In [1]:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import sqlite3
import numpy as np
from numpy import random
database = 'database.sqlite'
conn = sqlite3.connect(database)
```

In [2]:

```
query = """select * from match where home_player_X1 is not null and 'home_player_Y1
and  home_player_X2 is not null and 'home_player_Y2' is not null
and  home_player_X3 is not null and home_player_Y3 is not null
and  home_player_X4 is not null and home_player_Y4 is not null
and  home_player_X5 is not null and home_player_Y5 is not null
and  home_player_X6 is not null and home_player_Y6 is not null
and  home_player_X7 is not null and home_player_Y7 is not null
and  home_player_X8 is not null and home_player_Y8 is not null
and  home_player_X9 is not null and home_player_Y9 is not null
and  home_player_X10 is not null and home_player_Y10 is not null
and  home_player_X11 is not null and home_player_Y11 is not null
and  away_player_X1 is not null and away_player_Y1 is not null
and  away_player_X2 is not null and away_player_Y2 is not null
and  away_player_X3 is not null and away_player_Y3 is not null
and  away_player_X4 is not null and away_player_Y4 is not null
and  away_player_X5 is not null and away_player_Y5 is not null
and  away_player_X6 is not null and away_player_Y6 is not null
and  away_player_X7 is not null and away_player_Y7 is not null
and  away_player_X8 is not null and away_player_Y8 is not null
and  away_player_X9 is not null and away_player_Y9 is not null
and  away_player_X10 is not null and away_player_Y10 is not null
and  away_player_X11 is not null and away_player_Y11 is not null
and home_team_goal is not null and away_team_goal is not null
and home_player_1 is not null
and home_player_2 is not null
and home_player_3 is not null
and home_player_4 is not null
and home_player_5 is not null
and home_player_6 is not null
and home_player_7 is not null
and home_player_8 is not null
and home_player_9 is not null
and home_player_10 is not null
and home_player_11 is not null
and away_player_1 is not null
and away_player_2 is not null
and away_player_3 is not null
and away_player_4 is not null
and away_player_5 is not null
and away_player_6 is not null
and away_player_7 is not null
and away_player_8 is not null
and away_player_9 is not null
and away_player_10 is not null
and away_player_11 is not null
and  B365H is not null and B365D is not null and B365A is not null;"""
matches = pd.read_sql(query, conn)
matches
```

Out[2]:

	id	country_id	league_id	season	stage	date	match_api_id	home_team_api_id	a
0	146	1	1	2008/2009	24	2009-02-27 00:00:00	493017	8203	9
1	154	1	1	2008/2009	25	2009-03-08	493025	9984	8

1	154	1	1	2008/2009	25	03-08 00:00:00	493025	9984	8
2	156	1	1	2008/2009	25	2009-03-07 00:00:00	493027	8635	1
3	163	1	1	2008/2009	26	2009-03-13	493034	8203	8

In [3]:

player_api_id	player_api_id	date_stat

```
query = """ select * from Player as p ,Player_Stats as s where p.player_api_id = s.
player = pd.read_sql(query, conn)
player[['player_api_id','date_stat']]
```

Out[3]:

	player_api_id	player_api_id	date_stat
0	505942	505942	2016-02-18 00:00:00
1	505942	505942	2015-11-19 00:00:00
2	505942	505942	2015-09-21 00:00:00
3	505942	505942	2015-03-20 00:00:00
4	505942	505942	2007-02-22 00:00:00
5	155782	155782	2016-04-21 00:00:00
6	155782	155782	2016-04-07 00:00:00
7	155782	155782	2016-01-07 00:00:00
8	155782	155782	2015-12-24 00:00:00
9	155782	155782	2015-12-17 00:00:00
10	155782	155782	2015-10-16 00:00:00
11	155782	155782	2015-09-25 00:00:00
12	155782	155782	2015-09-21 00:00:00
13	155782	155782	2015-01-09 00:00:00
14	155782	155782	2014-12-05 00:00:00
15	155782	155782	2014-11-07 00:00:00
16	155782	155782	2014-09-18 00:00:00
17	155782	155782	2014-05-02 00:00:00
18	155782	155782	2014-04-04 00:00:00
19	155782	155782	2014-03-14 00:00:00
20	155782	155782	2013-12-13 00:00:00
21	155782	155782	2013-11-08 00:00:00
22	155782	155782	2013-10-04 00:00:00
23	155782	155782	2013-09-20 00:00:00
24	155782	155782	2013-05-03 00:00:00
25	155782	155782	2013-03-22 00:00:00
26	155782	155782	2013-03-15 00:00:00
27	155782	155782	2013-02-22 00:00:00
28	155782	155782	2013-02-15 00:00:00

	player_api_id	player_api_id	date_stat
29	155782	155782	2012-08-31 00:00:00
...	...	...	...
183223	108760	108760	2014-09-18 00:00:00
183224	108760	108760	2013-12-06 00:00:00
183225	108760	108760	2013-11-22 00:00:00
183226	108760	108760	2013-09-20 00:00:00
183227	108760	108760	2009-08-30 00:00:00
183228	108760	108760	2007-02-22 00:00:00
183229	39494	39494	2016-04-14 00:00:00
183230	39494	39494	2016-02-11 00:00:00
183231	39494	39494	2015-10-09 00:00:00
183232	39494	39494	2015-09-21 00:00:00
183233	39494	39494	2015-01-09 00:00:00
183234	39494	39494	2014-10-31 00:00:00
183235	39494	39494	2014-09-18 00:00:00
183236	39494	39494	2014-03-28 00:00:00
183237	39494	39494	2014-03-14 00:00:00
183238	39494	39494	2014-02-28 00:00:00
183239	39494	39494	2013-09-20 00:00:00
183240	39494	39494	2013-04-26 00:00:00
183241	39494	39494	2013-02-15 00:00:00
183242	39494	39494	2012-08-31 00:00:00
183243	39494	39494	2012-02-22 00:00:00
183244	39494	39494	2011-08-30 00:00:00
183245	39494	39494	2011-02-22 00:00:00
183246	39494	39494	2010-08-30 00:00:00
183247	39494	39494	2010-02-22 00:00:00
183248	39494	39494	2009-08-30 00:00:00
183249	39494	39494	2009-02-22 00:00:00
183250	39494	39494	2008-08-30 00:00:00
183251	39494	39494	2007-08-30 00:00:00
183252	39494	39494	2007-02-22 00:00:00

183253 rows × 3 columns

In [4]:

```
drop = matches.columns.values[-27:-1]
print drop
#Removing other betting houses odds
matches = matches.drop(drop,1)
matches= matches.drop('BSA',1)
#Raw features
matches.columns.values
```

```
['BWH' 'BWD' 'BWA' 'IWH' 'IWD' 'IWA' 'LBH' 'LBD' 'LBA' 'PSH' 'PSD' 'PS
A'
 'WHH' 'WHD' 'WHA' 'SJH' 'SJD' 'SJA' 'VCH' 'VCD' 'VCA' 'GBH' 'GBD' 'GB
A'
 'BSH' 'BSD']
```

Out[4]:

```
array(['id', 'country_id', 'league_id', 'season', 'stage', 'date',
       'match_api_id', 'home_team_api_id', 'away_team_api_id',
       'home_team_goal', 'away_team_goal', 'home_player_X1',
       'home_player_X2', 'home_player_X3', 'home_player_X4',
       'home_player_X5', 'home_player_X6', 'home_player_X7',
       'home_player_X8', 'home_player_X9', 'home_player_X10',
       'home_player_X11', 'away_player_X1', 'away_player_X2',
       'away_player_X3', 'away_player_X4', 'away_player_X5',
       'away_player_X6', 'away_player_X7', 'away_player_X8',
       'away_player_X9', 'away_player_X10', 'away_player_X11',
       'home_player_Y1', 'home_player_Y2', 'home_player_Y3',
       'home_player_Y4', 'home_player_Y5', 'home_player_Y6',
       'home_player_Y7', 'home_player_Y8', 'home_player_Y9',
       'home_player_Y10', 'home_player_Y11', 'away_player_Y1',
       'away_player_Y2', 'away_player_Y3', 'away_player_Y4',
       'away_player_Y5', 'away_player_Y6', 'away_player_Y7',
       'away_player_Y8', 'away_player_Y9', 'away_player_Y10',
       'away_player_Y11', 'home_player_1', 'home_player_2',
       'home_player_3', 'home_player_4', 'home_player_5', 'home_player
_6',
       'home_player_7', 'home_player_8', 'home_player_9', 'home_player
_10',
       'home_player_11', 'away_player_1', 'away_player_2', 'away_playe
r_3',
       'away_player_4', 'away_player_5', 'away_player_6', 'away_player
_7',
       'away_player_8', 'away_player_9', 'away_player_10',
       'away_player_11', 'goal', 'shoton', 'shotoff', 'foulcommit', 'c
ard',
       'cross', 'corner', 'possession', 'B365H', 'B365D', 'B365A'], dt
ype=object)
```

In [5]:

```
matches.columns.values
```

Out[5]:

```
array(['id', 'country_id', 'league_id', 'season', 'stage', 'date',
       'match_api_id', 'home_team_api_id', 'away_team_api_id',
       'home_team_goal', 'away_team_goal', 'home_player_X1',
       'home_player_X2', 'home_player_X3', 'home_player_X4',
       'home_player_X5', 'home_player_X6', 'home_player_X7',
       'home_player_X8', 'home_player_X9', 'home_player_X10',
       'home_player_X11', 'away_player_X1', 'away_player_X2',
       'away_player_X3', 'away_player_X4', 'away_player_X5',
       'away_player_X6', 'away_player_X7', 'away_player_X8',
       'away_player_X9', 'away_player_X10', 'away_player_X11',
       'home_player_Y1', 'home_player_Y2', 'home_player_Y3',
       'home_player_Y4', 'home_player_Y5', 'home_player_Y6',
       'home_player_Y7', 'home_player_Y8', 'home_player_Y9',
       'home_player_Y10', 'home_player_Y11', 'away_player_Y1',
       'away_player_Y2', 'away_player_Y3', 'away_player_Y4',
       'away_player_Y5', 'away_player_Y6', 'away_player_Y7',
       'away_player_Y8', 'away_player_Y9', 'away_player_Y10',
       'away_player_Y11', 'home_player_1', 'home_player_2',
       'home_player_3', 'home_player_4', 'home_player_5', 'home_player
_6',
       'home_player_7', 'home_player_8', 'home_player_9', 'home_player
_10',
       'home_player_11', 'away_player_1', 'away_player_2', 'away_playe
r_3',
       'away_player_4', 'away_player_5', 'away_player_6', 'away_player
_7',
       'away_player_8', 'away_player_9', 'away_player_10',
       'away_player_11', 'goal', 'shoton', 'shotoff', 'foulcommit', 'c
ard',
       'cross', 'corner', 'possession', 'B365H', 'B365D', 'B365A'], dt
ype=object)
```

In [6]:

```
matches= matches.drop(['goal', 'shoton', 'shotoff', 'foulcommit', 'card',
       'cross', 'corner', 'possession'],1)
```

In [7]:

```
#Transforming date column into a date type
matches['date']=pd.to_datetime(matches['date'], format='%Y-%m-%d %H:%M:%S.%f')
#
matches['date']=matches['date']-matches['date'].unique()[0]
matches['date']=matches['date'].astype('timedelta64[D]')
```

In [8]:

```
#matches=matches.drop(['home_player_1', 'home_player_2','home_player_3', 'home_play
#       'home_player_7', 'home_player_8', 'home_player_9', 'home_player_10',
#       'home_player_11', 'away_player_1', 'away_player_2', 'away_player_3',
#       'away_player_4', 'away_player_5', 'away_player_6', 'away_player_7',
#       'away_player_8', 'away_player_9', 'away_player_10',
#       'away_player_11'])


#Transforming season column into categorical value
from sklearn import preprocessing
le = preprocessing.LabelEncoder()
matches['season'] = le.fit_transform(matches['season'].astype('str'))
```

In [9]:

```
matches['season'].describe()
```

Out[9]:

```
count    19600.000000
mean         3.714541
std          2.215884
min          0.000000
25%          2.000000
50%          4.000000
75%          6.000000
max          7.000000
Name: season, dtype: float64
```

In [10]:

```
matches.columns.values
```

Out[10]:

```
array(['id', 'country_id', 'league_id', 'season', 'stage', 'date',
       'match_api_id', 'home_team_api_id', 'away_team_api_id',
       'home_team_goal', 'away_team_goal', 'home_player_X1',
       'home_player_X2', 'home_player_X3', 'home_player_X4',
       'home_player_X5', 'home_player_X6', 'home_player_X7',
       'home_player_X8', 'home_player_X9', 'home_player_X10',
       'home_player_X11', 'away_player_X1', 'away_player_X2',
       'away_player_X3', 'away_player_X4', 'away_player_X5',
       'away_player_X6', 'away_player_X7', 'away_player_X8',
       'away_player_X9', 'away_player_X10', 'away_player_X11',
       'home_player_Y1', 'home_player_Y2', 'home_player_Y3',
       'home_player_Y4', 'home_player_Y5', 'home_player_Y6',
       'home_player_Y7', 'home_player_Y8', 'home_player_Y9',
       'home_player_Y10', 'home_player_Y11', 'away_player_Y1',
       'away_player_Y2', 'away_player_Y3', 'away_player_Y4',
       'away_player_Y5', 'away_player_Y6', 'away_player_Y7',
       'away_player_Y8', 'away_player_Y9', 'away_player_Y10',
       'away_player_Y11', 'home_player_1', 'home_player_2',
       'home_player_3', 'home_player_4', 'home_player_5', 'home_player
_6',
       'home_player_7', 'home_player_8', 'home_player_9', 'home_player
_10',
       'home_player_11', 'away_player_1', 'away_player_2', 'away_playe
r_3',
       'away_player_4', 'away_player_5', 'away_player_6', 'away_player
_7',
       'away_player_8', 'away_player_9', 'away_player_10',
       'away_player_11', 'B365H', 'B365D', 'B365A'], dtype=object)
```

In [ ]:

```

```

In [11]:

```
import graphlab as gl
gl.canvas.set_target('ipynb')
```

In [12]:

```

```

In [13]:

```
#train,data,test_data = matches_without_ids.random_split(.9, seed=0)
#matches['H']=(matches['home_team_goal']>matches['away_team_goal']).astype(int)
#matches['A']=(matches['home_team_goal']<matches['away_team_goal']).astype(int)
matches_without_ids = matches.drop(['home_player_1', 'home_player_2',
       'home_player_3', 'home_player_4', 'home_player_5', 'home_player_6',
       'home_player_7', 'home_player_8', 'home_player_9', 'home_player_10',
       'home_player_11', 'away_player_1', 'away_player_2', 'away_player_3',
       'away_player_4', 'away_player_5', 'away_player_6', 'away_player_7',
       'away_player_8', 'away_player_9', 'away_player_10',
       'away_player_11','id','home_team_goal','away_team_goal'],1)
```

In [14]:

```
matches_without_ids.columns.values
```

Out[14]:

```
array(['country_id', 'league_id', 'season', 'stage', 'date',
       'match_api_id', 'home_team_api_id', 'away_team_api_id',
       'home_player_X1', 'home_player_X2', 'home_player_X3',
       'home_player_X4', 'home_player_X5', 'home_player_X6',
       'home_player_X7', 'home_player_X8', 'home_player_X9',
       'home_player_X10', 'home_player_X11', 'away_player_X1',
       'away_player_X2', 'away_player_X3', 'away_player_X4',
       'away_player_X5', 'away_player_X6', 'away_player_X7',
       'away_player_X8', 'away_player_X9', 'away_player_X10',
       'away_player_X11', 'home_player_Y1', 'home_player_Y2',
       'home_player_Y3', 'home_player_Y4', 'home_player_Y5',
       'home_player_Y6', 'home_player_Y7', 'home_player_Y8',
       'home_player_Y9', 'home_player_Y10', 'home_player_Y11',
       'away_player_Y1', 'away_player_Y2', 'away_player_Y3',
       'away_player_Y4', 'away_player_Y5', 'away_player_Y6',
       'away_player_Y7', 'away_player_Y8', 'away_player_Y9',
       'away_player_Y10', 'away_player_Y11', 'B365H', 'B365D', 'B365
A',
       'Output'], dtype=object)
```

In [15]:

```
def determine_home_result(match):
    if match['home_team_goal'] > match['away_team_goal']:
        return 'H'
    elif match['home_team_goal'] < match['away_team_goal']:
        return 'A'
    else:
        return 'D'
matches['Output']=matches.apply(determine_home_result, axis=1)
```

In [15]:

```
pure_data = gl.SFrame(matches_without_ids)
train_data,test_data = pure_data.random_split(.8, seed=0)
#train,valid=train_data.random_split(.8,seed=0)
folds = gl.cross_validation.KFold(train_data, 5)
```

```
This non-commercial license of GraphLab Create for academic use is ass
igned to felipeapfernandes@gmail.com and will expire on August 26, 201
7.

[INFO] graphlab.cython.cy_server: GraphLab Create v2.1 started. Loggin
g: /tmp/graphlab_server_1476546209.log
```

In [16]:

```
model_kfolds=[]
for train,valid in folds:
    #(train,valid) = folds[i]
    model = gl.logistic_classifier.create(train,
                                                  target='Output',
                                                 features=['country_id', 'league_id
       'match_api_id', 'home_team_api_id', 'away_team_api_id',
       'home_player_X1', 'home_player_X2', 'home_player_X3',
       'home_player_X4', 'home_player_X5', 'home_player_X6',
       'home_player_X7', 'home_player_X8', 'home_player_X9',
       'home_player_X10', 'home_player_X11', 'away_player_X1',
       'away_player_X2', 'away_player_X3', 'away_player_X4',
       'away_player_X5', 'away_player_X6', 'away_player_X7',
       'away_player_X8', 'away_player_X9', 'away_player_X10',
       'away_player_X11', 'home_player_Y1', 'home_player_Y2',
       'home_player_Y3', 'home_player_Y4', 'home_player_Y5',
       'home_player_Y6', 'home_player_Y7', 'home_player_Y8',
       'home_player_Y9', 'home_player_Y10', 'home_player_Y11',
       'away_player_Y1', 'away_player_Y2', 'away_player_Y3',
       'away_player_Y4', 'away_player_Y5', 'away_player_Y6',
       'away_player_Y7', 'away_player_Y8', 'away_player_Y9',
       'away_player_Y10', 'away_player_Y11','B365H', 'B365D', 'B365A'],
                                                     validation_set=valid)
    model_kfolds.append(model)
```

```
WARNING: Detected extremely low variance for feature(s) 'home_player
_Y2', 'away_player_Y2', 'away_player_Y3' because all entries are nea
rly the same.
Proceeding with model training using all features. If the model does
 not provide results of adequate quality, exclude the above mentione
d feature(s) from the input dataset.

Logistic regression:

--------------------------------------------------------

Number of examples          : 12597

Number of classes           : 3

Number of feature columns   : 55

Number of unpacked features : 55

Number of coefficients    : 112
```

In [17]:

```
#result=pure_model.evaluate(test_data)
result=[]
for model in model_kfolds:
    result.append(model.evaluate(test_data))
```

In [18]:

```
#pure_model.show(view='Evaluation')
result
```

Out[18]:

```
[{'accuracy': 0.5284194134440695,
  'auc': 0.6526528046893878,
  'confusion_matrix': Columns:
        target_label    str
        predicted_label str
        count   int

  Rows: 9

  Data:
  +--------------+-----------------+-------+
  | target_label | predicted_label | count |
  +--------------+-----------------+-------+
  |      H       |        D        |   8   |
  |      A       |        D        |   12  |
  |      D       |        D        |   8   |
  |      D       |        H        |  710  |
  |      H       |        H        |  1495 |
```

In [19]:

```
top = model.predict_topk(test_data, output_type='probability', k = 3)
```

In [20]:

```
print top
```

```
+----+-------+----------------+
| id | class |  probability   |
+----+-------+----------------+
| 0  |   H   | 0.576467886553 |
| 0  |   D   | 0.234468528222 |
| 0  |   A   | 0.189063585225 |
| 1  |   H   | 0.388531413143 |
| 1  |   A   | 0.356242476119 |
| 1  |   D   | 0.255226110738 |
| 2  |   H   | 0.483082240589 |
| 2  |   D   | 0.272892422591 |
| 2  |   A   | 0.24402533682  |
| 3  |   H   | 0.502684594609 |
+----+-------+----------------+
[11559 rows x 3 columns]
Note: Only the head of the SFrame is printed.
You can use print_rows(num_rows=m, num_columns=n) to print more rows a
nd columns.
```

In [21]:

```
pred = model.predict(test_data)
```

In [22]:

```
(test_data['Output']==pred).sum()/(len(pred)*1.0)
```

Out[22]:

```
0.5258240332208669
```

In [ ]:

```

```

In [23]:

```
import re
def def_formations(matches_positions):
    pos=matches_positions.to_dataframe()
    form=[]
    for index,row in pos.iterrows():
        b= row.values
        dfs = (b <= 3).sum()
        mid1 = ((b >= 4) & (b<=6)).sum()
        mid2 = ((b >= 7) & (b<=9)).sum()
        atk1 = ((b >= 10)).sum()
        formation="%d-%d-%d-%d"%(dfs,mid1,mid2,atk1)
        formation = re.sub('0-','',formation)
        form.append(formation)
    return form
```

In [24]:

```
positions_home= pure_data[['home_player_Y2',
                      'home_player_Y3',
                      'home_player_Y4',
                      'home_player_Y5',
                      'home_player_Y6',
                      'home_player_Y7',
                      'home_player_Y8',
                      'home_player_Y9',
                      'home_player_Y10',
                      'home_player_Y11']]
positions_away = pure_data[['away_player_Y2',
                      'away_player_Y3',
                      'away_player_Y4',
                      'away_player_Y5',
                      'away_player_Y6',
                      'away_player_Y7',
                      'away_player_Y8',
                      'away_player_Y9',
                      'away_player_Y10',
                      'away_player_Y11']]
formation_home=def_formations(positions_home)
formation_away=def_formations(positions_away)
```

In [25]:

```
pure_data['formation_h']=formation_home
pure_data['formation_a']=formation_away

pure_data['formation_a'].unique()
```

Out[25]:

```
dtype: str
Rows: 19
['3-2-3-2', '3-5-2', '4-2-2-2', '3-1-4-2', '3-4-3', '4-4-2', '4-1-2-
3', '4-2-1-3', '3-3-3-1', '4-3-2-1', '4-1-4-1', '4-3-1-2', '4-5-1', '5
-3-2', '4-1-3-2', '4-3-3', '4-2-3-1', '3-6-1', '5-4-1']
```

In [26]:

```
data= pure_data
```

In [27]:

```
data=pure_data.to_dataframe().drop(['home_player_X1', 'home_player_X2', 'home_playe
       'home_player_X4', 'home_player_X5', 'home_player_X6',
       'home_player_X7', 'home_player_X8', 'home_player_X9',
       'home_player_X10', 'home_player_X11', 'away_player_X1',
       'away_player_X2', 'away_player_X3', 'away_player_X4',
       'away_player_X5', 'away_player_X6', 'away_player_X7',
       'away_player_X8', 'away_player_X9', 'away_player_X10',
       'away_player_X11', 'home_player_Y1', 'home_player_Y2',
       'home_player_Y3', 'home_player_Y4', 'home_player_Y5',
       'home_player_Y6', 'home_player_Y7', 'home_player_Y8',
       'home_player_Y9', 'home_player_Y10', 'home_player_Y11',
       'away_player_Y1', 'away_player_Y2', 'away_player_Y3',
       'away_player_Y4', 'away_player_Y5', 'away_player_Y6',
       'away_player_Y7', 'away_player_Y8', 'away_player_Y9',
       'away_player_Y10', 'away_player_Y11'],1)
```

In [28]:

```
data = gl.SFrame(data)
data.show()
```

country_id		league_id		season	
dtype:	**int**	dtype:	**int**	dtype:	**int**
num_unique (est.):	**9**	num_unique (est.):	**9**	num_unique (est.):	**8**
num_undefined:	**0**	num_undefined:	**0**	num_undefined:	**0**
min:	**1**	min:	**1**	min:	**0**
max:	**21,484**	max:	**21,484**	max:	**7**
median:	**10,223**	median:	**10,223**	median:	**4**
mean:	**10,322.024**	mean:	**10,322.024**	mean:	**3.71**
std:	**7,150.496**	std:	**7,150.496**	std:	**2.21**
distribution of values:		distribution of values:		distribution of values:	

In [29]:

```
train_data,test_data = data.random_split(.8, seed=0)
train,valid=train_data.random_split(.8,seed=0)
model = gl.logistic_classifier.create(train,target='Output',
                                      features=['country_id','league_id','season','
                                                'home_team_api_id','away_team_api_i
                                                'formation_h','formation_a'], valid
```

```
Logistic regression:

--------------------------------------------------------

Number of examples          : 12674

Number of classes           : 3

Number of feature columns   : 13

Number of unpacked features : 13

Number of coefficients    : 96

Starting Newton Method

--------------------------------------------------------

+-----------+----------+--------------+-------------------+-----------
----------+

| Iteration | Passes   | Elapsed Time | Training-accuracy | Validation
-accuracy |

+-----------+----------+--------------+-------------------+-----------
----------+

| 1         | 2        | 0.089409     | 0.530614          | 0.526196
          |

| 2         | 3        | 0.149522     | 0.534007          | 0.529450
          |

| 3         | 4        | 0.209564     | 0.533139          | 0.530426
          |

| 4         | 5        | 0.270951     | 0.533297          | 0.530101
          |

| 5         | 6        | 0.332927     | 0.533297          | 0.530101
          |

| 6         | 7        | 0.393710     | 0.533297          | 0.530101
          |

+-----------+----------+--------------+-------------------+-----------
----------+
```

In [30]:

```
model.evaluate(test_data)
```

```
SUCCESS: Optimal solution found.
```

Out[30]:

```
{'accuracy': 0.5315338697119127,
 'auc': 0.6526445952145282,
 'confusion_matrix': Columns:
        target_label    str
        predicted_label str
        count   int

 Rows: 9

 Data:
 +--------------+-----------------+-------+
 | target_label | predicted_label | count |
 +--------------+-----------------+-------+
 |      H       |        D        |   13  |
 |      A       |        D        |   15  |
 |      D       |        D        |   18  |
 |      H       |        H        |  1511 |
 |      D       |        H        |  708  |
 |      D       |        A        |  244  |
 |      H       |        A        |  228  |
 |      A       |        A        |  519  |
 |      A       |        H        |  597  |
 +--------------+-----------------+-------+
 [9 rows x 3 columns],
 'f1_score': 0.39538430285888615,
 'log_loss': 0.9864334331563913,
 'precision': 0.4838648243862084,
 'recall': 0.4466285216833064,
 'roc_curve': Columns:
        threshold       float
        fpr     float
        tpr     float
        p       int
        n       int
        class   int

 Rows: 300003

 Data:
 +-----------+----------------+-----+------+------+-------+
 | threshold |      fpr       | tpr |  p   |  n   | class |
 +-----------+----------------+-----+------+------+-------+
 |    0.0    |      1.0       | 1.0 | 1131 | 2722 |   0   |
 |   1e-05   |      1.0       | 1.0 | 1131 | 2722 |   0   |
 |   2e-05   | 0.999632623071 | 1.0 | 1131 | 2722 |   0   |
 |   3e-05   | 0.999632623071 | 1.0 | 1131 | 2722 |   0   |
 |   4e-05   | 0.999632623071 | 1.0 | 1131 | 2722 |   0   |
 |   5e-05   | 0.999265246143 | 1.0 | 1131 | 2722 |   0   |
 |   6e-05   | 0.999265246143 | 1.0 | 1131 | 2722 |   0   |
 |   7e-05   | 0.999265246143 | 1.0 | 1131 | 2722 |   0   |
 |   8e-05   | 0.999265246143 | 1.0 | 1131 | 2722 |   0   |
 |   9e-05   | 0.999265246143 | 1.0 | 1131 | 2722 |   0   |
 +-----------+----------------+-----+------+------+-------+
 [300003 rows x 6 columns]
 Note: Only the head of the SFrame is printed.
 You can use print_rows(num_rows=m, num_columns=n) to print more rows
 and columns.}
```

In [31]:

```
pred = model.predict(test_data)
(test_data['Output']==pred).sum()/(len(pred)*1.0)
```

Out[31]:

0.5315338697119127

Feature Engineering

In [40]:

```
def ExtractGoalTendency(values):
    #data = values.to_dataframe()
    values.sort_values(by=['league_id','season','stage'])
    return values




data_goals = ExtractGoalTendency(data.to_dataframe())
data_goals
```

Out[40]:

	country_id	league_id	season	stage	date	match_api_id	home_team_api_id
0	1	1	0	24	0.0	493017	8203
1	1	1	0	25	9.0	493025	9984
2	1	1	0	25	8.0	493027	8635
3	1	1	0	26	14.0	493034	8203
4	1	1	0	26	15.0	493040	10000
5	1	1	0	27	23.0	493045	9991
6	1	1	0	27	22.0	493048	9999
7	1	1	0	29	44.0	493061	8635
8	1	1	0	29	42.0	493062	9999
9	1	1	0	31	58.0	493082	9999
10	1	1	0	32	64.0	493089	10000
11	1	1	0	32	64.0	493092	9991
12	1	1	0	32	64.0	493094	10001
13	1	1	0	33	71.0	493097	9985
14	1	1	0	33	71.0	493103	8635
15	1	1	0	33	71.0	493105	9984
16	1	1	0	34	78.0	493106	9987
17	1	1	0	34	78.0	493107	9991
18	1	1	1	1	156.0	665321	9984
19	1	1	1	1	155.0	665322	9994
20	1	1	1	1	155.0	665323	8571
21	1	1	1	10	219.0	665411	8342
22	1	1	1	10	218.0	665417	8203
23	1	1	1	10	218.0	665421	9993
24	1	1	1	11	233.0	665425	8342
25	1	1	1	11	233.0	665427	9987

	country_id	league_id	season	stage	date	match_api_id	home_team_api_id
26	1	1	1	11	232.0	665429	10000
27	1	1	1	11	232.0	665430	9994
28	1	1	1	11	232.0	665435	10001
29	1	1	1	12	239.0	665438	9985
...	...	...	...	...	...	...	...
19570	21484	21484	7	6	2403.0	2030140	9783
19571	21484	21484	7	6	2403.0	2030141	9869
19572	21484	21484	7	7	2409.0	2030142	8302
19573	21484	21484	7	7	2408.0	2030144	9910
19574	21484	21484	7	7	2410.0	2030145	8581
19575	21484	21484	7	7	2410.0	2030146	9906
19576	21484	21484	7	7	2409.0	2030147	9864
19577	21484	21484	7	7	2410.0	2030148	8315
19578	21484	21484	7	7	2409.0	2030149	7878
19579	21484	21484	7	7	2409.0	2030150	8558
19580	21484	21484	7	7	2410.0	2030151	8370
19581	21484	21484	7	8	2423.0	2030152	8634
19582	21484	21484	7	8	2423.0	2030153	8372
19583	21484	21484	7	8	2424.0	2030155	10205
19584	21484	21484	7	8	2423.0	2030156	8633
19585	21484	21484	7	8	2424.0	2030157	8560
19586	21484	21484	7	8	2423.0	2030158	10267
19587	21484	21484	7	8	2424.0	2030159	9783
19588	21484	21484	7	8	2425.0	2030160	9869
19589	21484	21484	7	8	2423.0	2030161	8603
19590	21484	21484	7	9	2431.0	2030162	8634
19591	21484	21484	7	9	2430.0	2030163	8302
19592	21484	21484	7	9	2431.0	2030164	8306
19593	21484	21484	7	9	2430.0	2030165	9910
19594	21484	21484	7	9	2431.0	2030166	8581
19595	21484	21484	7	9	2431.0	2030167	9906
19596	21484	21484	7	9	2430.0	2030168	9864
19597	21484	21484	7	9	2432.0	2030169	8315
19598	21484	21484	7	9	2430.0	2030170	7878
19599	21484	21484	7	9	2429.0	2030171	8370

19600 rows × 14 columns