Final Project for UCLA MAS 405

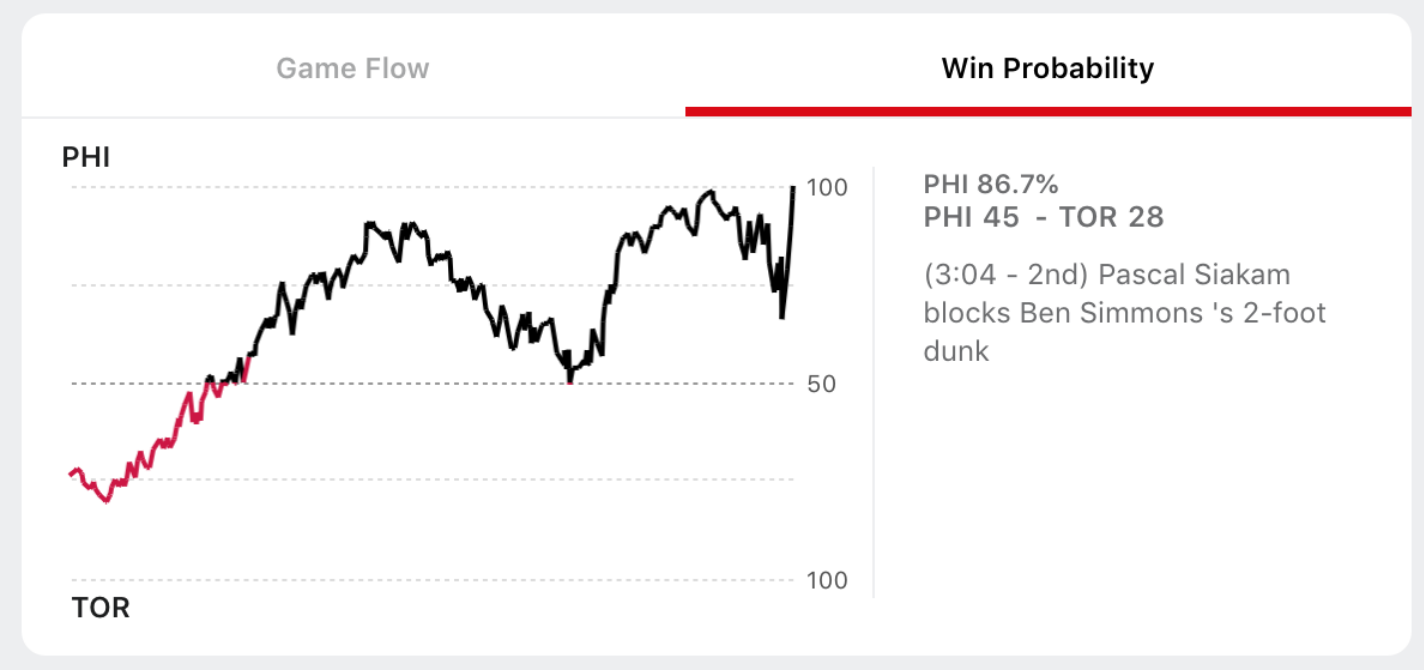
Andrew Sang - 505256314

2019-07-04

Introduction

For my final project in 405, I wanted to really work on a project that would let me delve into a game show that I've always been really interested in, "Jeopardy!". Growing up, I was always so impressed with the ability of the contestants to quickly recall facts from such a diverse body of areas. In contrast to most other game shows, the winner will come back for the next show. My college roommate had "Jeopardy!"" on Playstation 3 and we spent many nights buzzing in and attempting to best each other.

As an aside, I also really enjoy watching sports. One of the recent developments has been a win probability output that is generate throughout the course of the game.



ESPN Win Probability

Business Goal/Question

When I'm in the middle of an episode, I often catch myself being curious about who will win. Sure, the person with the most points is the likely winner…but, how likely?

Can if I develop a proof of concept model to generate a "realtime" win probability for any standard* game of "Jeopardy!"?

Technical Goals

I also wanted to see if I could use a lot of the concepts in the course to manage the data used for the model. These include, web scraping, storing data in SQL, and doing manipulations in R.

Set up Workspace & Libraries

Scraping

I made the decision to scrape the data in Python. I've included the ipynb converted to py script (scrape.py). The source of the data is http://www.j-archive.com/ (http://www.j-archive.com/), which is a site where ardent Jeopardy users track/input the information associated with each game of jeopardy. I decided to loop through each season, each episode, and find the scores (e.g. http://www.j-archive.com/showscores.php?game_id=6344 (http://www.j-archive.com/showscores.php?game_id=6344)). Once I had that data, I stored it as a JSON object. Using Python, I did some manipulations to get the data at a season/episode/round/question, with the scores of the players after the question concluded as the values. I saved this data as a CSV and am reading it into R.

A future enhancement would be to set up the script as a cron job so that it can pull automatically and update the CSV.

Sending Data to MySQL Database

Next, I'd like to store this information in a MySQL database. If/when the data grows too large, we can always just pull the games of Jeopardy that do not currently exist in the database and then do an insert statement. For now, it seems like there isn't a performance issue with just pulling together all of the existing games.

```
# import & quick cleanup
pivot <- read.csv('pivot_generated_20190702095603.csv', stringsAsFactors = FALSE)
pivot <- pivot[, colnames(pivot)!='X']
pivot <- pivot %>% select(season, game_id, rnd, question, player0, player1, player2, pla
yer3)
pivot %>% head(2)
```

```
##   season game_id rnd question player0 player1 player2 player3
## 1      1     173   0        1     100       0       0      NA
## 2      1     173   0        2     100       0     200      NA
```

```
# write data to mysql table
con <- dbConnect(MySQL(), user='root', password='account123', dbname='production', host=
'localhost', port=3306)
dbWriteTable(con,
             name="scores",
             value=pivot,
             field.types=c(season="INTEGER",
                           game_id="INTEGER",
                           rnd="INTEGER",
                           question="INTEGER",
                           player0="INTEGER",
                           player1="INTEGER",
                           player2="INTEGER",
                           player3="INTEGER"),
             row.names=FALSE,
             overwrite=TRUE)
```

```
## [1] TRUE
```

```
# can do some basic queries for fun
dbGetQuery(con, "SELECT count(1) FROM scores")
```

```
##   count(1)
## 1   314665
```

```
dbGetQuery(con, "SELECT * FROM scores LIMIT 10")
```

```
##    season game_id rnd question player0 player1 player2 player3
## 1       1     173   0        1     100       0       0      NA
## 2       1     173   0        2     100       0     200      NA
## 3       1     173   0        3     100       0     300      NA
## 4       1     173   0        4     100       0     500      NA
## 5       1     173   0        5     100       0     800      NA
## 6       1     173   0        6     100       0    1200      NA
## 7       1     173   0        7     600       0     700      NA
## 8       1     173   0        8     600    -300     400      NA
## 9       1     173   0        9    1000    -300     400      NA
## 10      1     173   0       10    1500    -300     400      NA
```

```
dbGetQuery(con, "SELECT count(distinct season) as season_cnt, count(distinct game_id) as
game_cnt FROM scores")
```

```
##   season_cnt game_cnt
## 1         36     5286
```

```
dbGetQuery(con, "SELECT
                    greatest(max(player0), max(player1), max(player2), max(player3)) as
 max_game_score,
                    least(min(player0), min(player1), min(player2), min(player3)) as min
_game_score
                FROM scores")
```

```
##   max_game_score min_game_score
## 1         131127          -6800
```

```
dbDisconnect(con)
```

```
## [1] TRUE
```

R Manipulation and Data Checking

Now, we have the data in a SQL database but we need to do some data cleaning before proceeding to any modeling.

```
con <- dbConnect(MySQL(), user='root', password='account123', dbname='production', host=
'localhost', port=3306)
scores <- dbGetQuery(con, "SELECT * FROM scores")
scores %>% summary()
```

```
##      season         game_id          rnd            question
##  Min.   : 0.00   Min.   :   1   Min.   :0.0000   Min.   : 0.00
##  1st Qu.:16.00   1st Qu.:1557   1st Qu.:0.0000   1st Qu.: 7.00
##  Median :23.00   Median :2939   Median :1.0000   Median :15.00
##  Mean   :22.49   Mean   :3023   Mean   :0.5225   Mean   :14.92
##  3rd Qu.:29.00   3rd Qu.:4499   3rd Qu.:1.0000   3rd Qu.:22.00
##  Max.   :35.00   Max.   :6347   Max.   :2.0000   Max.   :30.00
##
##     player0          player1         player2         player3
##  Min.   : -4600   Min.   :-6400   Min.   :-6800   Min.   :    0
##  1st Qu.:  1400   1st Qu.:  900   1st Qu.:  900   1st Qu.:    0
##  Median :  3600   Median : 2700   Median : 2600   Median :16801
##  Mean   :  5310   Mean   : 4016   Mean   : 3963   Mean   :14489
##  3rd Qu.:  7400   3rd Qu.: 5800   3rd Qu.: 5600   3rd Qu.:25201
##  Max.   :131127   Max.   :49699   Max.   :68000   Max.   :28400
##                                                   NA's   :314656
```

```
#We can utilize the DataMaid package to get a quick overview
# makeDataReport(data = scores,
#                mode = c("summarize","visualize","check"),
#                smartNum = FALSE,
#                file = "codebook_subset.Rmd",
#                replace = TRUE,
#                checks = list(character = "showAllFactorLevels",
#                              factor = "showAllFactorLevels",
#                              labelled = "showAllFactorLevels",
#                              haven_labelled = "showAllFactorLevels",
#                              numeric = NULL,
#                              integer = NULL,
#                              logical = NULL,
#                              Date = NULL),
#                listChecks = FALSE,
#                maxProbVals = 100,
#                codebook = TRUE,
#                reportTitle = "Codebook for Final Project")

# look into the player3 issue
scores %>% filter(!is.na(player3)) %>% head()
```

```
##   season game_id rnd question player0 player1 player2 player3
## 1      0    1347   2        0   22900       0   19800       0
## 2      0    1348   2        0    6600    1599    6799   28400
## 3      0    1349   2        0   20601   13300    -500   25201
## 4      0    1933   2        0    8999    9200   26000       0
## 5      0    1936   2        0    2001   22801       0   22800
## 6      0    1940   2        0   20901   24000   20900   16801
```

```
scores %>% filter(game_id==1933)
```

```
##    season game_id rnd question player0 player1 player2 player3
## 1       0    1933   0        1       0     200       0      NA
## 2       0    1933   0        2       0     400       0      NA
## 3       0    1933   0        3     400     400       0      NA
## 4       0    1933   0        4     400     400       0      NA
## 5       0    1933   0        5     400     400       0      NA
## 6       0    1933   0        6     400     400     800      NA
## 7       0    1933   0        7     400     400    1600      NA
## 8       0    1933   0        8     400    1200    1600      NA
## 9       0    1933   0        9     400    1200    1000      NA
## 10      0    1933   0       10     400    2200    1000      NA
## 11      0    1933   0       11     400    2200    1000      NA
## 12      0    1933   0       12     400    2200    2000      NA
## 13      0    1933   0       13     800    2200    2000      NA
## 14      0    1933   0       14    1200    2200    1600      NA
## 15      0    1933   0       15    1200    2200    2200      NA
## 16      0    1933   0       16    1200    2200    2200      NA
## 17      0    1933   0       17    2000    2200    2200      NA
## 18      0    1933   0       18    2400    2200    2200      NA
## 19      0    1933   0       19    2400    2800    2200      NA
## 20      0    1933   0       20    2400    3600    2200      NA
## 21      0    1933   0       21    2400    4600    2200      NA
## 22      0    1933   0       22    2400    4600    3000      NA
## 23      0    1933   0       23    2400    4600    4000      NA
## 24      0    1933   0       24    2400    4600    4000      NA
## 25      0    1933   0       25    2400    4600    4000      NA
## 26      0    1933   0       26    3000    4600    4000      NA
## 27      0    1933   0       27    3000    4800    4000      NA
## 28      0    1933   0       28    3000    4800    4000      NA
## 29      0    1933   0       29    4000    4800    4000      NA
## 30      0    1933   0       30    4000    5200    4000      NA
## 31      0    1933   1        1    4000    5200    4000      NA
## 32      0    1933   1        2    5000    5200    4000      NA
## 33      0    1933   1        3    5000    5200    4000      NA
## 34      0    1933   1        4    5000    5200    2500      NA
## 35      0    1933   1        5    5000    5200    2500      NA
## 36      0    1933   1        6    5000    7700    2500      NA
## 37      0    1933   1        7    5000    7700    2500      NA
## 38      0    1933   1        8    5000    7700    2500      NA
## 39      0    1933   1        9    5000    7700    4500      NA
## 40      0    1933   1       10    5000    7700    9000      NA
## 41      0    1933   1       11    5000    7700    9500      NA
## 42      0    1933   1       12    5000    7700    9500      NA
## 43      0    1933   1       13    5000    7700   10500      NA
## 44      0    1933   1       14    5000    7700   10500      NA
## 45      0    1933   1       15    7000    7700   10500      NA
## 46      0    1933   1       16    9500    7700   10500      NA
## 47      0    1933   1       17   11000    7700   10500      NA
## 48      0    1933   1       18   13000    7700   10500      NA
## 49      0    1933   1       19   15500    7700   10500      NA
## 50      0    1933   1       20   15500    7700   11000      NA
## 51      0    1933   1       21   15500    7700   11000      NA
## 52      0    1933   1       22   15500   10200   11000      NA
```

```
## 53      0    1933   1       23   15500    8200   13000      NA
## 54      0    1933   1       24   15500    9200   13000      NA
## 55      0    1933   1       25   15500    9200   13000      NA
## 56      0    1933   1       26   17500    9200   13000      NA
## 57      0    1933   2        0    8999    9200   26000       0
```

```
scores <- scores[,colnames(scores)!='player3']
scores %>% summary()
```

```
##      season         game_id          rnd            question
##  Min.   : 0.00   Min.   :   1   Min.   :0.0000   Min.   : 0.00
##  1st Qu.:16.00   1st Qu.:1557   1st Qu.:0.0000   1st Qu.: 7.00
##  Median :23.00   Median :2939   Median :1.0000   Median :15.00
##  Mean   :22.49   Mean   :3023   Mean   :0.5225   Mean   :14.92
##  3rd Qu.:29.00   3rd Qu.:4499   3rd Qu.:1.0000   3rd Qu.:22.00
##  Max.   :35.00   Max.   :6347   Max.   :2.0000   Max.   :30.00
##     player0          player1         player2
##  Min.   : -4600   Min.   :-6400   Min.   :-6800
##  1st Qu.:  1400   1st Qu.:  900   1st Qu.:  900
##  Median :  3600   Median : 2700   Median : 2600
##  Mean   :  5310   Mean   : 4016   Mean   : 3963
##  3rd Qu.:  7400   3rd Qu.: 5800   3rd Qu.: 5600
##  Max.   :131127   Max.   :49699   Max.   :68000
```

```
# how many games have X questions?
scores %>%
  group_by(game_id) %>%
  summarise(rnd_cnt = n_distinct(rnd),
            question_cnt = n_distinct(paste(rnd,question,sep='-'))) %>%
  ungroup() %>%
  group_by(question_cnt) %>%
  summarise(game_id_cnt = n_distinct(game_id)) %>%
  arrange(desc(game_id_cnt))
```

```
## # A tibble: 27 x 2
##    question_cnt game_id_cnt
##           <int>       <int>
##  1           61        3016
##  2           59         564
##  3           60         488
##  4           58         429
##  5           57         270
##  6           56         188
##  7           55         127
##  8           54          62
##  9           53          32
## 10           51          27
## # … with 17 more rows
```

```
# all have 3 rounds

# looks like the vast majority have more than 50+ questions. Going to filter anything wi
th less than that.
scores <- scores %>% mutate(rnd_quest = paste(rnd,question,sep="_"))
scores <- data.table(scores)[,c("game_question_cnt"):=list(n_distinct(rnd_quest)), by=li
st(game_id)]
scores <- scores %>% filter(game_question_cnt>50)
```

Example of a quick model for the other piece

```
# get the winner for each game
last_q <- scores %>% filter(rnd_quest == '2_0')
last_q$winner <- apply(last_q[,c('player0','player1','player2')],1,which.max) - 1
last_q <- last_q %>% select(game_id, winner)

scores_winner <- scores %>% left_join(last_q, by = 'game_id')
scores_winner <- scores_winner %>% mutate(winner=as.factor(winner))

# add a rank for each row number that we can parse
scores_winner <- scores_winner %>% group_by(game_id) %>% mutate(row_num = row_number())

# split 75/25
set.seed(1234)
game_list <- scores_winner$game_id %>% unique() %>% sample()
train_split <- floor(length(game_list)*.75)
train_games <- game_list[1:train_split]
test_games <- game_list[train_split:length(game_list)]

# convert frames to h2o
train_frame <- scores_winner %>% filter(game_id %in% train_games)
test_frame <- scores_winner %>% filter(game_id %in% test_games)

X <- colnames(train_frame)[3:length(colnames(train_frame))-1]
y <- "winner"
model <- h2o.glm(x = X, y = y,
                 family = "multinomial",
                 training_frame = as.h2o(train_frame), validation_frame = as.h2o(test_fr
ame),
                 seed = 1234, nfolds = 5)
```

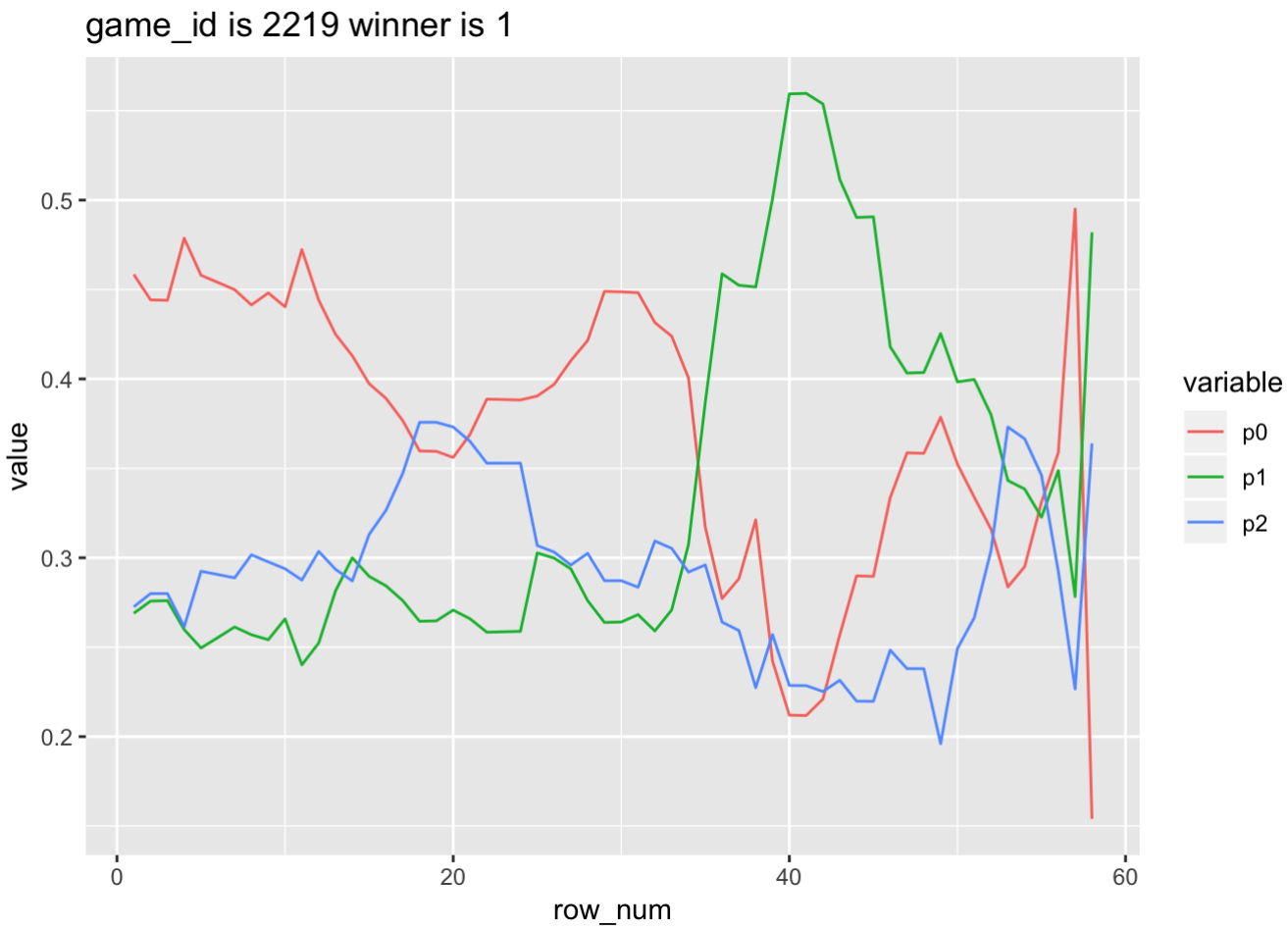
```
## Warning in .verify_dataxy(training_frame, x, y): removing response variable
## from the explanatory variables
```

```
## Warning in .h2o.startModelJob(algo, params, h2oRestApiVersion): Dropping bad and cons
tant columns: [rnd_quest].
```

```
predict_frame <- h2o.predict(model, as.h2o(test_frame)) %>% as.data.frame()
test_w_preds <- test_frame %>% copy() %>% as.data.frame()
test_w_preds[, c("p0","p1","p2")] <- predict_frame[, c("p0","p1","p2")]
```

```
# reshape dataframe to help plotting
graph_game_preds <- function(data, id, ...){
  data <- data %>% filter(game_id==id) %>% select(game_id, winner, row_num, p0, p1, p2)
  data <- melt(data, id=c("row_num","game_id","winner"))
  win <- (data$winner %>% unique())[[1]]
  graph <- ggplot(data) + geom_line(aes(x=row_num, y=value, colour=variable)) +
    ggtitle(paste('game_id is',id,'winner is',win,sep=" "))
  return(graph)
}

graph_game_preds(test_w_preds, id=test_games %>% sample(size=1))
```




```
h2o.removeAll() ## clean slate - just in case the cluster was already running
h2o.shutdown()
```

```
## Are you sure you want to shutdown the H2O instance running at http://localhost:54321/
(Y/N)?
```