames(final_mse) <- c("train_percent", "train", "test")
#head(final_accuracy)
rownames(final_mse) <- NULL

final_mse <- as.data.frame(final_mse)
print(paste("the best mse for train set with transformed categorical variable is:", min(final_mse$train
```

```
## [1] "the best mse for train set with transformed categorical variable is: 11904204506913200"
```

```r
print(paste("the best mse for test set with only transformed categorical variable is:", min(final_mse$t
```

```
## [1] "the best mse for test set with only transformed categorical variable is: 13834598941554508"
```

```r
final_mse$train_percent <- factor(as.character(final_mse$train_percent))
head(final_mse)
```

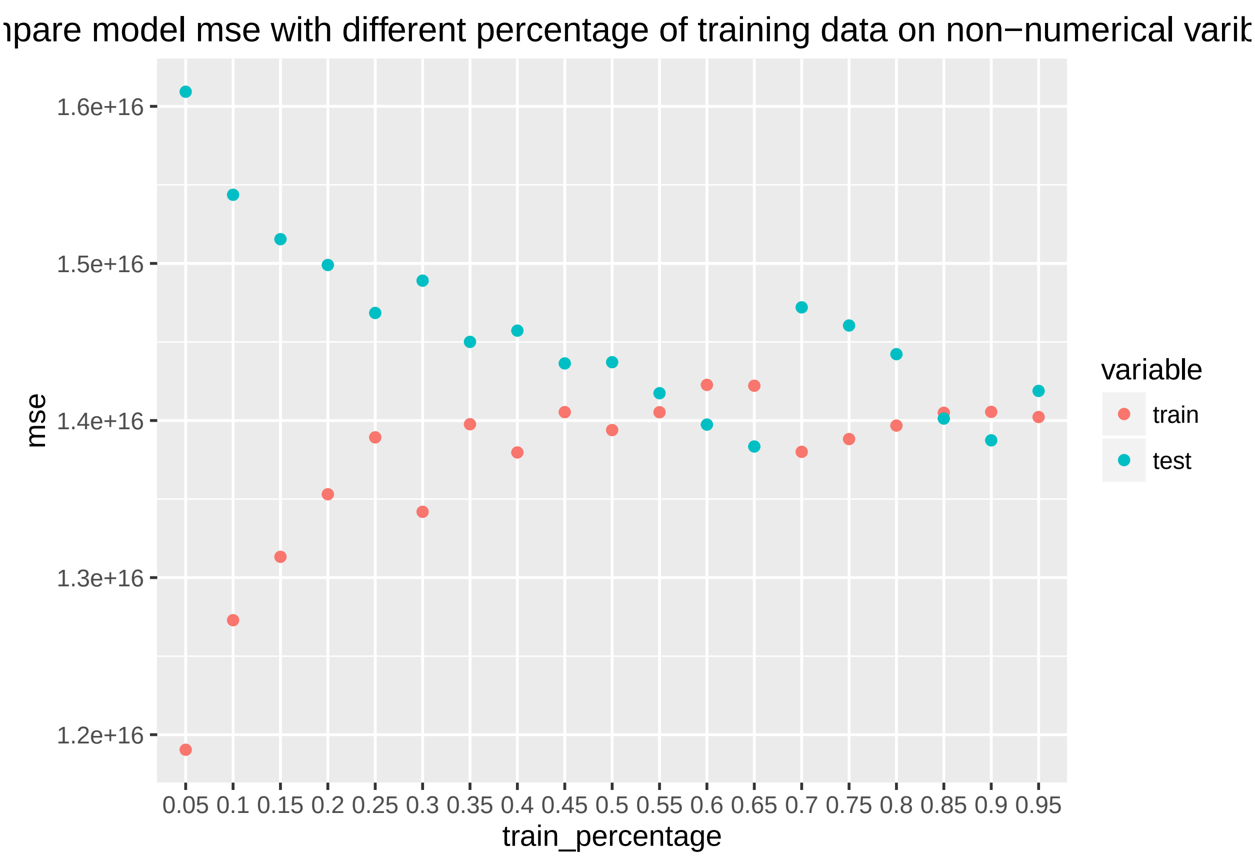
```
##   train_percent        train         test
## 1          0.05 1.190420e+16 1.609294e+16
## 2           0.1 1.272887e+16 1.543696e+16
## 3          0.15 1.313232e+16 1.515435e+16
## 4           0.2 1.353055e+16 1.499003e+16
## 5          0.25 1.389277e+16 1.468467e+16
## 6           0.3 1.341886e+16 1.489021e+16
```

```r
melt_mse <- melt(final_mse, id = "train_percent")

colnames(melt_mse) <- c("train_percentage", "variable", "mse")

ggplot(melt_mse,
       aes(x = train_percentage, y=mse, color = variable)) +
  geom_point() +

  ggtitle("Compare model mse with different percentage of training data on non-numerical varible")
```



5. Try to improve the prediction quality in (1) as much as possible by using both numeric and non- numeric variables as well as creating additional transformed features including interaction features (for example is_genre_comedy x is_budget_greater_than_3M). Explain which transformations you used and why you chose them. Graph the train and test MSE as a function of the train set size (averaged over 10 random data partitions as described above)?

I first combine 1) all the numeric variables from original data, 2) transfromed numeric varibles from question 2, and 3) transformed categorical variables from question 3 and 4. Then build a lm model to remove the non-significant variables. Based on these variables, the mse of this model for test data can reach 8.9e+15. Then I create variables for all possible interactions btween numerical variables and categorical variables. Then build a lm model to remove the non-significant variables. Based on these variables containing selected interaction variables, the mse for test data can reach 8.7e+15.

```
movies_combined2$Profit <- NULL
movies_combined3 <- cbind(movies_combined2, created_categorical)
lm_model_3 <- lm(Profit~., data = movies_combined3)
summary(lm_model_3)
```

```
##
## Call:
## lm(formula = Profit ~ ., data = movies_combined3)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -450145864  -36438706   -2529328   27573206 1633675580
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)            3.490e+08  2.119e+08   1.647 0.099673 .
## Runtime               -7.931e+04  1.055e+05  -0.752 0.452387
## Metascore              5.240e+05  2.772e+05   1.891 0.058779 .
## imdbRating            -1.484e+07  2.798e+06  -5.303 1.22e-07 ***
## imdbVotes              4.163e+02  2.335e+01  17.829  < 2e-16 ***
## tomatoRating          -1.080e+08  3.797e+07  -2.845 0.004463 **
## tomatoFresh            5.869e+05  2.920e+05   2.010 0.044546 *
## tomatoRotten           4.042e+05  9.671e+04   4.180 3.00e-05 ***
## tomatoUserMeter       -8.610e+05  3.527e+05  -2.441 0.014680 *
## tomatoUserRating       9.450e+07  8.708e+06  10.852  < 2e-16 ***
## tomatoUserReviews      1.130e+02  9.528e+00  11.860  < 2e-16 ***
## log_tomatoRotten      -1.882e+07  3.707e+06  -5.076 4.06e-07 ***
## sqr_tomatoRating       4.110e+08  1.127e+08   3.647 0.000269 ***
## sqr_tomatoFresh       -5.558e+07  2.826e+07  -1.967 0.049295 *
## sqr_tomatoUserReviews -1.137e+09  9.057e+07 -12.558  < 2e-16 ***
## cub_tomatoRating      -3.072e+08  6.219e+07  -4.939 8.23e-07 ***
## cub_tomatoFresh        6.296e+07  1.556e+07   4.045 5.35e-05 ***
## cub_tomatoUserMeter   -1.096e+07  5.482e+06  -1.999 0.045696 *
## cub_tomatoUserReviews  7.865e+08  7.013e+07  11.215  < 2e-16 ***
## writer_david_s._goyer -4.025e+07  2.265e+07  -1.777 0.075659 .
## drama                 -2.193e+07  3.897e+06  -5.626 2.00e-08 ***
## action                -9.547e+06  4.510e+06  -2.117 0.034371 *
## adventure              4.909e+07  5.066e+06   9.690  < 2e-16 ***
## actor_johnny_depp      2.419e+07  2.026e+07   1.194 0.232637
## usa                    6.536e+06  5.252e+06   1.244 0.213413
## award_l_1             -5.255e+07  6.832e+06  -7.692 1.91e-14 ***
## award_l_2             -5.091e+07  6.518e+06  -7.812 7.53e-15 ***
## award_l_3             -4.463e+07  5.881e+06  -7.590 4.15e-14 ***
## pro_warner            -1.231e+07  6.945e+06  -1.772 0.076475 .
## pro_universal          7.062e+06  7.038e+06   1.003 0.315730
## pro_20th_cen           2.121e+07  7.035e+06   3.015 0.002590 **
## top_1                  6.657e+07  3.014e+07   2.209 0.027266 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 93420000 on 3300 degrees of freedom
## Multiple R-squared:  0.5845, Adjusted R-squared:  0.5806
## F-statistic: 149.8 on 31 and 3300 DF,  p-value: < 2.2e-16
```

```
# from model summary, we can see some varialbes did not contribute, we could drop it.
movies_combined4 <- subset(movies_combined3, select = -c(actor_johnny_depp, usa, pro_universal))

lm_model_4 <- lm(Profit~., data = movies_combined4)
summary(lm_model_4)
```

```
##
## Call:
## lm(formula = Profit ~ ., data = movies_combined4)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -445524581  -36346596   -2841499   27180539 1631033598
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)            3.410e+08  2.119e+08   1.609 0.107641
## Runtime               -7.489e+04  1.053e+05  -0.711 0.477187
## Metascore              5.120e+05  2.771e+05   1.848 0.064732 .
## imdbRating            -1.508e+07  2.781e+06  -5.423 6.27e-08 ***
## imdbVotes              4.175e+02  2.331e+01  17.911  < 2e-16 ***
## tomatoRating          -1.058e+08  3.794e+07  -2.789 0.005314 **
## tomatoFresh            5.961e+05  2.920e+05   2.042 0.041274 *
## tomatoRotten           4.267e+05  9.605e+04   4.443 9.17e-06 ***
## tomatoUserMeter       -8.349e+05  3.522e+05  -2.371 0.017821 *
## tomatoUserRating       9.426e+07  8.706e+06  10.827  < 2e-16 ***
## tomatoUserReviews      1.143e+02  9.501e+00  12.031  < 2e-16 ***
## log_tomatoRotten      -1.918e+07  3.703e+06  -5.180 2.35e-07 ***
## sqr_tomatoRating       4.043e+08  1.126e+08   3.591 0.000334 ***
## sqr_tomatoFresh       -5.249e+07  2.822e+07  -1.860 0.062919 .
## sqr_tomatoUserReviews -1.143e+09  9.052e+07 -12.622  < 2e-16 ***
## cub_tomatoRating      -3.042e+08  6.213e+07  -4.896 1.02e-06 ***
## cub_tomatoFresh        6.068e+07  1.552e+07   3.909 9.44e-05 ***
## cub_tomatoUserMeter   -1.123e+07  5.481e+06  -2.049 0.040535 *
## cub_tomatoUserReviews  7.877e+08  7.013e+07  11.232  < 2e-16 ***
## writer_david_s._goyer -4.159e+07  2.264e+07  -1.837 0.066283 .
## drama                 -2.209e+07  3.888e+06  -5.681 1.45e-08 ***
## action                -9.901e+06  4.503e+06  -2.199 0.027962 *
## adventure              4.910e+07  5.063e+06   9.698  < 2e-16 ***
## award_l_1             -5.146e+07  6.790e+06  -7.578 4.54e-14 ***
## award_l_2             -4.988e+07  6.470e+06  -7.709 1.66e-14 ***
## award_l_3             -4.376e+07  5.854e+06  -7.475 9.83e-14 ***
## pro_warner            -1.295e+07  6.887e+06  -1.881 0.060120 .
## pro_20th_cen           2.061e+07  7.000e+06   2.945 0.003257 **
## top_1                  6.541e+07  3.014e+07   2.170 0.030070 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 93440000 on 3303 degrees of freedom
```

```
## Multiple R-squared:  0.584,  Adjusted R-squared:  0.5805
## F-statistic: 165.6 on 28 and 3303 DF,  p-value: < 2.2e-16
```

```r
final_mse<- vector(, 3)
for(i in 0:18){

  percent <- 0.05 + i * 0.05
  each_mse <- calculate_MSE_mean( movies_combined4, iter = 100, percent = percent)

  each_mse <- c(percent, each_mse)

  final_mse <- rbind(final_mse, each_mse)

}

# remove the first placeholder row
final_mse <- final_mse[-1, ]
colnames(final_mse) <- c("train_percent", "train", "test")
#head(final_accuracy)
rownames(final_mse) <- NULL

final_mse <- as.data.frame(final_mse)
print(paste("the best mse for train set with combined numeric and transformed categorical variable is:"
```

```
## [1] "the best mse for train set with combined numeric and transformed categorical variable is: 628109
```

```r
print(paste("the best mse for test set with combined numeric and  transformed categorical variable is:"
```

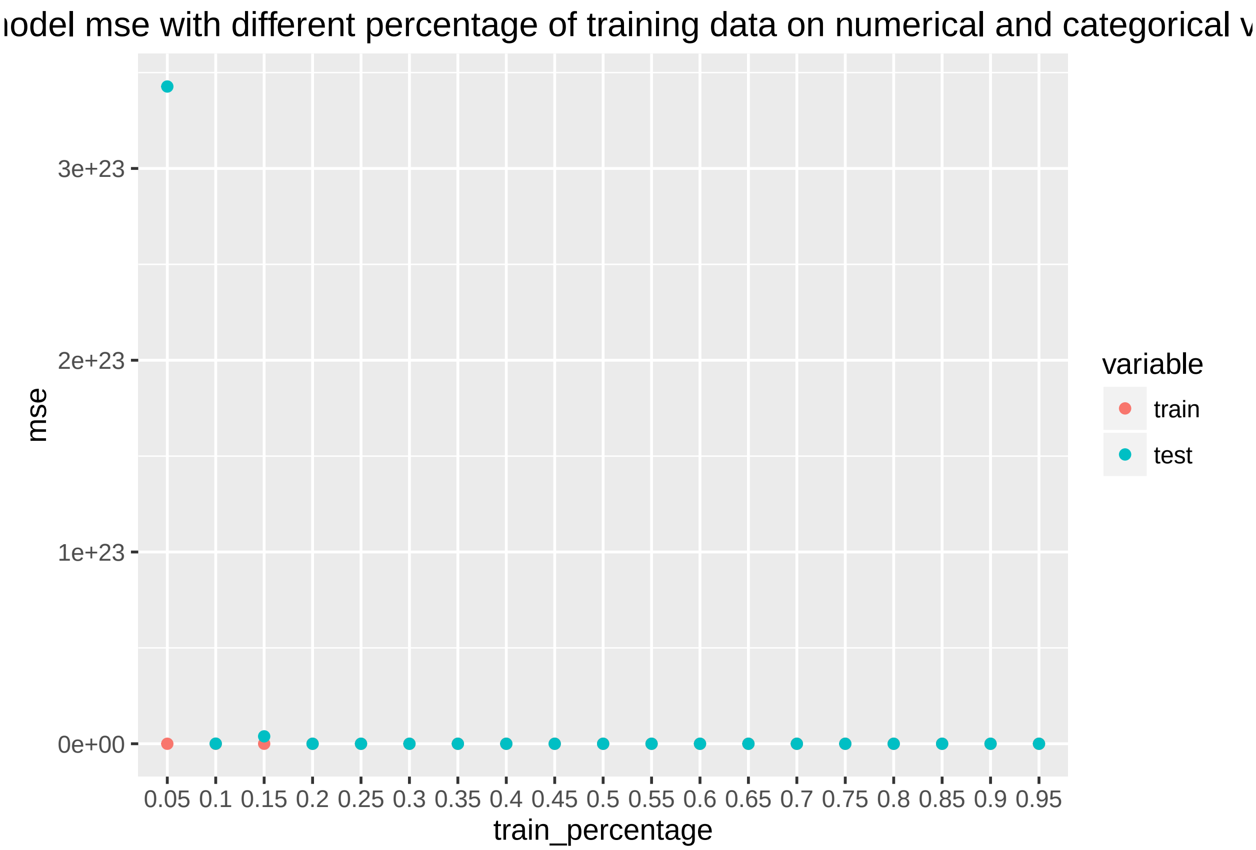
```
## [1] "the best mse for test set with combined numeric and  transformed categorical variable is: 815873
```

```r
final_mse$train_percent <- factor(as.character(final_mse$train_percent))
#head(final_mse)
melt_mse <- melt(final_mse, id = "train_percent")

colnames(melt_mse) <- c("train_percentage", "variable", "mse")

ggplot(melt_mse,
       aes(x = train_percentage, y=mse, color = variable)) +
  geom_point() +

  ggtitle("Compare model mse with different percentage of training data on numerical and categorical va
```




```
# then I tried to further improve the mse by creating the interaction between numeric variables and cat
numeric_v <- 1:18
categorical_v <- 19:28
inter_data <- data.frame(matrix(0, nrow = dim(movies_numeric)[1], ncol= 0))
inter_name <- c()
for(i in numeric_v){
  name1 <- colnames(movies_combined4)[i]

  for(j in categorical_v){
    name2 <- colnames(movies_combined4)[j]
    new_name <- paste("inter", name1, name2, sep = "_")
    inter_name <- c(inter_name, new_name)
    new_col = movies_combined4[i] * movies_combined4[j]
    inter_data <- cbind(inter_data, new_col)
  }
}
colnames(inter_data) <- inter_name

movies_combined5 <- cbind(movies_combined4, inter_data)
# run a lm model on it
lm_model_5 <- lm(Profit~., data = movies_combined5)
# then only select the significant columns
movies_combined6 <- movies_combined5[, (summary(lm_model_5)$coefficients[, 4] < 0.05)]
movies_combined6 <- cbind(movies_combined6, Profit = movies_merged$Profit)


final_mse<- vector(, 3)
```

```r
for(i in 0:18){

  percent <- 0.05 + i * 0.05
  each_mse <- calculate_MSE_mean( movies_combined6, iter = 100, percent = percent)

  each_mse <- c(percent, each_mse)

  final_mse <- rbind(final_mse, each_mse)

}

# remove the first placeholder row
final_mse <- final_mse[-1, ]
colnames(final_mse) <- c("train_percent", "train", "test")
#head(final_accuracy)
rownames(final_mse) <- NULL

final_mse <- as.data.frame(final_mse)
print(paste("the best mse for train set with combined numeric and transformed categorical variable is:"
```

```
## [1] "the best mse for train set with combined numeric and transformed categorical variable is: 310043
```

```r
print(paste("the best mse for test set with combined numeric and  transformed categorical variable is:"
```

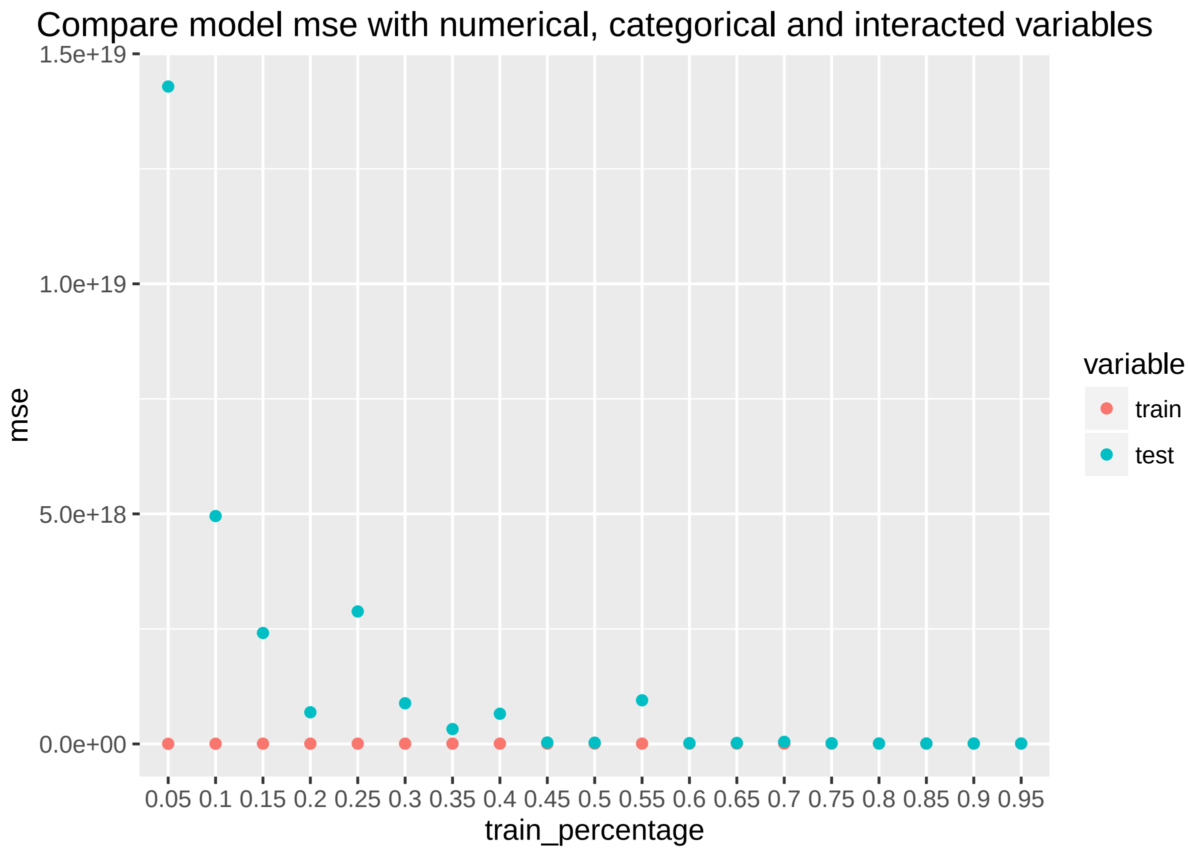
```
## [1] "the best mse for test set with combined numeric and  transformed categorical variable is: 930299
```

```r
final_mse$train_percent <- factor(as.character(final_mse$train_percent))
#head(final_mse)
melt_mse <- melt(final_mse, id = "train_percent")

colnames(melt_mse) <- c("train_percentage", "variable", "mse")

ggplot(melt_mse,
       aes(x = train_percentage, y=mse, color = variable)) +
  geom_point() +

  ggtitle("Compare model mse with numerical, categorical and interacted variables")
```


Compare model mse with numerical, categorical and interacted variables
1.5e+19
1.0e+19
5.0e+18
0.0e+00
mse
0.05 0.1 0.15 0.2 0.25 0.3 0.35 0.4 0.45 0.5 0.55 0.6 0.65 0.7 0.75 0.8 0.85 0.9 0.95
train_percentage
variable
train
test