DataComp - R Modeling with Text Embeddings and Financial Metrics (0116)

Team

1/11/2021

Contents

Dataset

```
df_financialmetrics_3 = read.csv("df_financialmetrics_3.csv")
df_textembeddings_2 = read.csv("df_textembeddings_2.csv")
df_model_0116 = merge(df_financialmetrics_3, df_textembeddings_2, by=c("Symbol", "ID"))
#df_model_0116 = df_model_0116[,-c(16)]
#head(df_model_0116)
df_model_0116$Stock = paste(df_model_0116$Symbol, df_model_0116$ID, sep = '-')
df_model_0116 = df_model_0116[,-c(1,2,3,5,6,17)]
#df_model_0116

df_model_0116_2 = df_model_0116

# tryout base point
df_model_0116_2$PercentChg = df_model_0116_2$PercentChg*100
#df_model_0116_2

# tryout feature scaling
df_model_0116_3 = df_model_0116
df_model_0116_3[,c(3:11)] = scale(df_model_0116_3[,c(3:11)])
#df_model_0116_3
```

```
library(caTools)
set.seed(123)
split = sample.split(df_model_0116, SplitRatio = 0.8)
df_train = subset(df_model_0116, split == TRUE)
df_test = subset(df_model_0116, split == FALSE)
```

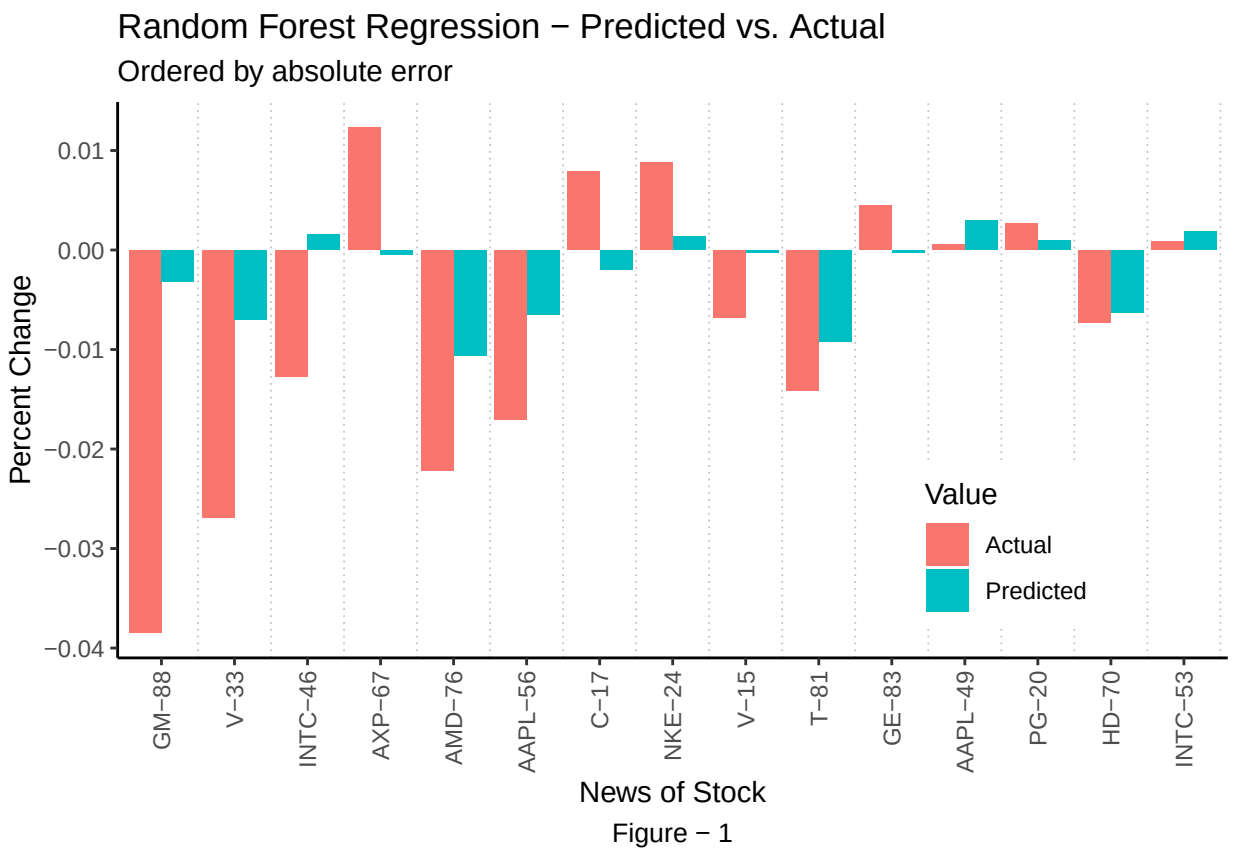
```
df_train = na.omit(df_train)
df_test = na.omit(df_test)

#df_train_2 = df_train_2[,-c(1:5)]
#df_test_2 = df_test_2[,-c(1:5)]
```

Models

Random Forest

```
library(randomForest)
model_rf = randomForest(x = df_train[,-c(1,2)], y = df_train[,2], ntree = 200)

#plot(model_rf)
# ntree >= 150

#model_rf

pred_rf = predict(model_rf, df_test[, -c(1,2)])
#pred_rf

mse_rf = mean((df_test[,2] - pred_rf)^2)
mae_rf = caret::MAE(df_test[,2], pred_rf)
rmse_rf = caret::RMSE(df_test[,2], pred_rf)
#cat("Model_rf ---- MSE: ", mse_rf, "MAE: ", mae_rf, " RMSE: ", rmse_rf)


# viz
df_out_rf = cbind(df_test$Stock, df_test$PercentChg, pred_rf)
df_out_rf = data.frame(df_out_rf)
df_out_rf$Stock = df_out_rf$V1
df_out_rf$Actual = df_out_rf$V2
df_out_rf$Predicted = df_out_rf$pred_rf
df_out_rf = df_out_rf[,-c(1,2,3)]
df_out_rf$Actual = as.numeric(df_out_rf$Actual)
df_out_rf$Predicted = as.numeric(df_out_rf$Predicted)
df_out_rf$Error = abs(df_out_rf$Predicted-df_out_rf$Actual)
#df_out_rf
library(reshape2)
df_out_rf2 = melt(df_out_rf[,-4], id.vars = 'Stock')
df_out_rf3 = merge(df_out_rf2,df_out_rf, by=c('Stock'))
df_out_rf3 = df_out_rf3[,c('Stock', 'variable', 'value' ,'Error')]
#df_out_rf3

library(ggplot2)
theme_set(
  theme_classic() +
    theme(legend.position = "top")
  )
ggplot(df_out_rf3, aes(x=reorder(Stock, -Error), y=value, fill=variable))+
  geom_bar(stat='identity', position='dodge')+
  geom_vline(xintercept = c(1.5:24.5), linetype="dotted", color = "grey", size=0.3)+
```

```
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1),
        legend.position = c(0.8, 0.2),
        plot.caption = element_text(hjust=0.5, size = 10))+
  labs(title = "Random Forest Regression - Predicted vs. Actual",
        subtitle = "Ordered by absolute error",
        caption = "Figure - 1",
        x = "News of Stock", y = "Percent Change", fill = "Value")+
  scale_y_continuous(breaks=c(-0.04, -0.03, -0.02, -0.01, 0, 0.01, 0.02, 0.03))
```




```
# ---
# Model_rf ---- MSE:  0.0001961043 MAE:  0.010555  RMSE:  0.01400372
```

PCA

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

set.seed (1234)

# scale=TRUE: standardize
model_pcr = pcr(PercentChg~., data = df_train[,-c(1)], scale = FALSE, validation = "CV")

#model_pcr

pred_pcr = predict(model_pcr, ncomp = model_pcr$ncomp ,df_test[, -c(1,2)], CV = "kfold", kfold = 10)
#pred_pcr
```

```r
mse_pca = mean((df_test[,2] - pred_pcr)^2)
mae_pca = caret::MAE(df_test[,2], pred_pcr)
rmse_pca = caret::RMSE(df_test[,2], pred_pcr)
#cat("Model_pca ---- MSE: ", mse_pca, "MAE: ", mae_pca, " RMSE: ", rmse_pca)

#summary(model_pcr)

#predplot(model_pcr)
#coefplot(model_pcr)

df_out_pcr = cbind(df_test$Stock, df_test$PercentChg, pred_pcr)
df_out_pcr = data.frame(df_out_pcr)
df_out_pcr$Stock = df_out_pcr$V1
df_out_pcr$Actual = df_out_pcr$V2
df_out_pcr$Predicted = df_out_pcr$pred_pcr
df_out_pcr = df_out_pcr[,-c(1,2,3)] #
df_out_pcr$Actual = as.numeric(df_out_pcr$Actual)
df_out_pcr$Predicted = as.numeric(df_out_pcr$Predicted)
df_out_pcr$Error = abs(df_out_pcr$Predicted-df_out_pcr$Actual)
#df_out_pcr
library(reshape2)
df_out_pcr2 = melt(df_out_pcr[,-4], id.vars = 'Stock')
df_out_pcr3 = merge(df_out_pcr2,df_out_pcr, by=c('Stock'))
df_out_pcr3 = df_out_pcr3[,c('Stock', 'variable', 'value' ,'Error')]
#df_out_pcr3

library(ggplot2)
theme_set(
  theme_classic() +
    theme(legend.position = "top")
  )
ggplot(df_out_pcr3, aes(x=reorder(Stock, -Error), y=value, fill=variable))+
  geom_bar(stat='identity', position='dodge')+
  geom_vline(xintercept = c(1.5:24.5), linetype="dotted", color = "grey", size=0.3)+
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1),
        legend.position = c(0.8, 0.2),
        plot.caption = element_text(hjust=0.5, size = 10))+
  labs(title = "Principal Component Regression - Predicted vs. Actual",
        subtitle = "Ordered by absolute error",
        caption = "Figure - 2",
        x = "News of Stock", y = "Percent Change", fill = "Value")+
  scale_y_continuous(breaks=c(-0.04, -0.03, -0.02, -0.01, 0, 0.01, 0.02, 0.03))
```

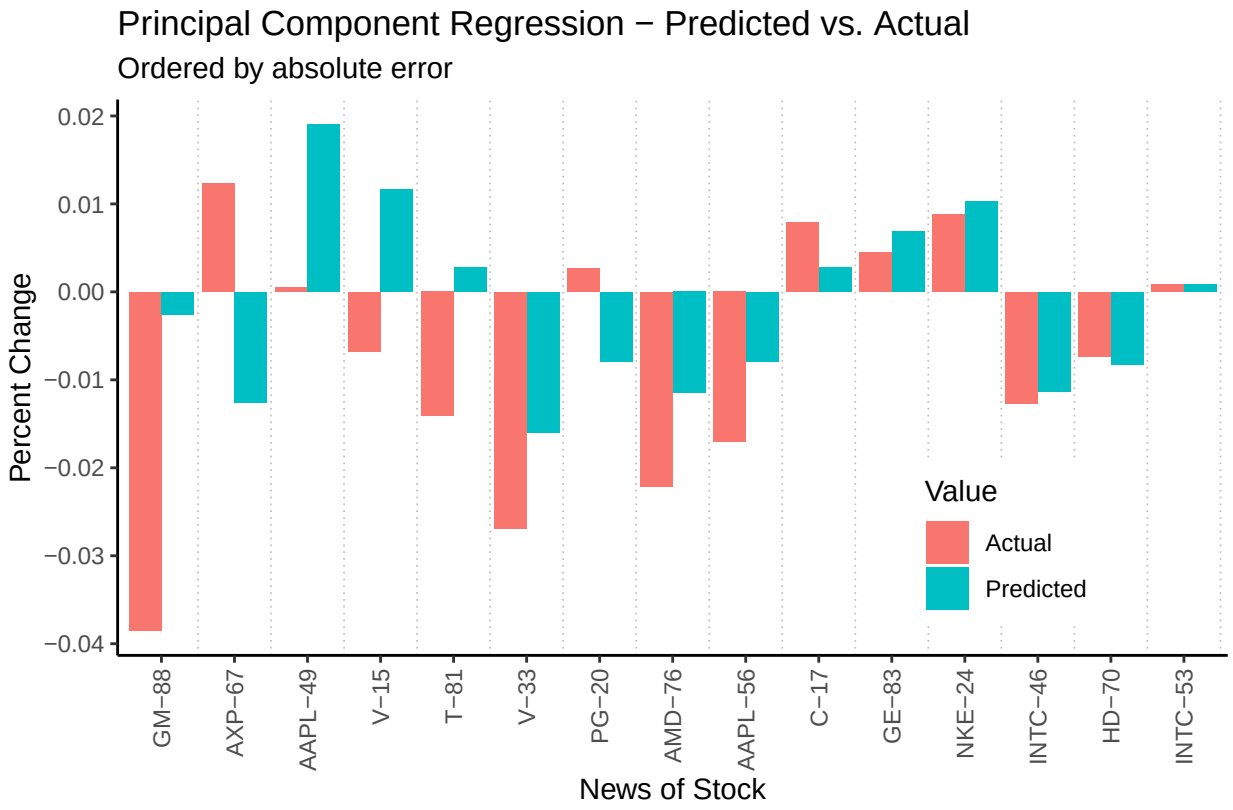


Figure − 2

```
# ----
# Model_pca ---- MSE:  0.0002117531 MAE:  0.01184545  RMSE:  0.01455174
```

XGBoost

```
#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_xgb = data.matrix(df_train[,-c(1,2)])
y_train_xgb = df_train[,2]
x_test_xgb = data.matrix(df_test[,-c(1,2)])
```

```r
y_test_xgb = df_test[,2]

xgb_train = xgb.DMatrix(data = x_train_xgb, label = y_train_xgb)
xgb_test = xgb.DMatrix(data = x_test_xgb, label = y_test_xgb)

#default parameters
#params <- list(booster = "gblinear",  eta=0.3, gamma=5, alpha=1)
#xgbcv <- xgb.cv( params = params, data = xgb_train, nrounds = 100, nfold = 10, showsd = T, stratified
#best iteration = 52


#first default - model training
xgb = xgboost(data = xgb_train, nrounds = 100, max.depth = 1, eta = 0.5, nfold=10, fold=5)
```

```r
#xgb

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

mse_xgb = mean((y_test_xgb - xgbpred)^2)
mae_xgb = caret::MAE(y_test_xgb, xgbpred)
rmse_xgb = caret::RMSE(y_test_xgb, xgbpred)
#cat("Model_XGBoost ---- MSE: ", mse_xgb, "MAE: ", mae_xgb, " RMSE: ", rmse_xgb)

df_out_xgb = cbind(df_test$Stock, df_test$PercentChg, xgbpred)
df_out_xgb = data.frame(df_out_xgb)
df_out_xgb$Stock = df_out_xgb$V1
df_out_xgb$Actual = df_out_xgb$V2
df_out_xgb$Predicted = df_out_xgb$xgbpred
df_out_xgb = df_out_xgb[,-c(1,2,3)] #
df_out_xgb$Actual = as.numeric(df_out_xgb$Actual)
df_out_xgb$Predicted = as.numeric(df_out_xgb$Predicted)
df_out_xgb$Error = abs(df_out_xgb$Predicted-df_out_xgb$Actual)
#df_out_xgb
library(reshape2)
df_out_xgb2 = melt(df_out_xgb[,-4], id.vars = 'Stock')
df_out_xgb3 = merge(df_out_xgb2,df_out_xgb, by=c('Stock'))
df_out_xgb3 = df_out_xgb3[,c('Stock', 'variable', 'value' ,'Error')]
#df_out_xgb3

library(ggplot2)
theme_set(
  theme_classic() +
    theme(legend.position = "top")
  )
ggplot(df_out_xgb3, aes(x=reorder(Stock, -Error), y=value, fill=variable))+
  geom_bar(stat='identity', position='dodge')+
  geom_vline(xintercept = c(1.5:24.5), linetype="dotted", color = "grey", size=0.3)+
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1),
        legend.position = c(0.8, 0.2),
        plot.caption = element_text(hjust=0.5, size = 10))+
  labs(title = "XGBoost - Predicted vs. Actual",
        subtitle = "Ordered by absolute error",
        caption = "Figure - 3",
```

```
      x = "News of Stock", y = "Percent Change", fill = "Value")+
  scale_y_continuous(breaks=c(-0.04, -0.03, -0.02, -0.01, 0, 0.01, 0.02, 0.03))
```

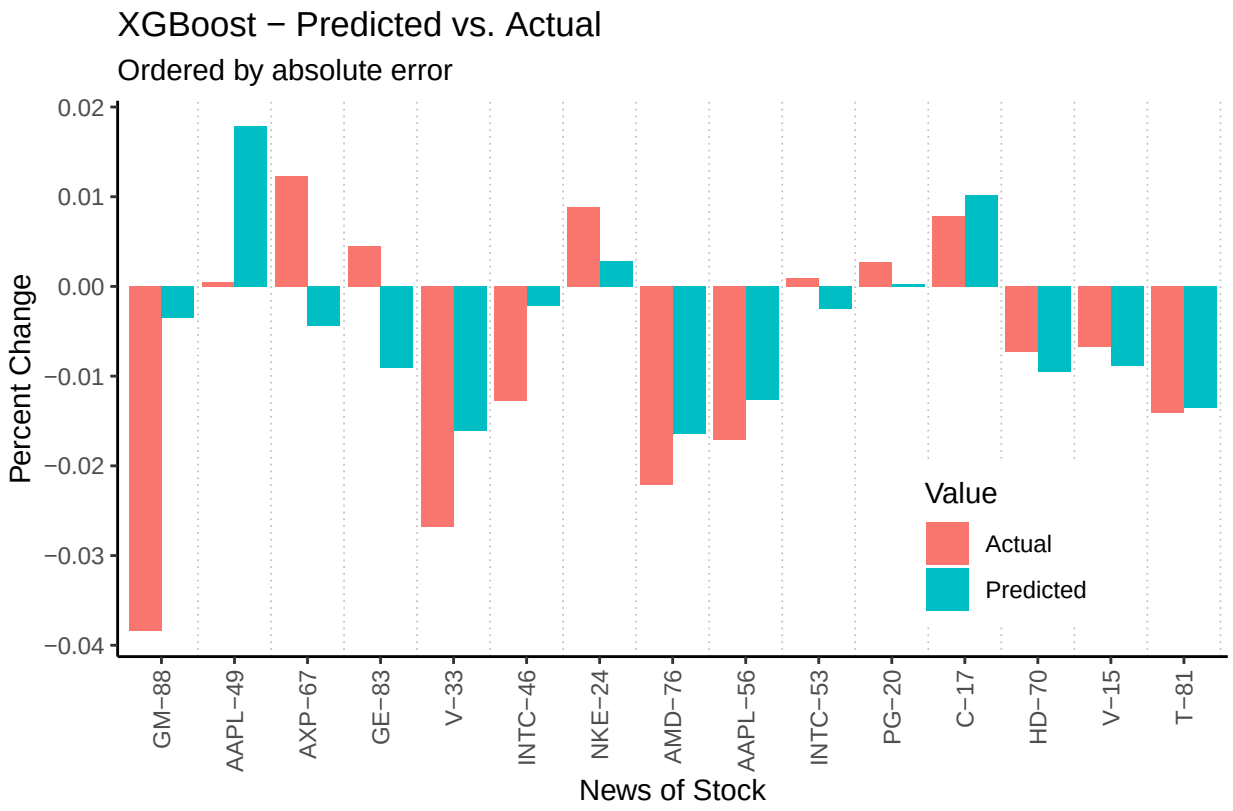


Figure − 3

```
# ----
# Model_XGBoost ---- MSE:  0.000155489 MAE:  0.008873421  RMSE:  0.01246952
```

Model Evaluations

```
#Model_rf ---- MSE:  0.0001641795 MAE:  0.009213431  RMSE:  0.01281325 n:200
#Model_pca ---- MSE: 0.0002225742 MAE: 0.01114589 RMSE: 0.01491892
#Model_XGBoost ---- MSE:  0.000155489 MAE:  0.008873421  RMSE:  0.01246952


Model = c('RF', 'PCR', 'XGBoost')
MSE = c(0.0001641795, 0.0002225742, 0.000155489)
MAE = c(0.009213431, 0.01114589, 0.008873421)
RMSE = c(0.01281325, 0.01491892, 0.01246952)
df_eval = data.frame(cbind(Model, MSE, MAE, RMSE))

library(knitr)

kable(df_eval, caption = "Model Performances")
```

Table 1: Model Performances

Model	MSE	MAE	RMSE
RF	0.0001641795	0.009213431	0.01281325
PCR	0.0002225742	0.01114589	0.01491892
XGBoost	0.000155489	0.008873421	0.01246952

```
####### Text Only
Model_to = c('RF', 'PCR', 'XGBoost')
MSE_to = c(0.0001657844, 0.0002354437, 0.0001710762)
MAE_to = c(0.009290567, 0.01233536, 0.009270712)
RMSE_to = c(0.01287573, 0.01534418, 0.01307961)
df_eval_to = data.frame(cbind(Model_to, MSE_to, MAE_to, RMSE_to))

library(knitr)

kable(df_eval_to, caption = "Model Performances (Text Only)")
```

Table 2: Model Performances (Text Only)

Model_to	MSE_to	MAE_to	RMSE_to
RF	0.0001657844	0.009290567	0.01287573
PCR	0.0002354437	0.01233536	0.01534418
XGBoost	0.0001710762	0.009270712	0.01307961

```
###### Text Embeddings by Bag of Words
Model_bow = c('RF', 'PCR', 'XGBoost')
MSE_bow = c(0.0002356849, '-', 0.0002356849)
MAE_bow = c(0.01232258, '-', 0.01232258)
RMSE_bow = c(0.01535203, '-', 0.01535203)
df_eval_bow = data.frame(cbind(Model_bow, MSE_bow, MAE_bow, RMSE_bow))

library(knitr)

kable(df_eval_bow, caption = "Model Performances (BOW)")
```

Table 3: Model Performances (BOW)

Model_bow	MSE_bow	MAE_bow	RMSE_bow
RF	0.0002356849	0.01232258	0.01535203
PCR	-	-	-
XGBoost	0.0002356849	0.01232258	0.01535203