Individual Project: Wine Review and Recommendation

Author: Bin Dong

Email: bindong2@illinois.edu

I. Project Description

This project is aming at two questions

II. Summary

A glance at all data

The basic properties of all fields from raw data are:

Numerics

Field	average	var	min	max
points	88.44714	9.23996	80	100
price	35.36339	1682.822	4	3300

Discoveries from numeric fields:

1. Points(ratings) are ranged from 80 to 100, which are very concentrated on the upper bound of range 0-100. It is better to scale this field
2. There are certain wines with no price. and price varies a lot between wines.

Factors

Field	levels	empty entries
country	44	63
description	119955	0
designation	37980	37465
province	426	63
region_1	1230	21247
region_2	18	79460
taster_name	20	26244
taster_twitter_handle	16	31213
title	118840	0
variety	708	1
winery	16757	0

Discoveries from factor fields:

1. Very limited tasters are involved in this training data.

2. Country, Description, variety and winery information are almost complete. Description are almost all distinguished between each row, so it should not be used as factor for regression.
3. There are many empty entries in designation, region_1, region_2. In my opinion there should not be significant difference between wine rating in such a geography granularity, but leaving out empty entries as the same factor would affect final regression model. They need to be further converted.

III. Data Processing

1. According to the description of `Title` column, A title can be discretized as `winery`+`vintage`+other information like desigination and province, so I extracted vintage from title according to title constructions. The basic strategy is to replace winery with empty string and parse the first 4-digit string as vintage.

2. Apply LDA to descriptions

```
library(text2vec)
library(tm)
```

```
## Loading required package: NLP
```

```
library(stringr)

descriptions <- stringr::str_replace_all(raw.data$description,"[^[:alpha:]]", " ")
descriptions <- stringr::str_replace_all(descriptions,"\\s+", " ")

stopwords <- c(tm::stopwords("en"))

tokens <- descriptions %>% tolower %>% word_tokenizer

it <- itoken(tokens)

v <- create_vocabulary(it, stopwords = stopwords) %>% prune_vocabulary(term_count_min = 10)

vectorizer <- vocab_vectorizer(v)

dtm <- create_dtm(it, vectorizer)
```

```
lda_model <- LDA$new(n_topics = 10,
          doc_topic_prior = 0.1, topic_word_prior = 0.01)
doc_topic_distr <- lda_model$fit_transform(dtm, n_iter = 1000,
                                           convergence_tol <- 0.01,
                                           check_convergence_every_n = 10)
```

```
## INFO [2019-12-02 00:05:01] iter 10 loglikelihood = -22788716.591
## INFO [2019-12-02 00:05:06] iter 20 loglikelihood = -22084759.902
## INFO [2019-12-02 00:05:10] iter 30 loglikelihood = -21577551.668
## INFO [2019-12-02 00:05:14] iter 40 loglikelihood = -21264674.653
## INFO [2019-12-02 00:05:19] iter 50 loglikelihood = -21059801.147
## INFO [2019-12-02 00:05:19] early stopping at 50 iteration
```

```
raw.data.cleaned <- data.frame(raw.data.cleaned, doc_topic_distr)
```

Fianlly, remove all rows with na.

```
raw.data.cleaned <- read.csv("cleaned.csv")
head(raw.data.cleaned)
```

IV. Train-Test split

```
set.seed(1)

traintestsplit <- 0.7
n <- nrow(raw.data.cleaned)
row.names(raw.data.cleaned) <- 1:n

train.index <- sample(1:n, floor(n*traintestsplit))
train.data <- raw.data.cleaned[train.index, ]
test.data <- raw.data.cleaned[-train.index, ]
```

V. Modeling

Model 1: Prediction with linear model

1.1 Linear model with only country, taster_name, price and vintage.

```
test.data.x <- test.data[,c("country", "price", "vintage", "taster_name")]
test.data.y <- test.data$points

lm.model <- lm(points~country+price+vintage+taster_name, raw.data.cleaned)

result <- predict.lm(lm.model, newdata = test.data.x)

rmse <- sqrt(sum((result-test.data.y)^2)/length(result))

print(rmse)
```

```
## [1] 2.596058
```

1.2. Linear model with above plus word vector

```
test.data.x2 <- test.data[,c("country", "price", "vintage",
                             "taster_name", "X1", "X2", "X3",
                             "X4", "X5", "X6", "X7", "X8",
                             "X9", "X10")]
test.data.y2 <- test.data$points

lm.model2 <- lm(points~country+price+vintage+
                  taster_name+X1+X2+X3+X4+X5+
                  X6+X7+X8+X9+X10, raw.data.cleaned)
```

```
result2 <- predict.lm(lm.model2, newdata = test.data.x2)

rmse <- sqrt(sum((result2-test.data.y2)^2)/length(result2))

print(rmse)
```

```
## [1] 2.510201
```

Model 2: Predict with

```
library(randomForest)
```

```
## randomForest 4.6-14

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

```
rf.model <- randomForest(points~price+vintage, raw.data.cleaned, ntree=10, mtry=2)
```

```
rf.result <- predict(rf.model, test.data.x)

rmse <- sqrt(sum((rf.result-test.data.y2)^2)/length(rf.result))

print(rmse)
```

```
## [1] 2.322746
```

```
library(randomForest)
test.data.x2 <- test.data[,c("price",
                             "vintage", "X1", "X2", "X3",
                             "X4", "X5", "X6", "X7", "X8",
                             "X9", "X10")]
rf.model2 <- randomForest(points~price+vintage+
                 X1+X2+X3+X4+X5+
                 X6+X7+X8+X9+X10, raw.data.cleaned, ntree=10, mtry=2)
```

```
rf.result2 <- predict(rf.model2, newdata = test.data.x2)

rmse <- sqrt(sum((rf.result2-test.data.y)^2)/length(rf.result2))

print(rmse)
```

```
## [1] 1.583712
```