PRoj3 part2

Joon

May 4, 2018

PART 2 CLASSIFICATION

This code just prepares the data as shown in the project description. Feel free to ignore this part as its just the copied code

```
library(tidyverse)
```

```
## -- Attaching packages ----------------------------------------------------- tidyverse 1.2.1 --
```

```
## v ggplot2 2.2.1     v purrr   0.2.4
## v tibble  1.4.2     v dplyr   0.7.4
## v tidyr   0.7.2     v stringr 1.2.0
## v readr   1.1.1     v forcats 0.2.0
```

```
## -- Conflicts -------------------------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following object is masked from 'package:base':
##
##     date
```

```
theme_set(theme_bw())

csv_file <- "Affordability_Wide_2017Q4_Public.csv"
tidy_afford <- read_csv(csv_file) %>%
  filter(Index == "Mortgage Affordability") %>%
  drop_na() %>%
  filter(RegionID != 0, RegionName != "United States") %>%
  dplyr::select(RegionID, RegionName, matches("^[1|2]")) %>%
  gather(time, affordability, matches("^[1|2]")) %>%
  type_convert(col_types=cols(time=col_date(format="%Y-%m")))
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   RegionID = col_integer(),
##   RegionName = col_character(),
##   SizeRank = col_integer(),
##   Index = col_character()
## )
```

```
## See spec(...) for full column specifications.
```

```
tidy_afford
```

```
## # A tibble: 12,480 x 4
```

```
##    RegionID RegionName                           time       affordability
##       <int> <chr>                                <date>             <dbl>
##  1   394913 New York, NY                         1979-03-01         0.262
##  2   753899 Los Angeles-Long Beach-Anaheim, CA 1979-03-01         0.358
##  3   394463 Chicago, IL                          1979-03-01         0.262
##  4   394514 Dallas-Fort Worth, TX                1979-03-01         0.301
##  5   394974 Philadelphia, PA                     1979-03-01         0.204
##  6   394692 Houston, TX                          1979-03-01         0.243
##  7   395209 Washington, DC                       1979-03-01         0.254
##  8   394856 Miami-Fort Lauderdale, FL            1979-03-01         0.268
##  9   394347 Atlanta, GA                          1979-03-01         0.248
## 10   394404 Boston, MA                           1979-03-01         0.222
## # ... with 12,470 more rows
```

More preparation of data

Same thing, preparation of data copied from the project description

```
outcome_df <- tidy_afford %>%
  mutate(yq = quarter(time, with_year=TRUE)) %>%
  filter(yq %in% c("2016.4", "2017.4")) %>%
  select(RegionID, RegionName, yq, affordability) %>%
  spread(yq, affordability) %>%
  mutate(diff = `2017.4` - `2016.4`) %>%
  mutate(Direction = ifelse(diff>0, "up", "down")) %>%
  select(RegionID, RegionName, Direction)
outcome_df
```

```
## # A tibble: 80 x 3
##    RegionID RegionName      Direction
##       <int> <chr>           <chr>
##  1   394304 Akron, OH       down
##  2   394312 Albuquerque, NM down
##  3   394318 Allentown, PA   down
##  4   394347 Atlanta, GA     up
##  5   394355 Austin, TX      up
##  6   394357 Bakersfield, CA down
##  7   394358 Baltimore, MD   down
##  8   394367 Baton Rouge, LA up
##  9   394378 Bellingham, WA  up
## 10   394388 Birmingham, AL  down
## # ... with 70 more rows
```

DISCUSSION OF INITIAL THOUGHTS

I chose to compare two random forest classifiers based on data for the entire time series vs the time series for the last few years (2011-2016). So I first split up the original data into 2 data frames where df1 is predicting the data over all the time (1980-2016) vs df2 which uses data from the past five years (2011-2016). MY initial assumption going into this is that the smaller dataset is a bit more accurate as it will reflect more recent market changes and will thus better predict affordability.

```
predictor_df1 <- tidy_afford %>%
  filter(year(time) <= 2016)
```

```r
predictor_df2 <- tidy_afford %>%
  filter(year(time) <=2016 & year(time)  >= 2011)
```

COde for the models

```r
wide_df1 <- predictor_df1 %>%
  select(RegionID,time,affordability) %>%
  tidyr::spread(time,affordability)

wide_df2 <- predictor_df2 %>%
  select(RegionID,time,affordability) %>%
  tidyr::spread(time,affordability)
```

I am now widening the data to prepare it so I can turn it into quarterly differences

```r
matrix_1 <- wide_df1 %>%
  select(-RegionID) %>%
  as.matrix() %>%
  .[,-1]

matrix_2 <- wide_df1 %>%
  select(-RegionID) %>%
  as.matrix() %>%
  .[,-ncol(.)]

matrix_3 <- wide_df2 %>%
  select(-RegionID) %>%
  as.matrix() %>%
  .[,-1]

matrix_4 <- wide_df2 %>%
  select(-RegionID) %>%
  as.matrix() %>%
  .[,-ncol(.)]

diff_df1 <- (matrix_1 - matrix_2) %>%
  magrittr::set_colnames(NULL) %>%
  as_data_frame() %>%
  mutate(RegionID = wide_df1$RegionID)

diff_df2 <- (matrix_3 - matrix_4) %>%
  magrittr::set_colnames(NULL) %>%
  as_data_frame() %>%
  mutate(RegionID = wide_df2$RegionID)
```

I join the two dataframes and label them either up or down based on the quarterly differences

```r
final_df1 <- diff_df1 %>%
  inner_join(outcome_df %>% select(RegionID, Direction), by="RegionID") %>%
  mutate(Direction=factor(Direction, levels=c("down", "up")))

final_df2 <- diff_df2 %>%
  inner_join(outcome_df %>% select(RegionID, Direction), by="RegionID") %>%
```

```
  mutate(Direction=factor(Direction, levels=c("down", "up")))
```

Code preparing the data for training using two random forests

```
set.seed(5555)

test_rf_df1 <- final_df1 %>%
  group_by(Direction) %>%
  sample_frac(.2) %>%
  ungroup()

test_rf_df2 <- final_df2 %>%
  group_by(Direction) %>%
  sample_frac(.2) %>%
  ungroup()

train_rf_df1 <- final_df1 %>%
  anti_join(test_rf_df1,by="RegionID")

train_rf_df2 <- final_df2 %>%
  anti_join(test_rf_df2,by="RegionID")
```

I used random forests to classify my data

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.4.4
```

```
## randomForest 4.6-14
```

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

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
rf1 <- randomForest(Direction~., data=train_rf_df1 %>% select(-RegionID))

rf2 <- randomForest(Direction~., data=train_rf_df2 %>% select(-RegionID))

rf1
```

```
##
## Call:
##  randomForest(formula = Direction ~ ., data = train_rf_df1 %>%      select(-RegionID))
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 12
##
##         OOB estimate of  error rate: 32.81%
## Confusion matrix:
##      down up class.error
```

```
## down    8 15   0.6521739
## up      6 35   0.1463415
```

```
rf2
```

```
##
## Call:
##  randomForest(formula = Direction ~ ., data = train_rf_df2 %>%      select(-RegionID))
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 25%
## Confusion matrix:
##      down up class.error
## down   10 13  0.56521739
## up      3 38  0.07317073
```

Using random forests to predict data

```
library(randomForest)
rf1_predictions <- predict(rf1, newdata=test_rf_df1 %>% select(-RegionID))

rf2_predictions <- predict(rf2, newdata=test_rf_df2 %>% select(-RegionID))

table(pred=rf1_predictions, observed=test_rf_df1$Direction)
```

```
##       observed
## pred   down up
##   down    2  0
##   up      4 10
```

```
table(pred=rf2_predictions, observed=test_rf_df2$Direction)
```

```
##       observed
## pred   down up
##   down    2  3
##   up      4  7
```

The data suggests that the error percentage for the larger dataset is 25% while the error percentage for the more recent dataset (2011-2016) is around 43.75% By there percentages we are given an initial assumption that the more recent dataset is less useful at classifying.

Cross validation part of the project.

I did two cross validations, one for the data including the entire time series and one including data from just the past 5 years

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.4.4
```

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

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
```

```
##     lift
```

```
library(sqldf)
```

```
## Warning: package 'sqldf' was built under R version 3.4.4
```

```
## Loading required package: gsubfn
```

```
## Loading required package: proto
```

```
## Warning: package 'proto' was built under R version 3.4.4
```

```
## Loading required package: RSQLite
```

```
result_df1 <- createFolds(final_df1$Direction,k=10) %>%
  purrr::imap(function(test_indices,fold_number){
    train_cross_df1 <- final_df1 %>%
      select(-RegionID) %>%
      slice(-test_indices)

    test_cross_df1 <- final_df1 %>%
      select(-RegionID) %>%
      slice(test_indices)

    rf1_cross <- randomForest(Direction~., data=train_cross_df1)

    test_cross_df1 %>%
      select(observed_label = Direction) %>%
      mutate(fold = fold_number) %>%
      mutate(prob_positive_rf1 = predict(rf1_cross, newdata = test_cross_df1,type ="prob")[,"up"]) %>%
      mutate(predicted_label_rf1 = ifelse(prob_positive_rf1 > 0.5, "up", "down"))
  }) %>%
  purrr::reduce(bind_rows)

result_df2 <- createFolds(final_df2$Direction,k=10) %>%
  purrr::imap(function(test_indices,fold_number){
    train_cross_df2 <- final_df2 %>%
      select(-RegionID) %>%
      slice(-test_indices)

    test_cross_df2 <- final_df2 %>%
      select(-RegionID) %>%
      slice(test_indices)

    rf2_cross <- randomForest(Direction~., data=train_cross_df2)

    test_cross_df2 %>%
      select(observed_label = Direction) %>%
      mutate(fold = fold_number) %>%
      mutate(prob_positive_rf2 = predict(rf2_cross, newdata = test_cross_df2,type ="prob")[,"up"]) %>%
      mutate(predicted_label_rf2 = ifelse(prob_positive_rf2 > 0.5, "up", "down"))
  }) %>%
  purrr::reduce(bind_rows)

result_df1
```

```
## # A tibble: 80 x 4
```

```
##    observed_label fold   prob_positive_rf1 predicted_label_rf1
##    <fct>          <chr>              <dbl> <chr>
##  1 up             Fold01             0.840 up
##  2 up             Fold01             0.688 up
##  3 up             Fold01             0.366 down
##  4 up             Fold01             0.604 up
##  5 up             Fold01             0.544 up
##  6 down           Fold01             0.470 down
##  7 down           Fold01             0.618 up
##  8 down           Fold01             0.516 up
##  9 down           Fold02             0.764 up
## 10 up             Fold02             0.544 up
## # ... with 70 more rows
```

```
result_df2
```

```
## # A tibble: 80 x 4
##    observed_label fold   prob_positive_rf2 predicted_label_rf2
##    <fct>          <chr>              <dbl> <chr>
##  1 down           Fold01             0.338 down
##  2 up             Fold01             0.596 up
##  3 down           Fold01             0.396 down
##  4 up             Fold01             0.424 down
##  5 up             Fold01             0.728 up
##  6 up             Fold01             0.650 up
##  7 down           Fold01             0.430 down
##  8 up             Fold01             0.824 up
##  9 down           Fold02             0.428 down
## 10 up             Fold02             0.564 up
## # ... with 70 more rows
```

COmputing the auroc curve based on the two dataframes

```
library(ROCR)
```

```
## Warning: package 'ROCR' was built under R version 3.4.4
```

```
## Loading required package: gplots
```

```
## Warning: package 'gplots' was built under R version 3.4.4
```

```
##
## Attaching package: 'gplots'
```

```
## The following object is masked from 'package:stats':
##
##     lowess
```

```
labels_df1 <- split(result_df1$observed_label, result_df1$fold)

predictions_rf1 <- split(result_df1$prob_positive_rf1, result_df1$fold) %>% prediction(labels_df1)

labels_df2 <- split(result_df2$observed_label, result_df2$fold)

predictions_rf2 <- split(result_df2$prob_positive_rf2, result_df2$fold) %>% prediction(labels_df2)

# compute average AUC for the first RF
mean_auc_rf1 <- predictions_rf1 %>%
```

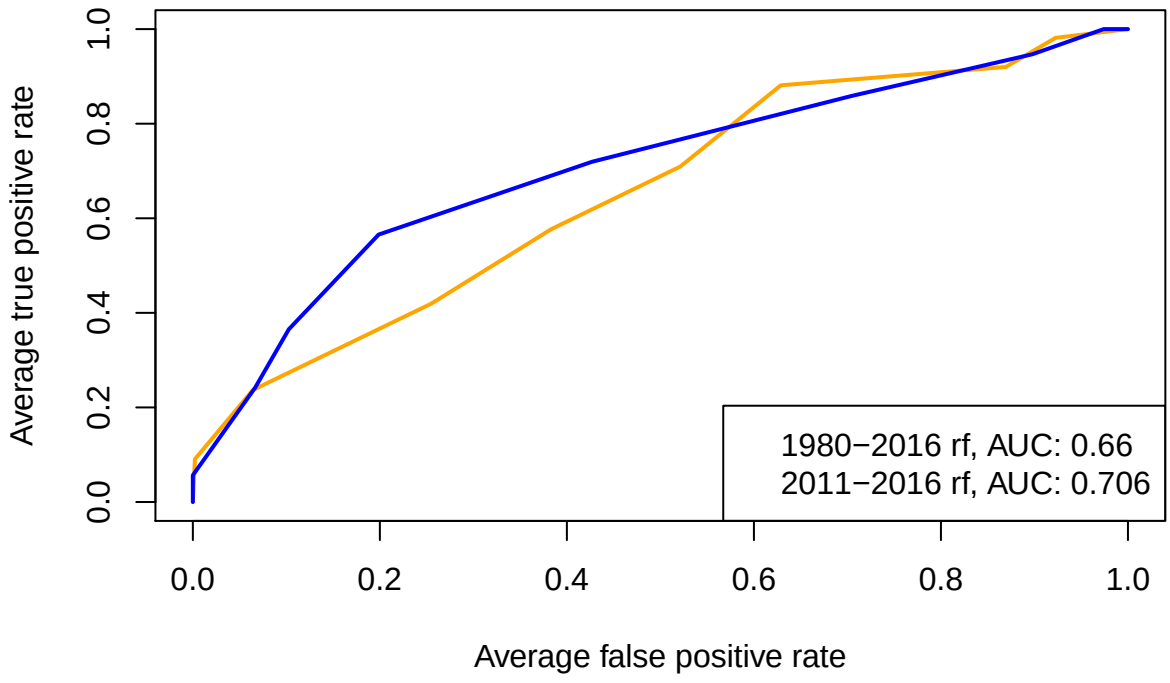
```
  performance(measure="auc") %>%
  # I know, this line is ugly, but that's how it is
  slot("y.values") %>% unlist() %>%
  mean()

# compute average AUC for the second RF
mean_auc_rf2 <- predictions_rf2 %>%
  performance(measure="auc") %>%
  slot("y.values") %>% unlist() %>%
  mean()

# plot the ROC curve for the first RF
predictions_rf1 %>%
  performance(measure="tpr", x.measure="fpr") %>%
  plot(avg="threshold", col="orange", lwd=2)

# plot the ROC curve for the second RF
predictions_rf2 %>%
  performance(measure="tpr", x.measure="fpr") %>%
  plot(avg="threshold", col="blue", lwd=2, add=TRUE)

legend("bottomright",
       legend=paste(c("1980-2016", "2011-2016"), "rf, AUC:", round(c(mean_auc_rf1, mean_auc_rf2), digits
       col=c("orange", "blue"))
```



INTERPRETATION AND DISCUSSION

overall it seems that the more recent dataset (2011-2016) is slightly more accurate. This is reflected in the auroc curve shown above as the closer the curve is to the lefthand corner, the more accurate it is. HOwever this is in contrast to the random forest classification done above in which the error rate was 25% for the larger dataset while the error rate of the more recent dataset was 43.75%. However our oerall AUROC curve which was based on the predictions made from our corss validation seem to hold true