Models Analysis of Car information from eBay

Kaiwen Wang, Yuhua Ding

12/10/2019

1. Introduction

The aim of this project is to build and improve Machine Learning models that can predict prices of cars listed on eBay.com. We scrapped 200 pages of car listings from eBay, then reformatted and transformed the obtained dataset. In order to obtain the best prediction we tried to fit different models on the data and chose the best model that offers the lowest Root Mean Squared Error for the testing data - the Random Forest model and we will elaborate more on in the following sections.

2. Data Acquisition

The data was scraped from eBay car website. We chose eBay because it has a comprehensive used car market information across the nation and at the same time very user-friendly for web-scrapping. After we scraped and clean the 10050 posts (200 pages worth of data, or the maximum amount our scrapper would allow) on the eBay, we got the eBay car dataset that contains 19 columns and 10050 rows. The 19 columns are car-related parameters including url,price, year, mileage, brand, model, body type, drive type, engine, fuel type, transmission, color, condition, cylinder, sale by, options, title, vehicle title, and warranty. During the scraping process pacakge BeautifulSoup and xpath were implemented. Scraping details are included in file *DataMining_DataScrape.ipynb*. Overall, the quality of data we obtained from eBay is not steady - there had been a lot of entries with parameters that is either miskeyed or intentionally entered incorrectly. Some of the entries we observed are simply counterintuitive. For instance, a close-to-new Mercedez-benz sedan won't be on sale for $500, and a Toyota shouldn't worth 80k USD after a million miles (not sure if a car can run a million miles anyway). Therefore, we decided to remove those entries to ensure better prediction in data processing stage.

3. Data Processing

3.1 load pacakges

```
library(readr)
library(ggplot2)
library(stringr)
library(dplyr)
library(caret)
library(randomForest)
library(gbm)
```

3.2 import data

```
setwd("D:/QMSS_Fall_2019/5058_Data_Mining/Final_project_sample")
df <- read.csv("project_info.csv", header = T, na.strings = c("", "NA"))
df <- df %>% rename(Fuel_type = "ï..Fuel_type")
```

3.3 clean variables

clean variable price: we firsted transformed the original variable `price`, which was a string variable to `pricex`, a numerical variable, then filled the NAs with the mean price. The mean price of all cars in our dataset was $40,395.00. Some of the listings prices are extreme outliers (from a few hundred dollars to half a million). We ran our initial regressions with those outliers and the model performance was far from ideal. Additionally, those extreme outliers caused problem in training/testing set partitioning - when those outliers are assigned to training or testing based on different random seeding, the testing set prediction performance varies a lot. Therefore, we decided to remove 2.5% of data at each wing of the price distribution as outliers. After the operation, we have 9562 observations left. That is a definitely compromise, but allowed us to produce far better predictions.

We also generated a new variable `lnrpicex` as the logarithm transformed `pricex` to try models in log form, since the prices are normally large numbers and log transformed prices may have a less-skewed distribution.

```
df$pricex = as.numeric(gsub("[\\US $,]", "", df$price))
```

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

```
df$pricex[is.na(df$pricex)]<-mean(df$pricex, na.rm = TRUE) #with mean 40395

quantile(df$pricex, c(0.025, 0.975))
```

```
##     2.5%    97.5%
##   3500.0 164998.9
```

```
df <- df %>% filter(pricex >= 3500, pricex <= 164999)
nrow(df)
```

```
## [1] 9562
```

```
df$lnpricex <- log(df$pricex)
```

clean variable year: there was one observation had irregular year value, so we manually fixed that.

```
df$year[df$year == 91] = "1991"
```

In the following chunks, we generated a lot of dummy variables for many non-quantifiable car parameters including fuel type, body type or make. Before generating the dummy variables for each type, we filled the NAs with "other" to avoid missingness in the dummy variables. We considered using car model as predictor but since the number of different car models in out model is in thousands and many of the models are misentries, utilizing the `model` would be unrealistic.

clean variable fuel type

```
df$Fuel_type <- tolower(as.character(df$Fuel_type))
df$Fuel_type[is.na(df$Fuel_type)]<- "other"
df$gas <- if_else(str_detect(df$Fuel_type, regex(".*gas.*")) == TRUE, 1, 0)
df$diesel <- if_else(str_detect(df$Fuel_type, regex(".*diesel.*")) == TRUE, 1, 0)
df$flex <- if_else(str_detect(df$Fuel_type, regex(".*flex.*")) == TRUE, 1, 0)
df$electric <- if_else(str_detect(df$Fuel_type, regex(".*electric.*")) == TRUE, 1, 0)
```

```
df$body_type = tolower(df$body_type)
new_df <- df
new_df$body_type <- tolower(new_df$body_type)
new_df$body_type[is.na(new_df$body_type)]<-'other'
new_df$sedan <- if_else(str_detect(new_df$body_type, regex(".*sedan.*")) == TRUE, 1, 0)
new_df$coupe <- if_else(str_detect(new_df$body_type, regex(".*coupe.*|.*2.*dr.*")) == TRUE, 1, 0)
new_df$hatchback <- if_else(str_detect(new_df$body_type, regex(".*hatchback.*")) == TRUE, 1, 0)
new_df$suv <- if_else(str_detect(new_df$body_type, regex(".*sport uti.*|.*suv.*")) == TRUE, 1, 0)
new_df$convertible <- if_else(str_detect(new_df$body_type, regex(".*convert.*")) == TRUE, 1, 0)
new_df$van <- if_else(str_detect(new_df$body_type, regex(".*van.*")) == TRUE &
                        str_detect(new_df$body_type, regex(".*avant.*")) == FALSE, 1, 0)
new_df$pickup <- if_else(str_detect(new_df$body_type, regex(".*pickup.*")) == TRUE, 1, 0)
new_df$wagon <- if_else(str_detect(new_df$body_type, regex(".*wagon.*")) == TRUE, 1, 0)
new_df$fastback <- if_else(str_detect(new_df$body_type, regex(".*fastback.*")) == TRUE, 1, 0)
```

clean variable color

```
new_df$color = as.character(tolower(new_df$color))
new_df$color[is.na(new_df$color)]<-'other'
## the most popular colors are black and white
new_df$most_popular_color <- if_else(str_detect(new_df$color,
                                                (".*black.*|.*white.*")) == TRUE, 1, 0)
## we were curious to see if really long color description would make a car more expensive
## and it worked.
new_df$long_desc_color <- if_else(str_length(new_df$color) >= 15, 1, 0)
```

clean variable mileage

```
new_df$mileagex <- as.numeric(str_extract(new_df$mileage, "(\\d)+"))
new_df$mileagex[new_df$mileagex >= 999999 ] <- 70580 #3rd quartile
#with mean 44776, similar tactic to how we processed missingness price
new_df$mileagex[is.na(new_df$mileagex)]<- mean(new_df$mileagex, na.rm = TRUE)
```

clean variable condition

```
new_df$condition <- tolower(as.character(new_df$condition))
new_df$condition[is.na(new_df$condition)]<- "other"
new_df$condi_new <- if_else(str_detect(new_df$condition, ("new")) == TRUE, 1, 0)
new_df$condi_used <- if_else(str_detect(new_df$condition, ("used")) == TRUE, 1, 0)
new_df$condi_cpo <- if_else(str_detect(new_df$condition, ("certified pre-owned")) == TRUE, 1, 0)

new_df <- filter(new_df, condi_new == 1 | mileagex != 0)
```

clean variable cylinder

```
table(is.na(new_df$cylinder))
```

```
##
## FALSE  TRUE
##  6932  2532
```

```
new_df$cylinderx <- tolower(new_df$cylinder) %>% str_extract("(\\d)+")
new_df$cylinderx <- as.numeric(new_df$cylinderx)
new_df$cylinderx[is.na(new_df$cylinderx)]<- median(new_df$cylinderx, na.rm = TRUE) # median of 6
```

create variable turbo or Supercharged

```
new_df$engine <- tolower(as.character(new_df$engine))
new_df$engine[is.na(new_df$engine)]<- "other"
new_df$turbo <- if_else(str_detect(new_df$engine, ".*turbo.*") == TRUE, 1, 0)
new_df$scharged <- if_else(str_detect(new_df$engine, ".*supercharge.*") == TRUE, 1, 0)
```

clean variable sale by

```
new_df$for_sale_by <- tolower(as.character(new_df$for_sale_by))
new_df$for_sale_by[is.na(new_df$for_sale_by)]<- "other"
new_df$by_dealer <- if_else(str_detect(new_df$for_sale_by, "dealer"), 1, 0)
new_df$by_private <- if_else(str_detect(new_df$for_sale_by, regex("private seller|.*owner.*")), 1, 0)
```

clean variable drive type

```
new_df$drive_type <- tolower(as.character(new_df$drive_type))
new_df$drive_type[is.na(new_df$drive_type)]<- "other"
new_df$rwd <- if_else(str_detect(new_df$drive_type, regex(".*rwd.*")) == TRUE, 1, 0)
new_df$awd <- if_else(str_detect(new_df$drive_type, regex(".*awd.*")) == TRUE, 1, 0)
new_df$fourwd = if_else(str_detect(new_df$drive_type, regex(".*4wd.*")) == TRUE, 1, 0)
new_df$fwd <- if_else(str_detect(new_df$drive_type, regex(".*fwd.*")) == TRUE, 1, 0)
```

clean variable transmission

```
new_df$transmission <- tolower(as.character(new_df$transmission))
new_df$transmission[is.na(new_df$transmission)]<- "other"
new_df$auto <- if_else(str_detect(new_df$transmission, regex(".*auto.*")) == TRUE, 1, 0)
new_df$manu <- if_else(str_detect(new_df$transmission, regex(".*manu.*")) == TRUE, 1, 0)
```

clean variable warranty

```
new_df$warranty <- tolower(as.character(new_df$warranty))
new_df$warranty[is.na(new_df$warranty)]<- "other"
new_df$havwar <- if_else(str_detect(new_df$warranty, regex(".*has an existing warranty.*")) == TRUE, 1,
```

clean variable make

```
new_df$make = as.character(tolower(new_df$make))
new_df$make[is.na(new_df$make)]<-'other'
new_df$ford <- if_else(str_detect(new_df$make, regex(".*ford.*")) == TRUE, 1, 0)
new_df$chevrolet <- if_else(str_detect(new_df$make, regex(".*chevrolet.*")) == TRUE, 1, 0)
new_df$mercedes <- if_else(str_detect(new_df$make, regex(".*mercedes.*")) == TRUE, 1, 0)
new_df$bmw <- if_else(str_detect(new_df$make, regex(".*bmw.*")) == TRUE, 1, 0)
new_df$honda <- if_else(str_detect(new_df$make, regex(".*honda.*")) == TRUE, 1, 0)
new_df$toyota<- if_else(str_detect(new_df$make, regex(".*toyota.*")) == TRUE, 1, 0)
```

```
new_df$ram <- if_else(str_detect(new_df$make, regex(".*ram.*")) == TRUE, 1, 0)
new_df$jeep <- if_else(str_detect(new_df$make, regex(".*jeep.*")) == TRUE, 1, 0)
new_df$porsche <- if_else(str_detect(new_df$make, regex(".*porsche.*")) == TRUE, 1, 0)
new_df$dodge <- if_else(str_detect(new_df$make, regex(".*dodge.*")) == TRUE, 1, 0)
new_df$nissan <- if_else(str_detect(new_df$make, regex(".*nissan.*")) == TRUE, 1, 0)
new_df$audi <- if_else(str_detect(new_df$make, regex(".*audi.*")) == TRUE, 1, 0)
new_df$cadillac <- if_else(str_detect(new_df$make, regex(".*cadillac.*")) == TRUE, 1, 0)
new_df$ferrari <- if_else(str_detect(new_df$make, regex(".*ferrar.*")) == TRUE, 1, 0)
new_df$lexus  <- if_else(str_detect(new_df$make, regex(".*lexus.*")) == TRUE, 1, 0)
new_df$gmc <- if_else(str_detect(new_df$make, regex(".*gmc.*")) == TRUE, 1, 0)
new_df$bentley <- if_else(str_detect(new_df$make, regex(".*bentley.*")) == TRUE, 1, 0)
new_df$jaguar <- if_else(str_detect(new_df$make, regex(".*jaguar.*")) == TRUE, 1, 0)
new_df$subaru  <- if_else(str_detect(new_df$make, regex(".*subaru.*")) == TRUE, 1, 0)
new_df$volkswagen <- if_else(str_detect(new_df$make, regex(".*volkswagen.*")) == TRUE, 1, 0)
new_df$chrysler <- if_else(str_detect(new_df$make, regex(".*chrysler.*")) == TRUE, 1, 0)
new_df$landrover <- if_else(str_detect(new_df$make, regex(".*land rover.*")) == TRUE, 1, 0)
new_df$mazda <- if_else(str_detect(new_df$make, regex(".*mazda.*")) == TRUE, 1, 0)
new_df$maserati <- if_else(str_detect(new_df$make, regex(".*maserati.*")) == TRUE, 1, 0)
new_df$lincoln <- if_else(str_detect(new_df$make, regex(".*lincoln.*")) == TRUE, 1, 0)
new_df$hyundai <- if_else(str_detect(new_df$make, regex(".*hyundai.*")) == TRUE, 1, 0)
new_df$pontiac <- if_else(str_detect(new_df$make, regex(".*pontiac.*")) == TRUE, 1, 0)
new_df$kia <- if_else(str_detect(new_df$make, regex(".*kia.*")) == TRUE, 1, 0)
new_df$infiniti <-if_else(str_detect(new_df$make, regex(".*infiniti.*")) == TRUE, 1, 0)
new_df$hummer <- if_else(str_detect(new_df$make, regex(".*hummer.*")) == TRUE, 1, 0)
new_df$buick <- if_else(str_detect(new_df$make, regex(".*buick.*")) == TRUE, 1, 0)
new_df$volvo <- if_else(str_detect(new_df$make, regex(".*volvo.*")) == TRUE, 1, 0)
new_df$mitsubishi <- if_else(str_detect(new_df$make, regex(".*mitsubishi.*")) == TRUE, 1, 0)
new_df$acura <- if_else(str_detect(new_df$make, regex(".*acura.*")) == TRUE, 1, 0)
new_df$mini <- if_else(str_detect(new_df$make, regex(".*mini.*")) == TRUE, 1, 0)
new_df$mclaren <- if_else(str_detect(new_df$make, regex(".*mclaren.*")) == TRUE, 1, 0)
new_df$lotus <- if_else(str_detect(new_df$make, regex(".*lotus.*")) == TRUE, 1, 0)
new_df$alfa <- if_else(str_detect(new_df$make, regex(".*alfa.*")) == TRUE, 1, 0)
new_df$shelby <- if_else(str_detect(new_df$make, regex(".*shelby.*")) == TRUE, 1, 0)
new_df$mercury <- if_else(str_detect(new_df$make, regex(".*mercury.*")) == TRUE, 1, 0)
new_df$aston <- if_else(str_detect(new_df$make, regex(".*aston.*")) == TRUE, 1, 0)
```

create variable for air conditioning and power seat

```
new_df$options <- tolower(new_df$options)
new_df$options[is.na(new_df$options)]<-'did not specify'
new_df$ac <- if_else(str_detect(new_df$options, regex(".*air condition.*")) == TRUE, 1, 0)
new_df$pwr_seat <- if_else(str_detect(new_df$options, regex(".*power seat.*")) == TRUE, 1, 0)
```

create variable age and variable mileage per year And lastly we calculated the age of the cars using 2020 - year of the car. If it is a year 2020 car, then we manually mark the age with 1. Additionally, we added the variable mileage per year `mpy` to see if the overuse of a car would affect the price.

```
new_df$age <- round(if_else(new_df$year == 2020, 1, 2020 - as.numeric(new_df$year)), 0)
new_df$mpy <- round(new_df$mileagex / new_df$age, 0)
```

3.4 prepare data for regression

```
dataset <- new_df %>% select(-1:-19, -21)
nrow(dataset)
```

```
## [1] 9464
```

set seed and split training and testing data

```
set.seed(20191125)
in_train <- createDataPartition(y = dataset$pricex, p = 0.8, list = FALSE)
training <- dataset[in_train, ]
testing <- dataset[-in_train, ]
```

4. Fit Models

4.1 Model fitting on pricex

Linear regression

```
#lm
fit_lm <- train(pricex ~ ., data = training, method = "lm",
                preProcess = c("center", "scale"))
y_hat_lm <- predict(fit_lm, newdata = testing)
round(defaultSummary(data.frame(obs = testing$pricex,pred = y_hat_lm)),4)
```

```
##       RMSE   Rsquared        MAE
## 19279.5697     0.5048 13098.7488
```

Partial Least Squares Regression

```
#pls
fit_pls <- train(pricex ~., data = training, method = "pls",
                 preProcess = c("center", "scale"))
y_hat_pls <- predict(fit_pls, newdata = testing)
round(defaultSummary(data.frame(obs = testing$pricex, pred = y_hat_pls)),4)
```

```
##       RMSE   Rsquared        MAE
## 19312.3936     0.5023 13284.7252
```

Generalized Linear Models

```
#glm
fit_glm <- train(pricex ~., data = training, method = "glm",
                 preProcess = c("center", "scale"))
y_hat_glm <- predict(fit_glm, newdata = testing)
round(defaultSummary(data.frame(obs = testing$pricex,pred = y_hat_glm)),4)
```

```
##       RMSE   Rsquared        MAE
## 19279.5697     0.5048 13098.7488
```

Generalized Linear Model via penalized maximum likelihood

```
#glmnet
fit_glmnet <- train(pricex ~., data = training, method = "glmnet",
                    preProcess = c("center", "scale"))
y_hat_glmnet <- predict(fit_glmnet, newdata = testing)
round(defaultSummary(data.frame(obs = testing$pricex,pred = y_hat_glmnet)),4)
```

```
##       RMSE   Rsquared        MAE
## 19255.6398     0.5053 13076.7276
```

Decision tree (CART method)

```
#cart method
trControl = trainControl(method = "cv", number = 10)
fit_tree <- train(pricex ~ ., data = training, method = "rpart2",
             tuneLength = 10, trControl = trControl)
y_hat_tree <- predict(fit_tree, newdata = testing)
round(defaultSummary(data.frame(obs = testing$pricex, pred = y_hat_tree)),4)
```

```
##       RMSE   Rsquared        MAE
## 20432.3614     0.4435 13453.6099
```

```
# Tree bag
fit_bag <- train(pricex ~ ., data = training, method = "treebag")
y_hat_bag <- predict(fit_bag, newdata = testing)
round(defaultSummary(data.frame(obs = testing$pricex, pred = y_hat_bag)),4)
```

```
##       RMSE   Rsquared        MAE
## 19899.2324     0.4706 13158.7998
```

Random Forest

```
#randomForest
set.seed(20191125)
fit_rf <- randomForest(pricex ~ .,ntree = 500, mtry = 35, data = training,
                       trControl=trControl, importance = TRUE)
y_hat_rf <- predict(fit_rf,newdata=testing)
round(defaultSummary(data.frame(obs = testing$pricex, pred = y_hat_rf)),4)
```

```
##       RMSE   Rsquared        MAE
## 13139.7829     0.7707  7488.5458
```

Generalized Boosted Regression Models

```
#boost
databoost = gbm(pricex ~ .,data = training, distribution = "gaussian", n.trees = 1000,
                interaction.depth = 6, shrinkage = 0.2)
preddataboost = predict(databoost, newdata = testing, n.trees = 1000)
round(defaultSummary(data.frame(obs = testing$pricex, pred = preddataboost)),4)
```

```
##      RMSE  Rsquared       MAE
## 14447.501     0.724  8845.151
```

Neural Networks and Multinomial Log-Linear Models

```
#neural network
nnetGrid <- expand.grid(.decay = c(0, 0.01, .1),
                        .size = c(1:10))
nn <- train(pricex ~ ., data = training, method = "nnet",
            trControl = trControl, tuneGrid = nnetGrid,
            preProcess = c("center", "scale"), trace = FALSE)
defaultSummary(data.frame(obs = testing$pricex,
                          pred = predict(nn, newdata = testing)))
```

```
##     RMSE Rsquared      MAE
## 44058.74       NA 34542.01
```

For all of the models we tried, Random Forest and Generalized Boosted Regression produced best prediction (lowest RMSE in the testing set) and the R-sq for those two models are over 0.70. Random Forest is slight better than the Generalized Boosted Model. And among other models we tried, linear regression, partial least squares, generalized linear regression as well as its penalized version, decision tree and treebag all produced similar RMSE in the testing set. Neural network had the worst prediction performance for this dataset.

4.2 Model fitting on logarithm transformed `pricex` (`lnpricex`)

Since the car prices are usually in large numbers and using log-transformed price would result in less-skewed distribution, we also tried the all of the models with `lnpricex` being the dependent variable (except for the neural network model, which offered exceptionally bad prediction for this dataset).

subsetting the dataset

```
lndataset <- new_df %>% select(-1:-20)
```

set seed and split training and testing data

```
set.seed(20191125)
in_train_ln <- createDataPartition(y = lndataset$lnpricex, p = 0.8, list = FALSE)
training_ln <- lndataset[in_train_ln, ]
testing_ln <- lndataset[-in_train_ln, ]
```

Linear Regression model

```
#lm
fit_lm <- train(lnpricex ~ ., data = training_ln, method = "lm",
                preProcess = c("center", "scale"))
y_hat_lm <- predict(fit_lm, newdata = testing_ln)
round(defaultSummary(data.frame(obs = testing_ln$lnpricex,pred = y_hat_lm)),4)
```

```
##     RMSE Rsquared      MAE
##   0.5196   0.5619   0.3675
```

Partial Least Squares Regression

```
#pls
fit_pls <- train(lnpricex ~., data = training_ln, method = "pls",
                 preProcess = c("center", "scale"))
y_hat_pls <- predict(fit_pls, newdata = testing_ln)
round(defaultSummary(data.frame(obs = testing_ln$lnpricex, pred = y_hat_pls)),4)
```

```
##     RMSE Rsquared      MAE
##   0.5226   0.5553   0.3842
```

Generalized Linear Models

```
#glm
fit_glm <- train(lnpricex ~., data = training_ln, method = "glm",
                  preProcess = c("center", "scale"))
y_hat_glm <- predict(fit_glm, newdata = testing_ln)
round(defaultSummary(data.frame(obs = testing_ln$lnpricex,pred = y_hat_glm)),4)
```

```
##     RMSE Rsquared      MAE
##   0.5196   0.5619   0.3675
```

Generalized Linear Model via penalized maximum likelihood

```
#glmnet
fit_glmnet <- train(lnpricex ~., data = training_ln, method = "glmnet",
                    preProcess = c("center", "scale"))
y_hat_glmnet <- predict(fit_glmnet, newdata = testing_ln)
round(defaultSummary(data.frame(obs = testing_ln$lnpricex,pred = y_hat_glmnet)),4)
```

```
##     RMSE Rsquared      MAE
##   0.5184   0.5626   0.3680
```

Decision tree (CART method)

```
#cart method
trControl = trainControl(method = "cv", number = 10)
fit_tree <- train(lnpricex ~ ., data = training_ln, method = "rpart2",
             tuneLength = 10, trControl = trControl)
y_hat_tree <- predict(fit_tree, newdata = testing_ln)
round(defaultSummary(data.frame(obs = testing_ln$lnpricex, pred = y_hat_tree)),4)
```

```
##     RMSE Rsquared      MAE
##   0.5687   0.4720   0.4315
```

Treebag

```
# Tree bag
fit_bag <- train(lnpricex ~ ., data = training_ln, method = "treebag")
y_hat_bag <- predict(fit_bag, newdata = testing_ln)
round(defaultSummary(data.frame(obs = testing_ln$lnpricex, pred = y_hat_bag)),4)
```

```
##     RMSE Rsquared      MAE
##   0.5566   0.4936   0.4210
```

Random Forest

```
#randomForest
set.seed(20191125)
fit_rf <- randomForest(lnpricex ~ .,ntree = 500, mtry = 35, data = training_ln, trControl=trControl,impo
y_hat_rf = predict(fit_rf,newdata=testing_ln)#21893.2329
round(defaultSummary(data.frame(obs = testing_ln$lnpricex, pred = y_hat_rf)),4)
```

```
##     RMSE Rsquared      MAE
##   0.3568   0.7928   0.2350
```

Generalized Boosted Regression Models

```
#boost
databoost = gbm(lnpricex ~ .,data = training_ln, distribution = "gaussian", n.trees = 1000,
                interaction.depth = 6, shrinkage = 0.2)
preddataboost = predict(databoost, newdata = testing_ln, n.trees = 1000)
round(defaultSummary(data.frame(obs = testing_ln$lnpricex, pred = preddataboost)),4)
```

```
##     RMSE Rsquared      MAE
##   0.3762   0.7701   0.2611
```

The regression results and prediction outcomes for those models with log-transformed `pricex` (`lnpricex`)being the dependent variable displayed similar pattern: the random forest model and generalized boosted regression model are superior to others, where the random forest won by a thin margin (RMSE 0.3568 vs 0.3762). The R-sq for those models are slightly higher than the models with `pricex` as the dependent, but it is harder to interpret because of the RMSE is also in logarithm form.

5.Conclusion

After cleaned, transformed and expanded data with additional variabels using various packages, we tested different models on the data. During the process of modeling fitting, we also tried logarithm transformed for the dependent variable price. We found that the best model is Random Forest model with ntree = 500 and mtry = 35. It has the lowest RMSE. The RMSE and R-sq for the Random Forest model is 13139.78 and 0.7707, respectively. The log-version of the Random Forest produced the higher R-sq (0.7928), yet the RMSE is much harder to interpret since it is in logarithm format.

To conclude, with our current knowledge, Random Forest is the best algorithm to predict for this dataset (prices of car listings on eBay). Our model performance already displayed quite significant improvement compare to models from related studies (many had RMSE between 25,000 - 30,000). Yet we admit there's room for further development - the RMSE and R-sq is still not ideal and outcomes may show different result if we were able to build a larger dataset. For future researches, we hope to employ more advanced methods, increase the number of observations and include more information as predictors for better results.