Predict Major League Baseball (MLB) game outcomes - win/lose

By Wes Harbert

MSDS 692 Data Science Practicum

In [1]:

```
import os

import glob
import pandas as pd
import numpy as np
import numpy.ma as ma
import matplotlib.pyplot as plt
%matplotlib inline

from sklearn.cross_validation import train_test_split
from sklearn.feature_selection import VarianceThreshold
from sklearn.ensemble import ExtraTreesClassifier, RandomForestClassifier, Ada
BoostClassifier, VotingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn import model_selection
from sklearn.model_selection import cross_val_score

# ann analysis packages
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, Activation
from keras.layers.normalization import BatchNormalization
from keras.utils import to_categorical
from keras import backend as K

K.set_image_dim_ordering( 'tf' )
```

```
/Users/wesharbert/anaconda3/envs/py36_keras_tf/lib/python3.6/site-
packages/sklearn/cross_validation.py:44: DeprecationWarning: This
module was deprecated in version 0.18 in favor of the model_select
ion module into which all the refactored classes and functions are
moved. Also note that the interface of the new CV iterators are di
fferent from that of this module. This module will be removed in 0
.20.
  "This module will be removed in 0.20.", DeprecationWarning)
Using TensorFlow backend.
```

Import data

In [2]:

```
#data source is retrosheet.org:  http://www.retrosheet.org/gamelogs/index.html
. I am using data from all MLB games for
#for the 1960 - 2017 seasons. Each season was downloaded as a '.txt' file with
each data record separated by a comma.

#local path
path1 = 'baseball/data/seasons/'
path2 = 'baseball/data/headers.txt'

#import all
files = glob.glob(path1 + "/*.TXT")
seasons = []
for file_ in files:
    df = pd.read_csv(file_, sep=',', index_col = False, encoding = 'latin1')
    seasons.append(df)

#data field descriptions downloaded as separate file. I am saving the headers
for later use.
headers = pd.read_csv(path2, sep='\n', header = None)
```

In [3]:

```
#number of seasons in data set
len(seasons)
```

Out[3]:

```
58
```

In [4]:

```
#number of records and features for each season
[season.shape for season in seasons]
```

Out[4]:

```
[(2427, 161),
 (2429, 161),
 (2104, 161),
 (2104, 161),
 (2102, 161),
 (2103, 161),
 (2101, 161),
 (2104, 161),
 (2268, 161),
 (2105, 161),
 (2102, 161),
 (2098, 161),
 (1945, 161),
 (2106, 161),
 (2266, 161),
 (2265, 161),
 (2108, 161),
 (1624, 161),
 (2016, 161),
```

```
 (1393, 161),
 (2104, 161),
 (1599, 161),
 (1858, 161),
 (1614, 161),
 (2427, 161),
 (2431, 161),
 (1619, 161),
 (1942, 161),
 (1622, 161),
 (1937, 161),
 (1943, 161),
 (1625, 161),
 (1235, 161),
 (1944, 161),
 (1933, 161),
 (1429, 161),
 (2102, 161),
 (1618, 161),
 (2099, 161),
 (2105, 161),
 (1620, 161),
 (1938, 161),
 (2428, 161),
 (2428, 161),
 (2429, 161),
 (2428, 161),
 (2427, 161),
 (2425, 161),
 (2429, 161),
 (2429, 161),
 (2430, 161),
 (2430, 161),
 (2428, 161),
 (2429, 161),
 (2427, 161),
 (2429, 161),
 (2428, 161),
 (2430, 161)]
```

In [5]:

```
#note, all seasons have the same number of features - which is good!
```

In [6]:

```
headers.head()
```

Out[6]:

	0
0	date
1	game_in_day
2	day_of_wk
3	visiting_team
4	visiting_league

In [7]:

```
#header shape matches number of features in data
headers.shape
```

Out[7]:

```
(161, 1)
```

Data field descriptions

(for reference)

(from http://www.retrosheet.org/gamelogs/glfields.txt) Field(s) Meaning 1 Date in the form "yyyymmdd" 2 Number of game: "0" -- a single game "1" -- the first game of a double (or triple) header including seperate admission doubleheaders "2" -- the second game of a double (or triple) header including seperate admission doubleheaders "3" -- the third game of a triple-header "A" -- the first game of a double-header involving 3 teams "B" -- the second game of a double-header involving 3 teams 3 Day of week ("Sun","Mon","Tue","Wed","Thu","Fri","Sat") 4-5 Visiting team and league 6 Visiting team game number For this and the home team game number, ties are counted as games and suspended games are counted from the starting rather than the ending date. 7-8 Home team and league 9 Home team game number 10-11 Visiting and home team score (unquoted) 12 Length of game in outs (unquoted). A full 9-inning game would have a 54 in this field. If the home team won without batting in the bottom of the ninth, this field would contain a 51. 13 Day/night indicator ("D" or "N") 14 Completion information. If the game was completed at a later date (either due to a suspension or an upheld protest) this field will include: "yyyymmdd,park,vs,hs,len" Where yyyymmdd -- the date the game was completed park -- the park ID where the game was completed vs -- the visitor score at the time of interruption hs -- the home score at the time of interruption len -- the length of the game in outs at time of interruption All the rest of the information in the record refers to the entire game. 15 Forfeit information: "V" -- the game was forfeited to the visiting team "H" -- the game was forfeited to the home team "T" -- the game was ruled a no-decision 16 Protest information: "P" -- the game was protested by an unidentified team "V" -- a disallowed protest was made by the visiting team "H" -- a disallowed protest was made by the home team "X" -- an upheld protest was made by the visiting team "Y" -- an upheld protest was made by the home team Note: two of these last four codes can appear in the field (if both teams protested the game). 17 Park ID 18 Attendance (unquoted) 19 Time of game in minutes (unquoted) 20-21 Visiting and home line scores. For example: "010000(10)0x" Would indicate a game where the home team scored a run in the second inning, ten in the seventh and didn't bat in the bottom of the ninth. 22-38 Visiting team offensive statistics (unquoted) (in order): at-bats hits doubles triples homeruns RBI sacrifice hits. This may include sacrifice flies for years prior to 1954 when sacrifice flies were allowed. sacrifice flies (since 1954) hit-by-pitch walks intentional walks strikeouts stolen bases caught stealing grounded into double plays awarded first on catcher's interference left on base 39-43 Visiting team pitching statistics (unquoted)(in order): pitchers used (1 means it was a complete game) individual earned runs team earned runs wild pitches balks 44-49 Visiting team defensive statistics (unquoted) (in order): putouts. Note: prior to 1931, this may not equal 3 times the number of innings pitched. Prior to that, no putout was awarded when a runner was declared out for being hit by a batted ball. assists errors passed balls double plays triple plays 50-66 Home team offensive statistics 67-71 Home team pitching statistics 72-77 Home team defensive statistics 78-79 Home plate umpire ID and name 80-81 1B umpire ID and name 82-83 2B umpire ID and name 84-85 3B umpire ID and name 86-87 LF umpire ID and name 88-89 RF umpire ID and name If any umpire positions were not filled for a particular game the fields will be "","(none)". 90-91 Visiting team manager ID and name 92-93 Home team manager ID and name 94-95 Winning pitcher ID and name 96-97 Losing pitcher ID and name 98-99 Saving pitcher ID and name--"","(none)" if none awarded 100-101 Game Winning RBI batter ID and name--"","(none)" if none awarded 102-103 Visiting starting pitcher ID and name 104-105 Home starting pitcher ID and name 106-132 Visiting starting players ID, name and defensive position, listed in the order (1-9) they appeared in the batting order. 133-159 Home starting players ID, name and defensive position listed in the order (1-9) they appeared in the batting order. 160 Additional information. This is a grab-bag of informational items that might not warrant a field on their own. The field is alpha-numeric. Some items are represented by tokens such as: "HTBF" -- home team batted first. Note: if "HTBF" is specified it would be possible to see something like "01002000x" in the visitor's line score. Changes in umpire positions during a game will also appear in this field. These will be in the form: umpchange,inning,umpPosition,umpid with the latter three repeated for each umpire. These changes occur with umpire injuries, late arrival of umpires or changes from completion of suspended games. Details of suspended games are in field 14. 161 Acquisition information: "Y" -- we have the complete game "N" -- we don't have any portion of the game "D" -- the game was derived from box score and game story "P" -- we have some portion of the game. We may be missing innings at the beginning, middle and end of the game. Missing fields will be NULL.

Format Data

In [8]:

```
#name columns per field descriptions
for season in seasons:
    season.columns = headers.iloc[:,0]
```

Subset features

In [9]:

```
#save team lables - will be removed from the core feature set, but needed late
r to sort by team and compute statistics
home_vis_labels_list = []
for season in seasons:
    home_vis_labels_list.append(season.loc[:,['visiting_team','home_team']])
```

In [10]:

```
home_vis_labels_list[0].head()
```

Out[10]:

	visiting_team	home_team
0	BOS	OAK
1	ATL	WAS
2	PIT	ATL
3	MIL	CHN
4	ARI	CIN

In [11]:

```
#save data for building target labels (game scores)
target_label_base_list = []
for season in seasons:
    target_label_base_list.append(season.iloc[:,9:11])
```

In [12]:

```
target_label_base_list[0].head()
```

Out[12]:

	visiting_team_score	home_team_score
0	1	5
1	2	3
2	12	11
3	4	3
4	4	2

In [13]:

```
#innitial feature slection - remove features that are not measures of performa
nce suitable for prediction.
#selection based on personal judgement and some preliminary model interations.
seasons_subset = []
for season in seasons:
    season = season.iloc[:,3:77]
    season = season.drop(['outs_in_game_54_standard','attendance','duration_in
_minutes','visiting_league',
                          'home_league','day_night','completion_info','forfeit
_info','protest_info', 'park_ID',
                          'vis_score_by_inning', 'home_score_by_inning','visit
ing_team','home_team',
                          'visiting_team_score','home_team_score','home_team_g
ame_number','vis_team_game_number'],
                         axis = 1)
    seasons_subset.append(season)
seasons = seasons_subset
```

In [14]:

```
seasons[0].shape
```

Out[14]:

```
(2427, 56)
```

Check data for NaN values

In [15]:

```
sum([season.isnull().sum() for season in seasons])
```

Out[15]:

```
0
vis_at_bats              1
vis_hits                 1
vis_doubles              1
vis_triples              1
vis_hr                   1
vis_RBI                  1
vis_sacrafice_hits       1
vis_sacrafice_flies      1
vis_hit_by_pitch         1
vis_walks                1
vis_intentional_walks    1
vis_strikeouts           1
vis_stolen_bases         1
vis_caught_stealing      1
vis_grnd_dbl_plys        1
vis_first_cath_intf      1
vis_left_on_base         1
vis_pitchers_used        1
vis_ind_earned_runs      1
```

```
vis_ind_earned_runs          1
vis_team_earned_runs         1
vis_wild_pitches             1
vis_balks                    1
vis_putouts                  1
vis_assists                  1
vis_errors                   1
vis_passed_balls             1
vis_double_plays             1
vis_triple_plays             1
home_at_bats                 1
home_hits                    1
home_doubles                 1
home_triples                 1
home_hr                      1
home_RBI                     1
home_sacrafice_hits          1
home_sacrafice_flies         1
home_hit_by_pitch            1
home_walks                   1
home_intentional_walks       1
home_strikeouts              1
home_stolen_bases            1
home_caught_stealing         1
home_grnd_dbl_plys           1
home_first_cath_intf         1
home_left_on_base            1
home_pitchers_used           1
home_ind_earned_runs         1
home_team_earned_runs        1
home_wild_pitches            1
home_balks                   1
home_putouts                 1
home_assists                 1
home_errors                  1
home_passed_balls            1
home_double_plays            1
home_triple_plays            1
dtype: int64
```

In [16]:

```
#...NaN detected
```

In [17]:

```
ssn_idx = 0
for season in seasons:
    print(ssn_idx)
    null_columns=season.columns[season.isnull().any()]
    print(season[season.isnull().any(axis=1)][null_columns].head())
    ssn_idx += 1
```

```
0
Empty DataFrame
Columns: []
Index: []
```

```
1
Empty DataFrame
Columns: []
Index: []
2
Empty DataFrame
Columns: []
Index: []
3
Empty DataFrame
Columns: []
Index: []
4
Empty DataFrame
Columns: []
Index: []
5
Empty DataFrame
Columns: []
Index: []
6
Empty DataFrame
Columns: []
Index: []
7
Empty DataFrame
Columns: []
Index: []
8
Empty DataFrame
Columns: []
Index: []
9
Empty DataFrame
Columns: []
Index: []
10
Empty DataFrame
Columns: []
Index: []
11
0     vis_at_bats  vis_hits  vis_doubles  vis_triples  vis_hr  vis
_RBI  \
1131          NaN       NaN          NaN          NaN     NaN
NaN

0     vis_sacrafice_hits  vis_sacrafice_flies  vis_hit_by_pitch  v
is_walks  \
1131                 NaN                  NaN               NaN
NaN

0           ...           home_ind_earned_runs  home_team_earned_ru
ns  \
1131        ...                            NaN                    N
aN

0     home_wild_pitches  home_balks  home_putouts  home_assists  h
```

```
ome_errors  \
1131                 NaN         NaN            NaN           NaN
NaN

0     home_passed_balls  home_double_plays  home_triple_plays
1131                 NaN                NaN                NaN

[1 rows x 56 columns]
12
Empty DataFrame
Columns: []
Index: []
13
Empty DataFrame
Columns: []
Index: []
14
Empty DataFrame
Columns: []
Index: []
15
Empty DataFrame
Columns: []
Index: []
16
Empty DataFrame
Columns: []
Index: []
17
Empty DataFrame
Columns: []
Index: []
18
Empty DataFrame
Columns: []
Index: []
19
Empty DataFrame
Columns: []
Index: []
20
Empty DataFrame
Columns: []
Index: []
21
Empty DataFrame
Columns: []
Index: []
22
Empty DataFrame
Columns: []
Index: []
23
Empty DataFrame
Columns: []
Index: []
24
```

```
Empty DataFrame
Columns: []
Index: []
25
Empty DataFrame
Columns: []
Index: []
26
Empty DataFrame
Columns: []
Index: []
27
Empty DataFrame
Columns: []
Index: []
28
Empty DataFrame
Columns: []
Index: []
29
Empty DataFrame
Columns: []
Index: []
30
Empty DataFrame
Columns: []
Index: []
31
Empty DataFrame
Columns: []
Index: []
32
Empty DataFrame
Columns: []
Index: []
33
Empty DataFrame
Columns: []
Index: []
34
Empty DataFrame
Columns: []
Index: []
35
Empty DataFrame
Columns: []
Index: []
36
Empty DataFrame
Columns: []
Index: []
37
Empty DataFrame
Columns: []
Index: []
38
Empty DataFrame
```

```
Columns: []
Index: []
39
Empty DataFrame
Columns: []
Index: []
40
Empty DataFrame
Columns: []
Index: []
41
Empty DataFrame
Columns: []
Index: []
42
Empty DataFrame
Columns: []
Index: []
43
Empty DataFrame
Columns: []
Index: []
44
Empty DataFrame
Columns: []
Index: []
45
Empty DataFrame
Columns: []
Index: []
46
Empty DataFrame
Columns: []
Index: []
47
Empty DataFrame
Columns: []
Index: []
48
Empty DataFrame
Columns: []
Index: []
49
Empty DataFrame
Columns: []
Index: []
50
Empty DataFrame
Columns: []
Index: []
51
Empty DataFrame
Columns: []
Index: []
52
Empty DataFrame
Columns: []
```

```
Index: []
53
Empty DataFrame
Columns: []
Index: []
54
Empty DataFrame
Columns: []
Index: []
55
Empty DataFrame
Columns: []
Index: []
56
Empty DataFrame
Columns: []
Index: []
57
Empty DataFrame
Columns: []
Index: []
```

In [18]:

```
#Season 11, record 1131 contains NaN values
seasons[11].iloc[1131,:].head()
```

Out[18]:

```
0
vis_at_bats   NaN
vis_hits      NaN
vis_doubles   NaN
vis_triples   NaN
vis_hr        NaN
Name: 1131, dtype: float64
```

In [19]:

```
seasons[11].shape
```

Out[19]:

```
(2098, 56)
```

In [20]:

```
#drop game with NaN, also drop game from teams label list
seasons[11] = seasons[11].drop(seasons[11].index[1131])
home_vis_labels_list[11] = home_vis_labels_list[11].drop(home_vis_labels_list[
11].index[1131])
seasons[11].iloc[1130:1133,:]
```

Out[20]:

	vis_at_bats	vis_hits	vis_doubles	vis_triples	vis_hr	vis_RBI	vis_sacrafice_hi
1130	35.0	9.0	0.0	1.0	0.0	3.0	0.0
1132	40.0	14.0	2.0	0.0	2.0	6.0	0.0
1133	46.0	19.0	4.0	0.0	2.0	12.0	0.0

3 rows × 56 columns

In [21]:

```
#one less record in season 11 now
seasons[11].shape
```

Out[21]:

```
(2097, 56)
```

In [22]:

```
#check for NaN again, to confirm
sum([season.isnull().sum() for season in seasons])
```

Out[22]:

```
0
vis_at_bats              0
vis_hits                 0
vis_doubles              0
vis_triples              0
vis_hr                   0
vis_RBI                  0
vis_sacrafice_hits       0
vis_sacrafice_flies      0
vis_hit_by_pitch         0
vis_walks                0
vis_intentional_walks    0
vis_strikeouts           0
vis_stolen_bases         0
vis_caught_stealing      0
vis_grnd_dbl_plys        0
vis_first_cath_intf      0
vis_left_on_base         0
vis_pitchers_used        0
vis_ind_earned_runs      0
vis_team_earned_runs     0
vis_wild_pitches         0
vis_balks                0
```

```
vis_putouts                0
vis_assists                0
vis_errors                 0
vis_passed_balls           0
vis_double_plays           0
vis_triple_plays           0
home_at_bats               0
home_hits                  0
home_doubles               0
home_triples               0
home_hr                    0
home_RBI                   0
home_sacrafice_hits        0
home_sacrafice_flies       0
home_hit_by_pitch          0
home_walks                 0
home_intentional_walks     0
home_strikeouts            0
home_stolen_bases          0
home_caught_stealing       0
home_grnd_dbl_plys         0
home_first_cath_intf       0
home_left_on_base          0
home_pitchers_used         0
home_ind_earned_runs       0
home_team_earned_runs      0
home_wild_pitches          0
home_balks                 0
home_putouts               0
home_assists               0
home_errors                0
home_passed_balls          0
home_double_plays          0
home_triple_plays          0
dtype: int64
```

Create target class labels

In [23]:

```
target_label_base_list[11].shape
```

Out[23]:

```
(2098, 2)
```

In [24]:

```
#before computing target labels, remove record from game eleven that containte
d NaN values
target_label_base_list[11] = target_label_base_list[11].drop(target_label_base
_list[11].index[1131])
```

In [25]:

```
target_label_base_list[11].shape
```

Out[25]:

```
(2097, 2)
```

In [26]:

```
target_label_base_list[0].columns
```

Out[26]:

```
Index(['visiting_team_score', 'home_team_score'], dtype='object',
name=0)
```

In [27]:

```
#check for nullls
sum([labels.isnull().sum() for labels in target_label_base_list])
```

Out[27]:

```
0
visiting_team_score    0
home_team_score        0
dtype: int64
```

Binary data labels: visitor win = 0, home win = 1

In [28]:

```
#visitor wins = 0
#home wins = 1
target_labels_list = []
for label_base in target_label_base_list:
    label = pd.DataFrame(np.where(label_base["visiting_team_score"] > label_ba
se["home_team_score"], 0,1))
    label.columns = ['winner']
    target_labels_list.append(label)
```

In [29]:

```
[labels.hist() for labels in target_labels_list]
```

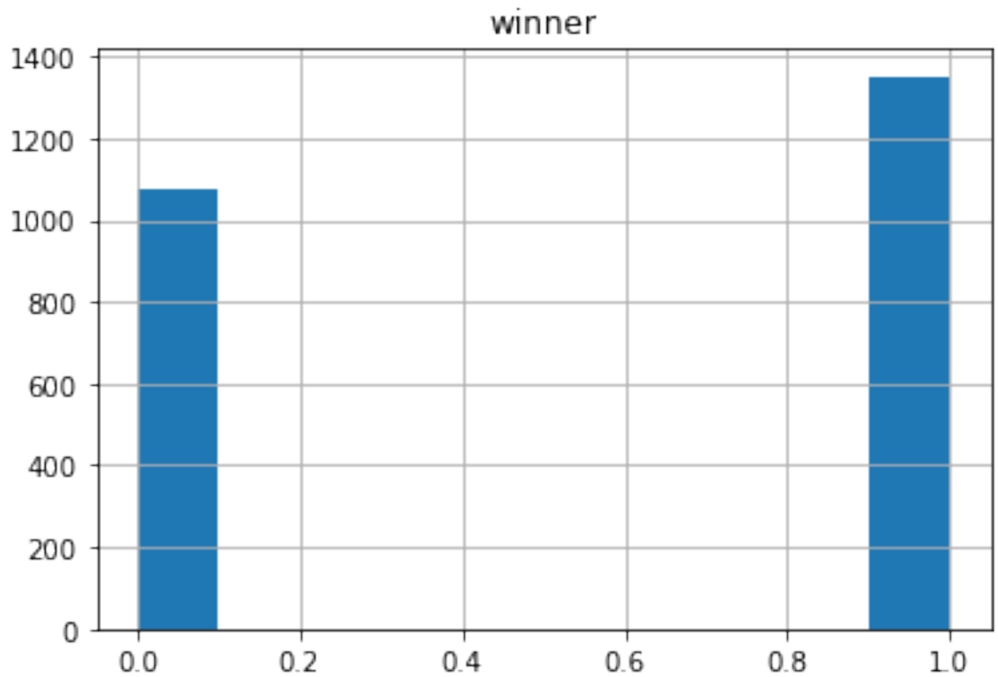
```
/Users/wesharbert/anaconda3/envs/py36_keras_tf/lib/python3.6/site-
packages/matplotlib/pyplot.py:524: RuntimeWarning: More than 20 fi
gures have been opened. Figures created through the pyplot interfa
ce (`matplotlib.pyplot.figure`) are retained until explicitly clos
ed and may consume too much memory. (To control this warning, see
the rcParam `figure.max_open_warning`).
  max_open_warning, RuntimeWarning)
```

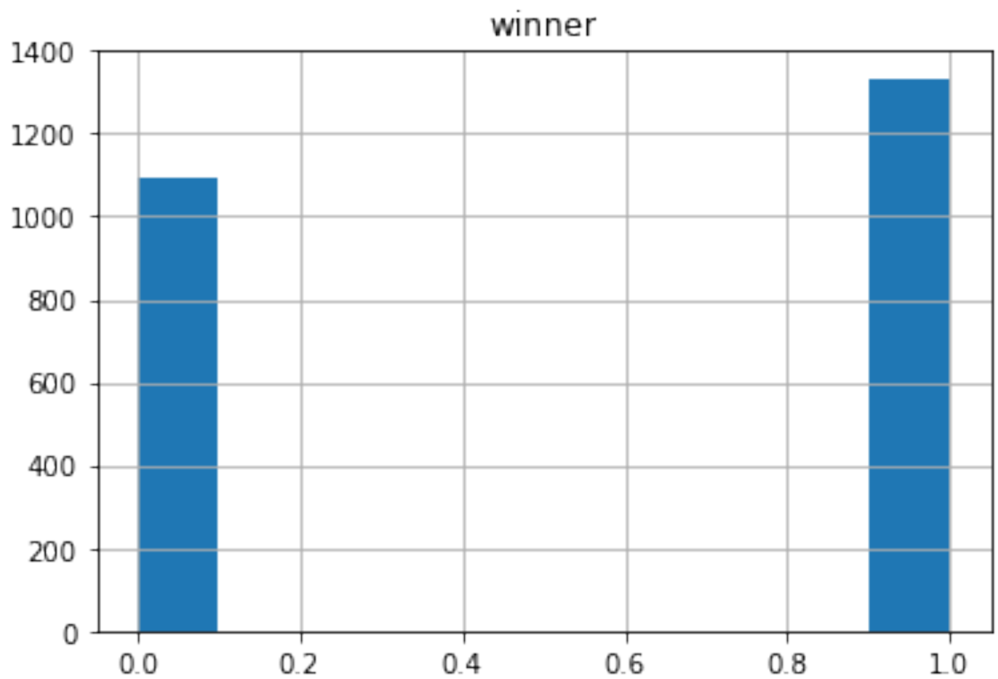
Out[29]:

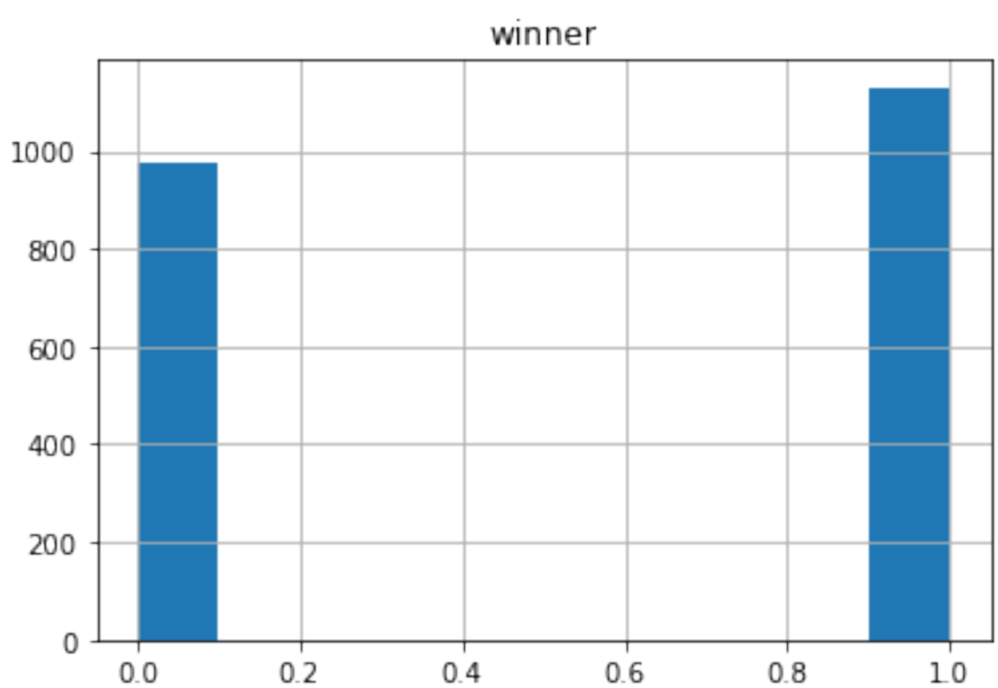
```
[array([[<matplotlib.axes._subplots.AxesSubplot object at 0x123c30
518>]], dtype=object),
```

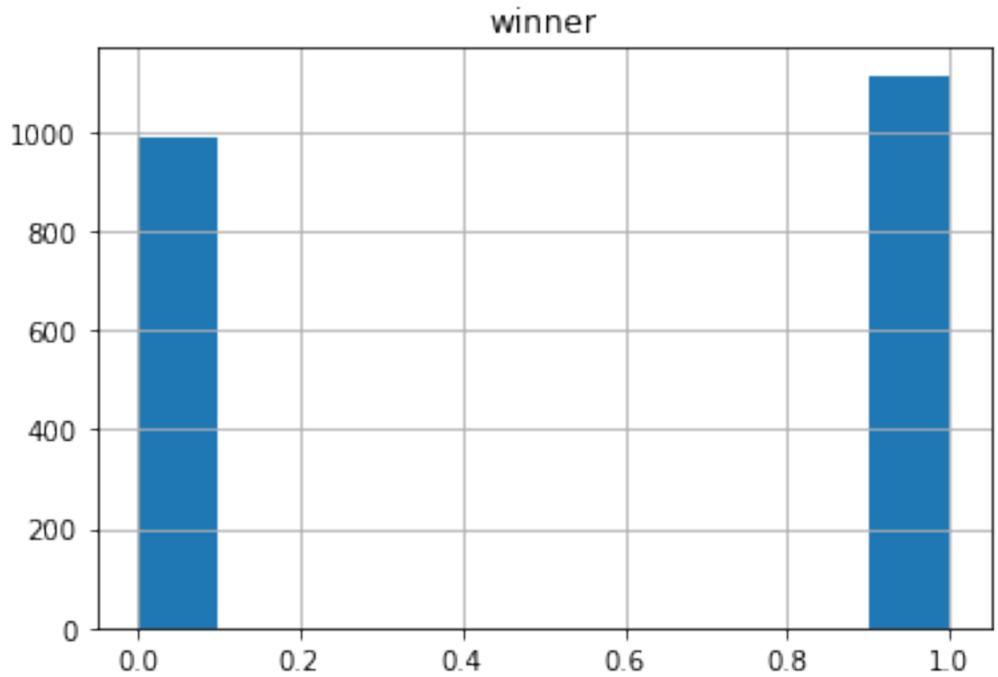
```
 array([[<matplotlib.axes._subplots.AxesSubplot object at 0x11f218
c18>]], dtype=object),
 array([[<matplotlib.axes._subplots.AxesSubplot object at 0x119ef3
390>]], dtype=object),
 array([[<matplotlib.axes._subplots.AxesSubplot object at 0x12d87b
160>]], dtype=object),
 array([[<matplotlib.axes._subplots.AxesSubplot object at 0x12d8ea
f28>]], dtype=object),
 array([[<matplotlib.axes._subplots.AxesSubplot object at 0x118ff0
4e0>]], dtype=object),
 array([[<matplotlib.axes._subplots.AxesSubplot object at 0x11dd7d
1d0>]], dtype=object),
 array([[<matplotlib.axes._subplots.AxesSubplot object at 0x11bf69
6d8>]], dtype=object),
 array([[<matplotlib.axes._subplots.AxesSubplot object at 0x121d0e
a20>]], dtype=object),
 array([[<matplotlib.axes._subplots.AxesSubplot object at 0x11fad8
4e0>]], dtype=object),
 array([[<matplotlib.axes._subplots.AxesSubplot object at 0x11cebd
128>]], dtype=object),
 array([[<matplotlib.axes._subplots.AxesSubplot object at 0x11aaa5
e48>]], dtype=object),
 array([[<matplotlib.axes._subplots.AxesSubplot object at 0x118f40
1d0>]], dtype=object),
 array([[<matplotlib.axes._subplots.AxesSubplot object at 0x124d59
d68>]], dtype=object),
 array([[<matplotlib.axes._subplots.AxesSubplot object at 0x1242de
e48>]], dtype=object),
 array([[<matplotlib.axes._subplots.AxesSubplot object at 0x121b29
be0>]], dtype=object),
 array([[<matplotlib.axes._subplots.AxesSubplot object at 0x119207
f28>]], dtype=object),
 array([[<matplotlib.axes._subplots.AxesSubplot object at 0x1200c4
b00>]], dtype=object),
 array([[<matplotlib.axes._subplots.AxesSubplot object at 0x11e3da
dd8>]], dtype=object),
 array([[<matplotlib.axes._subplots.AxesSubplot object at 0x10639a
dd8>]], dtype=object),
 array([[<matplotlib.axes._subplots.AxesSubplot object at 0x11fb8a
358>]], dtype=object),
 array([[<matplotlib.axes._subplots.AxesSubplot object at 0x1210a6
e48>]], dtype=object),
 array([[<matplotlib.axes._subplots.AxesSubplot object at 0x11ae69
160>]], dtype=object),
 array([[<matplotlib.axes._subplots.AxesSubplot object at 0x10618c
240>]], dtype=object),
 array([[<matplotlib.axes._subplots.AxesSubplot object at 0x121ff1
588>]], dtype=object),
 array([[<matplotlib.axes._subplots.AxesSubplot object at 0x118fe5
f28>]], dtype=object),
 array([[<matplotlib.axes._subplots.AxesSubplot object at 0x1238f5
438>]], dtype=object),
 array([[<matplotlib.axes._subplots.AxesSubplot object at 0x11c65d
400>]], dtype=object),
 array([[<matplotlib.axes._subplots.AxesSubplot object at 0x118eb2
748>]], dtype=object),
 array([[<matplotlib.axes._subplots.AxesSubplot object at 0x123993
```

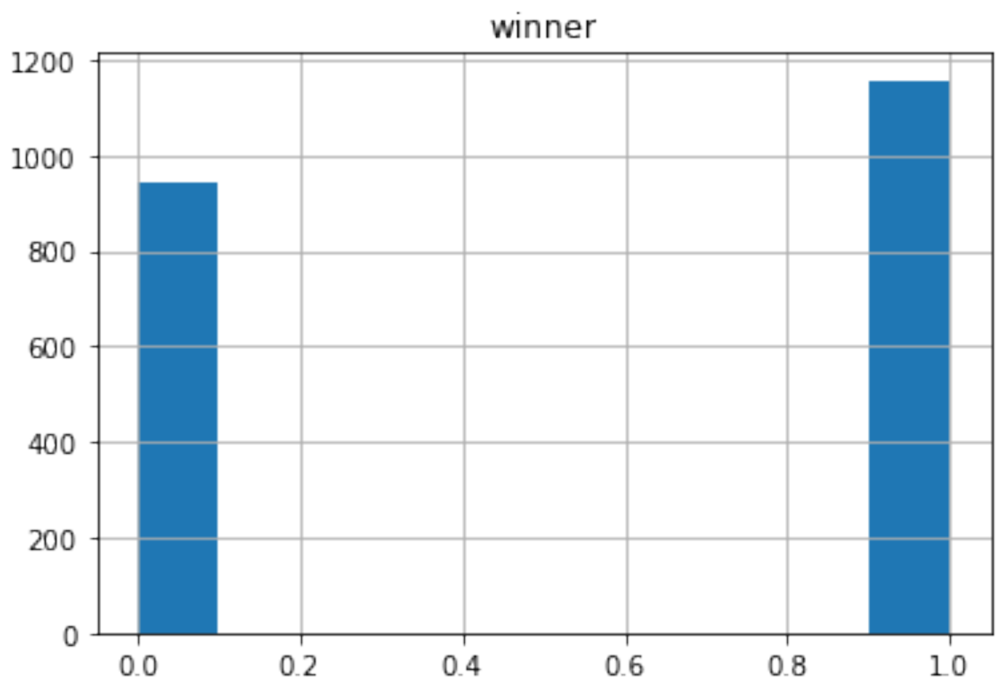
```
320>]], dtype=object),
 array([[<matplotlib.axes._subplots.AxesSubplot object at 0x11ba9a
5f8>]], dtype=object),
 array([[<matplotlib.axes._subplots.AxesSubplot object at 0x11a9ff
6d8>]], dtype=object),
 array([[<matplotlib.axes._subplots.AxesSubplot object at 0x1061f0
a20>]], dtype=object),
 array([[<matplotlib.axes._subplots.AxesSubplot object at 0x122529
400>]], dtype=object),
 array([[<matplotlib.axes._subplots.AxesSubplot object at 0x1223dc
c50>]], dtype=object),
 array([[<matplotlib.axes._subplots.AxesSubplot object at 0x120aba
f98>]], dtype=object),
 array([[<matplotlib.axes._subplots.AxesSubplot object at 0x11f213
320>]], dtype=object),
 array([[<matplotlib.axes._subplots.AxesSubplot object at 0x11b9ab
eb8>]], dtype=object),
 array([[<matplotlib.axes._subplots.AxesSubplot object at 0x118c08
1d0>]], dtype=object),
 array([[<matplotlib.axes._subplots.AxesSubplot object at 0x1268b7
2b0>]], dtype=object),
 array([[<matplotlib.axes._subplots.AxesSubplot object at 0x1267d4
5f8>]], dtype=object),
 array([[<matplotlib.axes._subplots.AxesSubplot object at 0x121fc9
f98>]], dtype=object),
 array([[<matplotlib.axes._subplots.AxesSubplot object at 0x121c89
4a8>]], dtype=object),
 array([[<matplotlib.axes._subplots.AxesSubplot object at 0x124cf5
588>]], dtype=object),
 array([[<matplotlib.axes._subplots.AxesSubplot object at 0x1241e4
8d0>]], dtype=object),
 array([[<matplotlib.axes._subplots.AxesSubplot object at 0x11a31d
2b0>]], dtype=object),
 array([[<matplotlib.axes._subplots.AxesSubplot object at 0x123882
780>]], dtype=object),
 array([[<matplotlib.axes._subplots.AxesSubplot object at 0x1224aa
860>]], dtype=object),
 array([[<matplotlib.axes._subplots.AxesSubplot object at 0x121a92
ba8>]], dtype=object),
 array([[<matplotlib.axes._subplots.AxesSubplot object at 0x1205a4
588>]], dtype=object),
 array([[<matplotlib.axes._subplots.AxesSubplot object at 0x118f1d
a90>]], dtype=object),
 array([[<matplotlib.axes._subplots.AxesSubplot object at 0x11f150
dd8>]], dtype=object),
 array([[<matplotlib.axes._subplots.AxesSubplot object at 0x11f25a
160>]], dtype=object),
 array([[<matplotlib.axes._subplots.AxesSubplot object at 0x11a3b4
cf8>]], dtype=object),
 array([[<matplotlib.axes._subplots.AxesSubplot object at 0x11d796
fd0>]], dtype=object),
 array([[<matplotlib.axes._subplots.AxesSubplot object at 0x11dd8f
0f0>]], dtype=object),
 array([[<matplotlib.axes._subplots.AxesSubplot object at 0x11c516
438>]], dtype=object),
 array([[<matplotlib.axes._subplots.AxesSubplot object at 0x11b2b6
dd8>]], dtype=object)]
```

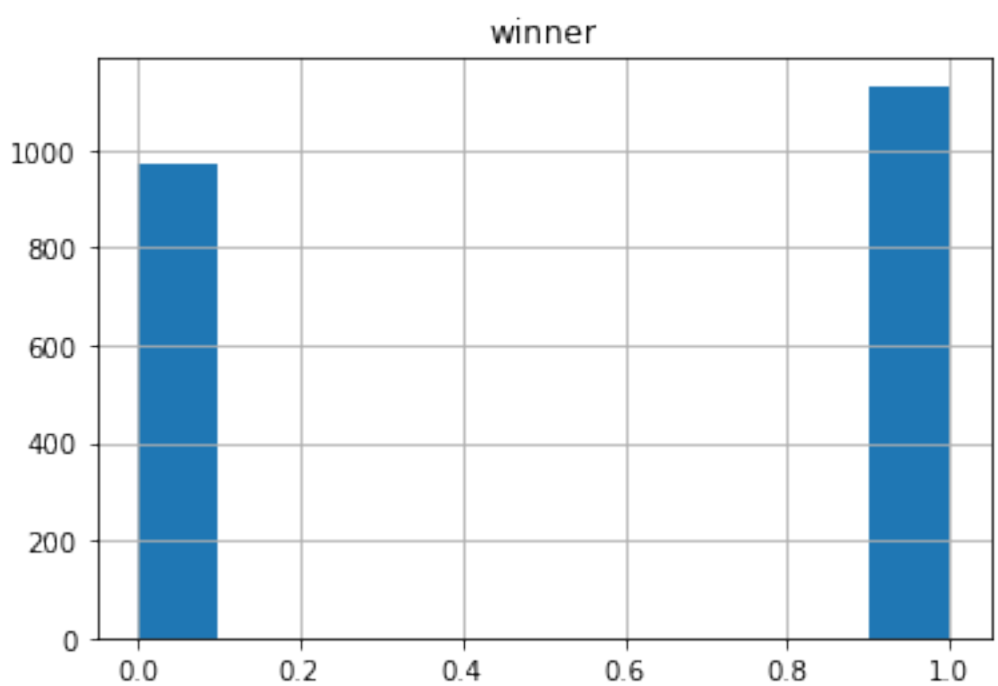

winner
0
200
400
600
800
1000
1200
1400
0.0
0.2
0.4
0.6
0.8
1.0

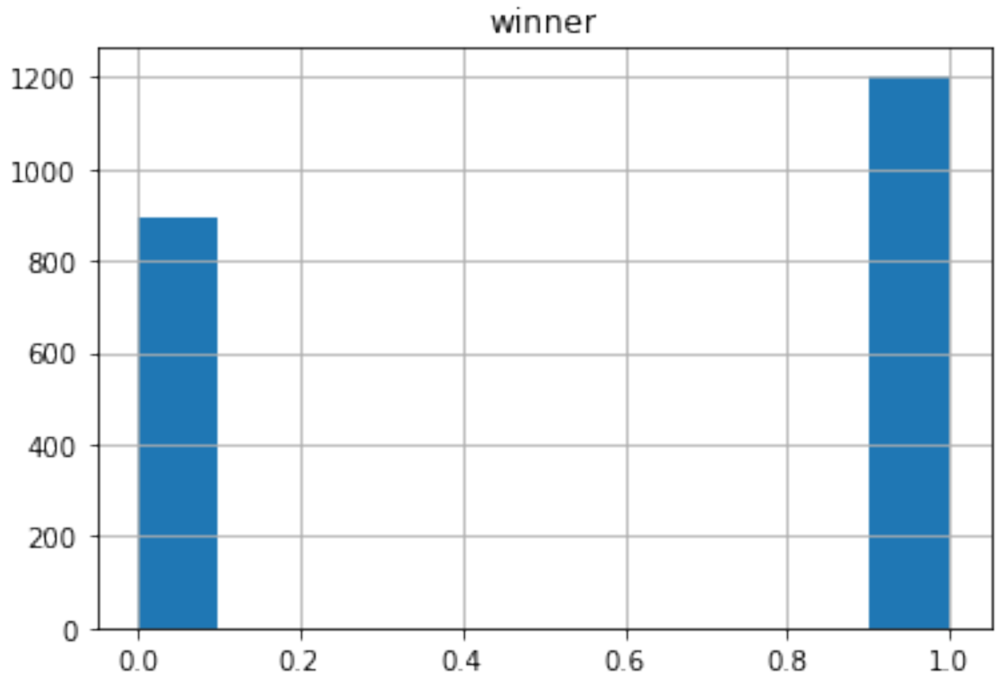

winner
0
200
400
600
800
1000
1200
1400
0.0
0.2
0.4
0.6
0.8
1.0

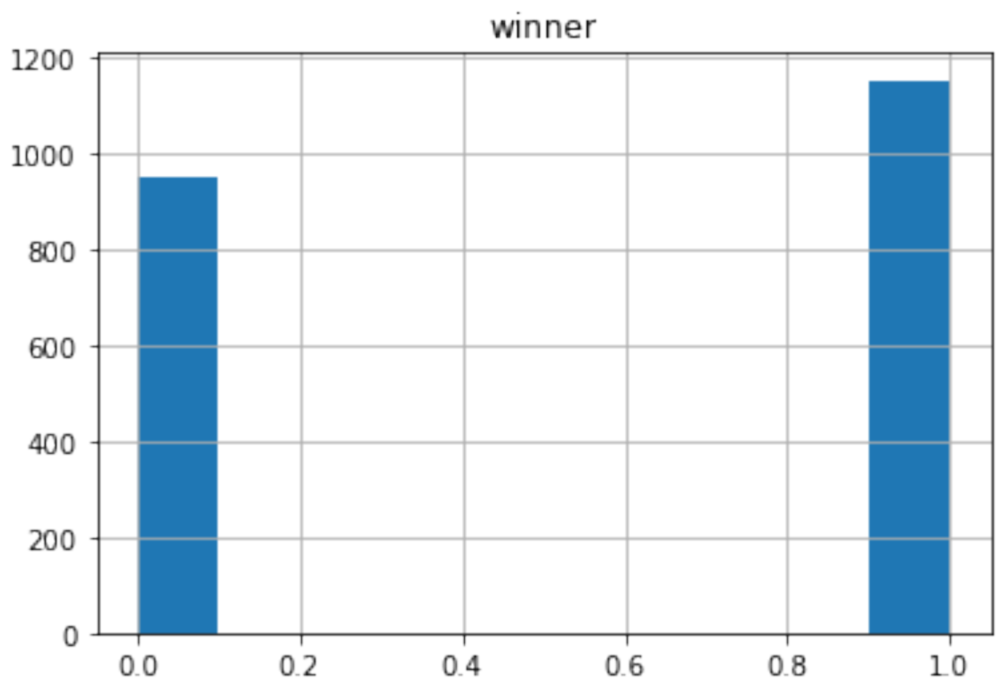

winner
0
200
400
600
800
1000
0.0
0.2
0.4
0.6
0.8
1.0

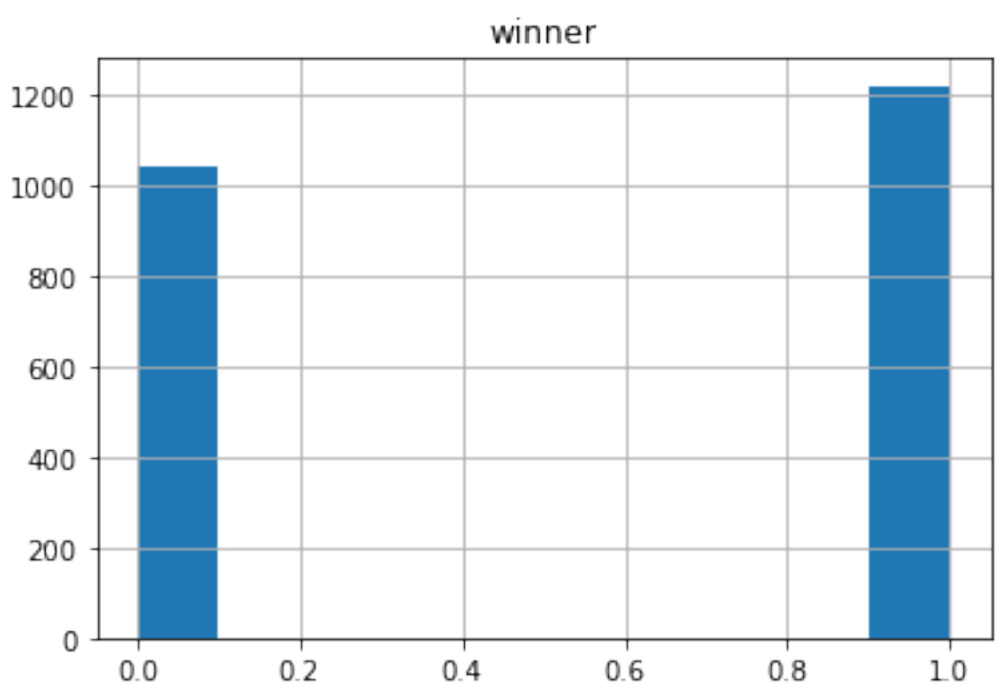

winner
0
200
400
600
800
1000
0.0
0.2
0.4
0.6
0.8
1.0

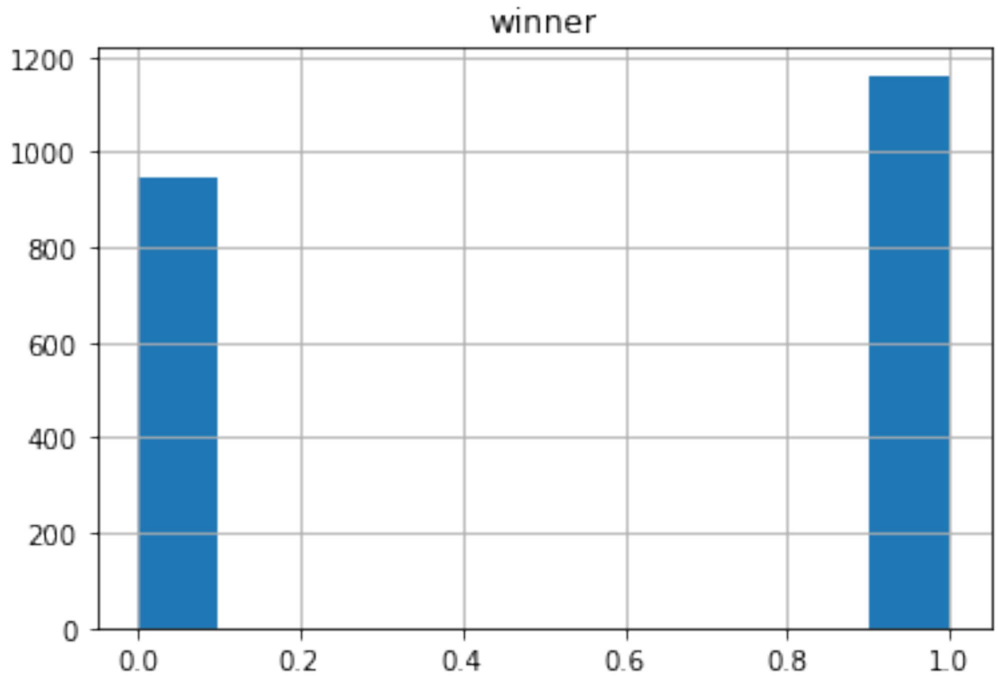

winner
0
200
400
600
800
1000
1200
0.0
0.2
0.4
0.6
0.8
1.0

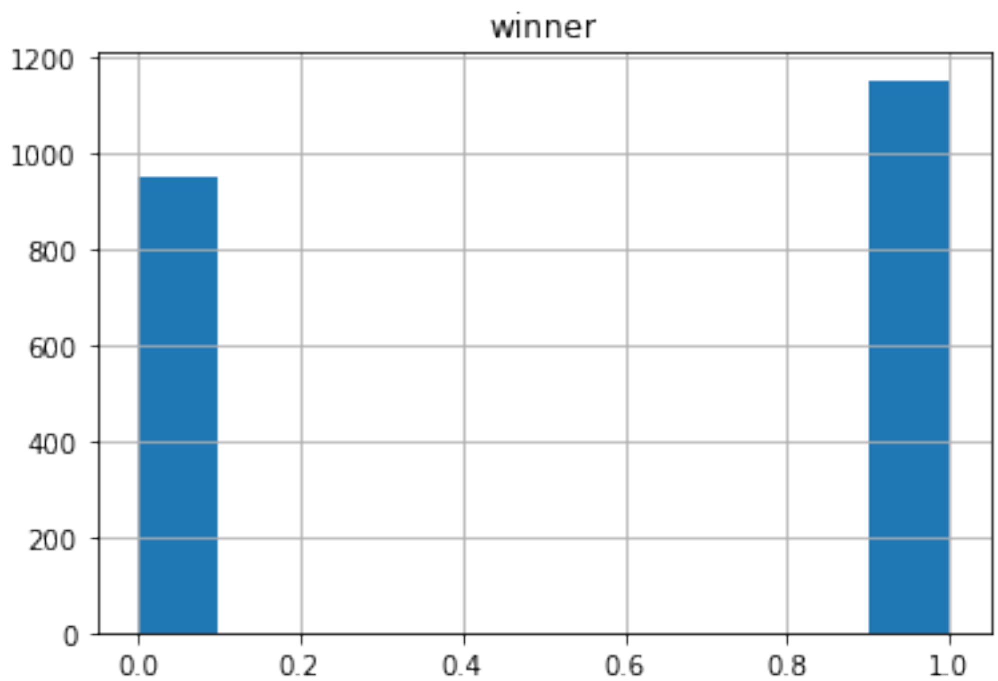

winner
0
200
400
600
800
1000
0.0
0.2
0.4
0.6
0.8
1.0

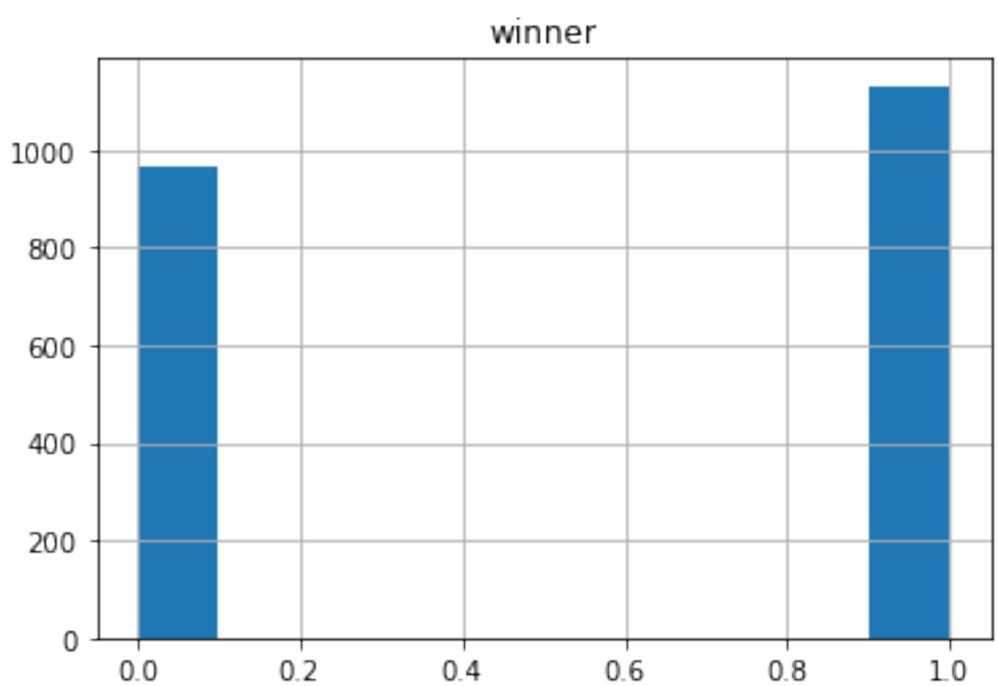

winner
0
200
400
600
800
1000
1200
0.0
0.2
0.4
0.6
0.8
1.0

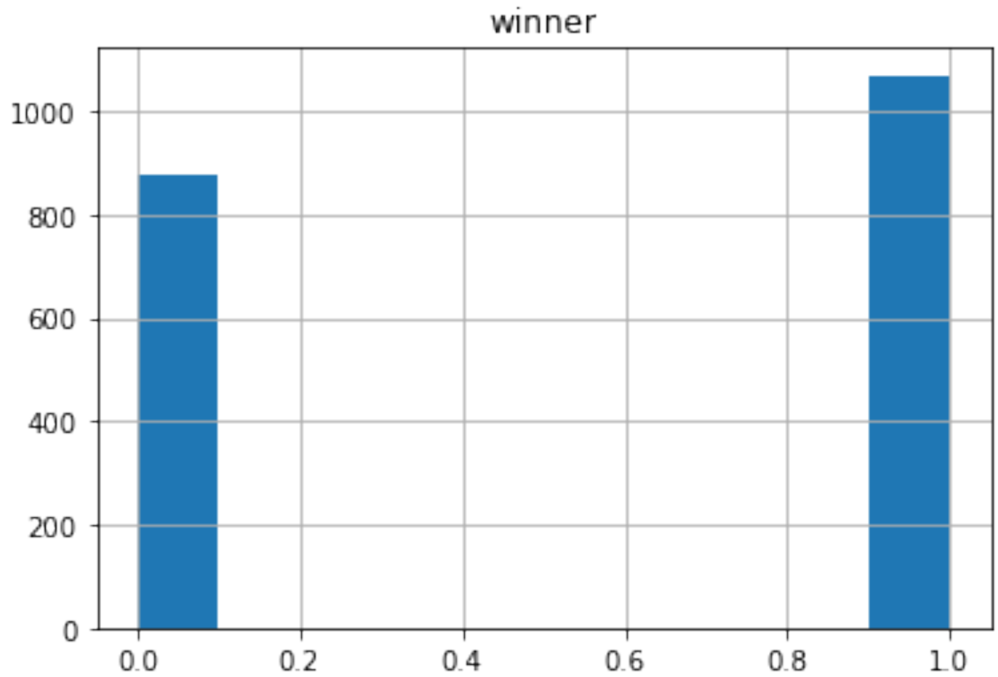

winner
0
200
400
600
800
1000
1200
0.0
0.2
0.4
0.6
0.8
1.0

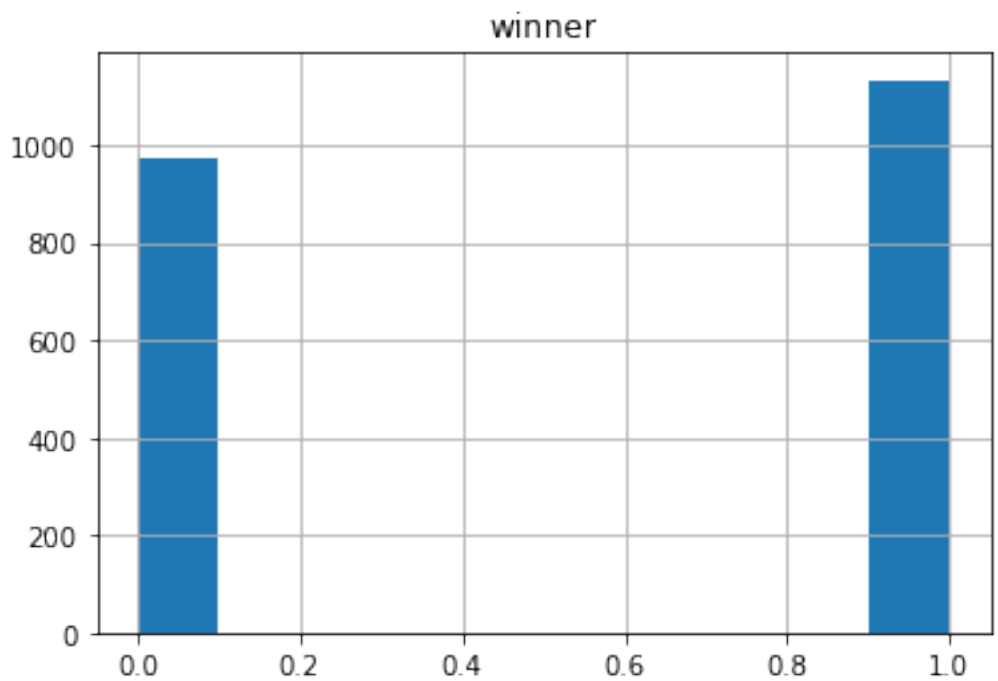

winner
0
200
400
600
800
1000
1200
0.0
0.2
0.4
0.6
0.8
1.0

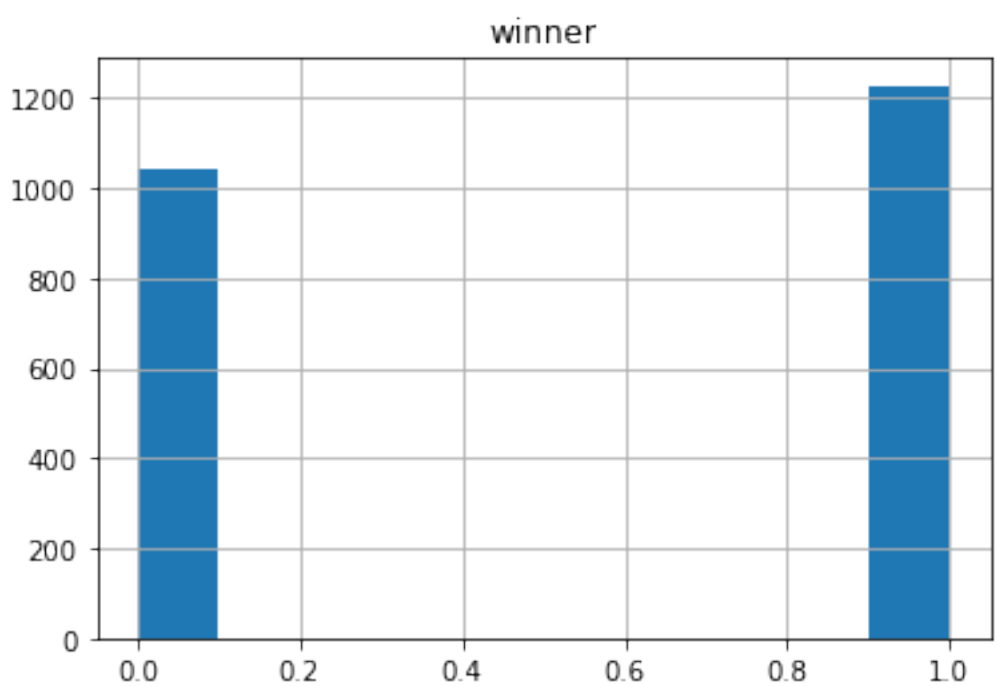

winner
0
200
400
600
800
1000
1200
0.0
0.2
0.4
0.6
0.8
1.0

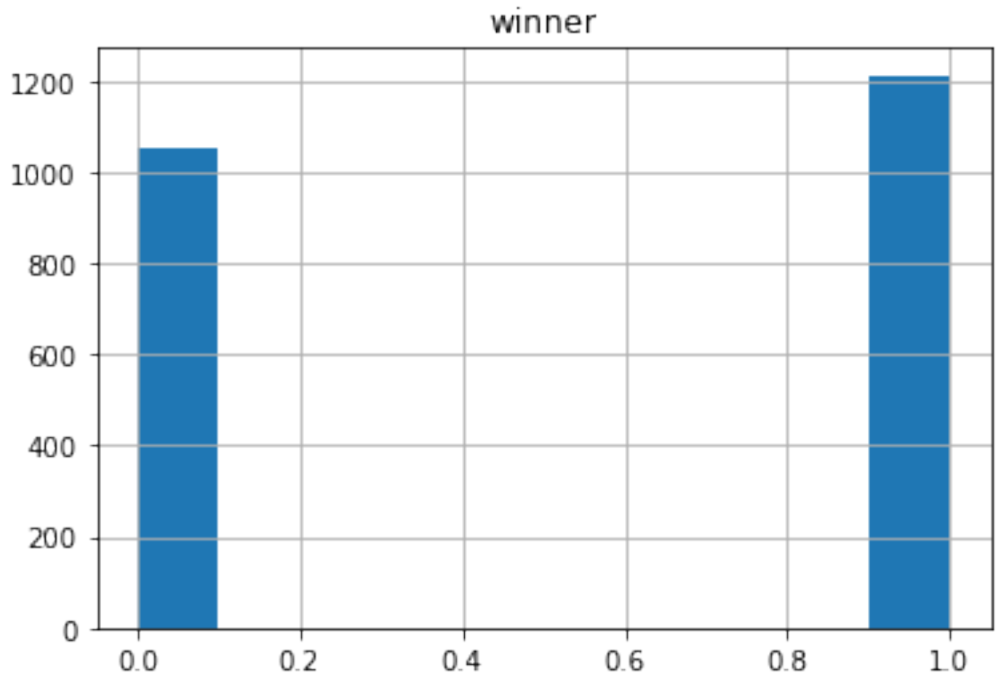

winner
0
200
400
600
800
1000
1200
0.0
0.2
0.4
0.6
0.8
1.0

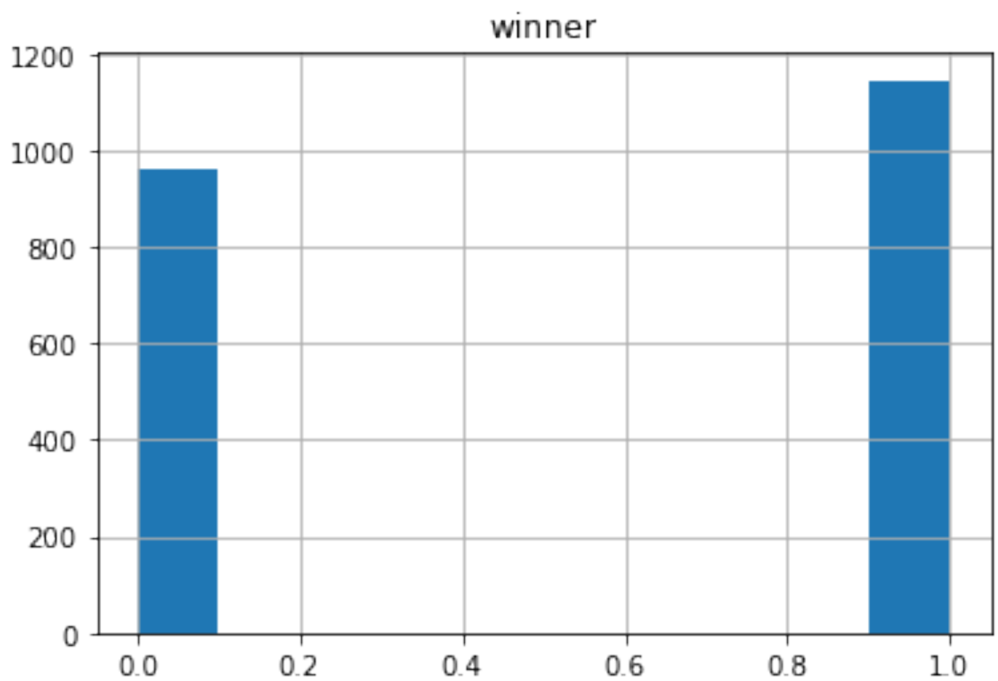

winner
0
200
400
600
800
1000
0.0
0.2
0.4
0.6
0.8
1.0

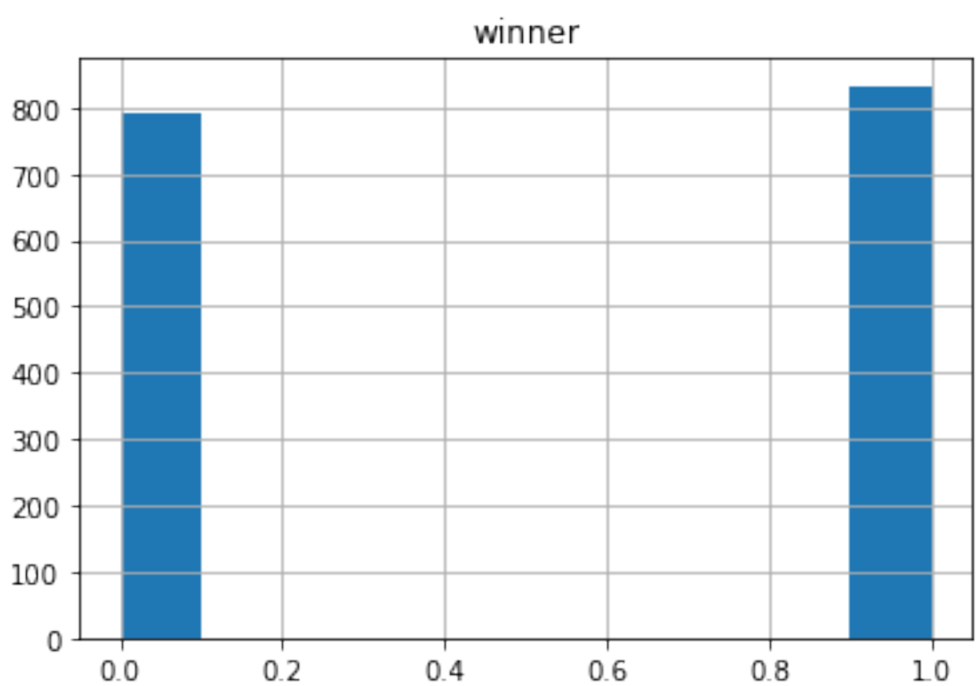

winner
0
200
400
600
800
1000
0.0
0.2
0.4
0.6
0.8
1.0

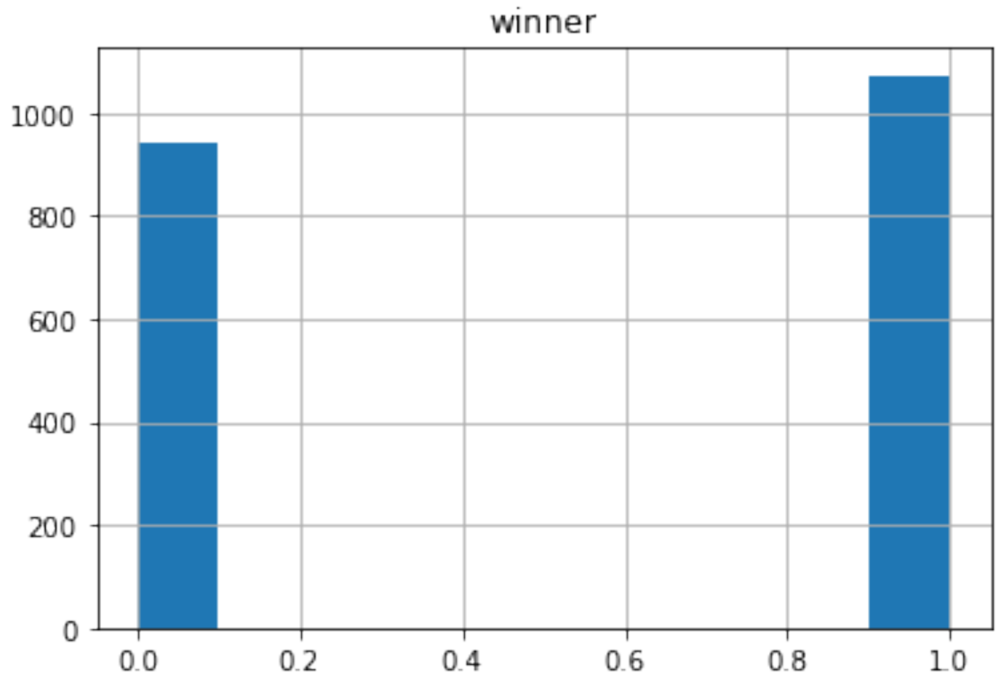

winner
0
200
400
600
800
1000
0.0
0.2
0.4
0.6
0.8
1.0

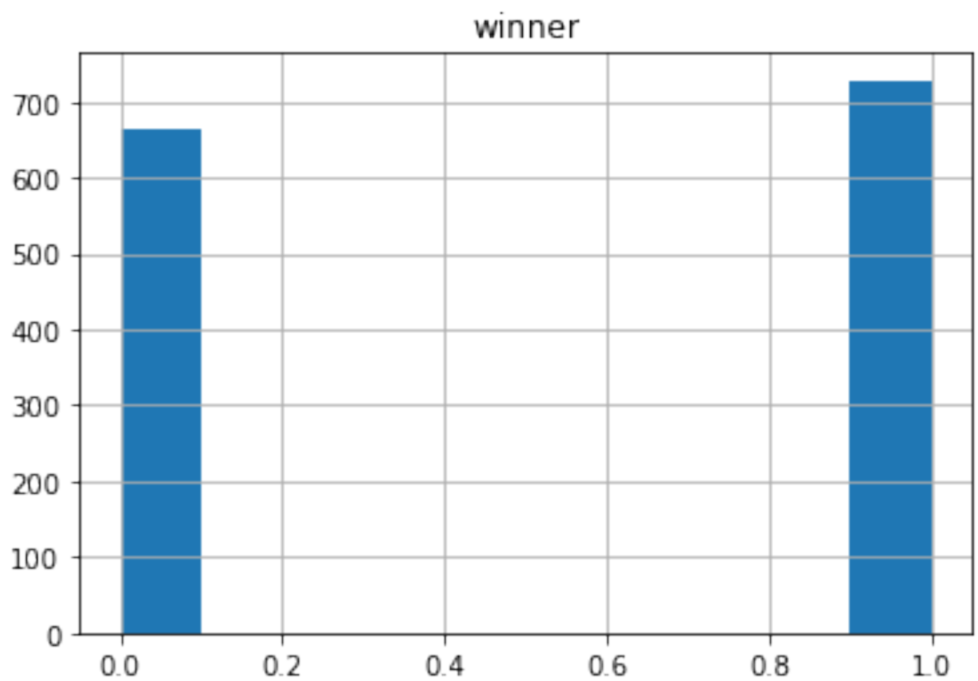

winner
0
200
400
600
800
1000
1200
0.0
0.2
0.4
0.6
0.8
1.0

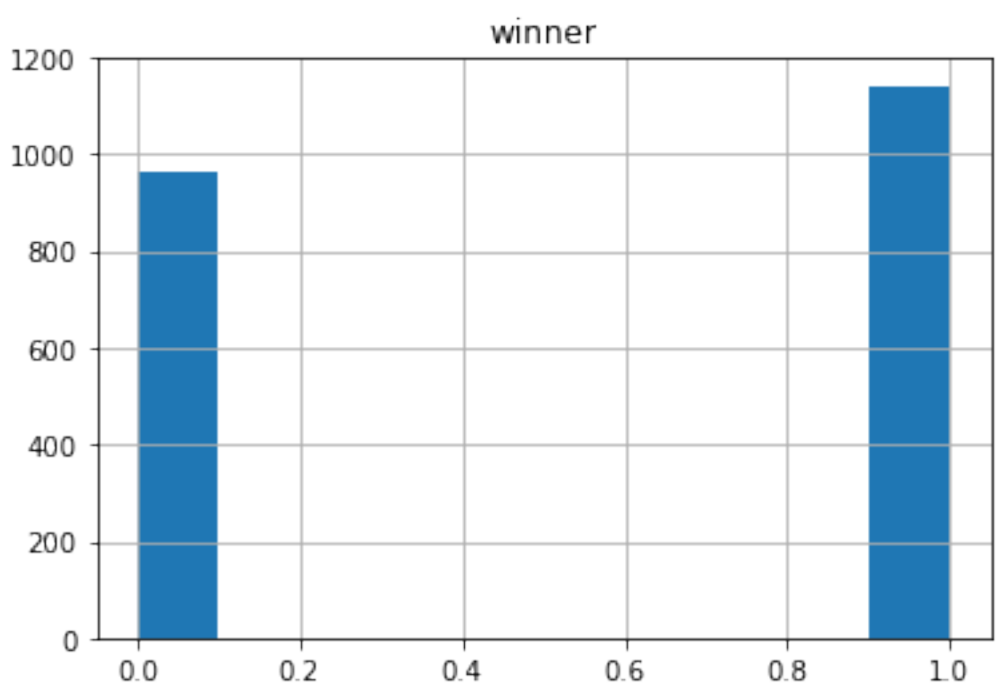

winner
1200
1000
800
600
400
200
0
0.0
0.2
0.4
0.6
0.8
1.0

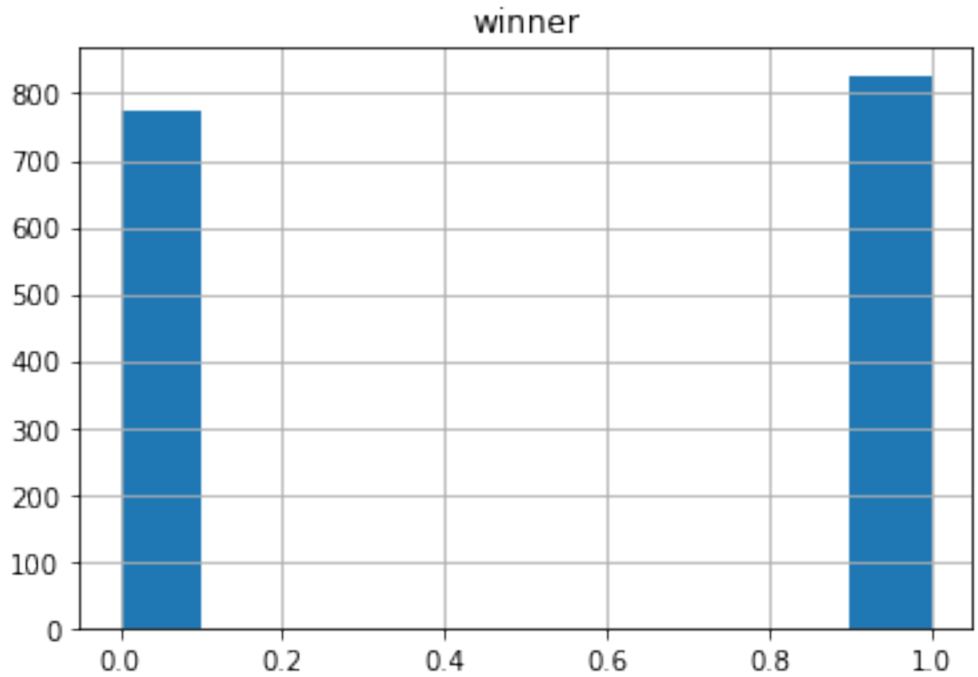

winner
1200
1000
800
600
400
200
0
0.0
0.2
0.4
0.6
0.8
1.0

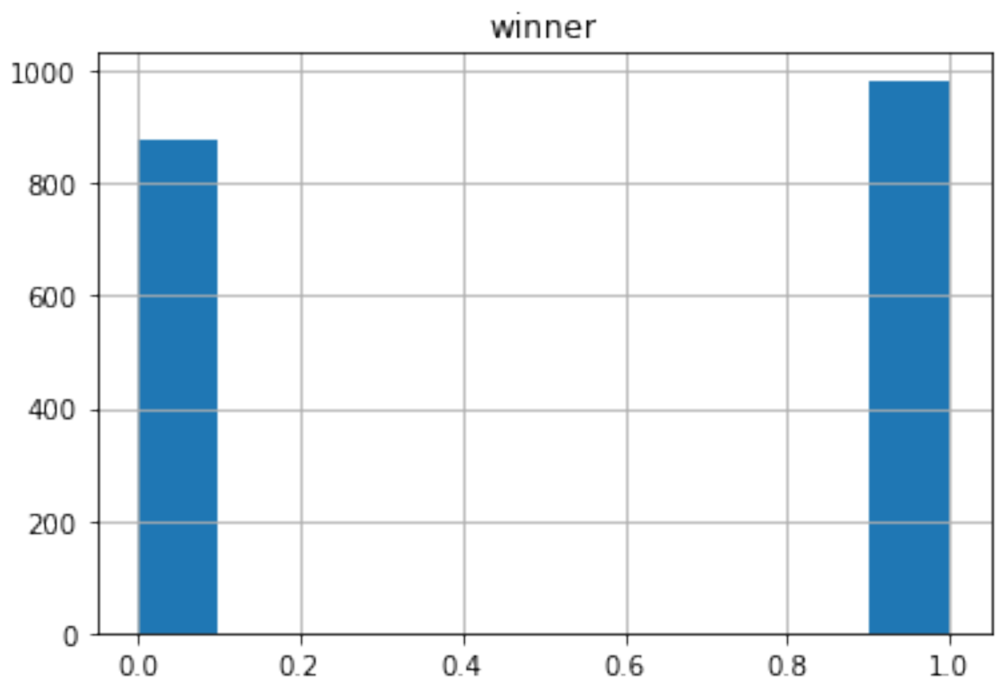

winner
800
700
600
500
400
300
200
100
0
0.0
0.2
0.4
0.6
0.8
1.0

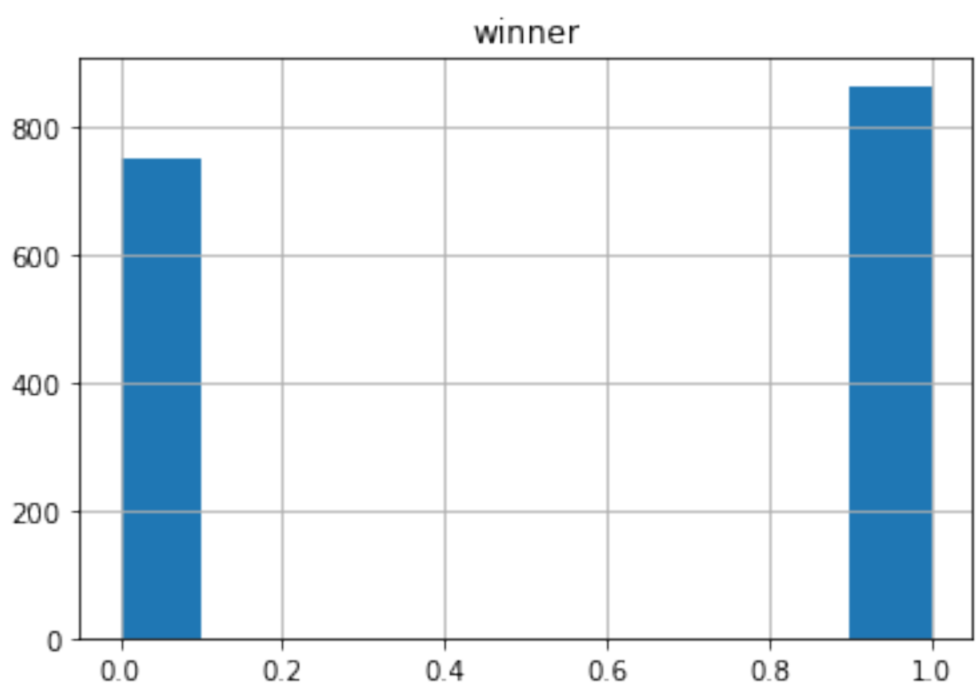

winner
0
200
400
600
800
1000
0.0
0.2
0.4
0.6
0.8
1.0

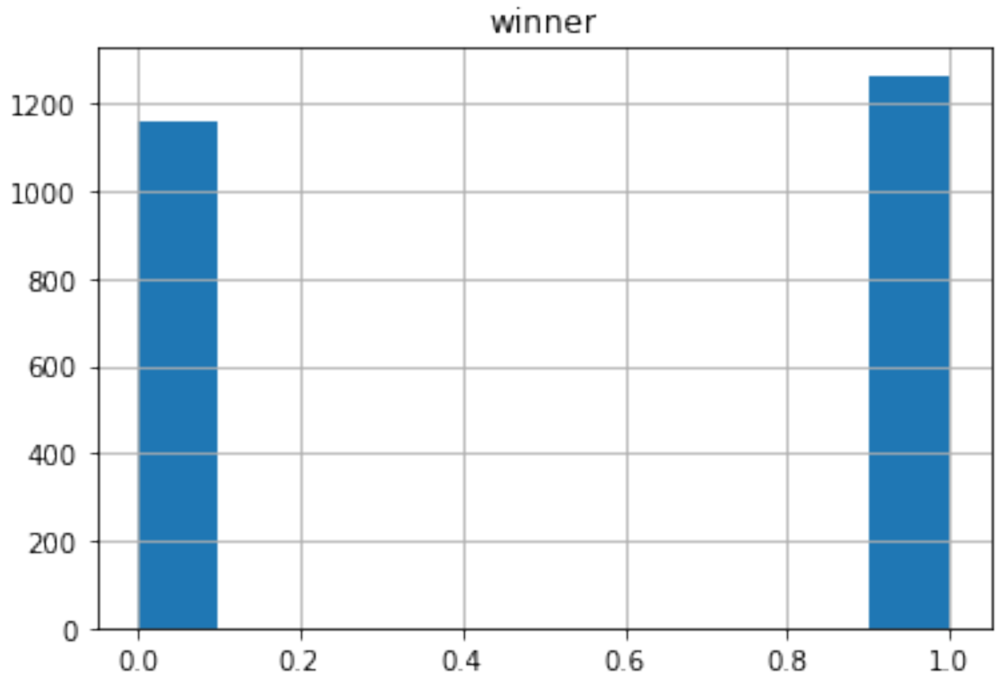

winner
0
100
200
300
400
500
600
700
0.0
0.2
0.4
0.6
0.8
1.0

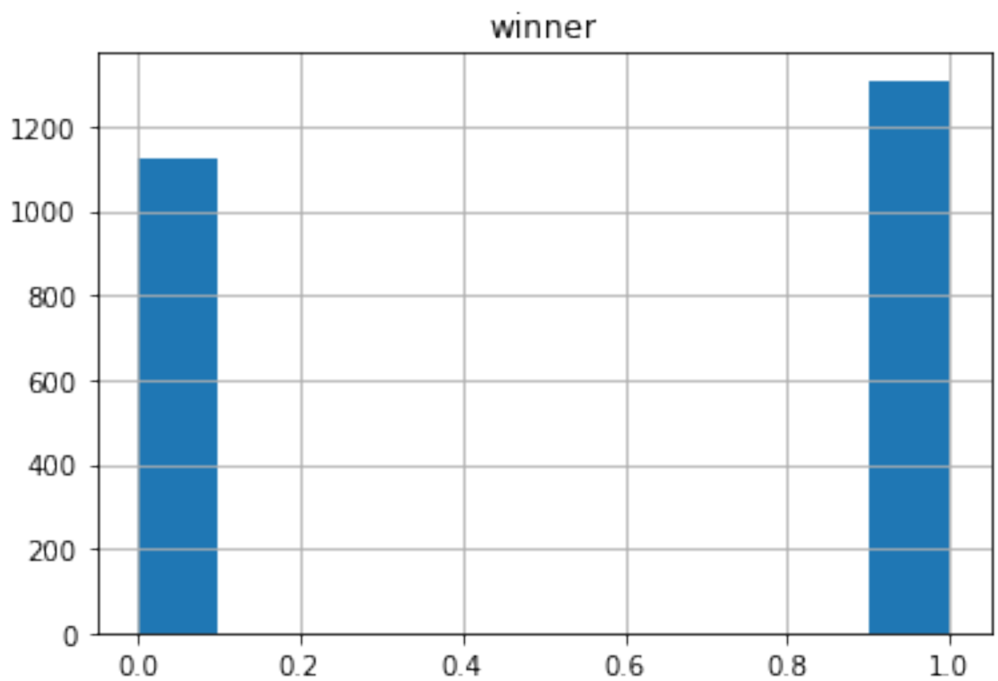

winner
0
200
400
600
800
1000
1200
0.0
0.2
0.4
0.6
0.8
1.0

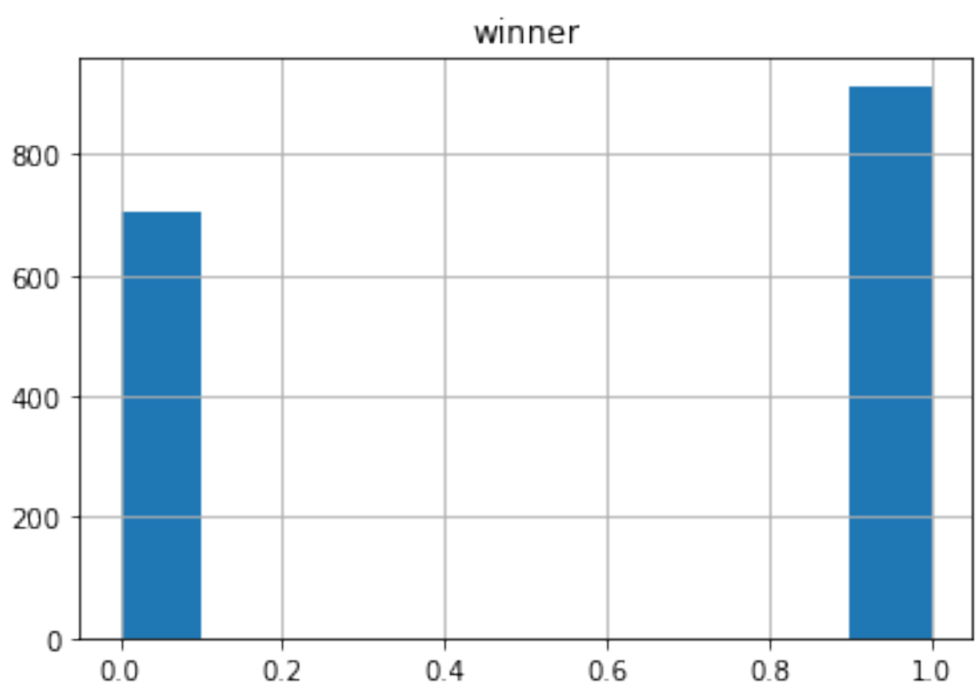

winner
0
100
200
300
400
500
600
700
800
0.0
0.2
0.4
0.6
0.8
1.0

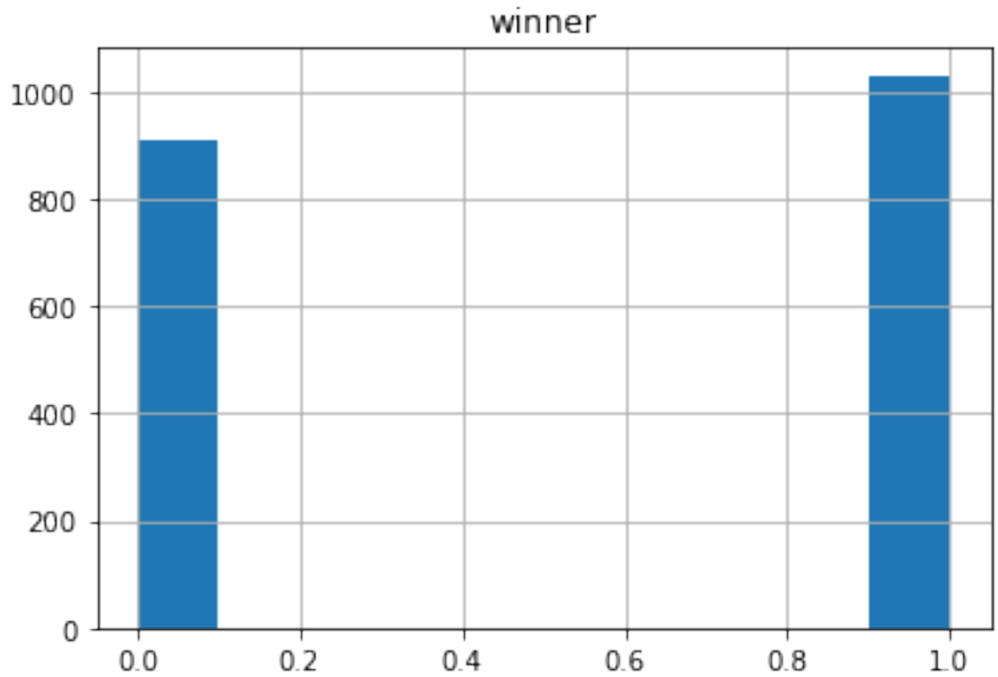

winner
0
200
400
600
800
1000
0.0
0.2
0.4
0.6
0.8
1.0

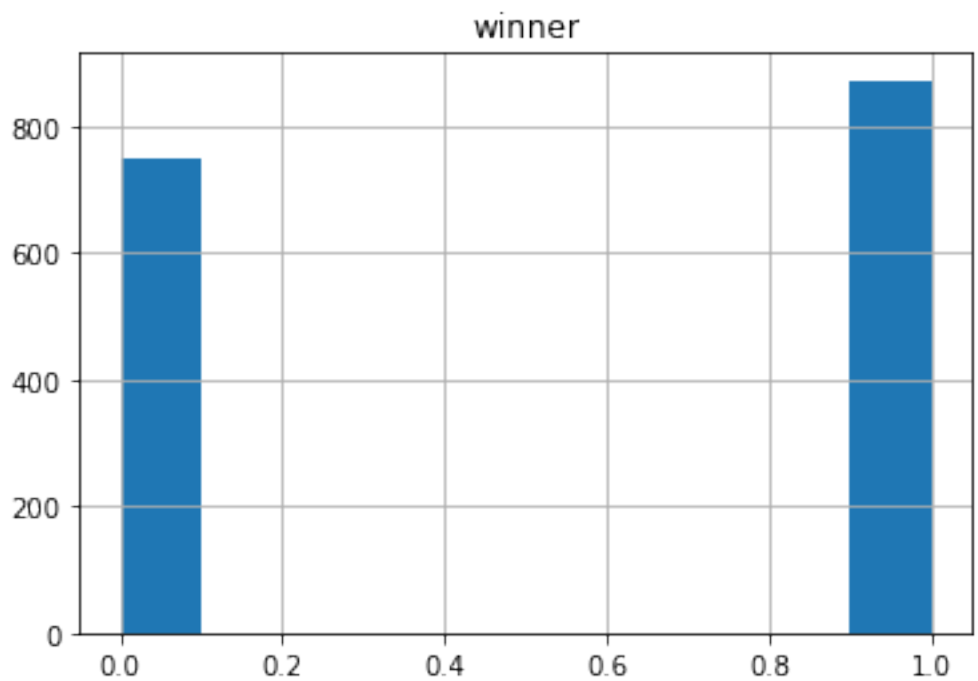

winner
0
200
400
600
800
0.0
0.2
0.4
0.6
0.8
1.0

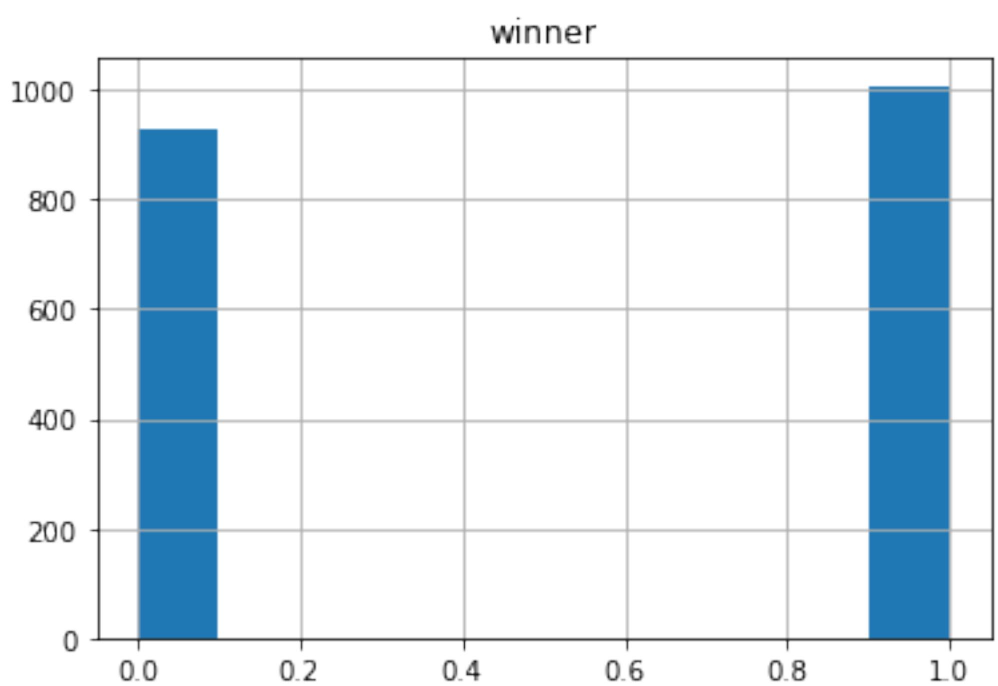

winner
0
200
400
600
800
1000
1200
0.0
0.2
0.4
0.6
0.8
1.0

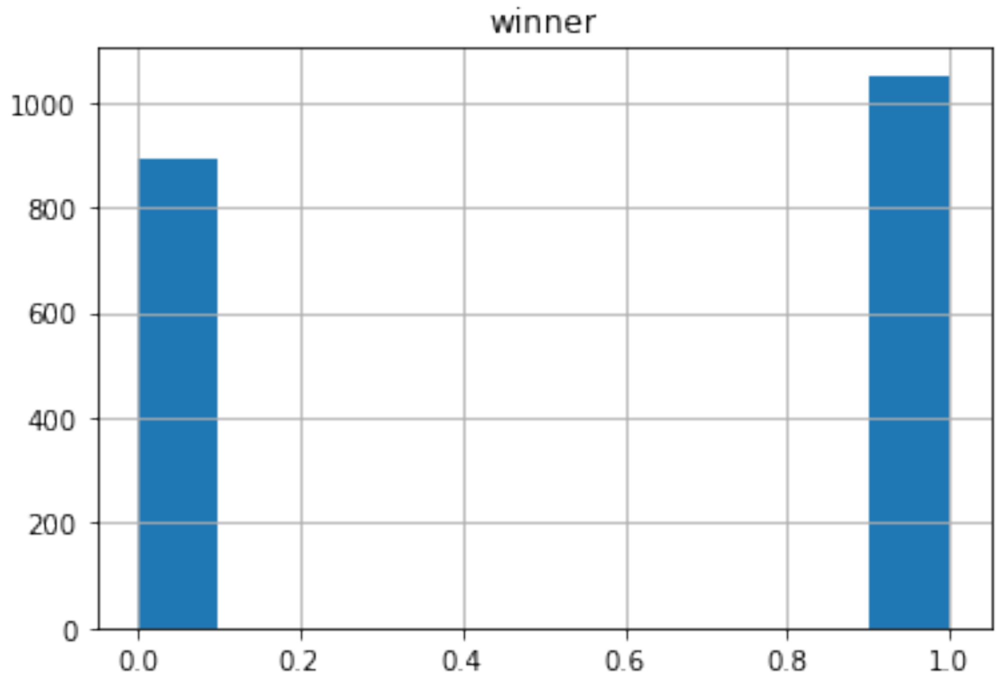

winner
0
200
400
600
800
1000
1200
0.0
0.2
0.4
0.6
0.8
1.0

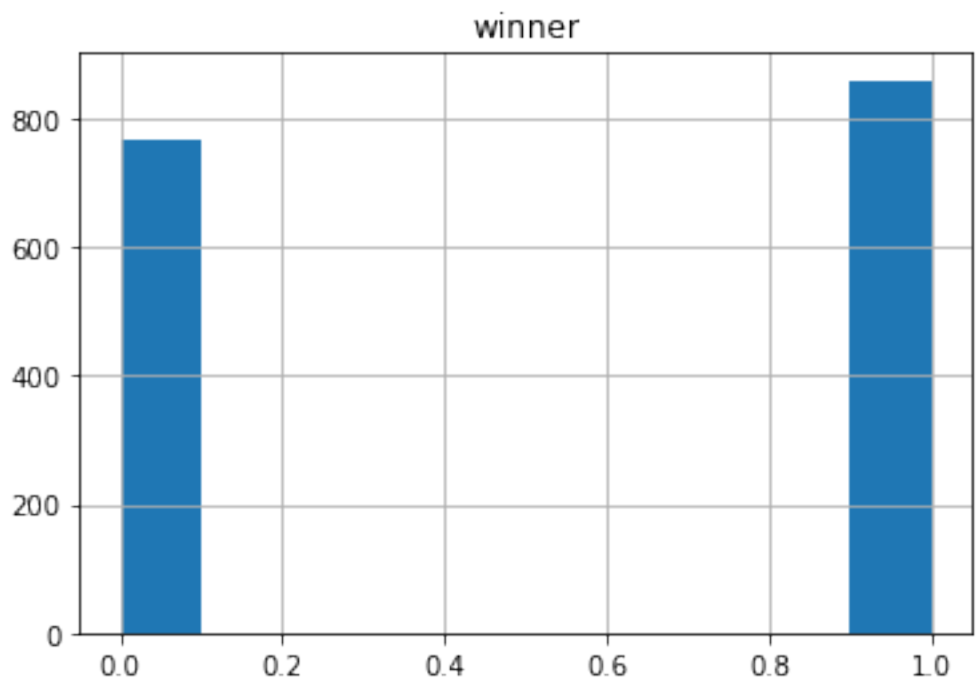

winner
0
200
400
600
800
0.0
0.2
0.4
0.6
0.8
1.0

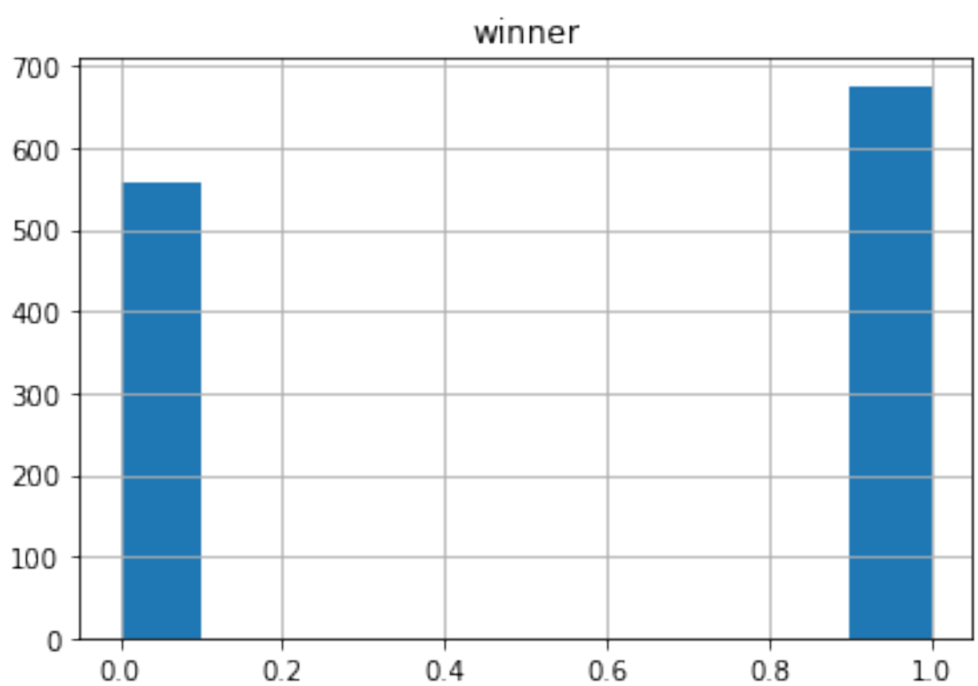

winner
0
200
400
600
800
1000
0.0
0.2
0.4
0.6
0.8
1.0

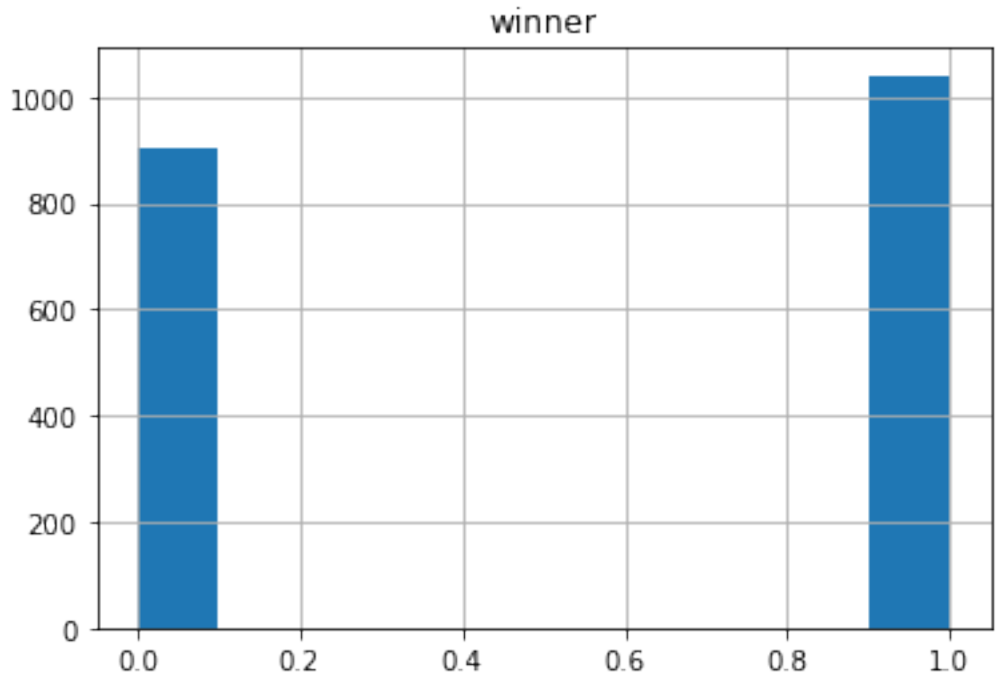

winner
0
200
400
600
800
0.0
0.2
0.4
0.6
0.8
1.0

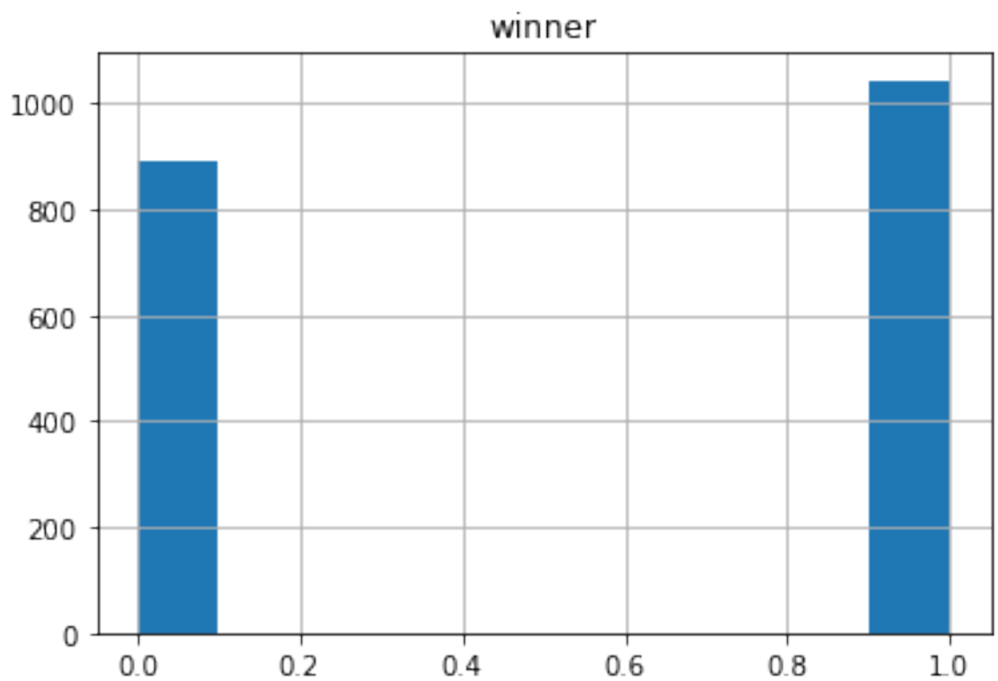

winner
0
200
400
600
800
1000
0.0
0.2
0.4
0.6
0.8
1.0

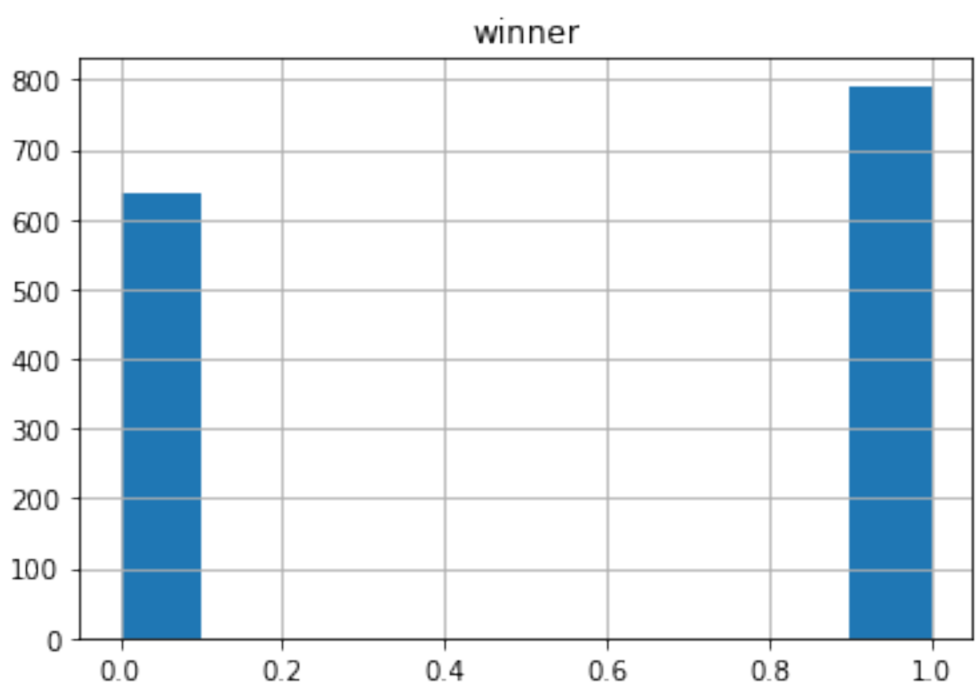

winner
0
200
400
600
800
1000
0.0
0.2
0.4
0.6
0.8
1.0

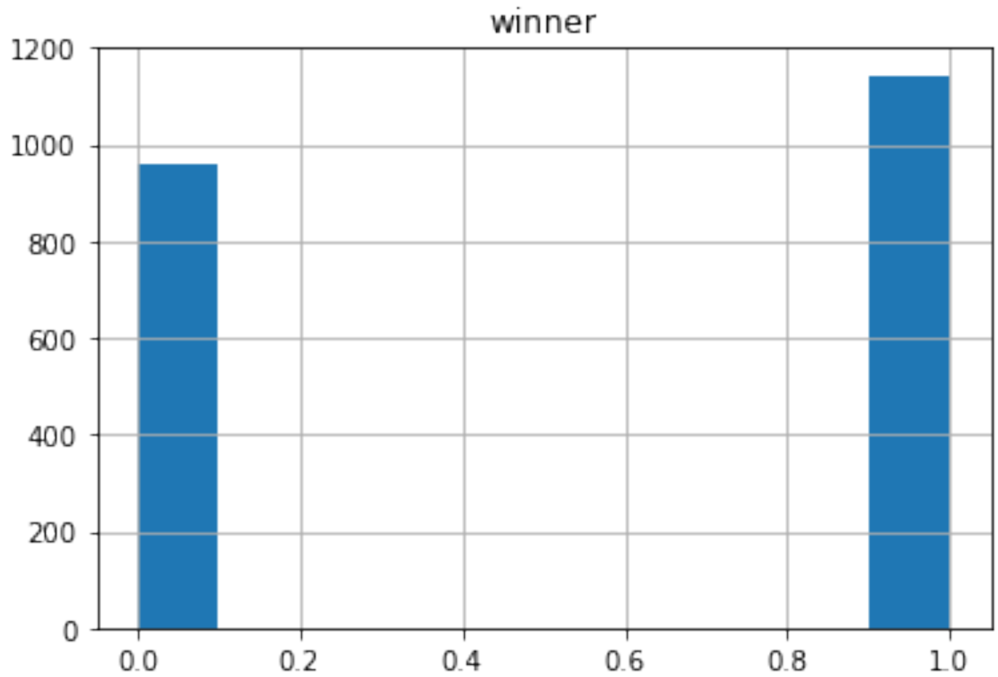

winner
0
200
400
600
800
0.0
0.2
0.4
0.6
0.8
1.0

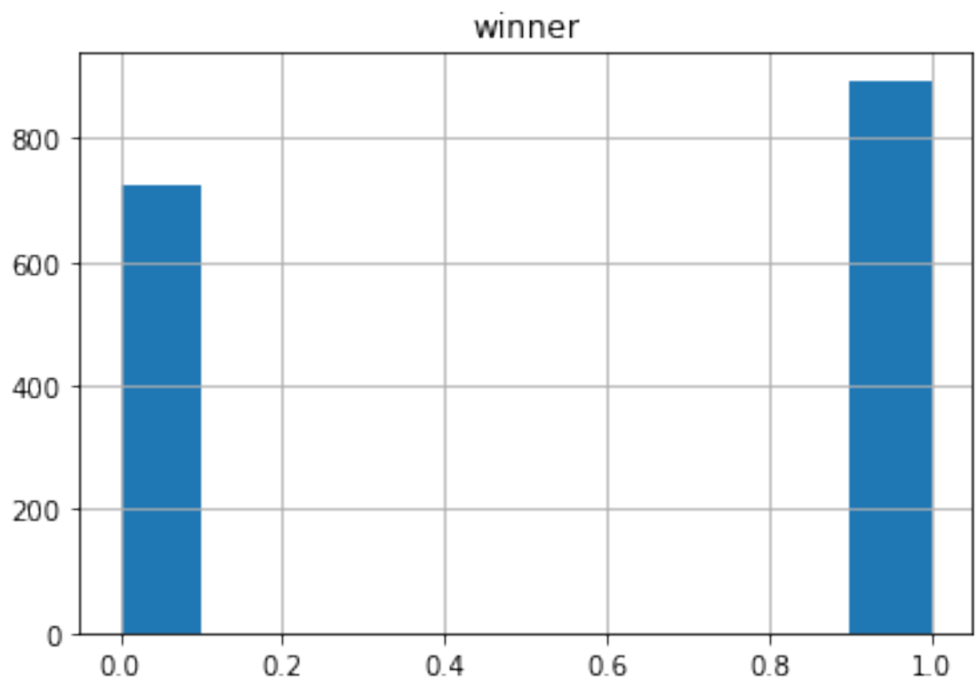

winner
0
100
200
300
400
500
600
700
0.0
0.2
0.4
0.6
0.8
1.0

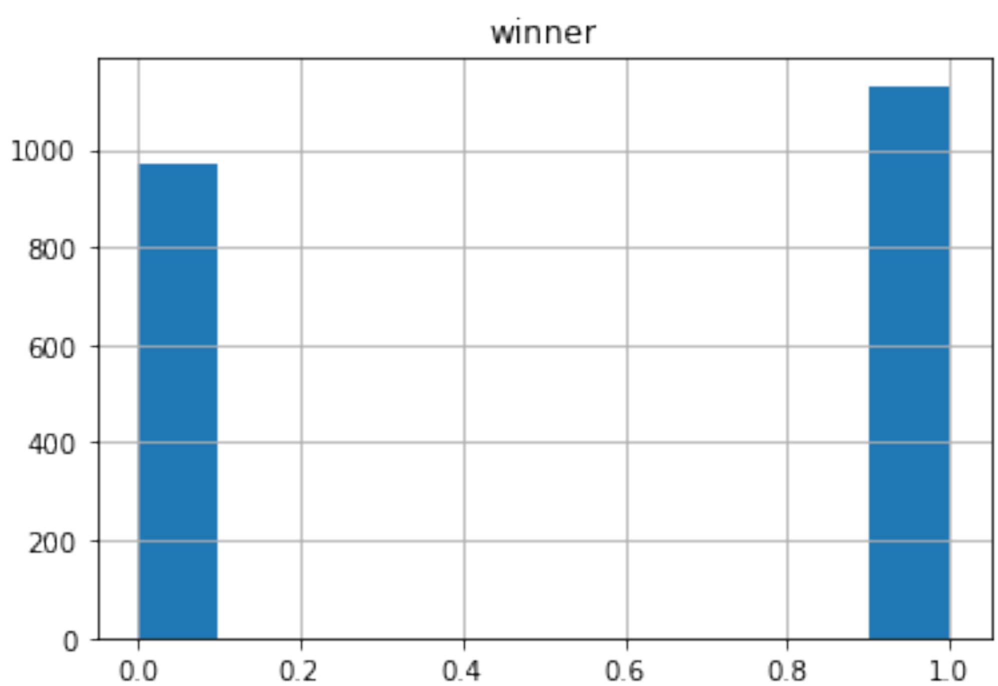

winner
0
200
400
600
800
1000
0.0
0.2
0.4
0.6
0.8
1.0

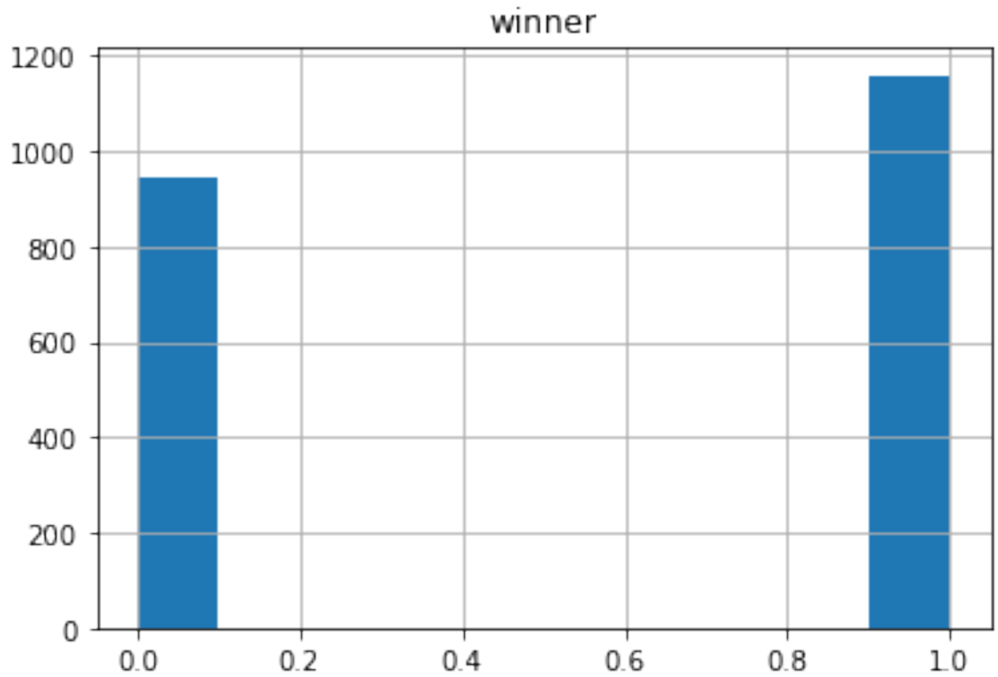

winner
0
200
400
600
800
1000
0.0
0.2
0.4
0.6
0.8
1.0

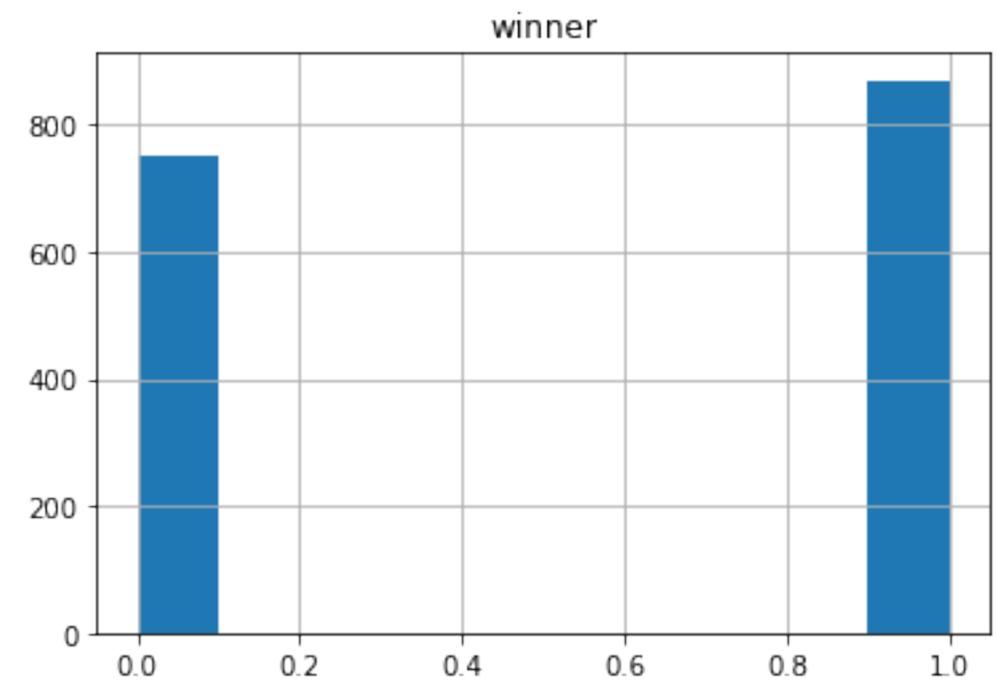

winner
0
100
200
300
400
500
600
700
800
0.0
0.2
0.4
0.6
0.8
1.0

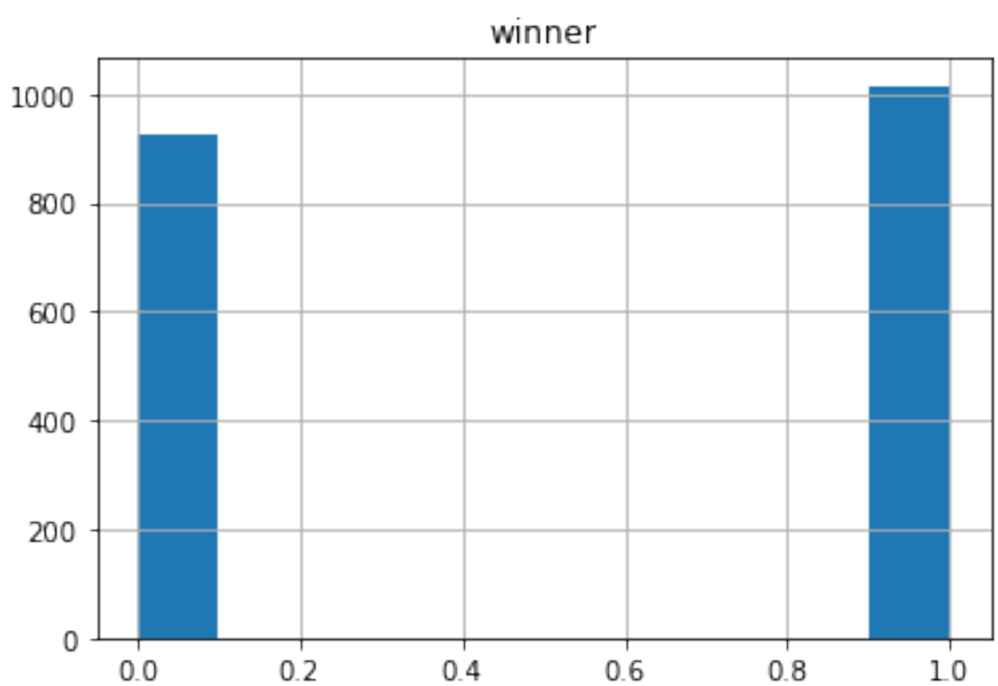

winner
1200
1000
800
600
400
200
0
0.0
0.2
0.4
0.6
0.8
1.0

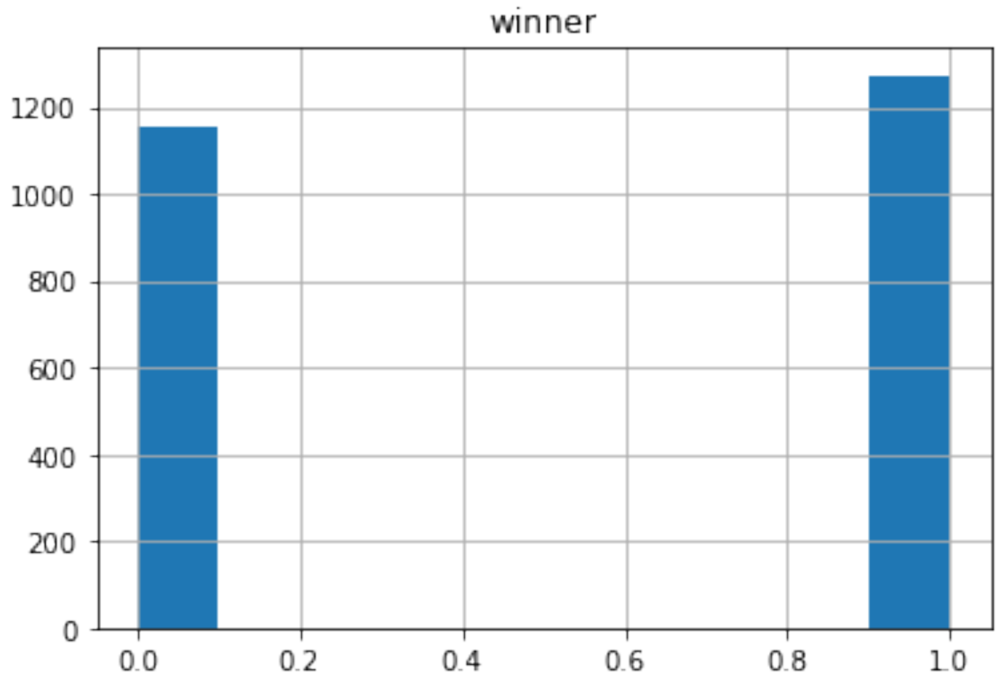

winner
800
600
400
200
0
0.0
0.2
0.4
0.6
0.8
1.0

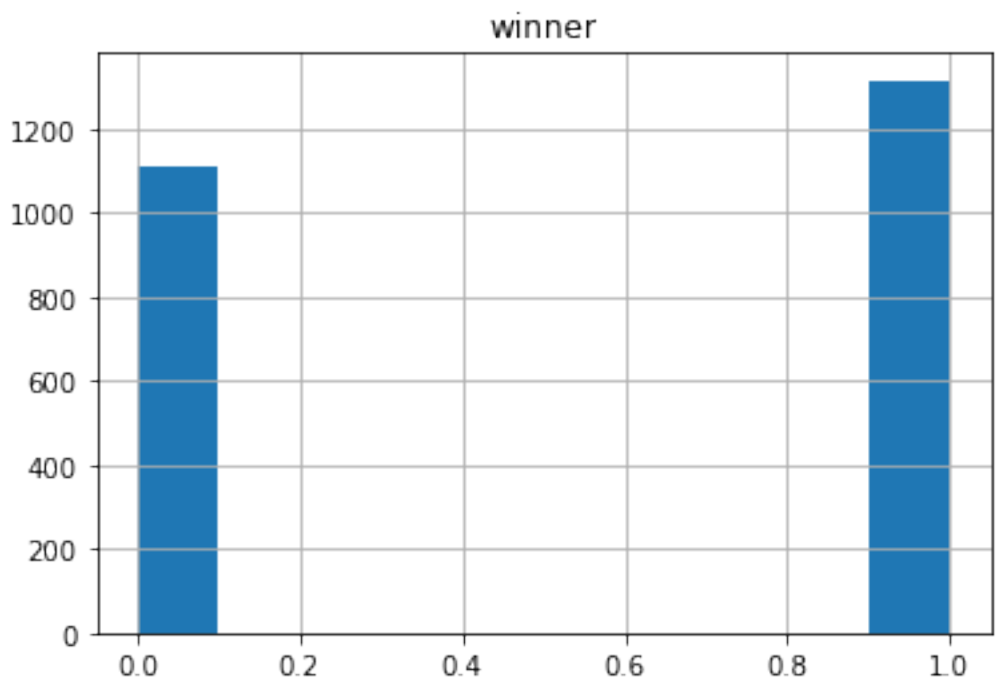

winner
1000
800
600
400
200
0
0.0
0.2
0.4
0.6
0.8
1.0

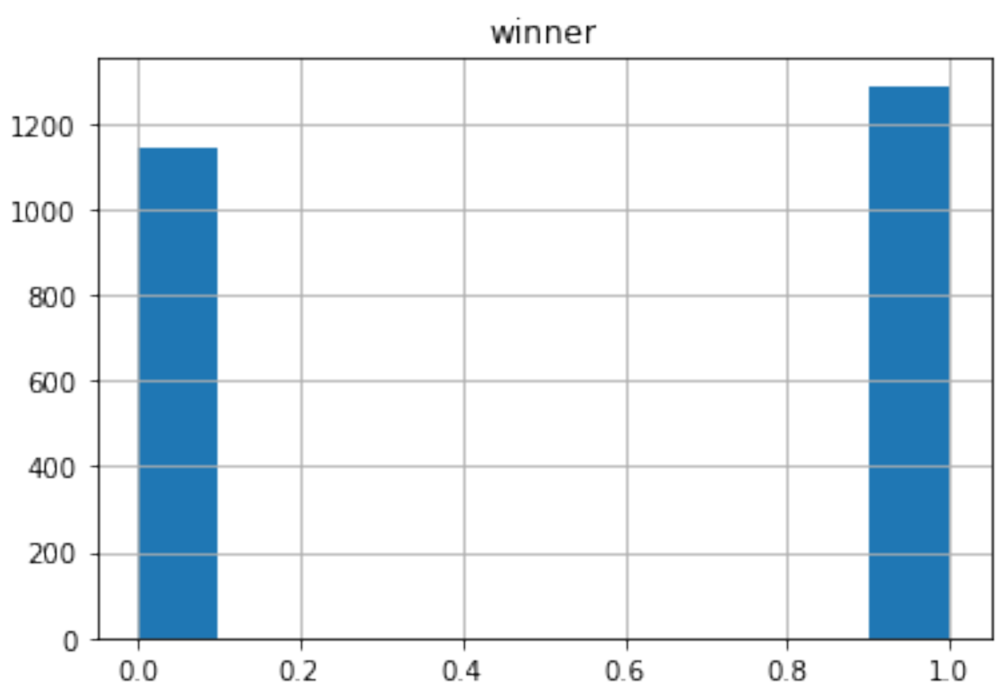

winner
0
200
400
600
800
1000
1200
0.0
0.2
0.4
0.6
0.8
1.0

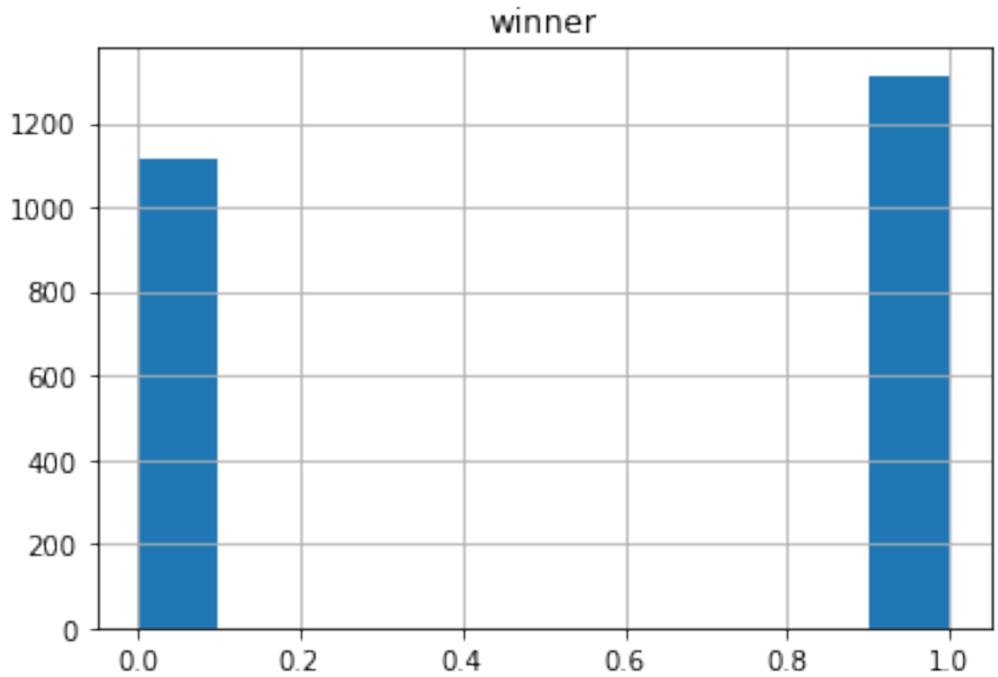

winner
0
200
400
600
800
0.0
0.2
0.4
0.6
0.8
1.0

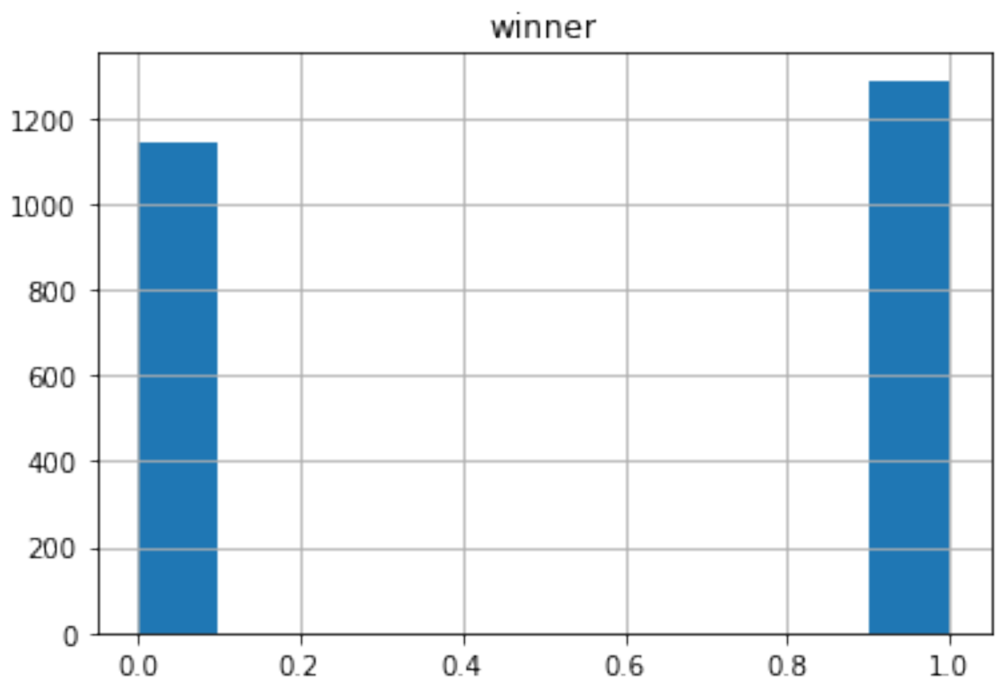

winner
0
200
400
600
800
1000
0.0
0.2
0.4
0.6
0.8
1.0

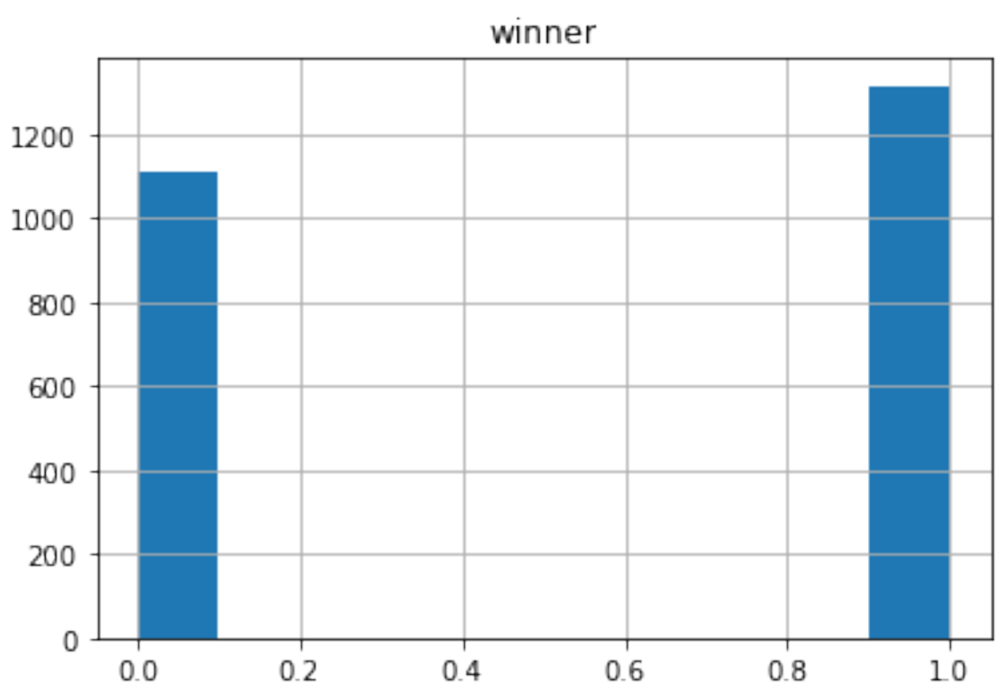

winner
0
200
400
600
800
1000
1200
0.0
0.2
0.4
0.6
0.8
1.0

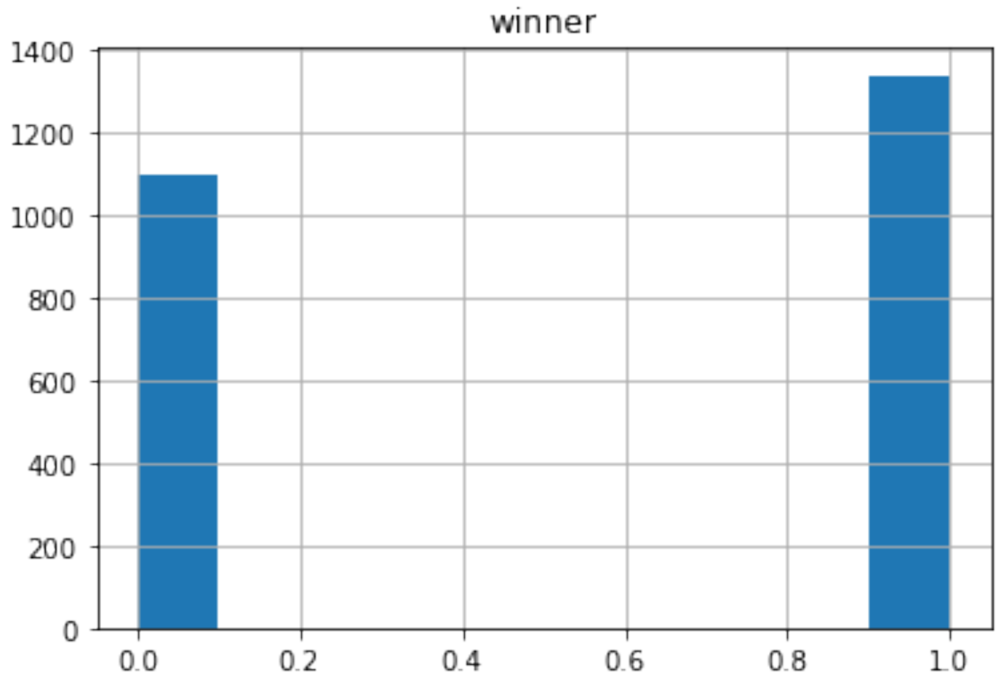

winner
0
200
400
600
800
1000
1200
0.0
0.2
0.4
0.6
0.8
1.0

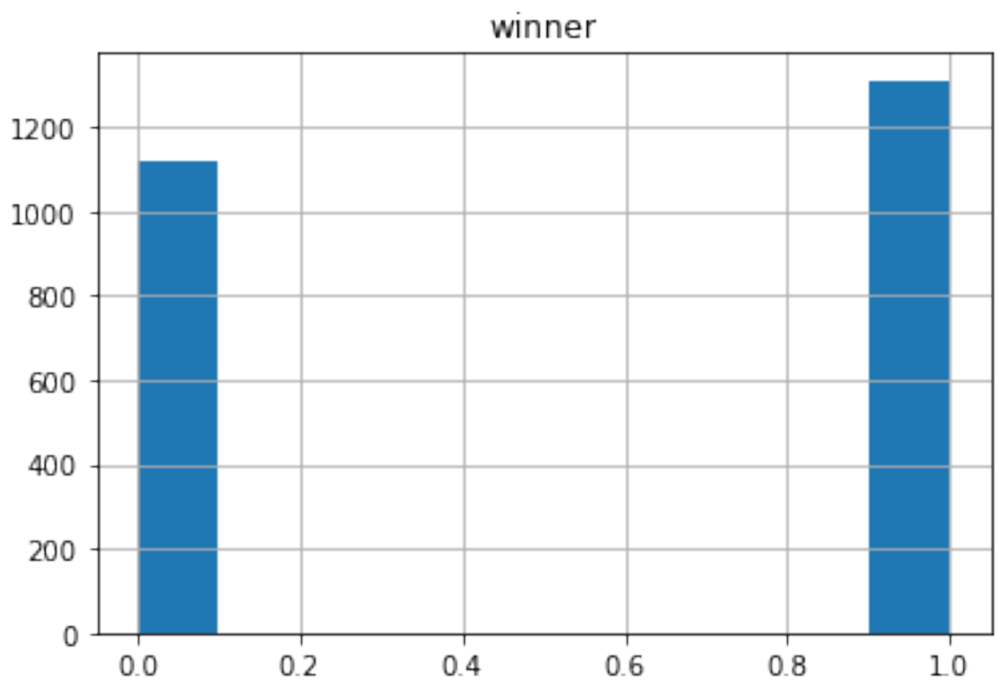

winner
0
200
400
600
800
1000
1200
0.0
0.2
0.4
0.6
0.8
1.0

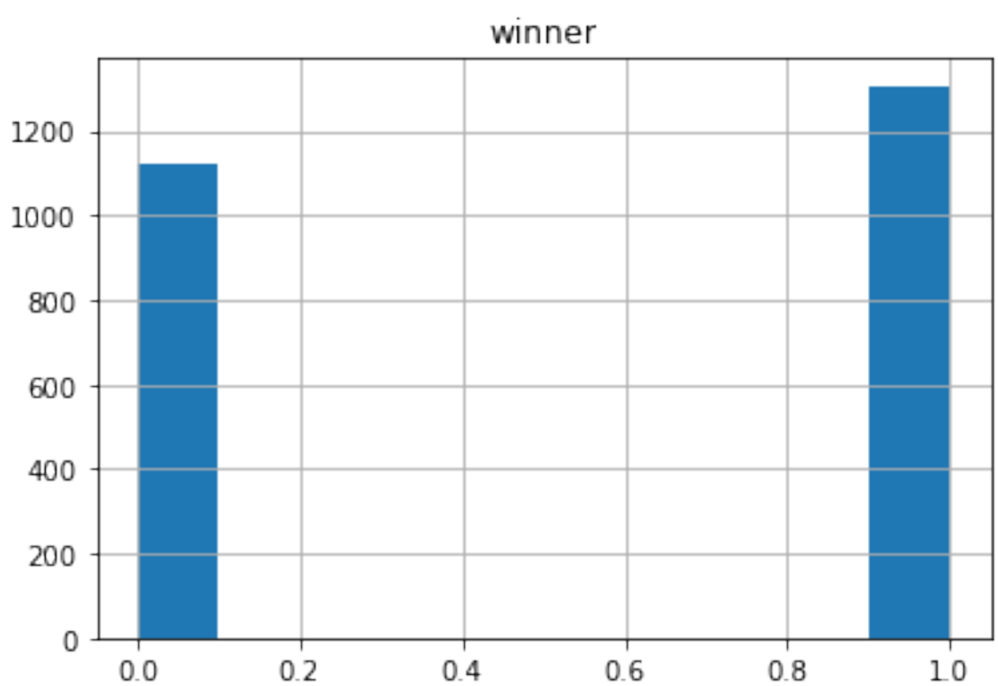

winner
0
200
400
600
800
1000
1200
0.0
0.2
0.4
0.6
0.8
1.0

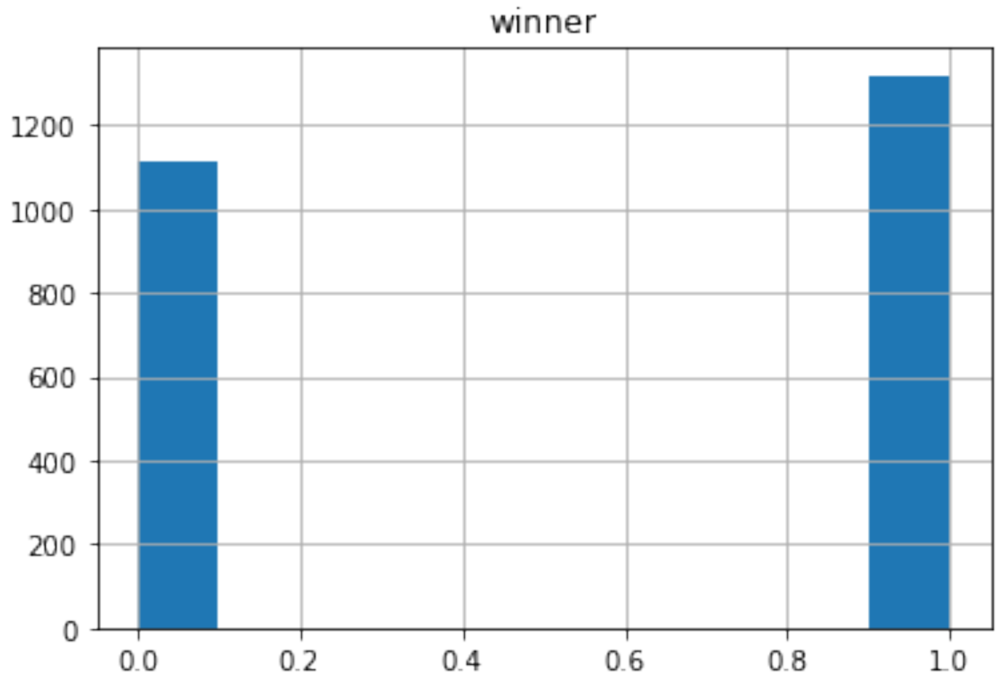

winner
0
200
400
600
800
1000
1200
0.0
0.2
0.4
0.6
0.8
1.0

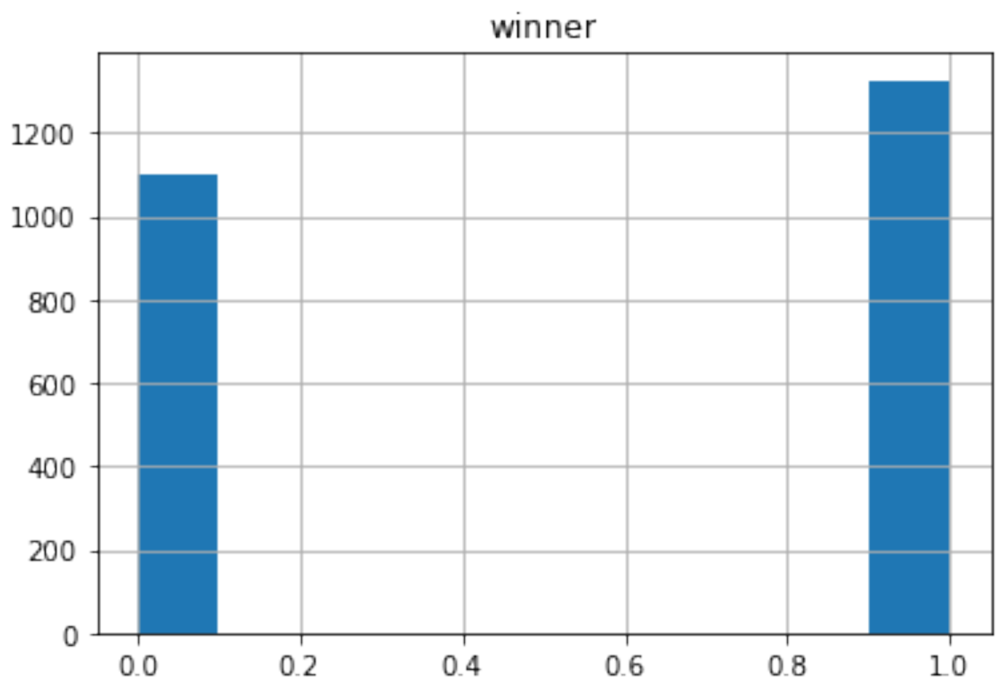

winner
0
200
400
600
800
1000
1200
0.0
0.2
0.4
0.6
0.8
1.0

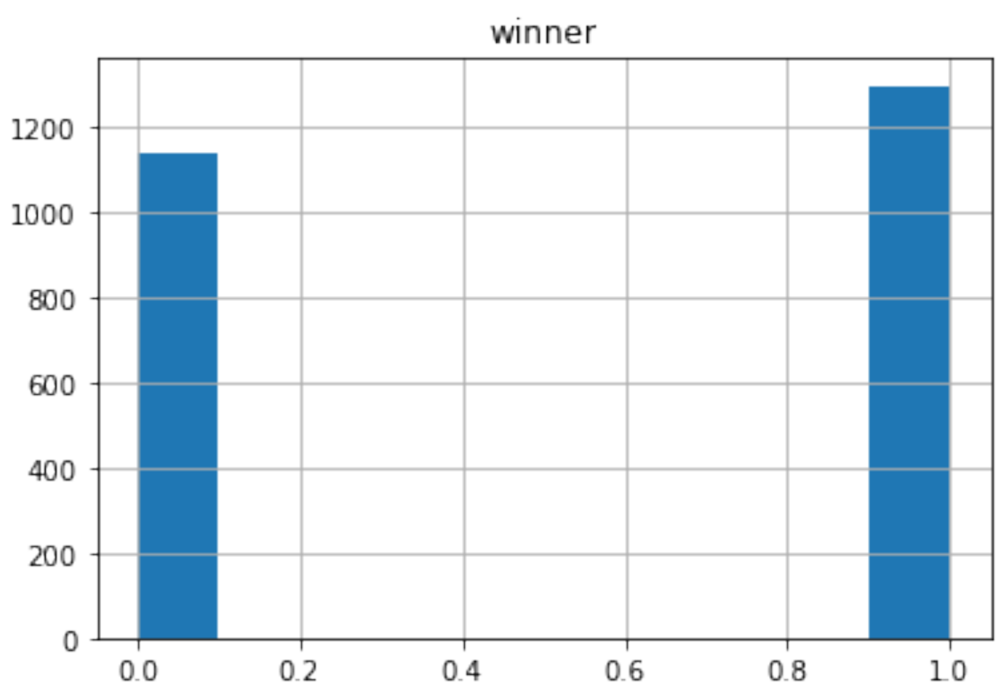

winner
1400
1200
1000
800
600
400
200
0
0.0
0.2
0.4
0.6
0.8
1.0

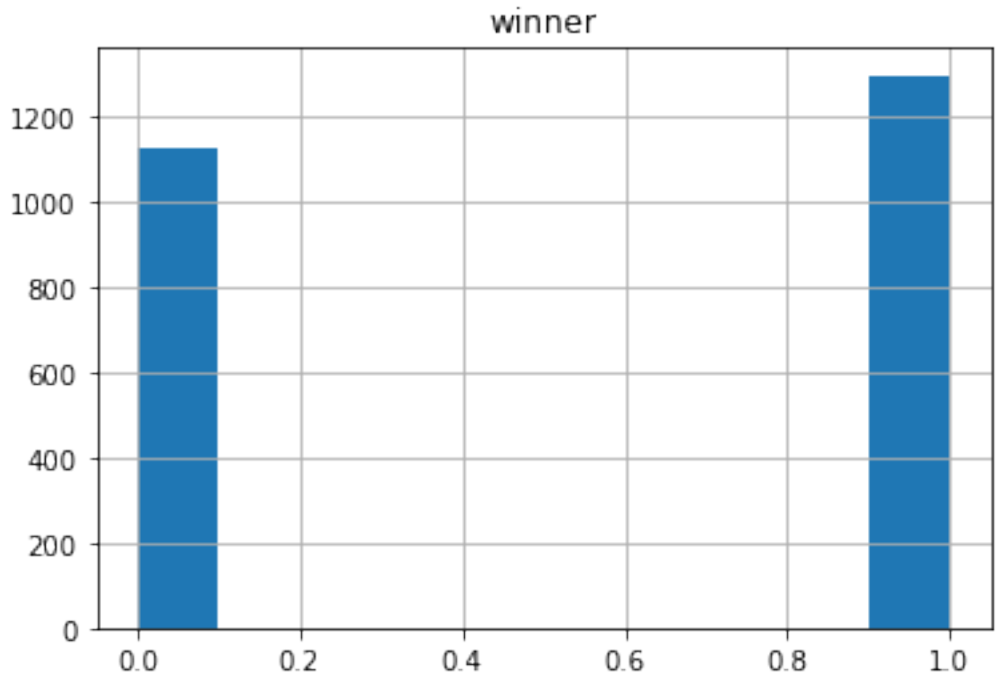

winner
1200
1000
800
600
400
200
0
0.0
0.2
0.4
0.6
0.8
1.0

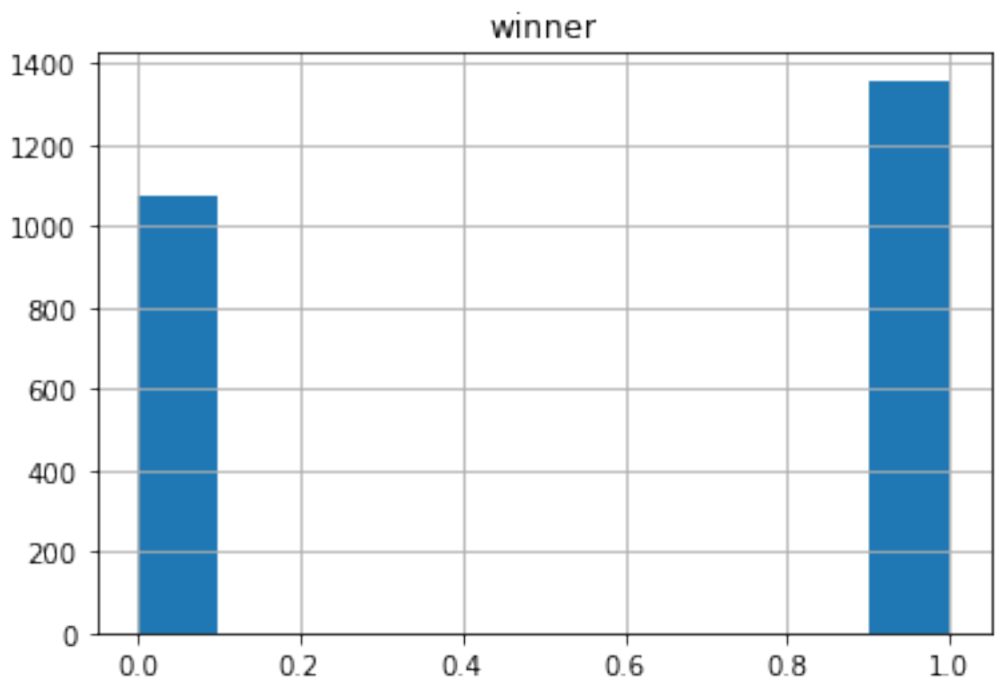

winner
1200
1000
800
600
400
200
0
0.0
0.2
0.4
0.6
0.8
1.0

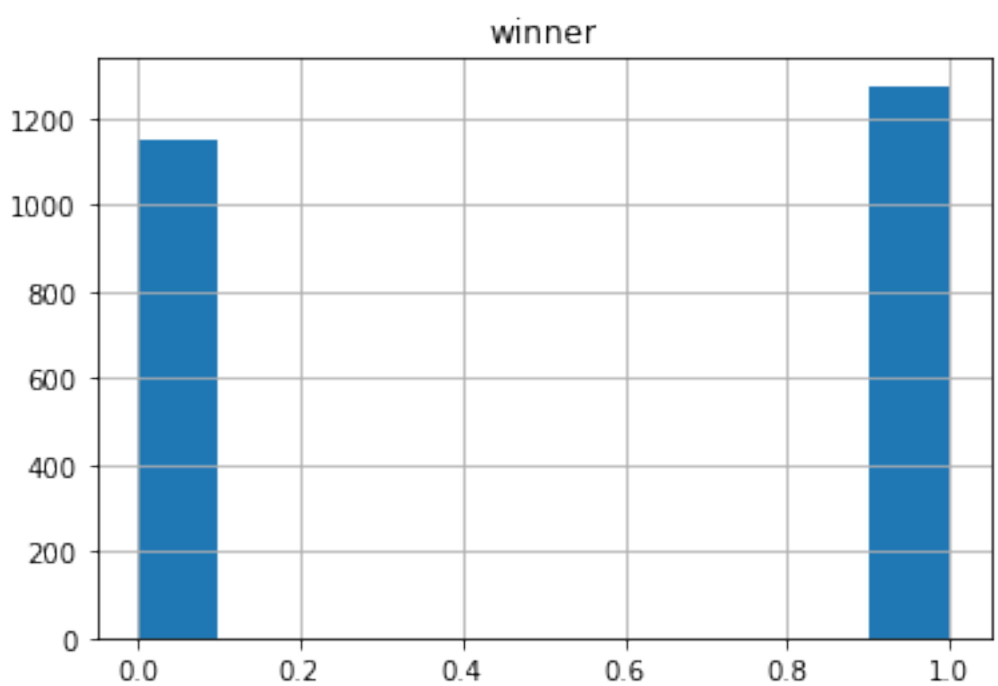

winner
0
200
400
600
800
1000
1200
0.0
0.2
0.4
0.6
0.8
1.0

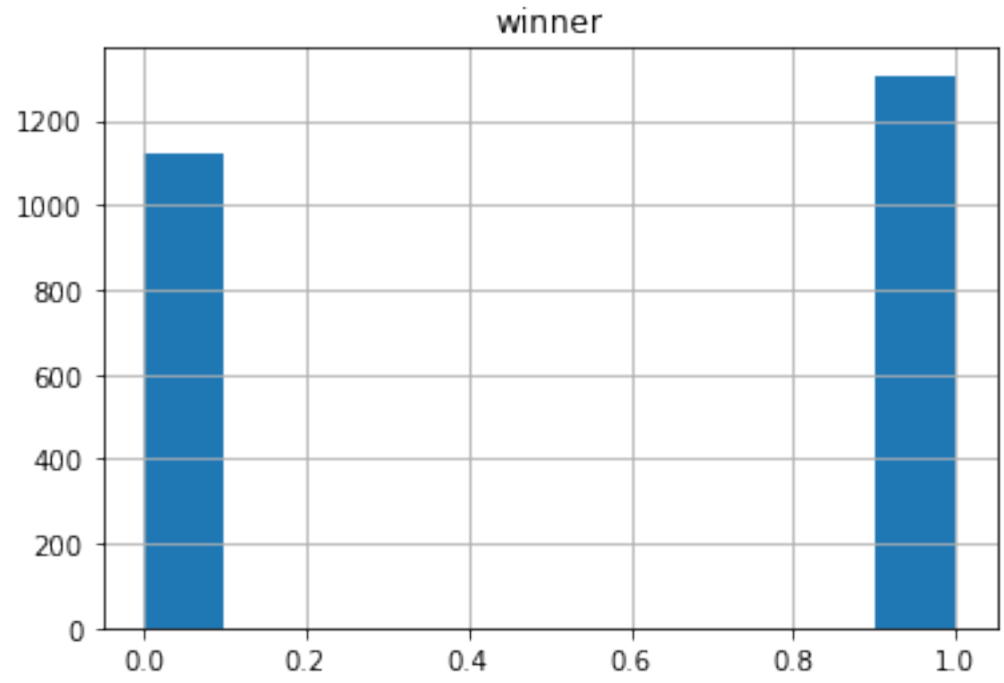

winner
0
200
400
600
800
1000
1200
0.0
0.2
0.4
0.6
0.8
1.0


winner
0
200
400
600
800
1000
1200
0.0
0.2
0.4
0.6
0.8
1.0


winner
0
200
400
600
800
1000
1200
0.0
0.2
0.4
0.6
0.8
1.0


winner
0
200
400
600
800
1000
1200
1400
0.0
0.2
0.4
0.6
0.8
1.0


winner
0
200
400
600
800
1000
1200
0.0
0.2
0.4
0.6
0.8
1.0



These histograms show a distinct pattern, the home team wins more than the visting team

In [126]:

```
#percent home wins for all games
all_scores = np.array(pd.concat(target_label_base_list))
home_wins = 0
for score in all_scores:
    if score[1] > score[0]:
        home_wins += 1
round(home_wins/len(all_scores)*100)
```

Out[126]:

54

Remove features with low variance

In [30]:

```
def variance_threshold_selector(data, threshold=0.5):
    selector = VarianceThreshold(threshold)
    selector.fit(data)
    return data[data.columns[selector.get_support(indices=True)]]
```

In [31]:

```
seasons_trimmed = []
for season in seasons:
    season = variance_threshold_selector(season)
    seasons_trimmed.append(season)
seasons = seasons_trimmed
```

In [32]:

```
#check number of features for each season
[season.shape for season in seasons]
```

Out[32]:

```
[(2427, 34),
 (2429, 34),
 (2104, 34),
 (2104, 34),
 (2102, 34),
 (2103, 34),
 (2101, 35),
 (2104, 34),
 (2268, 34),
 (2105, 34),
 (2102, 34),
 (2097, 35),
 (1945, 35),
 (2106, 34),
 (2266, 34),
 (2265, 34),
 (2108, 34),
 (1624, 36),
 (2016, 34),
 (1393, 34),
 (2104, 35),
 (1599, 34),
 (1858, 35),
 (1614, 34),
 (2427, 34),
 (2431, 34),
 (1619, 36),
 (1942, 34),
 (1622, 35),
 (1937, 35),
 (1943, 34),
 (1625, 33),
 (1235, 34),
 (1944, 34),
 (1933, 36),
 (1429, 33),
 (2102, 35),
 (1618, 32),
 (2099, 34),
 (2105, 34),
 (1620, 34),
 (1938, 35),
 (2428, 34),
 (2428, 34),
 (2429, 34),
 (2428, 34),
 (2427, 34),
 (2425, 34),
 (2429, 34),
 (2429, 34),
 (2430, 34),
 (2430, 34),
 (2428, 34),
 (2429, 34),
 (2427, 34),
 (2429, 34),
```

```
 (2428, 34),

 (2430, 34)]
```

In [33]:

```
#check that each season has the same features: total unique features should ma
tch individual season feature count
features_all = []
for season in seasons:
    for feat_name in season.columns:
        features_all.append(feat_name)
len(set(features_all))
```

Out[33]:

```
37
```

In [34]:

```
#season feature count varies accross seasons - the following code removes feat
ures that are not present in all seasons

#count the frequency of each feature accross all seasons
col_counts = {}
for season in seasons:
    for col in season.columns:
        col_counts[col] = col_counts.get(col,0) + 1

#if feature matches the max feature frequency, add it to keeper list
feat_list = []
for key, value in col_counts.items():
    if value == max(list(col_counts.values())):
        feat_list.append(key)
```

In [35]:

```
#the filtered feature list
feat_list
```

Out[35]:

```
['vis_at_bats',
 'vis_hits',
 'vis_doubles',
 'vis_hr',
 'vis_RBI',
 'vis_walks',
 'vis_strikeouts',
 'vis_grnd_dbl_plys',
 'vis_left_on_base',
 'vis_pitchers_used',
 'vis_ind_earned_runs',
 'vis_team_earned_runs',
 'vis_putouts',
 'vis_assists',
 'vis_errors',
 'vis_double_plays',
 'home_at_bats',
 'home_hits',
 'home_doubles',
 'home_hr',
 'home_RBI',
 'home_walks',
 'home_strikeouts',
 'home_grnd_dbl_plys',
 'home_left_on_base',
 'home_pitchers_used',
 'home_ind_earned_runs',
 'home_team_earned_runs',
 'home_putouts',
 'home_assists',
 'home_errors',
 'home_double_plays']
```

In [36]:

```python
#use filtered feature list to select standard features accross all seasons
for i in range(len(seasons)):
    seasons[i] = seasons[i].loc[:,feat_list]
```

In [37]:

```python
[season.shape for season in seasons]
```

Out[37]:

```
[(2427, 32),
 (2429, 32),
 (2104, 32),
 (2104, 32),
 (2102, 32),
 (2103, 32),
 (2101, 32),
 (2104, 32),
 (2268, 32),
 (2105, 32),
 (2102, 32),
```

```
(2102, 32),
(2097, 32),
(1945, 32),
(2106, 32),
(2266, 32),
(2265, 32),
(2108, 32),
(1624, 32),
(2016, 32),
(1393, 32),
(2104, 32),
(1599, 32),
(1858, 32),
(1614, 32),
(2427, 32),
(2431, 32),
(1619, 32),
(1942, 32),
(1622, 32),
(1937, 32),
(1943, 32),
(1625, 32),
(1235, 32),
(1944, 32),
(1933, 32),
(1429, 32),
(2102, 32),
(1618, 32),
(2099, 32),
(2105, 32),
(1620, 32),
(1938, 32),
(2428, 32),
(2428, 32),
(2429, 32),
(2428, 32),
(2427, 32),
(2425, 32),
(2429, 32),
(2429, 32),
(2430, 32),
(2430, 32),
(2428, 32),
(2429, 32),
(2427, 32),
(2429, 32),
(2428, 32),
(2430, 32)]
```

In [38]:

```
#check that each season has the same features: total unique features should ma
tch individual season feature count
features_all = []
for season in seasons:
    for feat_name in season.columns:
        features_all.append(feat_name)
len(set(features_all))
```

Out[38]:

```
32
```

confirmed

In [39]:

```
#select visitor feature names using 'vis_' prefix
vis_cols = [col for col in seasons[0] if col.startswith('vis_')]
del vis_cols[0]
```

In [40]:

```
vis_cols
```

Out[40]:

```
['vis_hits',
 'vis_doubles',
 'vis_hr',
 'vis_RBI',
 'vis_walks',
 'vis_strikeouts',
 'vis_grnd_dbl_plys',
 'vis_left_on_base',
 'vis_pitchers_used',
 'vis_ind_earned_runs',
 'vis_team_earned_runs',
 'vis_putouts',
 'vis_assists',
 'vis_errors',
 'vis_double_plays']
```

In [41]:

```
#select home feature names using 'home_' prefix
home_cols = [col for col in seasons[0] if col.startswith('home_')]
del home_cols[0]
```

In [42]:

```
home_cols
```

Out[42]:

```
['home_hits',
 'home_doubles',
 'home_hr',
 'home_RBI',
 'home_walks',
 'home_strikeouts',
 'home_grnd_dbl_plys',
 'home_left_on_base',
 'home_pitchers_used',
 'home_ind_earned_runs',
 'home_team_earned_runs',
 'home_putouts',
 'home_assists',
 'home_errors',
 'home_double_plays']
```

In [43]:

```
print(len(vis_cols))
print(len(home_cols))
```

```
15
15
```

In [44]:

```
#check if home and visitor feature sets match
vis_suffixlist = []
home_suffixlist = []
[vis_suffixlist.append(col.replace('vis_','')) for col in vis_cols]
[home_suffixlist.append(col.replace('home_','')) for col in home_cols]
set(vis_suffixlist)==set(home_suffixlist)
```

Out[44]:

```
True
```

Transform data distributions

In [45]:

```
#transform features so that distributions are normal
seasons = [np.log(season +1) for season in seasons]
#seasons = [np.arcsinh(season) for season in seasons]
```

More feature selection - now with random forest

In [46]:

```
#combine seasons in single dataframe, for use in random forest feature selecti
on below
X_feat = pd.concat(seasons)
Y_feat = pd.concat(target_labels_list)
Y_feat = Y_feat.values.ravel()
```

In [47]:

```
X_feat.shape
```

Out[47]:

```
(121367, 32)
```

In [48]:

```
Y_feat.shape
```

Out[48]:

```
(121367,)
```

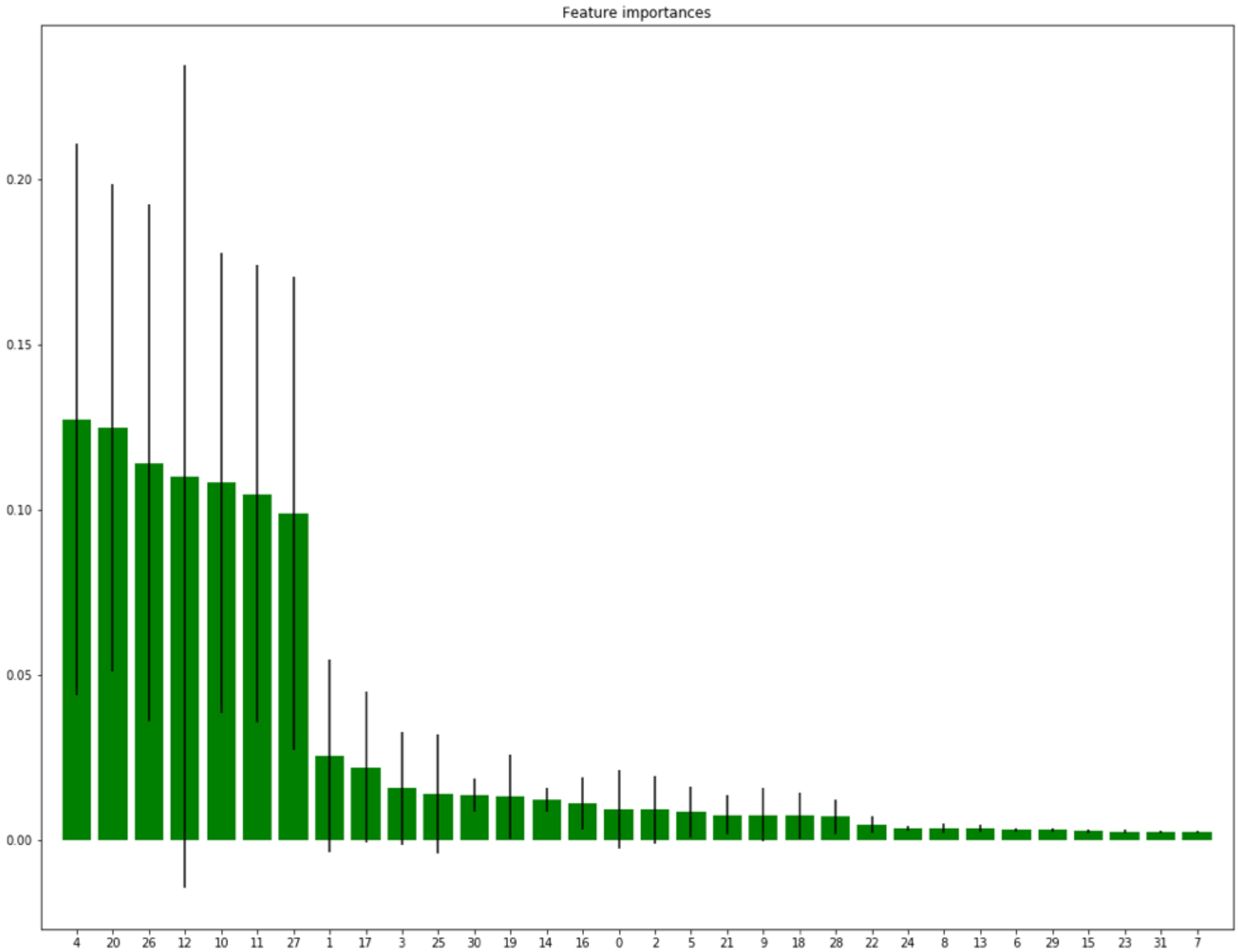
In [49]:

```
# Use random forest to compute feature importances
forest = ExtraTreesClassifier(n_estimators=500)
forest.fit(X_feat, Y_feat)
importances = forest.feature_importances_
std = np.std([tree.feature_importances_ for tree in forest.estimators_], axis=
0)
indices = np.argsort(importances)[::-1]

# Print the feature ranking
print("Feature ranking:")

for f in range(X_feat.shape[1]):
    print("%d. %s (%f)" % (f + 1, X_feat.columns[indices[f]], importances[indi
ces[f]]))
```

```
Feature ranking:
1. vis_RBI (0.127126)
2. home_RBI (0.124839)
3. home_ind_earned_runs (0.114062)
4. vis_putouts (0.109870)
5. vis_ind_earned_runs (0.108024)
6. vis_team_earned_runs (0.104672)
7. home_team_earned_runs (0.098839)
8. vis_hits (0.025326)
9. home_hits (0.021925)
10. vis_hr (0.015541)
11. home_pitchers_used (0.013925)
12. home_errors (0.013544)
13. home_hr (0.013056)
14. vis_errors (0.012072)
15. home_at_bats (0.010901)
16. vis_at_bats (0.009160)
17. vis_doubles (0.009085)
18. vis_walks (0.008311)
19. home_walks (0.007554)
20. vis_pitchers_used (0.007492)
21. home_doubles (0.007267)
22. home_putouts (0.006874)
23. home_strikeouts (0.004420)
24. home_left_on_base (0.003504)
25. vis_left_on_base (0.003504)
26. vis_assists (0.003358)
27. vis_strikeouts (0.002978)
28. home_assists (0.002943)
29. vis_double_plays (0.002581)
30. home_grnd_dbl_plys (0.002514)
31. home_double_plays (0.002413)
32. vis_grnd_dbl_plys (0.002318)
```

In [50]:

```
# Plot the feature importances of the forest - this is not my code, but I can'
t find where I took it from to provide
#a reference
plt.figure(figsize=(18,14))
plt.title("Feature importances")
plt.bar(range(X_feat.shape[1]), importances[indices], color="g", yerr=std[indi
ces], align="center")
plt.xticks(range(X_feat.shape[1]), indices)
plt.xlim([-1, X_feat.shape[1]])
plt.show()
```



In [51]:

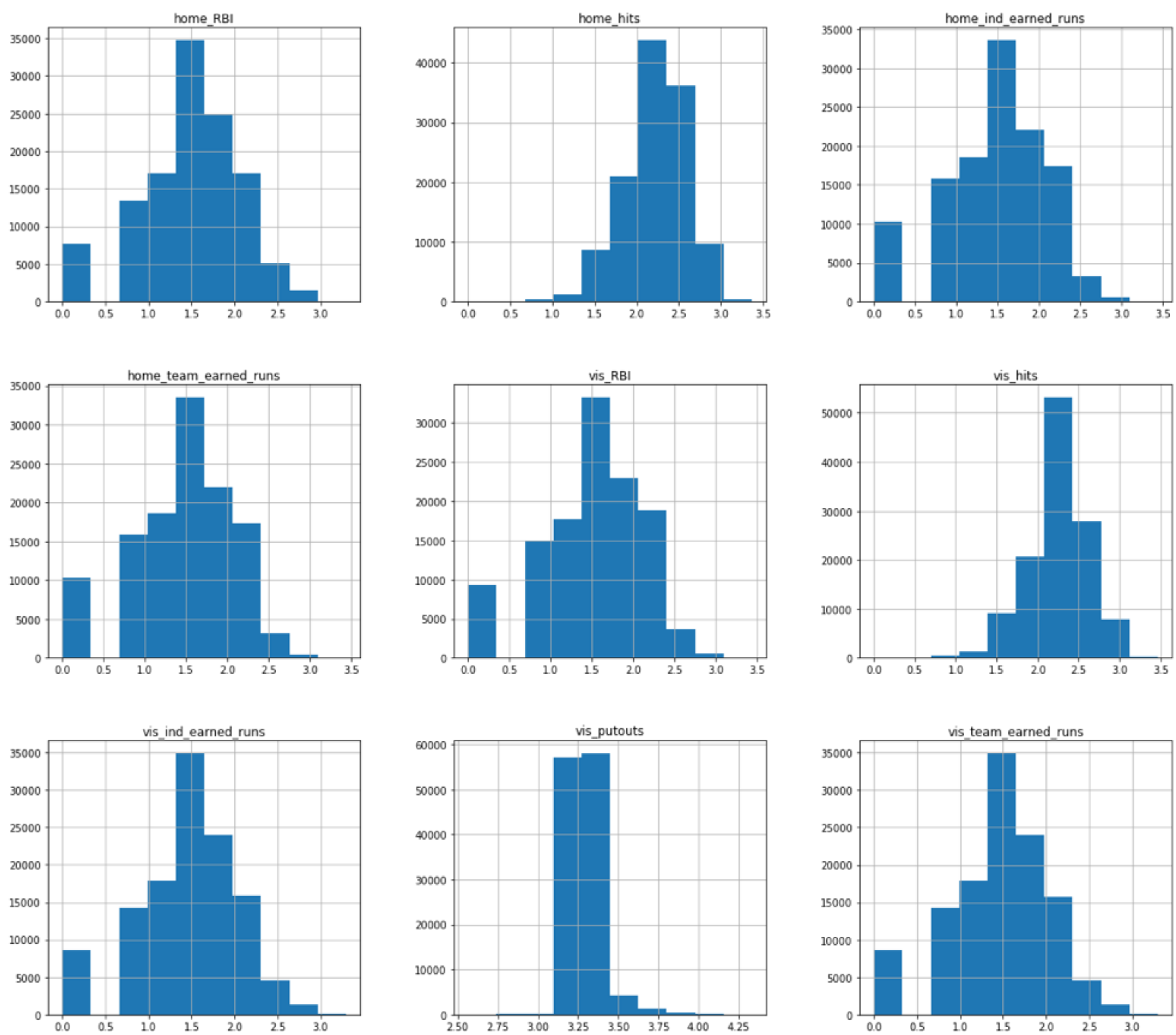
```
#select top 9 features
top_feat = seasons[0].iloc[:,indices[0:9]].columns
```

In [52]:

```
X_feat.loc[:,top_feat].hist(figsize=(20,18), bins=10)
```

Out[52]:

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x11b5160
48>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x11b6f12
78>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x11b7045
18>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x11b7c22
78>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x11b81da
20>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x11b81da
58>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x11be91f
d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x11b64a4
00>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x11b4a3a
20>]], dtype=object)
```



In [53]:

```
#reduce features in all seasons per features selected by random forest feature
importance
seasons = [season.loc[:,top_feat] for season in seasons]
```

In [54]:

```
#combine seasons with reduced reduced feature sets
pd.concat(seasons).shape
```

Out[54]:

```
(121367, 9)
```

In [55]:

```
seasons[0].head()
```

Out[55]:

	vis_RBI	home_RBI	home_ind_earned_runs	vis_putouts	vis_ind_earned_runs
0	0.693147	1.791759	0.693147	3.218876	1.791759
1	0.693147	1.386294	0.693147	3.295837	1.386294
2	2.397895	2.484907	2.397895	3.610918	2.484907
3	1.609438	1.386294	1.609438	3.433987	1.386294
4	1.609438	1.098612	1.386294	3.332205	1.098612

Compute team statistics for use in models

In [56]:

```
#redifine these variables based on reduced features (visitor or home team)
vis_cols = [col for col in seasons[0] if col.startswith('vis_')]
home_cols = [col for col in seasons[0] if col.startswith('home_')]
```

In [57]:

```
#function for computing statistics.
def season_stats(df, groupby_str, col_str, window):
    #rolling mean (or moving average)
    df_mean = df.groupby(groupby_str)[col_str].rolling(window).mean()
    df_mean.fillna(method='bfill', inplace = True)
    df_mean.index = df_mean.index.droplevel()
    df = df.join(df_mean, rsuffix= '_ma')

    #expanding mean
    df_mean_x = df.groupby(groupby_str)[col_str].expanding().mean()
    df_mean_x.fillna(method='bfill', inplace = True)
    df_mean_x.index = df_mean_x.index.droplevel()
    df = df.join(df_mean_x, rsuffix= '_ma_x')

    #rolling median
    df_median = df.groupby(groupby_str)[col_str].rolling(window).median()
    df_median.fillna(method='bfill', inplace = True)
    df_median.index = df_median.index.droplevel()
    df = df.join(df_median, rsuffix= '_mmed')

    #expanding median
    df_median_x = df.groupby(groupby_str)[col_str].expanding().median()
    df_median_x.fillna(method='bfill', inplace = True)
    df_median_x.index = df_median_x.index.droplevel()
    df = df.join(df_median_x, rsuffix= '_mmed_x')

    #rolling standard deviation
    df_std = df.groupby(groupby_str)[col_str].rolling(window).std()
    df_std.fillna(method='bfill', inplace = True)
    df_std.index = df_std.index.droplevel()
    df = df.join(df_std, rsuffix= '_mv_sd')

    #expanding standard deviation
    df_std_x = df.groupby(groupby_str)[col_str].expanding(window).std()
    df_std_x.fillna(method='bfill', inplace = True)
    df_std_x.index = df_std_x.index.droplevel()
    df = df.join(df_std_x, rsuffix= '_mv_sd_x')

    return df
```

In [58]:

```
# append team labels for each game in each season
for i in range(len(seasons)):
    season = home_vis_labels_list[i].join(seasons[i])
    seasons[i] = season
```

In [59]:

```
#compute home team statistics
for i in range(len(seasons)):
    for col in home_cols:
        seasons[i] = season_stats(seasons[i],'home_team', col, 5)
```

In [60]:

```
#compute visiting team statistics
for i in range(len(seasons)):
    for col in vis_cols:
        seasons[i] = season_stats(seasons[i],'visiting_team', col, 5)
```

In [61]:

```
#combine all win/lose target labels for all seeasons (single data frame)
Y = pd.concat(target_labels_list)
Y.shape
```

Out[61]:

```
(121367, 1)
```

In [62]:

```
#drop features used to compute statistics - will only use statistics for model
s
feat_drop = list(top_feat)
feat_drop.append('visiting_team')
feat_drop.append('home_team')
feat_drop
```

Out[62]:

```
['vis_RBI',
 'home_RBI',
 'home_ind_earned_runs',
 'vis_putouts',
 'vis_ind_earned_runs',
 'vis_team_earned_runs',
 'home_team_earned_runs',
 'vis_hits',
 'home_hits',
 'visiting_team',
 'home_team']
```

In [63]:

```
#combine seasons into single dataframe for modeling
X = pd.concat(seasons)
X = X.drop(feat_drop,
            axis=1)
X.shape
```

Out[63]:

```
(121367, 54)
```

In [64]:

```
X.columns
```

Out[64]:

```
Index(['home_RBI_ma', 'home_RBI_ma_x', 'home_RBI_mmed', 'home_RBI_
mmed_x',
       'home_RBI_mv_sd', 'home_RBI_mv_sd_x', 'home_ind_earned_runs
_ma',
       'home_ind_earned_runs_ma_x', 'home_ind_earned_runs_mmed',
       'home_ind_earned_runs_mmed_x', 'home_ind_earned_runs_mv_sd'
,
       'home_ind_earned_runs_mv_sd_x', 'home_team_earned_runs_ma',
       'home_team_earned_runs_ma_x', 'home_team_earned_runs_mmed',
       'home_team_earned_runs_mmed_x', 'home_team_earned_runs_mv_s
d',
       'home_team_earned_runs_mv_sd_x', 'home_hits_ma', 'home_hits
_ma_x',
       'home_hits_mmed', 'home_hits_mmed_x', 'home_hits_mv_sd',
       'home_hits_mv_sd_x', 'vis_RBI_ma', 'vis_RBI_ma_x', 'vis_RBI
_mmed',
       'vis_RBI_mmed_x', 'vis_RBI_mv_sd', 'vis_RBI_mv_sd_x', 'vis_
putouts_ma',
       'vis_putouts_ma_x', 'vis_putouts_mmed', 'vis_putouts_mmed_x
',
       'vis_putouts_mv_sd', 'vis_putouts_mv_sd_x', 'vis_ind_earned
_runs_ma',
       'vis_ind_earned_runs_ma_x', 'vis_ind_earned_runs_mmed',
       'vis_ind_earned_runs_mmed_x', 'vis_ind_earned_runs_mv_sd',
       'vis_ind_earned_runs_mv_sd_x', 'vis_team_earned_runs_ma',
       'vis_team_earned_runs_ma_x', 'vis_team_earned_runs_mmed',
       'vis_team_earned_runs_mmed_x', 'vis_team_earned_runs_mv_sd'
,
       'vis_team_earned_runs_mv_sd_x', 'vis_hits_ma', 'vis_hits_ma
_x',
       'vis_hits_mmed', 'vis_hits_mmed_x', 'vis_hits_mv_sd',
       'vis_hits_mv_sd_x'],
      dtype='object')
```

Random forest to reduce dimensionality...again

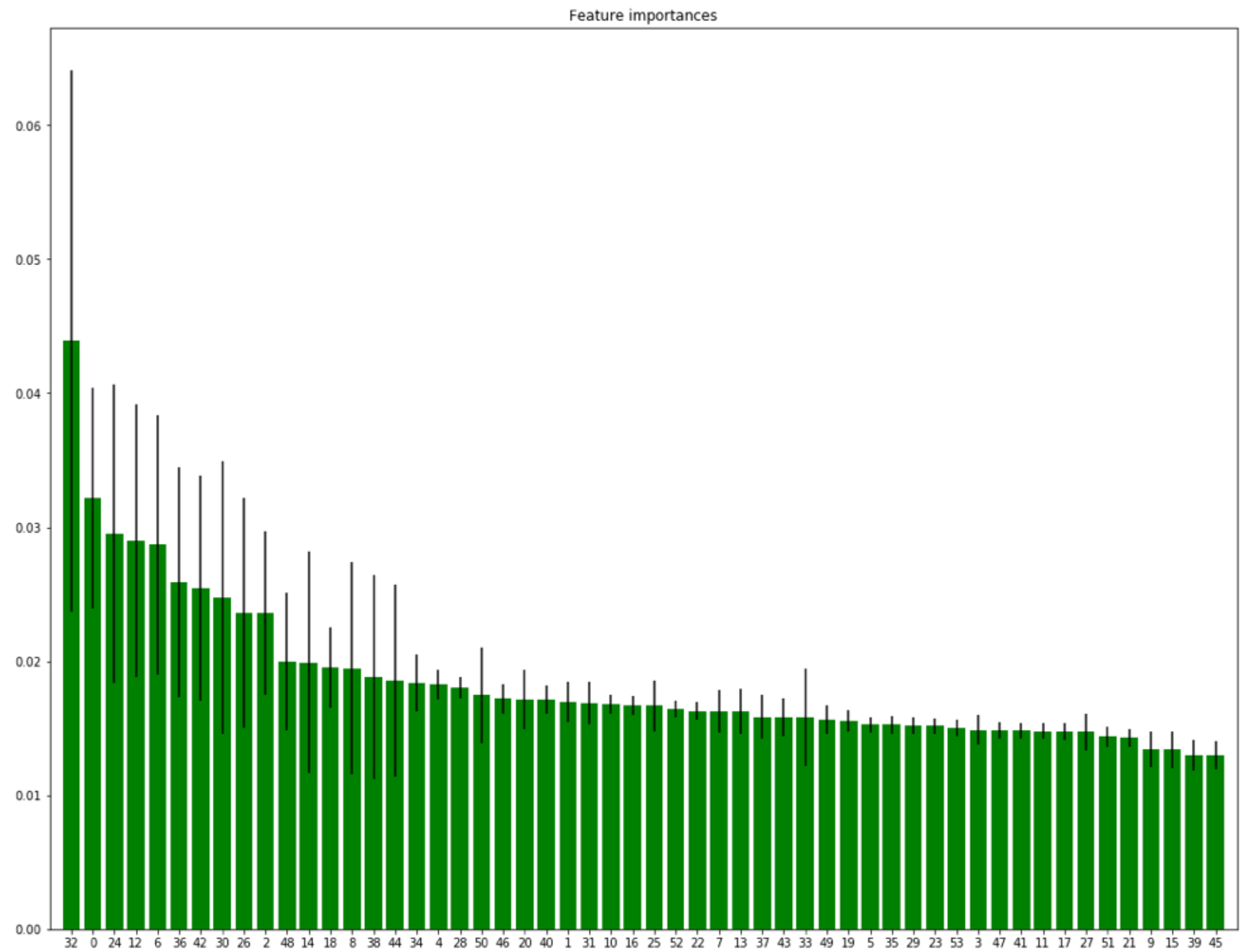
In [65]:

```
# Use random forest to compute feature importances on new feature set
forest = ExtraTreesClassifier(n_estimators=500)

Y_feat = Y.values.ravel()
forest.fit(X, Y_feat)
importances = forest.feature_importances_
std = np.std([tree.feature_importances_ for tree in forest.estimators_],
             axis=0)
indices = np.argsort(importances)[::-1]

# Print the feature ranking
print("Feature ranking:")

for f in range(X.shape[1]):
    print("%d. %s (%f)" % (f + 1, X.columns[indices[f]], importances[indices[f
]]))
```

```
Feature ranking:
1. vis_putouts_mmed (0.043883)
2. home_RBI_ma (0.032158)
3. vis_RBI_ma (0.029510)
4. home_team_earned_runs_ma (0.028981)
5. home_ind_earned_runs_ma (0.028682)
6. vis_ind_earned_runs_ma (0.025886)
7. vis_team_earned_runs_ma (0.025441)
8. vis_putouts_ma (0.024736)
9. vis_RBI_mmed (0.023588)
10. home_RBI_mmed (0.023585)
11. vis_hits_ma (0.019961)
12. home_team_earned_runs_mmed (0.019900)
13. home_hits_ma (0.019509)
14. home_ind_earned_runs_mmed (0.019475)
15. vis_ind_earned_runs_mmed (0.018791)
16. vis_team_earned_runs_mmed (0.018540)
17. vis_putouts_mv_sd (0.018387)
18. home_RBI_mv_sd (0.018260)
19. vis_RBI_mv_sd (0.018023)
20. vis_hits_mmed (0.017467)
21. vis_team_earned_runs_mv_sd (0.017197)
22. home_hits_mmed (0.017167)
23. vis_ind_earned_runs_mv_sd (0.017130)
24. home_RBI_ma_x (0.016972)
25. vis_putouts_ma_x (0.016864)
26. home_ind_earned_runs_mv_sd (0.016754)
27. home_team_earned_runs_mv_sd (0.016705)
28. vis_RBI_ma_x (0.016658)
29. vis_hits_mv_sd (0.016422)
30. home_hits_mv_sd (0.016287)
31. home_ind_earned_runs_ma_x (0.016249)
32. home_team_earned_runs_ma_x (0.016242)
33. vis_ind_earned_runs_ma_x (0.015834)
34. vis_team_earned_runs_ma_x (0.015819)
35. vis_putouts_mmed_x (0.015794)
36. vis_hits_ma_x (0.015641)
37. home_hits_ma_x (0.015553)
38. home_RBI_mv_sd_x (0.015249)
39. vis_putouts_mv_sd_x (0.015246)
40. vis_RBI_mv_sd_x (0.015166)
41. home_hits_mv_sd_x (0.015152)
42. vis_hits_mv_sd_x (0.014982)
43. home_RBI_mmed_x (0.014870)
44. vis_team_earned_runs_mv_sd_x (0.014808)
45. vis_ind_earned_runs_mv_sd_x (0.014793)
46. home_ind_earned_runs_mv_sd_x (0.014768)
47. home_team_earned_runs_mv_sd_x (0.014732)
48. vis_RBI_mmed_x (0.014719)
49. vis_hits_mmed_x (0.014363)
50. home_hits_mmed_x (0.014283)
51. home_ind_earned_runs_mmed_x (0.013447)
52. home_team_earned_runs_mmed_x (0.013401)
53. vis_ind_earned_runs_mmed_x (0.012995)
54. vis_team_earned_runs_mmed_x (0.012975)
```

In [66]:

```
# Plot the feature importances of the forest
plt.figure(figsize=(18,14))
plt.title("Feature importances")
plt.bar(range(X.shape[1]), importances[indices],
       color="g", yerr=std[indices], align="center")
plt.xticks(range(X.shape[1]), indices)
plt.xlim([-1, X.shape[1]])
plt.show()
```



In [67]:

```
#select top features
top_feat = X.iloc[:,indices[0:9]].columns
```

In [68]:

```
top_feat
```

Out[68]:

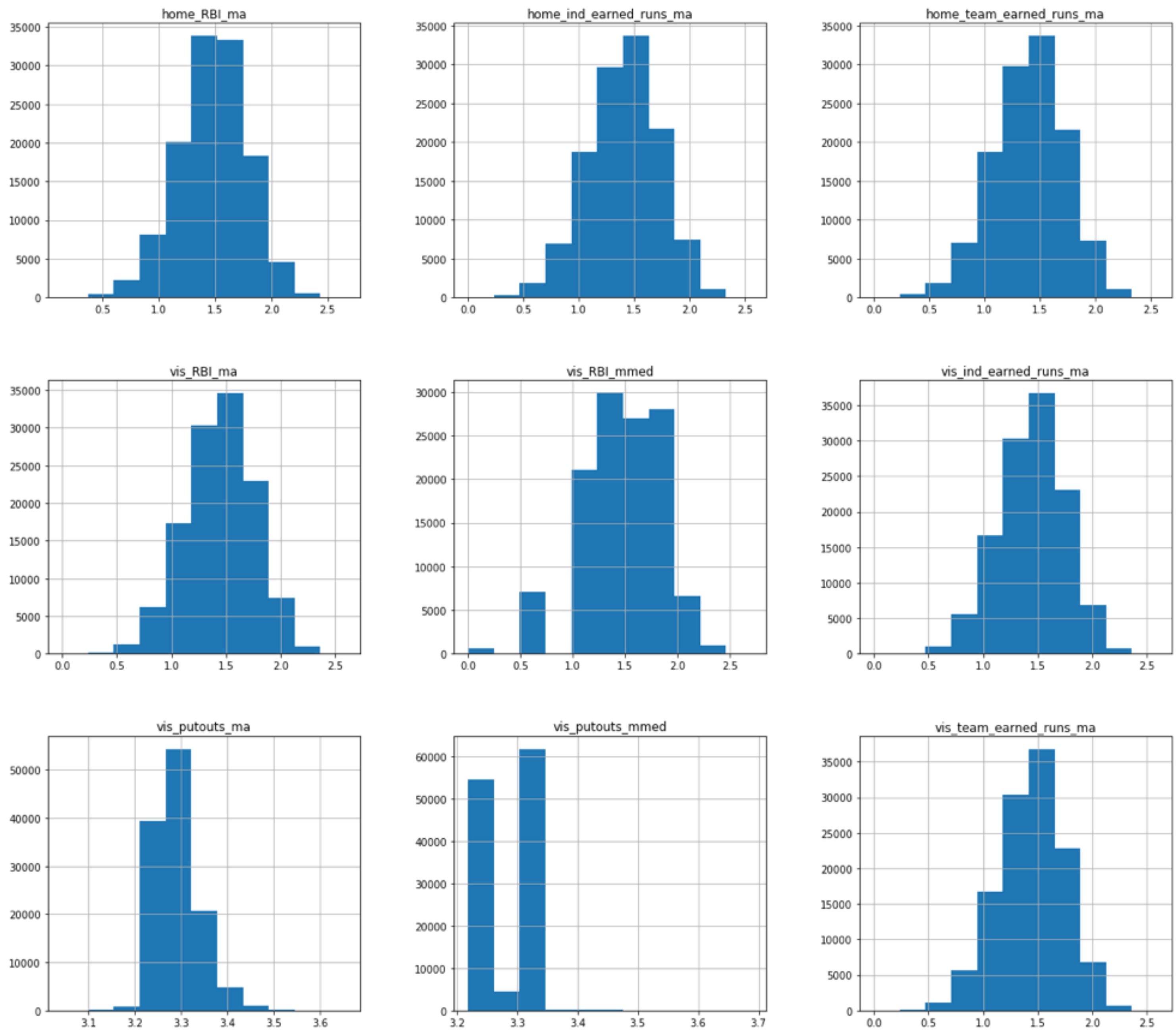
```
Index(['vis_putouts_mmed', 'home_RBI_ma', 'vis_RBI_ma',
       'home_team_earned_runs_ma', 'home_ind_earned_runs_ma',
       'vis_ind_earned_runs_ma', 'vis_team_earned_runs_ma', 'vis_p
utouts_ma',
       'vis_RBI_mmed'],
      dtype='object')
```

In [69]:

```
X.loc[:,top_feat].hist(figsize=(20,18), bins=11)
```

Out[69]:

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x11bfab5
f8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x11d1d93
90>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x11d2451
28>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x11d68f3
90>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x11d69e3
90>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x11d69e5
88>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x11e0b1d
d8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x11e1305
50>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x11e37f7
f0>]], dtype=object)
```



In [70]:

```
#reduce data to only include most important features
X = X.loc[:,top_feat]
```

In [71]:

```
#check that statistics computations did not produce NaN values
X.isnull().sum()
```

Out[71]:

```
vis_putouts_mmed            0
home_RBI_ma                 0
vis_RBI_ma                  0
home_team_earned_runs_ma    0
home_ind_earned_runs_ma     0
vis_ind_earned_runs_ma      0
vis_team_earned_runs_ma     0
vis_putouts_ma              0
vis_RBI_mmed                0
dtype: int64
```

Data ready for use in precdiction efforts

In [72]:

```
X.shape
```

Out[72]:

```
(121367, 9)
```

In [73]:

```
Y.shape
```

Out[73]:

```
(121367, 1)
```

Predict with: Logistic Regression, Random Forest, AdaBoost, and Ensemble voting

In [74]:

```
Y = Y.values.ravel()
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, rando
m_state=0)

#classifiers
num = 500
clf1 = LogisticRegression()
clf2 = RandomForestClassifier(n_estimators=num)
clf3 = AdaBoostClassifier(n_estimators=num)
eclf = VotingClassifier(estimators=[('lr', clf1), ('rf', clf2), ('ab', clf3)],
voting='hard')

#fit models
clf1 = clf1.fit(X_train,y_train)
clf2 = clf2.fit(X_train,y_train)
clf3 = clf3.fit(X_train,y_train)
eclf = eclf.fit(X_train,y_train)

#(code based on snippet from: http://scikit-learn.org/stable/modules/ensemble.
html)
```

Results (Logistic Regression, Random Forest, AdaBoost, Ensemble voting)

In [75]:

```
for clf, label in zip([clf1, clf2, clf3, eclf], ['Logistic Regression', 'Rando
m Forest', 'AdaBoost', 'Ensemble']):
     scores = cross_val_score(clf, X_test, y_test, cv=5, scoring='accuracy')
     print("Accuracy: %0.2f (+/- %0.2f) [%s]" % (scores.mean(), scores.std(),
label))
```

```
Accuracy: 0.70 (+/- 0.01) [Logistic Regression]
Accuracy: 0.69 (+/- 0.01) [Random Forest]
Accuracy: 0.70 (+/- 0.01) [AdaBoost]
Accuracy: 0.70 (+/- 0.01) [Ensemble]
```

Predict with ANN

In [83]:

```
def ANN_model():
    model = Sequential()
    model.add(Dense(20, activation='relu', input_dim=9))
    model.add(BatchNormalization())
    model.add(Dropout(0.2))
    model.add(Dense(12, activation='relu'))
    model.add(BatchNormalization())
    model.add(Dropout(0.2))
    model.add(Dense(6, activation='relu'))
    model.add(BatchNormalization())
    model.add(Dropout(0.2))
    model.add(Dense(4, activation='relu'))
    model.add(BatchNormalization())
    #model.add(Dropout(0.25))
    model.add(Dense(2, activation='relu'))
    model.add(Dense(1, activation='sigmoid'))
    return model
```

In [84]:

```
model_ANN = ANN_model()
# compile using cross entropy since this problem is a binary classification pr
oblem.
# using adam optimizer
model_ANN.compile(optimizer='adam', loss ='binary_crossentropy', metrics=['acc
uracy'])
model_ANN.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_7 (Dense)              (None, 20)                200       
_________________________________________________________________
batch_normalization_5 (Batch (None, 20)                80        
_________________________________________________________________
dropout_5 (Dropout)          (None, 20)                0         
_________________________________________________________________
dense_8 (Dense)              (None, 12)                252       
_________________________________________________________________
batch_normalization_6 (Batch (None, 12)                48        
_________________________________________________________________
dropout_6 (Dropout)          (None, 12)                0         
_________________________________________________________________
dense_9 (Dense)              (None, 6)                 78        
_________________________________________________________________
batch_normalization_7 (Batch (None, 6)                 24        
_________________________________________________________________
dropout_7 (Dropout)          (None, 6)                 0         
_________________________________________________________________
dense_10 (Dense)             (None, 4)                 28        
_________________________________________________________________
batch_normalization_8 (Batch (None, 4)                 16        
_________________________________________________________________
dense_11 (Dense)             (None, 2)                 10        
_________________________________________________________________
dense_12 (Dense)             (None, 1)                 3         
=================================================================
Total params: 739
Trainable params: 655
Non-trainable params: 84
_________________________________________________________________
```

In [85]:

```
ANN_log = model_ANN.fit(X, Y, validation_split = 0.2, epochs=500, batch_size=1
00, shuffle = True, verbose=2)
```

```
Train on 97093 samples, validate on 24274 samples
Epoch 1/500
 - 11s - loss: 0.6255 - acc: 0.6461 - val_loss: 0.5973 - val_acc:
0.6701
Epoch 2/500
 - 10s - loss: 0.5916 - acc: 0.6902 - val_loss: 0.5904 - val_acc:
0.6808
Epoch 3/500
 - 9s - loss: 0.5897 - acc: 0.6923 - val_loss: 0.5752 - val_acc: 0
.7009
Epoch 4/500
 - 9s - loss: 0.5879 - acc: 0.6946 - val_loss: 0.5814 - val_acc: 0
.6889
Epoch 5/500
 - 10s - loss: 0.5865 - acc: 0.6945 - val_loss: 0.5751 - val_acc:
0.7004
Epoch 6/500
```

```
 - 10s - loss: 0.5860 - acc: 0.6950 - val_loss: 0.5754 - val_acc:
0.7008
Epoch 7/500
 - 9s - loss: 0.5862 - acc: 0.6939 - val_loss: 0.5836 - val_acc: 0
.6935
Epoch 8/500
 - 10s - loss: 0.5853 - acc: 0.6949 - val_loss: 0.5809 - val_acc:
0.6993
Epoch 9/500
 - 9s - loss: 0.5851 - acc: 0.6949 - val_loss: 0.5747 - val_acc: 0
.6991
Epoch 10/500
 - 10s - loss: 0.5841 - acc: 0.6939 - val_loss: 0.5761 - val_acc:
0.7002
Epoch 11/500
 - 9s - loss: 0.5839 - acc: 0.6951 - val_loss: 0.5756 - val_acc: 0
.7010
Epoch 12/500
 - 10s - loss: 0.5841 - acc: 0.6951 - val_loss: 0.5769 - val_acc:
0.7004
Epoch 13/500
 - 10s - loss: 0.5839 - acc: 0.6954 - val_loss: 0.5862 - val_acc:
0.6881
Epoch 14/500
 - 10s - loss: 0.5838 - acc: 0.6958 - val_loss: 0.5771 - val_acc:
0.6969
Epoch 15/500
 - 9s - loss: 0.5830 - acc: 0.6962 - val_loss: 0.5763 - val_acc: 0
.7000
Epoch 16/500
 - 9s - loss: 0.5828 - acc: 0.6955 - val_loss: 0.5776 - val_acc: 0
.6968
Epoch 17/500
 - 9s - loss: 0.5837 - acc: 0.6962 - val_loss: 0.5758 - val_acc: 0
.6987
Epoch 18/500
 - 9s - loss: 0.5830 - acc: 0.6963 - val_loss: 0.5763 - val_acc: 0
.6991
Epoch 19/500
 - 9s - loss: 0.5827 - acc: 0.6962 - val_loss: 0.5863 - val_acc: 0
.6893
Epoch 20/500
 - 9s - loss: 0.5836 - acc: 0.6960 - val_loss: 0.5751 - val_acc: 0
.6998
Epoch 21/500
 - 10s - loss: 0.5825 - acc: 0.6961 - val_loss: 0.5780 - val_acc:
0.6996
Epoch 22/500
 - 9s - loss: 0.5824 - acc: 0.6951 - val_loss: 0.5808 - val_acc: 0
.6934
Epoch 23/500
 - 9s - loss: 0.5820 - acc: 0.6963 - val_loss: 0.5753 - val_acc: 0
.6997
Epoch 24/500
 - 9s - loss: 0.5821 - acc: 0.6960 - val_loss: 0.5821 - val_acc: 0
.6893
Epoch 25/500
```

```
 - 9s - loss: 0.5819 - acc: 0.6962 - val_loss: 0.5750 - val_acc: 0
.7004
Epoch 26/500
 - 9s - loss: 0.5820 - acc: 0.6960 - val_loss: 0.5744 - val_acc: 0
.7015
Epoch 27/500
 - 9s - loss: 0.5824 - acc: 0.6959 - val_loss: 0.5850 - val_acc: 0
.6854
Epoch 28/500
 - 10s - loss: 0.5822 - acc: 0.6961 - val_loss: 0.5786 - val_acc:
0.6962
Epoch 29/500
 - 10s - loss: 0.5824 - acc: 0.6958 - val_loss: 0.5812 - val_acc:
0.6912
Epoch 30/500
 - 9s - loss: 0.5813 - acc: 0.6965 - val_loss: 0.5770 - val_acc: 0
.6969
Epoch 31/500
 - 10s - loss: 0.5816 - acc: 0.6957 - val_loss: 0.5841 - val_acc:
0.6908
Epoch 32/500
 - 10s - loss: 0.5820 - acc: 0.6966 - val_loss: 0.5754 - val_acc:
0.7010
Epoch 33/500
 - 9s - loss: 0.5821 - acc: 0.6966 - val_loss: 0.5921 - val_acc: 0
.6757
Epoch 34/500
 - 10s - loss: 0.5819 - acc: 0.6955 - val_loss: 0.5838 - val_acc:
0.6886
Epoch 35/500
 - 10s - loss: 0.5822 - acc: 0.6952 - val_loss: 0.5747 - val_acc:
0.7009
Epoch 36/500
 - 10s - loss: 0.5827 - acc: 0.6952 - val_loss: 0.5801 - val_acc:
0.6963
Epoch 37/500
 - 10s - loss: 0.5822 - acc: 0.6955 - val_loss: 0.5762 - val_acc:
0.7001
Epoch 38/500
 - 9s - loss: 0.5824 - acc: 0.6953 - val_loss: 0.5769 - val_acc: 0
.6998
Epoch 39/500
 - 10s - loss: 0.5820 - acc: 0.6964 - val_loss: 0.5753 - val_acc:
0.7010
Epoch 40/500
 - 10s - loss: 0.5818 - acc: 0.6960 - val_loss: 0.5770 - val_acc:
0.6998
Epoch 41/500
 - 10s - loss: 0.5821 - acc: 0.6961 - val_loss: 0.5779 - val_acc:
0.6972
Epoch 42/500
 - 10s - loss: 0.5814 - acc: 0.6964 - val_loss: 0.5800 - val_acc:
0.6968
Epoch 43/500
 - 10s - loss: 0.5812 - acc: 0.6976 - val_loss: 0.5771 - val_acc:
0.7017
Epoch 44/500
```

```
 - 9s - loss: 0.5813 - acc: 0.6969 - val_loss: 0.5748 - val_acc: 0
.7004
Epoch 45/500
 - 9s - loss: 0.5818 - acc: 0.6962 - val_loss: 0.5769 - val_acc: 0
.7014
Epoch 46/500
 - 10s - loss: 0.5810 - acc: 0.6968 - val_loss: 0.5793 - val_acc:
0.6971
Epoch 47/500
 - 9s - loss: 0.5812 - acc: 0.6973 - val_loss: 0.5775 - val_acc: 0
.7009
Epoch 48/500
 - 10s - loss: 0.5817 - acc: 0.6962 - val_loss: 0.5758 - val_acc:
0.7020
Epoch 49/500
 - 10s - loss: 0.5815 - acc: 0.6970 - val_loss: 0.5786 - val_acc:
0.6994
Epoch 50/500
 - 10s - loss: 0.5812 - acc: 0.6968 - val_loss: 0.5816 - val_acc:
0.6975
Epoch 51/500
 - 9s - loss: 0.5810 - acc: 0.6973 - val_loss: 0.5776 - val_acc: 0
.7017
Epoch 52/500
 - 10s - loss: 0.5810 - acc: 0.6975 - val_loss: 0.5833 - val_acc:
0.6895
Epoch 53/500
 - 9s - loss: 0.5808 - acc: 0.6969 - val_loss: 0.5842 - val_acc: 0
.6902
Epoch 54/500
 - 10s - loss: 0.5805 - acc: 0.6961 - val_loss: 0.5753 - val_acc:
0.7015
Epoch 55/500
 - 10s - loss: 0.5810 - acc: 0.6965 - val_loss: 0.5813 - val_acc:
0.6939
Epoch 56/500
 - 10s - loss: 0.5810 - acc: 0.6980 - val_loss: 0.5783 - val_acc:
0.7006
Epoch 57/500
 - 9s - loss: 0.5809 - acc: 0.6976 - val_loss: 0.5789 - val_acc: 0
.7011
Epoch 58/500
 - 9s - loss: 0.5813 - acc: 0.6962 - val_loss: 0.5785 - val_acc: 0
.6972
Epoch 59/500
 - 10s - loss: 0.5811 - acc: 0.6974 - val_loss: 0.5772 - val_acc:
0.7001
Epoch 60/500
 - 9s - loss: 0.5809 - acc: 0.6964 - val_loss: 0.5810 - val_acc: 0
.6936
Epoch 61/500
 - 10s - loss: 0.5807 - acc: 0.6971 - val_loss: 0.5777 - val_acc:
0.6980
Epoch 62/500
 - 10s - loss: 0.5809 - acc: 0.6973 - val_loss: 0.5801 - val_acc:
0.7017
Epoch 63/500
```

```
 - 10s - loss: 0.5804 - acc: 0.6970 - val_loss: 0.5754 - val_acc:
0.7006
Epoch 64/500
 - 10s - loss: 0.5810 - acc: 0.6977 - val_loss: 0.5800 - val_acc:
0.7009
Epoch 65/500
 - 10s - loss: 0.5812 - acc: 0.6970 - val_loss: 0.5764 - val_acc:
0.7005
Epoch 66/500
 - 10s - loss: 0.5801 - acc: 0.6963 - val_loss: 0.5750 - val_acc:
0.6999
Epoch 67/500
 - 9s - loss: 0.5811 - acc: 0.6971 - val_loss: 0.5754 - val_acc: 0
.7008
Epoch 68/500
 - 10s - loss: 0.5805 - acc: 0.6957 - val_loss: 0.5806 - val_acc:
0.6912
Epoch 69/500
 - 10s - loss: 0.5806 - acc: 0.6972 - val_loss: 0.5778 - val_acc:
0.7013
Epoch 70/500
 - 10s - loss: 0.5801 - acc: 0.6967 - val_loss: 0.5761 - val_acc:
0.6994
Epoch 71/500
 - 10s - loss: 0.5803 - acc: 0.6965 - val_loss: 0.5769 - val_acc:
0.6994
Epoch 72/500
 - 10s - loss: 0.5804 - acc: 0.6967 - val_loss: 0.5745 - val_acc:
0.7006
Epoch 73/500
 - 10s - loss: 0.5805 - acc: 0.6969 - val_loss: 0.5761 - val_acc:
0.7003
Epoch 74/500
 - 10s - loss: 0.5810 - acc: 0.6980 - val_loss: 0.5754 - val_acc:
0.7015
Epoch 75/500
 - 10s - loss: 0.5803 - acc: 0.6980 - val_loss: 0.5745 - val_acc:
0.7004
Epoch 76/500
 - 10s - loss: 0.5801 - acc: 0.6974 - val_loss: 0.5952 - val_acc:
0.6745
Epoch 77/500
 - 10s - loss: 0.5803 - acc: 0.6982 - val_loss: 0.5849 - val_acc:
0.6902
Epoch 78/500
 - 9s - loss: 0.5805 - acc: 0.6960 - val_loss: 0.5777 - val_acc: 0
.6986
Epoch 79/500
 - 10s - loss: 0.5808 - acc: 0.6972 - val_loss: 0.5820 - val_acc:
0.6918
Epoch 80/500
 - 10s - loss: 0.5803 - acc: 0.6978 - val_loss: 0.5795 - val_acc:
0.6990
Epoch 81/500
 - 10s - loss: 0.5802 - acc: 0.6965 - val_loss: 0.5767 - val_acc:
0.6996
Epoch 82/500
```

```
 - 9s - loss: 0.5812 - acc: 0.6977 - val_loss: 0.5767 - val_acc: 0
.7011
Epoch 83/500
 - 10s - loss: 0.5804 - acc: 0.6973 - val_loss: 0.5768 - val_acc:
0.7001
Epoch 84/500
 - 10s - loss: 0.5803 - acc: 0.6968 - val_loss: 0.5769 - val_acc:
0.6999
Epoch 85/500
 - 9s - loss: 0.5810 - acc: 0.6977 - val_loss: 0.5752 - val_acc: 0
.7001
Epoch 86/500
 - 9s - loss: 0.5802 - acc: 0.6979 - val_loss: 0.5744 - val_acc: 0
.7020
Epoch 87/500
 - 10s - loss: 0.5804 - acc: 0.6967 - val_loss: 0.5747 - val_acc:
0.7012
Epoch 88/500
 - 10s - loss: 0.5801 - acc: 0.6967 - val_loss: 0.5758 - val_acc:
0.7005
Epoch 89/500
 - 10s - loss: 0.5806 - acc: 0.6962 - val_loss: 0.5758 - val_acc:
0.7009
Epoch 90/500
 - 10s - loss: 0.5810 - acc: 0.6970 - val_loss: 0.5861 - val_acc:
0.6795
Epoch 91/500
 - 10s - loss: 0.5802 - acc: 0.6966 - val_loss: 0.5998 - val_acc:
0.6652
Epoch 92/500
 - 10s - loss: 0.5811 - acc: 0.6971 - val_loss: 0.5786 - val_acc:
0.6970
Epoch 93/500
 - 10s - loss: 0.5804 - acc: 0.6978 - val_loss: 0.5794 - val_acc:
0.6974
Epoch 94/500
 - 10s - loss: 0.5805 - acc: 0.6956 - val_loss: 0.5750 - val_acc:
0.7025
Epoch 95/500
 - 10s - loss: 0.5806 - acc: 0.6978 - val_loss: 0.5791 - val_acc:
0.6977
Epoch 96/500
 - 10s - loss: 0.5803 - acc: 0.6980 - val_loss: 0.5798 - val_acc:
0.7006
Epoch 97/500
 - 10s - loss: 0.5803 - acc: 0.6968 - val_loss: 0.5738 - val_acc:
0.7025
Epoch 98/500
 - 10s - loss: 0.5802 - acc: 0.6968 - val_loss: 0.5880 - val_acc:
0.6841
Epoch 99/500
 - 9s - loss: 0.5800 - acc: 0.6980 - val_loss: 0.5756 - val_acc: 0
.6998
Epoch 100/500
 - 9s - loss: 0.5801 - acc: 0.6977 - val_loss: 0.5773 - val_acc: 0
.6997
Epoch 101/500
```

```
 - 9s - loss: 0.5800 - acc: 0.6983 - val_loss: 0.5784 - val_acc: 0
.6995
Epoch 102/500
 - 9s - loss: 0.5803 - acc: 0.6983 - val_loss: 0.5806 - val_acc: 0
.6956
Epoch 103/500
 - 9s - loss: 0.5803 - acc: 0.6969 - val_loss: 0.5768 - val_acc: 0
.7015
Epoch 104/500
 - 10s - loss: 0.5792 - acc: 0.6995 - val_loss: 0.5767 - val_acc:
0.7019
Epoch 105/500
 - 9s - loss: 0.5802 - acc: 0.6973 - val_loss: 0.5767 - val_acc: 0
.7000
Epoch 106/500
 - 9s - loss: 0.5808 - acc: 0.6968 - val_loss: 0.5756 - val_acc: 0
.7015
Epoch 107/500
 - 10s - loss: 0.5804 - acc: 0.6975 - val_loss: 0.5772 - val_acc:
0.7019
Epoch 108/500
 - 9s - loss: 0.5794 - acc: 0.6981 - val_loss: 0.5769 - val_acc: 0
.7009
Epoch 109/500
 - 9s - loss: 0.5806 - acc: 0.6977 - val_loss: 0.5794 - val_acc: 0
.6972
Epoch 110/500
 - 9s - loss: 0.5803 - acc: 0.6977 - val_loss: 0.5767 - val_acc: 0
.6998
Epoch 111/500
 - 10s - loss: 0.5798 - acc: 0.6966 - val_loss: 0.5773 - val_acc:
0.7010
Epoch 112/500
 - 9s - loss: 0.5804 - acc: 0.6970 - val_loss: 0.5745 - val_acc: 0
.7012
Epoch 113/500
 - 10s - loss: 0.5797 - acc: 0.6987 - val_loss: 0.5761 - val_acc:
0.6995
Epoch 114/500
 - 9s - loss: 0.5802 - acc: 0.6980 - val_loss: 0.5810 - val_acc: 0
.6917
Epoch 115/500
 - 10s - loss: 0.5799 - acc: 0.6985 - val_loss: 0.5785 - val_acc:
0.6979
Epoch 116/500
 - 9s - loss: 0.5797 - acc: 0.6968 - val_loss: 0.5760 - val_acc: 0
.7001
Epoch 117/500
 - 9s - loss: 0.5800 - acc: 0.6984 - val_loss: 0.5770 - val_acc: 0
.7001
Epoch 118/500
 - 10s - loss: 0.5800 - acc: 0.6967 - val_loss: 0.5795 - val_acc:
0.6991
Epoch 119/500
 - 9s - loss: 0.5807 - acc: 0.6975 - val_loss: 0.5804 - val_acc: 0
.6965
Epoch 120/500
```

```
 - 9s - loss: 0.5799 - acc: 0.6972 - val_loss: 0.5742 - val_acc: 0
.7000
Epoch 121/500
 - 10s - loss: 0.5799 - acc: 0.6978 - val_loss: 0.5825 - val_acc:
0.6912
Epoch 122/500
 - 9s - loss: 0.5803 - acc: 0.6990 - val_loss: 0.5768 - val_acc: 0
.6979
Epoch 123/500
 - 10s - loss: 0.5802 - acc: 0.6975 - val_loss: 0.5756 - val_acc:
0.6993
Epoch 124/500
 - 9s - loss: 0.5805 - acc: 0.6968 - val_loss: 0.5802 - val_acc: 0
.6989
Epoch 125/500
 - 10s - loss: 0.5807 - acc: 0.6978 - val_loss: 0.5767 - val_acc:
0.7007
Epoch 126/500
 - 9s - loss: 0.5803 - acc: 0.6966 - val_loss: 0.5765 - val_acc: 0
.6977
Epoch 127/500
 - 9s - loss: 0.5803 - acc: 0.6969 - val_loss: 0.5759 - val_acc: 0
.6984
Epoch 128/500
 - 9s - loss: 0.5800 - acc: 0.6979 - val_loss: 0.5770 - val_acc: 0
.6996
Epoch 129/500
 - 9s - loss: 0.5804 - acc: 0.6979 - val_loss: 0.5943 - val_acc: 0
.6797
Epoch 130/500
 - 9s - loss: 0.5798 - acc: 0.6984 - val_loss: 0.5755 - val_acc: 0
.6991
Epoch 131/500
 - 10s - loss: 0.5804 - acc: 0.6972 - val_loss: 0.5778 - val_acc:
0.6976
Epoch 132/500
 - 10s - loss: 0.5799 - acc: 0.6979 - val_loss: 0.5824 - val_acc:
0.6929
Epoch 133/500
 - 9s - loss: 0.5802 - acc: 0.6962 - val_loss: 0.5766 - val_acc: 0
.7001
Epoch 134/500
 - 9s - loss: 0.5799 - acc: 0.6979 - val_loss: 0.5773 - val_acc: 0
.7014
Epoch 135/500
 - 9s - loss: 0.5802 - acc: 0.6982 - val_loss: 0.5752 - val_acc: 0
.7015
Epoch 136/500
 - 9s - loss: 0.5802 - acc: 0.6971 - val_loss: 0.5770 - val_acc: 0
.7007
Epoch 137/500
 - 10s - loss: 0.5800 - acc: 0.6965 - val_loss: 0.5794 - val_acc:
0.6994
Epoch 138/500
 - 10s - loss: 0.5807 - acc: 0.6976 - val_loss: 0.5766 - val_acc:
0.7006
Epoch 139/500
```

```
 - 10s - loss: 0.5805 - acc: 0.6968 - val_loss: 0.5770 - val_acc:
0.7002
Epoch 140/500
 - 10s - loss: 0.5804 - acc: 0.6976 - val_loss: 0.5741 - val_acc:
0.7005
Epoch 141/500
 - 9s - loss: 0.5802 - acc: 0.6971 - val_loss: 0.5759 - val_acc: 0
.7007
Epoch 142/500
 - 9s - loss: 0.5805 - acc: 0.6973 - val_loss: 0.5886 - val_acc: 0
.6862
Epoch 143/500
 - 10s - loss: 0.5801 - acc: 0.6980 - val_loss: 0.5770 - val_acc:
0.7001
Epoch 144/500
 - 9s - loss: 0.5795 - acc: 0.6974 - val_loss: 0.5763 - val_acc: 0
.6997
Epoch 145/500
 - 10s - loss: 0.5798 - acc: 0.6976 - val_loss: 0.5752 - val_acc:
0.7007
Epoch 146/500
 - 10s - loss: 0.5798 - acc: 0.6979 - val_loss: 0.5912 - val_acc:
0.6855
Epoch 147/500
 - 10s - loss: 0.5797 - acc: 0.6986 - val_loss: 0.5770 - val_acc:
0.7004
Epoch 148/500
 - 9s - loss: 0.5802 - acc: 0.6978 - val_loss: 0.5747 - val_acc: 0
.7010
Epoch 149/500
 - 9s - loss: 0.5801 - acc: 0.6974 - val_loss: 0.5799 - val_acc: 0
.6973
Epoch 150/500
 - 9s - loss: 0.5800 - acc: 0.6981 - val_loss: 0.5844 - val_acc: 0
.6913
Epoch 151/500
 - 9s - loss: 0.5799 - acc: 0.6976 - val_loss: 0.5751 - val_acc: 0
.7009
Epoch 152/500
 - 10s - loss: 0.5796 - acc: 0.6981 - val_loss: 0.5793 - val_acc:
0.6988
Epoch 153/500
 - 10s - loss: 0.5799 - acc: 0.6975 - val_loss: 0.5761 - val_acc:
0.7000
Epoch 154/500
 - 9s - loss: 0.5805 - acc: 0.6976 - val_loss: 0.5769 - val_acc: 0
.7005
Epoch 155/500
 - 9s - loss: 0.5804 - acc: 0.6962 - val_loss: 0.5759 - val_acc: 0
.7012
Epoch 156/500
 - 10s - loss: 0.5801 - acc: 0.6978 - val_loss: 0.5791 - val_acc:
0.6990
Epoch 157/500
 - 10s - loss: 0.5800 - acc: 0.6971 - val_loss: 0.5753 - val_acc:
0.7009
Epoch 158/500
```

```
 - 10s - loss: 0.5797 - acc: 0.6975 - val_loss: 0.5936 - val_acc:
0.6783
Epoch 159/500
 - 10s - loss: 0.5804 - acc: 0.6971 - val_loss: 0.5792 - val_acc:
0.6993
Epoch 160/500
 - 10s - loss: 0.5801 - acc: 0.6986 - val_loss: 0.5759 - val_acc:
0.7012
Epoch 161/500
 - 9s - loss: 0.5802 - acc: 0.6977 - val_loss: 0.5778 - val_acc: 0
.6998
Epoch 162/500
 - 9s - loss: 0.5796 - acc: 0.6972 - val_loss: 0.5794 - val_acc: 0
.6972
Epoch 163/500
 - 9s - loss: 0.5798 - acc: 0.6975 - val_loss: 0.5740 - val_acc: 0
.7005
Epoch 164/500
 - 9s - loss: 0.5795 - acc: 0.6970 - val_loss: 0.5755 - val_acc: 0
.7016
Epoch 165/500
 - 10s - loss: 0.5801 - acc: 0.6977 - val_loss: 0.5855 - val_acc:
0.6896
Epoch 166/500
 - 10s - loss: 0.5797 - acc: 0.6977 - val_loss: 0.5747 - val_acc:
0.7017
Epoch 167/500
 - 10s - loss: 0.5803 - acc: 0.6969 - val_loss: 0.5740 - val_acc:
0.7019
Epoch 168/500
 - 9s - loss: 0.5806 - acc: 0.6975 - val_loss: 0.5766 - val_acc: 0
.7005
Epoch 169/500
 - 10s - loss: 0.5801 - acc: 0.6975 - val_loss: 0.5777 - val_acc:
0.6977
Epoch 170/500
 - 9s - loss: 0.5796 - acc: 0.6970 - val_loss: 0.5766 - val_acc: 0
.6998
Epoch 171/500
 - 9s - loss: 0.5795 - acc: 0.6975 - val_loss: 0.5755 - val_acc: 0
.7015
Epoch 172/500
 - 9s - loss: 0.5802 - acc: 0.6972 - val_loss: 0.5827 - val_acc: 0
.6936
Epoch 173/500
 - 10s - loss: 0.5801 - acc: 0.6974 - val_loss: 0.5749 - val_acc:
0.7016
Epoch 174/500
 - 10s - loss: 0.5805 - acc: 0.6973 - val_loss: 0.5760 - val_acc:
0.7015
Epoch 175/500
 - 9s - loss: 0.5796 - acc: 0.6970 - val_loss: 0.5779 - val_acc: 0
.7001
Epoch 176/500
 - 10s - loss: 0.5800 - acc: 0.6973 - val_loss: 0.5793 - val_acc:
0.6946
Epoch 177/500
```

```
 - 10s - loss: 0.5792 - acc: 0.6977 - val_loss: 0.5744 - val_acc:
0.7012
Epoch 178/500
 - 10s - loss: 0.5797 - acc: 0.6972 - val_loss: 0.5757 - val_acc:
0.7014
Epoch 179/500
 - 9s - loss: 0.5801 - acc: 0.6963 - val_loss: 0.5762 - val_acc: 0
.6991
Epoch 180/500
 - 10s - loss: 0.5801 - acc: 0.6971 - val_loss: 0.5757 - val_acc:
0.7002
Epoch 181/500
 - 9s - loss: 0.5799 - acc: 0.6977 - val_loss: 0.5739 - val_acc: 0
.7023
Epoch 182/500
 - 9s - loss: 0.5806 - acc: 0.6958 - val_loss: 0.5809 - val_acc: 0
.6964
Epoch 183/500
 - 9s - loss: 0.5804 - acc: 0.6962 - val_loss: 0.5762 - val_acc: 0
.7010
Epoch 184/500
 - 9s - loss: 0.5797 - acc: 0.6974 - val_loss: 0.5757 - val_acc: 0
.7011
Epoch 185/500
 - 9s - loss: 0.5797 - acc: 0.6967 - val_loss: 0.5767 - val_acc: 0
.7008
Epoch 186/500
 - 9s - loss: 0.5801 - acc: 0.6966 - val_loss: 0.5757 - val_acc: 0
.7019
Epoch 187/500
 - 10s - loss: 0.5796 - acc: 0.6977 - val_loss: 0.5764 - val_acc:
0.7001
Epoch 188/500
 - 9s - loss: 0.5796 - acc: 0.6980 - val_loss: 0.5758 - val_acc: 0
.6977
Epoch 189/500
 - 9s - loss: 0.5799 - acc: 0.6967 - val_loss: 0.5797 - val_acc: 0
.6983
Epoch 190/500
 - 9s - loss: 0.5795 - acc: 0.6974 - val_loss: 0.5780 - val_acc: 0
.6973
Epoch 191/500
 - 9s - loss: 0.5797 - acc: 0.6978 - val_loss: 0.5757 - val_acc: 0
.7014
Epoch 192/500
 - 9s - loss: 0.5791 - acc: 0.6981 - val_loss: 0.5771 - val_acc: 0
.6967
Epoch 193/500
 - 10s - loss: 0.5798 - acc: 0.6977 - val_loss: 0.5762 - val_acc:
0.6985
Epoch 194/500
 - 10s - loss: 0.5800 - acc: 0.6962 - val_loss: 0.5768 - val_acc:
0.7017
Epoch 195/500
 - 9s - loss: 0.5799 - acc: 0.6962 - val_loss: 0.5748 - val_acc: 0
.7021
Epoch 196/500
```

```
 - 9s - loss: 0.5794 - acc: 0.6978 - val_loss: 0.5760 - val_acc: 0
.7008
Epoch 197/500
 - 10s - loss: 0.5798 - acc: 0.6979 - val_loss: 0.5751 - val_acc:
0.7003
Epoch 198/500
 - 9s - loss: 0.5795 - acc: 0.6975 - val_loss: 0.5819 - val_acc: 0
.6945
Epoch 199/500
 - 9s - loss: 0.5794 - acc: 0.6979 - val_loss: 0.5752 - val_acc: 0
.6993
Epoch 200/500
 - 9s - loss: 0.5796 - acc: 0.6982 - val_loss: 0.5778 - val_acc: 0
.6995
Epoch 201/500
 - 10s - loss: 0.5802 - acc: 0.6964 - val_loss: 0.5756 - val_acc:
0.6999
Epoch 202/500
 - 9s - loss: 0.5793 - acc: 0.6977 - val_loss: 0.5788 - val_acc: 0
.6967
Epoch 203/500
 - 10s - loss: 0.5794 - acc: 0.6973 - val_loss: 0.5751 - val_acc:
0.7011
Epoch 204/500
 - 9s - loss: 0.5789 - acc: 0.6981 - val_loss: 0.5829 - val_acc: 0
.6915
Epoch 205/500
 - 9s - loss: 0.5796 - acc: 0.6980 - val_loss: 0.5769 - val_acc: 0
.7002
Epoch 206/500
 - 9s - loss: 0.5795 - acc: 0.6979 - val_loss: 0.5760 - val_acc: 0
.7007
Epoch 207/500
 - 9s - loss: 0.5797 - acc: 0.6980 - val_loss: 0.5776 - val_acc: 0
.7003
Epoch 208/500
 - 10s - loss: 0.5792 - acc: 0.6973 - val_loss: 0.5786 - val_acc:
0.7015
Epoch 209/500
 - 9s - loss: 0.5799 - acc: 0.6978 - val_loss: 0.5746 - val_acc: 0
.7014
Epoch 210/500
 - 10s - loss: 0.5798 - acc: 0.6969 - val_loss: 0.5780 - val_acc:
0.6997
Epoch 211/500
 - 9s - loss: 0.5797 - acc: 0.6976 - val_loss: 0.5801 - val_acc: 0
.6968
Epoch 212/500
 - 9s - loss: 0.5795 - acc: 0.6974 - val_loss: 0.5759 - val_acc: 0
.7012
Epoch 213/500
 - 9s - loss: 0.5794 - acc: 0.6979 - val_loss: 0.5756 - val_acc: 0
.7004
Epoch 214/500
 - 10s - loss: 0.5798 - acc: 0.6978 - val_loss: 0.5812 - val_acc:
0.6988
Epoch 215/500
```

```
 - 10s - loss: 0.5794 - acc: 0.6978 - val_loss: 0.5854 - val_acc:
0.6883
Epoch 216/500
 - 9s - loss: 0.5797 - acc: 0.6979 - val_loss: 0.5789 - val_acc: 0
.6994
Epoch 217/500
 - 9s - loss: 0.5802 - acc: 0.6972 - val_loss: 0.5811 - val_acc: 0
.6931
Epoch 218/500
 - 9s - loss: 0.5795 - acc: 0.6963 - val_loss: 0.5813 - val_acc: 0
.6953
Epoch 219/500
 - 9s - loss: 0.5796 - acc: 0.6977 - val_loss: 0.5761 - val_acc: 0
.7007
Epoch 220/500
 - 9s - loss: 0.5797 - acc: 0.6965 - val_loss: 0.5762 - val_acc: 0
.6995
Epoch 221/500
 - 9s - loss: 0.5802 - acc: 0.6960 - val_loss: 0.5767 - val_acc: 0
.7010
Epoch 222/500
 - 10s - loss: 0.5794 - acc: 0.6979 - val_loss: 0.5784 - val_acc:
0.6976
Epoch 223/500
 - 9s - loss: 0.5797 - acc: 0.6978 - val_loss: 0.5753 - val_acc: 0
.7002
Epoch 224/500
 - 9s - loss: 0.5802 - acc: 0.6967 - val_loss: 0.5767 - val_acc: 0
.7002
Epoch 225/500
 - 9s - loss: 0.5797 - acc: 0.6975 - val_loss: 0.5777 - val_acc: 0
.7012
Epoch 226/500
 - 10s - loss: 0.5799 - acc: 0.6978 - val_loss: 0.5800 - val_acc:
0.6982
Epoch 227/500
 - 10s - loss: 0.5799 - acc: 0.6971 - val_loss: 0.5738 - val_acc:
0.7008
Epoch 228/500
 - 9s - loss: 0.5795 - acc: 0.6976 - val_loss: 0.5808 - val_acc: 0
.6934
Epoch 229/500
 - 10s - loss: 0.5797 - acc: 0.6974 - val_loss: 0.5747 - val_acc:
0.7015
Epoch 230/500
 - 9s - loss: 0.5792 - acc: 0.6982 - val_loss: 0.5814 - val_acc: 0
.6929
Epoch 231/500
 - 9s - loss: 0.5803 - acc: 0.6964 - val_loss: 0.5771 - val_acc: 0
.7009
Epoch 232/500
 - 10s - loss: 0.5800 - acc: 0.6962 - val_loss: 0.5763 - val_acc:
0.6998
Epoch 233/500
 - 9s - loss: 0.5801 - acc: 0.6981 - val_loss: 0.5759 - val_acc: 0
.6989
Epoch 234/500
```

```
 - 9s - loss: 0.5792 - acc: 0.6981 - val_loss: 0.5854 - val_acc: 0
.6951
Epoch 235/500
 - 9s - loss: 0.5800 - acc: 0.6974 - val_loss: 0.5794 - val_acc: 0
.6984
Epoch 236/500
 - 10s - loss: 0.5797 - acc: 0.6980 - val_loss: 0.5750 - val_acc:
0.7004
Epoch 237/500
 - 9s - loss: 0.5799 - acc: 0.6974 - val_loss: 0.5827 - val_acc: 0
.6944
Epoch 238/500
 - 9s - loss: 0.5792 - acc: 0.6975 - val_loss: 0.5754 - val_acc: 0
.6994
Epoch 239/500
 - 10s - loss: 0.5796 - acc: 0.6972 - val_loss: 0.5743 - val_acc:
0.7022
Epoch 240/500
 - 9s - loss: 0.5794 - acc: 0.6975 - val_loss: 0.5814 - val_acc: 0
.6936
Epoch 241/500
 - 9s - loss: 0.5794 - acc: 0.6974 - val_loss: 0.5776 - val_acc: 0
.7011
Epoch 242/500
 - 10s - loss: 0.5803 - acc: 0.6974 - val_loss: 0.5757 - val_acc:
0.7011
Epoch 243/500
 - 10s - loss: 0.5798 - acc: 0.6969 - val_loss: 0.5797 - val_acc:
0.6954
Epoch 244/500
 - 9s - loss: 0.5791 - acc: 0.6974 - val_loss: 0.5821 - val_acc: 0
.6918
Epoch 245/500
 - 9s - loss: 0.5797 - acc: 0.6971 - val_loss: 0.5780 - val_acc: 0
.6986
Epoch 246/500
 - 9s - loss: 0.5792 - acc: 0.6979 - val_loss: 0.5754 - val_acc: 0
.6999
Epoch 247/500
 - 9s - loss: 0.5795 - acc: 0.6971 - val_loss: 0.5789 - val_acc: 0
.6982
Epoch 248/500
 - 9s - loss: 0.5789 - acc: 0.6966 - val_loss: 0.5772 - val_acc: 0
.7010
Epoch 249/500
 - 9s - loss: 0.5799 - acc: 0.6967 - val_loss: 0.5789 - val_acc: 0
.6962
Epoch 250/500
 - 10s - loss: 0.5794 - acc: 0.6974 - val_loss: 0.5800 - val_acc:
0.6987
Epoch 251/500
 - 9s - loss: 0.5796 - acc: 0.6977 - val_loss: 0.5773 - val_acc: 0
.7015
Epoch 252/500
 - 9s - loss: 0.5799 - acc: 0.6968 - val_loss: 0.5761 - val_acc: 0
.7003
Epoch 253/500
```

```
 - 9s - loss: 0.5802 - acc: 0.6970 - val_loss: 0.5773 - val_acc: 0
.7007
Epoch 254/500
 - 9s - loss: 0.5798 - acc: 0.6970 - val_loss: 0.5763 - val_acc: 0
.6994
Epoch 255/500
 - 9s - loss: 0.5792 - acc: 0.6967 - val_loss: 0.5762 - val_acc: 0
.7002
Epoch 256/500
 - 9s - loss: 0.5797 - acc: 0.6965 - val_loss: 0.5751 - val_acc: 0
.7003
Epoch 257/500
 - 10s - loss: 0.5793 - acc: 0.6974 - val_loss: 0.5769 - val_acc:
0.7008
Epoch 258/500
 - 9s - loss: 0.5797 - acc: 0.6970 - val_loss: 0.5753 - val_acc: 0
.7009
Epoch 259/500
 - 9s - loss: 0.5796 - acc: 0.6972 - val_loss: 0.5823 - val_acc: 0
.6954
Epoch 260/500
 - 10s - loss: 0.5795 - acc: 0.6965 - val_loss: 0.5760 - val_acc:
0.7002
Epoch 261/500
 - 9s - loss: 0.5803 - acc: 0.6968 - val_loss: 0.5764 - val_acc: 0
.6989
Epoch 262/500
 - 10s - loss: 0.5795 - acc: 0.6976 - val_loss: 0.5769 - val_acc:
0.6993
Epoch 263/500
 - 9s - loss: 0.5795 - acc: 0.6968 - val_loss: 0.5776 - val_acc: 0
.7009
Epoch 264/500
 - 10s - loss: 0.5794 - acc: 0.6970 - val_loss: 0.5741 - val_acc:
0.7010
Epoch 265/500
 - 10s - loss: 0.5793 - acc: 0.6977 - val_loss: 0.5859 - val_acc:
0.6900
Epoch 266/500
 - 9s - loss: 0.5799 - acc: 0.6970 - val_loss: 0.5774 - val_acc: 0
.7003
Epoch 267/500
 - 9s - loss: 0.5798 - acc: 0.6985 - val_loss: 0.5779 - val_acc: 0
.6976
Epoch 268/500
 - 9s - loss: 0.5787 - acc: 0.6974 - val_loss: 0.5764 - val_acc: 0
.6993
Epoch 269/500
 - 9s - loss: 0.5797 - acc: 0.6976 - val_loss: 0.5780 - val_acc: 0
.6968
Epoch 270/500
 - 9s - loss: 0.5792 - acc: 0.6983 - val_loss: 0.5767 - val_acc: 0
.7008
Epoch 271/500
 - 10s - loss: 0.5795 - acc: 0.6966 - val_loss: 0.5802 - val_acc:
0.6971
Epoch 272/500
```

```
 - 9s - loss: 0.5798 - acc: 0.6977 - val_loss: 0.5773 - val_acc: 0
.6998
Epoch 273/500
 - 9s - loss: 0.5788 - acc: 0.6982 - val_loss: 0.5762 - val_acc: 0
.7015
Epoch 274/500
 - 9s - loss: 0.5795 - acc: 0.6969 - val_loss: 0.5750 - val_acc: 0
.7012
Epoch 275/500
 - 9s - loss: 0.5793 - acc: 0.6969 - val_loss: 0.5748 - val_acc: 0
.7007
Epoch 276/500
 - 9s - loss: 0.5795 - acc: 0.6975 - val_loss: 0.5783 - val_acc: 0
.6988
Epoch 277/500
 - 9s - loss: 0.5791 - acc: 0.6982 - val_loss: 0.5793 - val_acc: 0
.6977
Epoch 278/500
 - 10s - loss: 0.5794 - acc: 0.6974 - val_loss: 0.5773 - val_acc:
0.7020
Epoch 279/500
 - 10s - loss: 0.5795 - acc: 0.6971 - val_loss: 0.5799 - val_acc:
0.6967
Epoch 280/500
 - 10s - loss: 0.5796 - acc: 0.6971 - val_loss: 0.5780 - val_acc:
0.7004
Epoch 281/500
 - 9s - loss: 0.5793 - acc: 0.6977 - val_loss: 0.5766 - val_acc: 0
.6998
Epoch 282/500
 - 9s - loss: 0.5786 - acc: 0.6973 - val_loss: 0.5769 - val_acc: 0
.6991
Epoch 283/500
 - 9s - loss: 0.5793 - acc: 0.6969 - val_loss: 0.5814 - val_acc: 0
.6955
Epoch 284/500
 - 9s - loss: 0.5800 - acc: 0.6958 - val_loss: 0.5752 - val_acc: 0
.7010
Epoch 285/500
 - 10s - loss: 0.5797 - acc: 0.6972 - val_loss: 0.5760 - val_acc:
0.7003
Epoch 286/500
 - 9s - loss: 0.5800 - acc: 0.6966 - val_loss: 0.5751 - val_acc: 0
.7009
Epoch 287/500
 - 10s - loss: 0.5791 - acc: 0.6982 - val_loss: 0.5766 - val_acc:
0.7013
Epoch 288/500
 - 9s - loss: 0.5790 - acc: 0.6967 - val_loss: 0.5772 - val_acc: 0
.7017
Epoch 289/500
 - 9s - loss: 0.5799 - acc: 0.6961 - val_loss: 0.5759 - val_acc: 0
.7025
Epoch 290/500
 - 10s - loss: 0.5794 - acc: 0.6959 - val_loss: 0.5801 - val_acc:
0.6950
Epoch 291/500
```

```
 - 9s - loss: 0.5792 - acc: 0.6965 - val_loss: 0.5769 - val_acc: 0
.7012
Epoch 292/500
 - 10s - loss: 0.5787 - acc: 0.6984 - val_loss: 0.5739 - val_acc:
0.7016
Epoch 293/500
 - 10s - loss: 0.5795 - acc: 0.6974 - val_loss: 0.5757 - val_acc:
0.6999
Epoch 294/500
 - 10s - loss: 0.5795 - acc: 0.6967 - val_loss: 0.5746 - val_acc:
0.7025
Epoch 295/500
 - 9s - loss: 0.5803 - acc: 0.6976 - val_loss: 0.5753 - val_acc: 0
.7022
Epoch 296/500
 - 10s - loss: 0.5800 - acc: 0.6974 - val_loss: 0.5780 - val_acc:
0.6968
Epoch 297/500
 - 10s - loss: 0.5803 - acc: 0.6971 - val_loss: 0.5868 - val_acc:
0.6859
Epoch 298/500
 - 11s - loss: 0.5799 - acc: 0.6963 - val_loss: 0.5770 - val_acc:
0.6988
Epoch 299/500
 - 10s - loss: 0.5791 - acc: 0.6971 - val_loss: 0.5810 - val_acc:
0.6964
Epoch 300/500
 - 10s - loss: 0.5798 - acc: 0.6966 - val_loss: 0.5794 - val_acc:
0.6956
Epoch 301/500
 - 9s - loss: 0.5803 - acc: 0.6960 - val_loss: 0.5780 - val_acc: 0
.6944
Epoch 302/500
 - 9s - loss: 0.5800 - acc: 0.6965 - val_loss: 0.5763 - val_acc: 0
.6995
Epoch 303/500
 - 9s - loss: 0.5799 - acc: 0.6967 - val_loss: 0.5785 - val_acc: 0
.6995
Epoch 304/500
 - 9s - loss: 0.5791 - acc: 0.6971 - val_loss: 0.5862 - val_acc: 0
.6856
Epoch 305/500
 - 10s - loss: 0.5800 - acc: 0.6962 - val_loss: 0.5789 - val_acc:
0.6953
Epoch 306/500
 - 9s - loss: 0.5799 - acc: 0.6959 - val_loss: 0.5782 - val_acc: 0
.6988
Epoch 307/500
 - 9s - loss: 0.5798 - acc: 0.6971 - val_loss: 0.5817 - val_acc: 0
.6909
Epoch 308/500
 - 10s - loss: 0.5799 - acc: 0.6967 - val_loss: 0.5759 - val_acc:
0.7005
Epoch 309/500
 - 10s - loss: 0.5794 - acc: 0.6955 - val_loss: 0.5743 - val_acc:
0.7010
Epoch 310/500
```

```
 - 10s - loss: 0.5799 - acc: 0.6960 - val_loss: 0.5784 - val_acc:
0.6948
Epoch 311/500
 - 10s - loss: 0.5801 - acc: 0.6980 - val_loss: 0.5763 - val_acc:
0.7018
Epoch 312/500
 - 10s - loss: 0.5799 - acc: 0.6966 - val_loss: 0.5761 - val_acc:
0.7011
Epoch 313/500
 - 10s - loss: 0.5796 - acc: 0.6975 - val_loss: 0.5784 - val_acc:
0.6956
Epoch 314/500
 - 10s - loss: 0.5803 - acc: 0.6965 - val_loss: 0.5770 - val_acc:
0.7026
Epoch 315/500
 - 9s - loss: 0.5795 - acc: 0.6979 - val_loss: 0.5809 - val_acc: 0
.6911
Epoch 316/500
 - 9s - loss: 0.5800 - acc: 0.6977 - val_loss: 0.5765 - val_acc: 0
.6979
Epoch 317/500
 - 10s - loss: 0.5808 - acc: 0.6968 - val_loss: 0.5839 - val_acc:
0.6899
Epoch 318/500
 - 9s - loss: 0.5802 - acc: 0.6978 - val_loss: 0.5760 - val_acc: 0
.7002
Epoch 319/500
 - 10s - loss: 0.5801 - acc: 0.6960 - val_loss: 0.5766 - val_acc:
0.7011
Epoch 320/500
 - 9s - loss: 0.5798 - acc: 0.6975 - val_loss: 0.5788 - val_acc: 0
.6973
Epoch 321/500
 - 9s - loss: 0.5807 - acc: 0.6964 - val_loss: 0.5749 - val_acc: 0
.7009
Epoch 322/500
 - 9s - loss: 0.5796 - acc: 0.6963 - val_loss: 0.5795 - val_acc: 0
.6989
Epoch 323/500
 - 9s - loss: 0.5801 - acc: 0.6966 - val_loss: 0.5755 - val_acc: 0
.7019
Epoch 324/500
 - 9s - loss: 0.5800 - acc: 0.6970 - val_loss: 0.5807 - val_acc: 0
.6940
Epoch 325/500
 - 9s - loss: 0.5798 - acc: 0.6962 - val_loss: 0.5779 - val_acc: 0
.6975
Epoch 326/500
 - 10s - loss: 0.5802 - acc: 0.6962 - val_loss: 0.5763 - val_acc:
0.7010
Epoch 327/500
 - 9s - loss: 0.5800 - acc: 0.6969 - val_loss: 0.5734 - val_acc: 0
.7023
Epoch 328/500
 - 9s - loss: 0.5795 - acc: 0.6968 - val_loss: 0.5796 - val_acc: 0
.6982
Epoch 329/500
```

```
 - 9s - loss: 0.5794 - acc: 0.6977 - val_loss: 0.5762 - val_acc: 0
.7004
Epoch 330/500
 - 9s - loss: 0.5796 - acc: 0.6968 - val_loss: 0.5790 - val_acc: 0
.6970
Epoch 331/500
 - 9s - loss: 0.5790 - acc: 0.6975 - val_loss: 0.5773 - val_acc: 0
.6950
Epoch 332/500
 - 9s - loss: 0.5794 - acc: 0.6975 - val_loss: 0.5839 - val_acc: 0
.6864
Epoch 333/500
 - 10s - loss: 0.5793 - acc: 0.6986 - val_loss: 0.5786 - val_acc:
0.6981
Epoch 334/500
 - 10s - loss: 0.5803 - acc: 0.6959 - val_loss: 0.5766 - val_acc:
0.7017
Epoch 335/500
 - 9s - loss: 0.5802 - acc: 0.6961 - val_loss: 0.5790 - val_acc: 0
.6972
Epoch 336/500
 - 9s - loss: 0.5791 - acc: 0.6970 - val_loss: 0.5755 - val_acc: 0
.7010
Epoch 337/500
 - 9s - loss: 0.5799 - acc: 0.6961 - val_loss: 0.5764 - val_acc: 0
.6998
Epoch 338/500
 - 10s - loss: 0.5793 - acc: 0.6968 - val_loss: 0.5829 - val_acc:
0.6886
Epoch 339/500
 - 9s - loss: 0.5795 - acc: 0.6973 - val_loss: 0.5779 - val_acc: 0
.6985
Epoch 340/500
 - 10s - loss: 0.5795 - acc: 0.6981 - val_loss: 0.5779 - val_acc:
0.6970
Epoch 341/500
 - 9s - loss: 0.5798 - acc: 0.6975 - val_loss: 0.5770 - val_acc: 0
.7001
Epoch 342/500
 - 9s - loss: 0.5799 - acc: 0.6967 - val_loss: 0.5779 - val_acc: 0
.6968
Epoch 343/500
 - 9s - loss: 0.5794 - acc: 0.6968 - val_loss: 0.5822 - val_acc: 0
.6931
Epoch 344/500
 - 9s - loss: 0.5800 - acc: 0.6958 - val_loss: 0.5818 - val_acc: 0
.6937
Epoch 345/500
 - 9s - loss: 0.5808 - acc: 0.6960 - val_loss: 0.5770 - val_acc: 0
.6977
Epoch 346/500
 - 9s - loss: 0.5793 - acc: 0.6972 - val_loss: 0.5752 - val_acc: 0
.7010
Epoch 347/500
 - 10s - loss: 0.5797 - acc: 0.6967 - val_loss: 0.5758 - val_acc:
0.6998
Epoch 348/500
```

```
 - 9s - loss: 0.5801 - acc: 0.6957 - val_loss: 0.5785 - val_acc: 0
.6982
Epoch 349/500
 - 9s - loss: 0.5803 - acc: 0.6970 - val_loss: 0.5753 - val_acc: 0
.7004
Epoch 350/500
 - 9s - loss: 0.5795 - acc: 0.6973 - val_loss: 0.5824 - val_acc: 0
.6931
Epoch 351/500
 - 9s - loss: 0.5798 - acc: 0.6960 - val_loss: 0.5812 - val_acc: 0
.6942
Epoch 352/500
 - 9s - loss: 0.5797 - acc: 0.6968 - val_loss: 0.5784 - val_acc: 0
.6954
Epoch 353/500
 - 10s - loss: 0.5796 - acc: 0.6969 - val_loss: 0.5798 - val_acc:
0.6924
Epoch 354/500
 - 10s - loss: 0.5798 - acc: 0.6970 - val_loss: 0.5758 - val_acc:
0.6988
Epoch 355/500
 - 9s - loss: 0.5797 - acc: 0.6965 - val_loss: 0.5755 - val_acc: 0
.7019
Epoch 356/500
 - 9s - loss: 0.5796 - acc: 0.6977 - val_loss: 0.5751 - val_acc: 0
.7016
Epoch 357/500
 - 9s - loss: 0.5795 - acc: 0.6963 - val_loss: 0.5787 - val_acc: 0
.6967
Epoch 358/500
 - 9s - loss: 0.5796 - acc: 0.6968 - val_loss: 0.5805 - val_acc: 0
.6940
Epoch 359/500
 - 9s - loss: 0.5794 - acc: 0.6983 - val_loss: 0.5754 - val_acc: 0
.7003
Epoch 360/500
 - 9s - loss: 0.5794 - acc: 0.6979 - val_loss: 0.5810 - val_acc: 0
.6930
Epoch 361/500
 - 10s - loss: 0.5792 - acc: 0.6964 - val_loss: 0.5762 - val_acc:
0.7005
Epoch 362/500
 - 9s - loss: 0.5798 - acc: 0.6965 - val_loss: 0.5779 - val_acc: 0
.6999
Epoch 363/500
 - 9s - loss: 0.5803 - acc: 0.6965 - val_loss: 0.5792 - val_acc: 0
.6927
Epoch 364/500
 - 9s - loss: 0.5797 - acc: 0.6964 - val_loss: 0.5779 - val_acc: 0
.7009
Epoch 365/500
 - 10s - loss: 0.5795 - acc: 0.6966 - val_loss: 0.5806 - val_acc:
0.6962
Epoch 366/500
 - 9s - loss: 0.5797 - acc: 0.6963 - val_loss: 0.5785 - val_acc: 0
.6956
Epoch 367/500
```

```
 - 9s - loss: 0.5799 - acc: 0.6965 - val_loss: 0.5797 - val_acc: 0
.6964
Epoch 368/500
 - 10s - loss: 0.5794 - acc: 0.6970 - val_loss: 0.5797 - val_acc:
0.6938
Epoch 369/500
 - 9s - loss: 0.5796 - acc: 0.6965 - val_loss: 0.5754 - val_acc: 0
.6991
Epoch 370/500
 - 9s - loss: 0.5792 - acc: 0.6966 - val_loss: 0.5781 - val_acc: 0
.7002
Epoch 371/500
 - 9s - loss: 0.5794 - acc: 0.6975 - val_loss: 0.5777 - val_acc: 0
.6982
Epoch 372/500
 - 9s - loss: 0.5798 - acc: 0.6976 - val_loss: 0.5796 - val_acc: 0
.6967
Epoch 373/500
 - 9s - loss: 0.5797 - acc: 0.6960 - val_loss: 0.5803 - val_acc: 0
.6980
Epoch 374/500
 - 10s - loss: 0.5795 - acc: 0.6975 - val_loss: 0.5755 - val_acc:
0.7008
Epoch 375/500
 - 10s - loss: 0.5798 - acc: 0.6962 - val_loss: 0.5809 - val_acc:
0.6940
Epoch 376/500
 - 9s - loss: 0.5800 - acc: 0.6967 - val_loss: 0.5770 - val_acc: 0
.6993
Epoch 377/500
 - 9s - loss: 0.5799 - acc: 0.6966 - val_loss: 0.5759 - val_acc: 0
.7008
Epoch 378/500
 - 10s - loss: 0.5794 - acc: 0.6968 - val_loss: 0.5770 - val_acc:
0.7008
Epoch 379/500
 - 9s - loss: 0.5789 - acc: 0.6974 - val_loss: 0.5752 - val_acc: 0
.7002
Epoch 380/500
 - 9s - loss: 0.5793 - acc: 0.6967 - val_loss: 0.5841 - val_acc: 0
.6902
Epoch 381/500
 - 9s - loss: 0.5798 - acc: 0.6970 - val_loss: 0.5790 - val_acc: 0
.6971
Epoch 382/500
 - 10s - loss: 0.5795 - acc: 0.6971 - val_loss: 0.5761 - val_acc:
0.7005
Epoch 383/500
 - 9s - loss: 0.5799 - acc: 0.6960 - val_loss: 0.5778 - val_acc: 0
.6983
Epoch 384/500
 - 9s - loss: 0.5796 - acc: 0.6968 - val_loss: 0.5762 - val_acc: 0
.7006
Epoch 385/500
 - 9s - loss: 0.5795 - acc: 0.6965 - val_loss: 0.5738 - val_acc: 0
.7012
Epoch 386/500
```

```
 - 9s - loss: 0.5795 - acc: 0.6960 - val_loss: 0.5765 - val_acc: 0
.6985
Epoch 387/500
 - 9s - loss: 0.5793 - acc: 0.6966 - val_loss: 0.5777 - val_acc: 0
.6990
Epoch 388/500
 - 10s - loss: 0.5796 - acc: 0.6967 - val_loss: 0.5786 - val_acc:
0.6983
Epoch 389/500
 - 10s - loss: 0.5793 - acc: 0.6971 - val_loss: 0.5739 - val_acc:
0.7016
Epoch 390/500
 - 9s - loss: 0.5796 - acc: 0.6956 - val_loss: 0.5747 - val_acc: 0
.7022
Epoch 391/500
 - 9s - loss: 0.5794 - acc: 0.6969 - val_loss: 0.5756 - val_acc: 0
.7003
Epoch 392/500
 - 9s - loss: 0.5800 - acc: 0.6968 - val_loss: 0.5751 - val_acc: 0
.7018
Epoch 393/500
 - 10s - loss: 0.5802 - acc: 0.6969 - val_loss: 0.5799 - val_acc:
0.6941
Epoch 394/500
 - 9s - loss: 0.5797 - acc: 0.6967 - val_loss: 0.5782 - val_acc: 0
.6970
Epoch 395/500
 - 10s - loss: 0.5795 - acc: 0.6962 - val_loss: 0.5782 - val_acc:
0.6990
Epoch 396/500
 - 10s - loss: 0.5793 - acc: 0.6971 - val_loss: 0.5810 - val_acc:
0.6937
Epoch 397/500
 - 10s - loss: 0.5790 - acc: 0.6963 - val_loss: 0.5768 - val_acc:
0.6992
Epoch 398/500
 - 9s - loss: 0.5796 - acc: 0.6966 - val_loss: 0.5772 - val_acc: 0
.6981
Epoch 399/500
 - 9s - loss: 0.5795 - acc: 0.6977 - val_loss: 0.5801 - val_acc: 0
.6944
Epoch 400/500
 - 9s - loss: 0.5790 - acc: 0.6970 - val_loss: 0.5758 - val_acc: 0
.7026
Epoch 401/500
 - 9s - loss: 0.5796 - acc: 0.6970 - val_loss: 0.5760 - val_acc: 0
.7001
Epoch 402/500
 - 10s - loss: 0.5791 - acc: 0.6978 - val_loss: 0.5849 - val_acc:
0.6889
Epoch 403/500
 - 10s - loss: 0.5797 - acc: 0.6971 - val_loss: 0.5746 - val_acc:
0.7014
Epoch 404/500
 - 9s - loss: 0.5800 - acc: 0.6959 - val_loss: 0.5745 - val_acc: 0
.7021
Epoch 405/500
```

```
 - 9s - loss: 0.5791 - acc: 0.6966 - val_loss: 0.5791 - val_acc: 0
.6973
Epoch 406/500
 - 10s - loss: 0.5794 - acc: 0.6960 - val_loss: 0.5785 - val_acc:
0.6980
Epoch 407/500
 - 9s - loss: 0.5795 - acc: 0.6978 - val_loss: 0.5776 - val_acc: 0
.7000
Epoch 408/500
 - 9s - loss: 0.5793 - acc: 0.6969 - val_loss: 0.5789 - val_acc: 0
.6949
Epoch 409/500
 - 10s - loss: 0.5793 - acc: 0.6969 - val_loss: 0.5806 - val_acc:
0.6958
Epoch 410/500
 - 9s - loss: 0.5788 - acc: 0.6978 - val_loss: 0.5788 - val_acc: 0
.6956
Epoch 411/500
 - 9s - loss: 0.5792 - acc: 0.6980 - val_loss: 0.5787 - val_acc: 0
.6944
Epoch 412/500
 - 10s - loss: 0.5793 - acc: 0.6964 - val_loss: 0.5776 - val_acc:
0.6978
Epoch 413/500
 - 9s - loss: 0.5798 - acc: 0.6970 - val_loss: 0.5798 - val_acc: 0
.6950
Epoch 414/500
 - 9s - loss: 0.5792 - acc: 0.6959 - val_loss: 0.5762 - val_acc: 0
.7013
Epoch 415/500
 - 10s - loss: 0.5791 - acc: 0.6976 - val_loss: 0.5761 - val_acc:
0.6990
Epoch 416/500
 - 10s - loss: 0.5797 - acc: 0.6979 - val_loss: 0.5764 - val_acc:
0.6998
Epoch 417/500
 - 9s - loss: 0.5798 - acc: 0.6963 - val_loss: 0.5743 - val_acc: 0
.7016
Epoch 418/500
 - 9s - loss: 0.5799 - acc: 0.6970 - val_loss: 0.5771 - val_acc: 0
.7014
Epoch 419/500
 - 9s - loss: 0.5796 - acc: 0.6967 - val_loss: 0.5757 - val_acc: 0
.7022
Epoch 420/500
 - 9s - loss: 0.5792 - acc: 0.6971 - val_loss: 0.5753 - val_acc: 0
.7012
Epoch 421/500
 - 9s - loss: 0.5796 - acc: 0.6966 - val_loss: 0.5771 - val_acc: 0
.6990
Epoch 422/500
 - 9s - loss: 0.5801 - acc: 0.6975 - val_loss: 0.5770 - val_acc: 0
.6977
Epoch 423/500
 - 10s - loss: 0.5797 - acc: 0.6986 - val_loss: 0.5756 - val_acc:
0.7017
Epoch 424/500
```

```
 - 9s - loss: 0.5796 - acc: 0.6969 - val_loss: 0.5770 - val_acc: 0
.7002
Epoch 425/500
 - 9s - loss: 0.5794 - acc: 0.6963 - val_loss: 0.5767 - val_acc: 0
.6993
Epoch 426/500
 - 9s - loss: 0.5792 - acc: 0.6967 - val_loss: 0.5761 - val_acc: 0
.7019
Epoch 427/500
 - 9s - loss: 0.5793 - acc: 0.6965 - val_loss: 0.5758 - val_acc: 0
.6986
Epoch 428/500
 - 9s - loss: 0.5786 - acc: 0.6971 - val_loss: 0.5751 - val_acc: 0
.7017
Epoch 429/500
 - 9s - loss: 0.5801 - acc: 0.6970 - val_loss: 0.5778 - val_acc: 0
.6988
Epoch 430/500
 - 10s - loss: 0.5798 - acc: 0.6975 - val_loss: 0.5802 - val_acc:
0.6976
Epoch 431/500
 - 9s - loss: 0.5795 - acc: 0.6973 - val_loss: 0.5750 - val_acc: 0
.6995
Epoch 432/500
 - 9s - loss: 0.5795 - acc: 0.6966 - val_loss: 0.5759 - val_acc: 0
.6980
Epoch 433/500
 - 9s - loss: 0.5802 - acc: 0.6981 - val_loss: 0.5785 - val_acc: 0
.6983
Epoch 434/500
 - 9s - loss: 0.5794 - acc: 0.6958 - val_loss: 0.5771 - val_acc: 0
.6996
Epoch 435/500
 - 9s - loss: 0.5794 - acc: 0.6968 - val_loss: 0.5751 - val_acc: 0
.6998
Epoch 436/500
 - 9s - loss: 0.5789 - acc: 0.6968 - val_loss: 0.5816 - val_acc: 0
.6935
Epoch 437/500
 - 10s - loss: 0.5792 - acc: 0.6973 - val_loss: 0.5821 - val_acc:
0.6939
Epoch 438/500
 - 9s - loss: 0.5796 - acc: 0.6961 - val_loss: 0.5765 - val_acc: 0
.7000
Epoch 439/500
 - 9s - loss: 0.5788 - acc: 0.6978 - val_loss: 0.5745 - val_acc: 0
.7009
Epoch 440/500
 - 9s - loss: 0.5788 - acc: 0.6969 - val_loss: 0.5785 - val_acc: 0
.6963
Epoch 441/500
 - 10s - loss: 0.5805 - acc: 0.6974 - val_loss: 0.5822 - val_acc:
0.6922
Epoch 442/500
 - 10s - loss: 0.5794 - acc: 0.6977 - val_loss: 0.5809 - val_acc:
0.6925
Epoch 443/500
```

```
 - 9s - loss: 0.5801 - acc: 0.6967 - val_loss: 0.5827 - val_acc: 0
.6900
Epoch 444/500
 - 10s - loss: 0.5793 - acc: 0.6968 - val_loss: 0.5757 - val_acc:
0.7009
Epoch 445/500
 - 9s - loss: 0.5798 - acc: 0.6971 - val_loss: 0.5834 - val_acc: 0
.6902
Epoch 446/500
 - 9s - loss: 0.5794 - acc: 0.6961 - val_loss: 0.5772 - val_acc: 0
.6972
Epoch 447/500
 - 9s - loss: 0.5795 - acc: 0.6969 - val_loss: 0.5775 - val_acc: 0
.7009
Epoch 448/500
 - 9s - loss: 0.5797 - acc: 0.6961 - val_loss: 0.5760 - val_acc: 0
.6991
Epoch 449/500
 - 9s - loss: 0.5792 - acc: 0.6974 - val_loss: 0.5778 - val_acc: 0
.7010
Epoch 450/500
 - 10s - loss: 0.5790 - acc: 0.6966 - val_loss: 0.5753 - val_acc:
0.7004
Epoch 451/500
 - 10s - loss: 0.5791 - acc: 0.6970 - val_loss: 0.5833 - val_acc:
0.6922
Epoch 452/500
 - 9s - loss: 0.5801 - acc: 0.6966 - val_loss: 0.5745 - val_acc: 0
.6999
Epoch 453/500
 - 9s - loss: 0.5795 - acc: 0.6972 - val_loss: 0.5748 - val_acc: 0
.7001
Epoch 454/500
 - 9s - loss: 0.5792 - acc: 0.6982 - val_loss: 0.5772 - val_acc: 0
.6991
Epoch 455/500
 - 9s - loss: 0.5789 - acc: 0.6972 - val_loss: 0.5753 - val_acc: 0
.7021
Epoch 456/500
 - 9s - loss: 0.5788 - acc: 0.6967 - val_loss: 0.5780 - val_acc: 0
.6949
Epoch 457/500
 - 9s - loss: 0.5797 - acc: 0.6968 - val_loss: 0.5769 - val_acc: 0
.6989
Epoch 458/500
 - 10s - loss: 0.5796 - acc: 0.6973 - val_loss: 0.5784 - val_acc:
0.6965
Epoch 459/500
 - 10s - loss: 0.5798 - acc: 0.6976 - val_loss: 0.5759 - val_acc:
0.7010
Epoch 460/500
 - 10s - loss: 0.5794 - acc: 0.6962 - val_loss: 0.5788 - val_acc:
0.6961
Epoch 461/500
 - 9s - loss: 0.5791 - acc: 0.6965 - val_loss: 0.5746 - val_acc: 0
.7023
Epoch 462/500
```

```
 - 9s - loss: 0.5792 - acc: 0.6967 - val_loss: 0.5855 - val_acc: 0
.6871
Epoch 463/500
 - 9s - loss: 0.5795 - acc: 0.6972 - val_loss: 0.5799 - val_acc: 0
.6985
Epoch 464/500
 - 9s - loss: 0.5796 - acc: 0.6962 - val_loss: 0.5752 - val_acc: 0
.7010
Epoch 465/500
 - 10s - loss: 0.5793 - acc: 0.6977 - val_loss: 0.5822 - val_acc:
0.6929
Epoch 466/500
 - 9s - loss: 0.5789 - acc: 0.6966 - val_loss: 0.5760 - val_acc: 0
.6983
Epoch 467/500
 - 9s - loss: 0.5797 - acc: 0.6974 - val_loss: 0.5773 - val_acc: 0
.6998
Epoch 468/500
 - 9s - loss: 0.5795 - acc: 0.6967 - val_loss: 0.5770 - val_acc: 0
.6995
Epoch 469/500
 - 9s - loss: 0.5793 - acc: 0.6982 - val_loss: 0.5773 - val_acc: 0
.6979
Epoch 470/500
 - 9s - loss: 0.5792 - acc: 0.6965 - val_loss: 0.5750 - val_acc: 0
.6998
Epoch 471/500
 - 9s - loss: 0.5794 - acc: 0.6968 - val_loss: 0.5769 - val_acc: 0
.6999
Epoch 472/500
 - 10s - loss: 0.5802 - acc: 0.6962 - val_loss: 0.5802 - val_acc:
0.6959
Epoch 473/500
 - 10s - loss: 0.5793 - acc: 0.6971 - val_loss: 0.5753 - val_acc:
0.7020
Epoch 474/500
 - 9s - loss: 0.5802 - acc: 0.6962 - val_loss: 0.5755 - val_acc: 0
.7009
Epoch 475/500
 - 9s - loss: 0.5797 - acc: 0.6963 - val_loss: 0.5746 - val_acc: 0
.7022
Epoch 476/500
 - 9s - loss: 0.5799 - acc: 0.6969 - val_loss: 0.5778 - val_acc: 0
.6979
Epoch 477/500
 - 9s - loss: 0.5800 - acc: 0.6971 - val_loss: 0.5826 - val_acc: 0
.6899
Epoch 478/500
 - 10s - loss: 0.5792 - acc: 0.6970 - val_loss: 0.5782 - val_acc:
0.7001
Epoch 479/500
 - 10s - loss: 0.5797 - acc: 0.6970 - val_loss: 0.5789 - val_acc:
0.6988
Epoch 480/500
 - 9s - loss: 0.5802 - acc: 0.6964 - val_loss: 0.5779 - val_acc: 0
.6970
Epoch 481/500
```

```
 - 9s - loss: 0.5796 - acc: 0.6964 - val_loss: 0.5808 - val_acc: 0
.6927
Epoch 482/500
 - 10s - loss: 0.5795 - acc: 0.6980 - val_loss: 0.5769 - val_acc:
0.6978
Epoch 483/500
 - 10s - loss: 0.5798 - acc: 0.6966 - val_loss: 0.5771 - val_acc:
0.7018
Epoch 484/500
 - 9s - loss: 0.5801 - acc: 0.6966 - val_loss: 0.5754 - val_acc: 0
.6987
Epoch 485/500
 - 10s - loss: 0.5792 - acc: 0.6979 - val_loss: 0.5780 - val_acc:
0.6970
Epoch 486/500
 - 9s - loss: 0.5789 - acc: 0.6976 - val_loss: 0.5750 - val_acc: 0
.7018
Epoch 487/500
 - 9s - loss: 0.5797 - acc: 0.6975 - val_loss: 0.5758 - val_acc: 0
.6996
Epoch 488/500
 - 10s - loss: 0.5796 - acc: 0.6979 - val_loss: 0.5731 - val_acc:
0.7030
Epoch 489/500
 - 9s - loss: 0.5802 - acc: 0.6963 - val_loss: 0.5782 - val_acc: 0
.6995
Epoch 490/500
 - 9s - loss: 0.5796 - acc: 0.6961 - val_loss: 0.5757 - val_acc: 0
.7015
Epoch 491/500
 - 10s - loss: 0.5794 - acc: 0.6962 - val_loss: 0.5793 - val_acc:
0.6960
Epoch 492/500
 - 9s - loss: 0.5799 - acc: 0.6969 - val_loss: 0.5753 - val_acc: 0
.7017
Epoch 493/500
 - 10s - loss: 0.5792 - acc: 0.6968 - val_loss: 0.5760 - val_acc:
0.7008
Epoch 494/500
 - 9s - loss: 0.5792 - acc: 0.6982 - val_loss: 0.5753 - val_acc: 0
.7022
Epoch 495/500
 - 10s - loss: 0.5794 - acc: 0.6975 - val_loss: 0.5792 - val_acc:
0.6974
Epoch 496/500
 - 9s - loss: 0.5789 - acc: 0.6967 - val_loss: 0.5780 - val_acc: 0
.6982
Epoch 497/500
 - 9s - loss: 0.5789 - acc: 0.6974 - val_loss: 0.5759 - val_acc: 0
.6997
Epoch 498/500
 - 9s - loss: 0.5794 - acc: 0.6975 - val_loss: 0.5770 - val_acc: 0
.7019
Epoch 499/500
 - 9s - loss: 0.5786 - acc: 0.6972 - val_loss: 0.5752 - val_acc: 0
.6997
Epoch 500/500
```

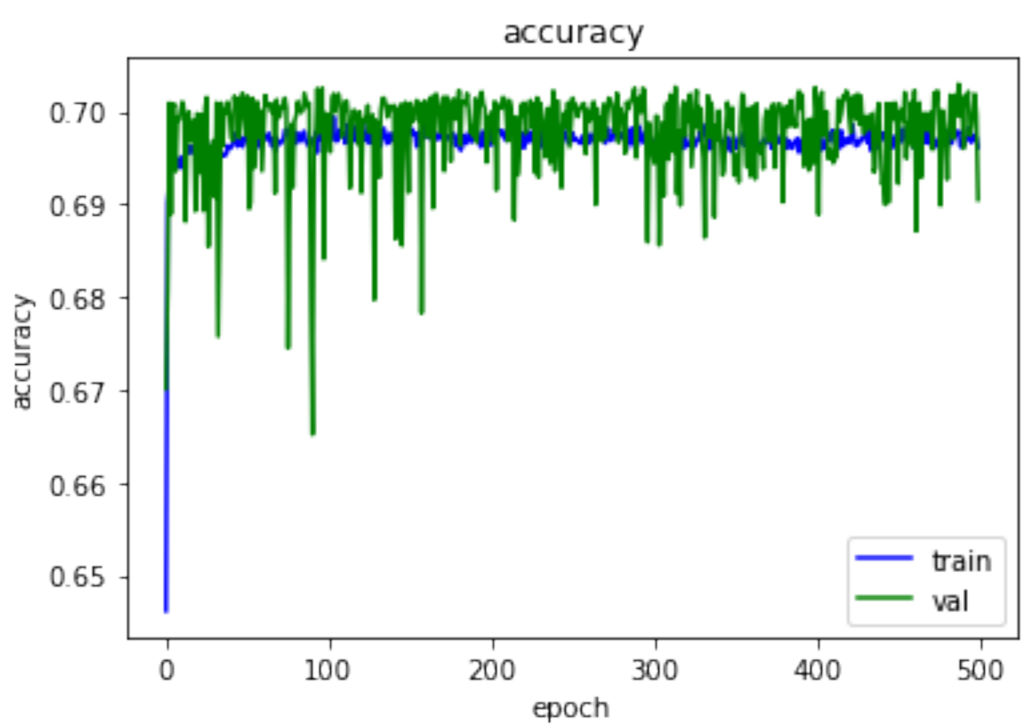
```
 - 10s - loss: 0.5787 - acc: 0.6961 - val_loss: 0.5808 - val_acc:
0.6905
```

In [86]:

```
#define accuracy & validation accuracy plot function
#print(model_record.history.keys())

def plot_model_accuracy(fit_model_obj, title):
    plt.plot(fit_model_obj.history['acc'], 'b')
    plt.plot(fit_model_obj.history['val_acc'], 'g')
    plt.title(title)
    plt.ylabel('accuracy')
    plt.xlabel('epoch')
    plt.legend(["train","val"], loc='lower right')
    plt.show()

    #define loss plot function
def plot_model_loss(fit_model_obj, title):
    plt.plot(fit_model_obj.history['loss'],'b')
    plt.plot(fit_model_obj.history['val_loss'], 'g')
    plt.title(title)
    plt.ylabel('loss')
    plt.xlabel('epoch')
    plt.legend(["train", "val"], loc='upper right')
    plt.show()
```

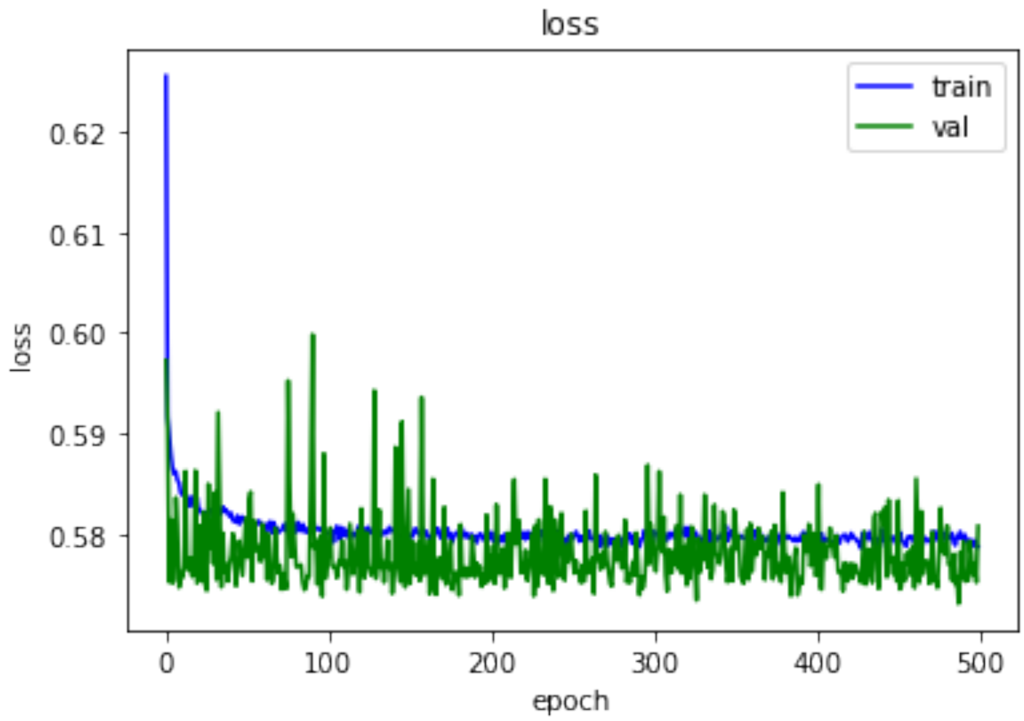
In [87]:

```
plot_model_accuracy(ANN_log, 'accuracy')
```



In [88]:

```
plot_model_loss(ANN_log, 'loss')
```



In [89]:

```
print("Accuracy: %0.2f [%s]" % (max(ANN_log.history['val_acc']), 'ANN'))
```

Accuracy: 0.70 [ANN]