DataComp - Data Preparation and Testing (0116)

Team

1/3/2021

Contents

Data Acquisition

```
df_1 = read.csv('data_v1.csv')
head(data_v1)
```

Response Variable - Extract The Price Change

```
# Available on CRAN
#install.packages('fmpcloudr')

# Pull the GitHub development version - currently same as CRAN
# install.packages("devtools")
#devtools::install_github("tonytrevisan/fmpcloudr")
library(fmpcloudr)

fmpc_set_token('b7c79cfacf2aec4f1db5e4cfaa16b849')
#symbols = c('AAPL','MSFT','BAC','TSLA')

#Bal = fmpc_financial_bs_is_cf(symbols,statement = 'balance')
#ratio = fmpc_financial_metrics(symbols, metric = 'ratios', quarterly = TRUE)

data_v2 = data_v1[,-c(1,4,6,10)] # with tickers
data_v2$`Publish date` = as.character(data_v2$`Publish date`)
```

```
#df = c()
#for (i in 1:nrow(data_v2)) {
#  df = fmpc_price_history(symbols = data_v2[i, "Symbol"],
#                          startDate = data_v2[i, "Publish date"],
#                          endDate = data_v2[i, "Publish date"] )
#}

#price_v1 = function(s, d1 , d2) {
#  return(fmpc_price_history(s, d1, d2)$changePercent)
#}
price_v2 = function(s, d1 , d2) {
  return(fmpc_price_history(s, d1, d2))
}
data_v2$Day1 = as.Date(data_v2$`Publish date`)+1
# publish day close price
mapply(price_v2, s = data_v2$Symbol,d1 = data_v2$`Publish date`, d2 = data_v2$`Publish date`)
# second day close price
mapply(price_v2, s = data_v2$Symbol,d1 = data_v2$Day1, d2 = data_v2$Day1)

# validation
fmpc_price_history(symbols = 'WMT', startDate = '2017-03-17', endDate = '2017-03-20')

# eliminated data records that have missing values in close price
head(data_v4)
```

Financial Metrics

```
head(data_v4)

data_v5=data_v4[,c(2,3,12)]
head(data_v5)

# extract financial metrics
df = c()
for (i in length(data_v5)){
  df = rbind(fmpc_financial_metrics(data_v5$Symbol, metric = 'ratios'))
}

head(df)
df_v2 = df[,c('symbol', 'date',
              'currentRatio', 'quickRatio', 'debtEquityRatio', 'interestCoverage', 'returnOnEquity',
              'priceEarningsRatio', 'receivablesTurnover', 'payablesTurnover', 'inventoryTurnover')]

# extract addition financial metric - eps
df2 = c()
for (i in length(data_v5)){
  df2 = rbind(fmpc_financial_bs_is_cf(data_v5$Symbol, statement = 'income'))
}

df2_v2 = df2[, c('symbol', 'date', 'period', 'eps')]

df3 <- merge(df_v2,df2_v2,by=c("symbol", "date"))
```

```r
df3$date = as.Date(df3$date, format = "%Y-%m-%d")
df4 = df3[df3$date>='2016-08-01',]

# get a column for quarter
library(tidyverse)
df5$year = year(as.Date(df5$date))

df5 <- df4 %>%
  select(period, everything())
df5 <- df5 %>%
  select(year, everything())
df5 <- df5 %>%
  select(symbol, everything())

df5$year = ifelse(df5$period=='Q1' & month(df5$date)==12, df5$year+1, df5$year)

# fix fiscal year
write_csv(df5, 'df5.csv')

df5

df6 = df5 %>%
  rename(
    Symbol = symbol,
    qtr = period,
    yr=year
    )

#df6$qtr = ifelse(month(df6$date) >= 1 & month(df6$date) <= 3, "Q1",
#                 ifelse(month(df6$date) >= 4  & month(df6$date) <= 6, "Q2",
#                        ifelse(month(df6$date) >= 7  & month(df6$date) <= 9, "Q3",
#                               ifelse(month(df6$date) >= 10  & month(df6$date) <= 12,"Q4", df6$qtr))))


data_v4$qtr = quarters(as.Date(data_v4$`Publish date`))
data_v4$yr = year(as.Date(data_v4$`Publish date`))
data = data_v4[,c('ID', 'Symbol', 'Stock', 'Publish date', 'PercentChg', 'qtr', 'yr')]
data$newsqtr = data$qtr
data$qtr = quarters(as.Date(data_v4$`Publish date`))

data1 = data %>%
  separate(qtr, c("qtr1", "qtr2"), sep = 1)

data1$qtr3 = as.numeric(data1$qtr2)-1

data1$yr2 = ifelse(data1$qtr3==0, data1$yr-1, data1$yr)
data1$qtr3 = ifelse(data1$qtr3==0, 4, data1$qtr3)

data2 = data1

data2$qtr = paste(data2$qtr1,data2$qtr3, sep = "")

data3 = data2[,c('ID', 'Symbol', 'Stock', 'Publish date', 'PercentChg', 'newsqtr', 'yr2', 'qtr')]
```

```r
data3$yr=data3$yr2

data3 = data3[,-c(6,7)]
data3$yr = as.integer(data3$yr)

data3
# qtr and yr are consistent with the respected columns
# in financial metrics for extracting the values in the previous quarters

df_fm_1 = merge(x = data3, y = df6, by = c("Symbol", "yr", "qtr"), all.x = TRUE)
df_fm_1

df_fm_1 <- df_fm_1 %>%
  select(Stock, everything())
df_fm_1 <- df_fm_1 %>%
  select(ID, everything())

df_fm_2 = df_fm_1 %>%
  rename(
    Fm_year = yr,
    Fm_quarter = qtr,
    Fm_date=date
    )
df_fm_2 = df_fm_2[,-c(2,3)]
df_fm_2 <- df_fm_2 %>%
  select(Fm_date, everything())
df_fm_2 <- df_fm_2 %>%
  select(Stock, everything())
df_fm_2 <- df_fm_2 %>%
  select(Symbol, everything())
df_fm_2 <- df_fm_2 %>%
  select(ID, everything())

df_fm_2 = df_fm_2 %>%
  rename(
    News_date = 'Publish date'
    )

df_fm_2$Fm_date<df_fm_2$News_date

write.csv(df_fm_2, 'df_fm_2.csv')
```

Data Cleaning

```r
summary(df_fm_2)

# take out inventoryTurnover (17/82 NA)
df_fm_3 = df_fm_2[, -c(15)]

# substitude remaining NAs with column mean
df_fm_3$interestCoverage = ifelse(is.na(df_fm_3$interestCoverage),
                                  mean(df_fm_3$interestCoverage, na.rm = TRUE),
```

```
                                    df_fm_3$interestCoverage)
df_fm_3$payablesTurnover = ifelse(is.na(df_fm_3$payablesTurnover),
                                    mean(df_fm_3$payablesTurnover, na.rm = TRUE),
                                    df_fm_3$payablesTurnover)

summary(df_fm_3)
write.csv(df_fm_3, "df_financialmetrics_3.csv")

# Model set 1 (FMs and Vectorized Text)
head(df_textembeddings_2) # news content
df_model_1 = merge(df_fm_3, df_textembeddings_2, by=c("Symbol", "ID"))
df_model_1 = df_model_1[,-c(16)]

head(df_textembeddings) # news headlines

write.csv(df_model_1, "df_model_1.csv")
head(df_model_1)

# Model set 2 (FMs and Original Text)
df_model_2 = merge(df_financialmetrics_3, data_v4, by=c("Symbol", "ID"))
df_model_2 = df_model_2[,-c(3,4)]
df_model_2 = df_model_2[,-c(15,16,20:24)]
df_model_2[,c(6:14)] = scale(df_model_2[,c(6:14)])
write.csv(df_model_2, "df_model_0113.csv")

df_model_3 = df_model_2
df_model_3$PercentChg.a = abs(df_model_3$PercentChg.x)
df_model_3$PercentChg.b = ifelse(df_model_3$PercentChg.x>0, 'Up', 'Down')
df_model_3[,c(6:14)] = scale(df_model_3[,c(6:14)])
df_model_3
write.csv(df_model_3, "df_model_3.csv")
```

EDA

Not performed

```
library(table1) # for stats summary

table1::label(gapminder$lifeExp) <- "Life Expectancy"
table1::label(gapminder$pop) <- "Population"
table1::label(gapminder$gdpPercap) <- "Gdp Per Capita"

table1::table1(~lifeExp + pop + gdpPercap | continent, data = gapminder)
```

Model Testings

Data Preparation

```
library(caTools)
set.seed(123)
split = sample.split(df_model_1, SplitRatio = 0.65)
df_train = subset(df_model_1, split == TRUE)
```

```
df_test = subset(df_model_1, split == FALSE)

df_train = na.omit(df_train)
df_test = na.omit(df_test)

df_train = df_train[,-c(1:5)]
df_test = df_test[,-c(1:5)]

df_train[,c(2:778)] = scale(df_train[,c(2:778)])
df_test[,c(2:778)] = scale(df_test[,c(2:778)])
```

Random Forest Regression

```
library(randomForest)
model_rf = randomForest(x = df_train[,-1], y = df_train[,1], ntree = 200)

plot(model_rf)
# ntree >= 150

model_rf

pred_rf = predict(model_rf, df_test[, -1])
pred_rf

#install.packages("Metrics")
library(Metrics)

mse = mean((df_test[,1] - pred_rf)^2)
mae = caret::MAE(df_test[,1], pred_rf)
rmse = caret::RMSE(df_test[,1], pred_rf)

cat("MSE: ", mse, "MAE: ", mae, " RMSE: ", rmse)

x = 1:length(y_test)
plot(x, y_test, col = "red", type = "l")
lines(x, pred_rf, col = "blue", type = "l")
legend(x = 1, y = 38,  legend = c("original test_y", "predicted test_y"),
       col = c("red", "blue"), box.lty = 1, cex = 0.8, lty = c(1, 1))
```

PCA

```
install.packages('pls')
library(pls)

set.seed (1234)

# scale=TRUE: standardize
model_pcr = pcr(PercentChg~., data = df_train, scale = TRUE, validation = "CV")

pred_pcr = predict(model_pcr, df_test[, -1])
pred_pcr
rmse_pcr = rmse(df_test[,1], pred_pcr)
```

```
rmse_pcr
mse_pcr = mse(df_test[,1], pred_pcr)
Adj_R2 <- 1-(mse_pcr/var(df_test[,1]))
Adj_R2

summary(model_pcr)
# Plot the root mean squared error
validationplot(model_pcr)
# Plot the cross validation MSE
validationplot(model_pcr, val.type="MSEP")
# Plot the R2
validationplot(model_pcr, val.type = "R2")

predplot(model_pcr)
coefplot(model_pcr)
```

XGBoost Regression (with LASSO Regularization enabled)

gblinear: generalized linear model with regularization (L1, L2) and gradient descent.

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

# Tuning Notes
#booster = gblinear
#gamma = 0
# or gamma = 5
#gamma Tune trick: Start with 0 and check CV error rate. If you see train error >>> test error, bring g
#alpha = 1 #L1 LASSO
#Objective = reg:linear
#eval_metric = RMSE

# -----
#preparing matrix
library(data.table)
#install.packages('mlr')
library(mlr)

x_train = data.matrix(df_train[,-1])
y_train = df_train[,1]
x_test = data.matrix(df_test[,-1])
y_test = df_test[,1]

xgb_train = xgb.DMatrix(data = x_train, label = y_train)
xgb_test = xgb.DMatrix(data = x_test, label = y_test)

#default parameters
#params <- list(booster = "gblinear", objective = "reg:linear", eta=0.3, gamma=0, alpha=1)

#Using the inbuilt xgb.cv function, let's calculate the best nround for this model. In addition, this f
#xgbcv <- xgb.cv( params = params, data = xgb_train, nrounds = 100, nfold = 5, showsd = T, stratified =
##best iteration = 79

#first default - model training
```

```r
xgb1 <- xgboost (data = xgb_train, nrounds = 50, max.depth = 2)
xgb1

#model prediction
xgbpred <- predict (xgb1,xgb_test)

mse = mean((y_test - xgbpred)^2)
mae = caret::MAE(y_test, xgbpred)
rmse = caret::RMSE(y_test, xgbpred)

cat("MSE: ", mse, "MAE: ", mae, " RMSE: ", rmse)

x = 1:length(y_test)
plot(x, y_test, col = "red", type = "l")
lines(x, xgbpred, col = "blue", type = "l")
legend(x = 1, y = 38,  legend = c("original test_y", "predicted test_y"),
       col = c("red", "blue"), box.lty = 1, cex = 0.8, lty = c(1, 1))

#view variable importance plot
 mat <- xgb.importance (feature_names = colnames(x_train),model = xgb1)
 xgb.plot.importance (importance_matrix = mat[1:50])
```