effectivepandaspl

March 18, 2023

[138]:
```
import pandas as pd
import numpy as np
%matplotlib inline
from IPython.display import display
```

[139]:
```
pd.__version__
```

[139]:
```
'1.4.4'
```

0.0.1 Loading data premier league matches for 3 seasons

[140]:
```
file_match = "match_pl_20230121.csv"
```

[141]:
```
matches = pd.read_csv(file_match, index_col=0, sep=';')
```

[142]:
```
matches
```

[142]:
```
          date   time            comp         round  day venue result   gf  \
0   13/08/2021  20:00  Premier League   Matchweek 1  Fri  Away      L  0.0
0   13/08/2021  20:00  Premier League   Matchweek 1  Fri  Home      W  2.0
0   14/08/2021  17:30  Premier League   Matchweek 1  Sat  Away      W  3.0
1   14/08/2021  15:00  Premier League   Matchweek 1  Sat  Home      W  3.0
0   14/08/2021  12:30  Premier League   Matchweek 1  Sat  Home      W  5.0
..         …      …               …           …  …   …    …   …
20  15/01/2023  14:00  Premier League  Matchweek 20  Sun  Away      L  0.0
28  18/01/2023  20:00  Premier League   Matchweek 7  Wed  Away      D  1.0
21  18/01/2023  20:00  Premier League   Matchweek 7  Wed  Home      D  1.0
29  19/01/2023  20:00  Premier League   Matchweek 7  Thu  Home      W  4.0
27  19/01/2023  20:00  Premier League   Matchweek 7  Thu  Away      L  2.0

     ga         opponent  …  match report  notes    sh  sot  dist   fk   pk  \
0   2.0        Brentford  …  Match Report    NaN  22.0  4.0  19.0  1.0  0.0
0   0.0          Arsenal  …  Match Report    NaN   8.0  3.0  12.1  0.0  0.0
0   0.0     Norwich City  …  Match Report    NaN  19.0  6.0  17.3  1.0  0.0
1   0.0   Crystal Palace  …  Match Report    NaN  13.0  6.0  21.0  4.0  0.0
0   1.0     Leeds United  …  Match Report    NaN  16.0  8.0  18.2  0.0  0.0
..  …               …  …           …     …    …   …     …   …   …
```

```
20  1.0          Chelsea  …  Match Report    NaN  10.0  5.0  17.9  0.0  0.0
28  1.0   Crystal Palace  …  Match Report    NaN  15.0  4.0  19.5  1.0  0.0
21  1.0   Manchester Utd  …  Match Report    NaN  10.0  5.0  18.2  2.0  0.0
29  2.0        Tottenham  …  Match Report    NaN  16.0  6.0  16.3  1.0  0.0
27  4.0  Manchester City  …  Match Report    NaN   9.0  3.0  16.5  0.0  0.0

   pkatt  season               team
0    0.0    2022            Arsenal
0    0.0    2022          Brentford
0    0.0    2022          Liverpool
1    0.0    2022            Chelsea
0    0.0    2022  Manchester United
..   …      …                   …
20   0.0    2023     Crystal Palace
28   0.0    2023  Manchester United
21   0.0    2023     Crystal Palace
29   0.0    2023    Manchester City
27   0.0    2023  Tottenham Hotspur

[1138 rows x 27 columns]
```

[143]:
```
matches.columns
```

[143]:
```
Index(['date', 'time', 'comp', 'round', 'day', 'venue', 'result', 'gf', 'ga',
       'opponent', 'xg', 'xga', 'poss', 'attendance', 'captain', 'formation',
       'referee', 'match report', 'notes', 'sh', 'sot', 'dist', 'fk', 'pk',
       'pkatt', 'season', 'team'],
      dtype='object')
```

[144]:
```
cols = ['team', 'opponent', 'round', 'date', 'venue','time', 'result']
```

0.0.2 Optimization of the memory used by the data

[145]:
```
matches[cols].dtypes
```

[145]:
```
team        object
opponent    object
round       object
date        object
venue       object
time        object
result      object
dtype: object
```

[146]:
```
matches[cols].memory_usage(deep=True)
```

```
[146]: Index        9104
       team        79467
       opponent    76359
       round       78182
       date        76246
       venue       69418
       time        70556
       result      66004
       dtype: int64
```

0.0.3 Original memory usage

```
[147]: #chaining
       (matches
        [cols]
        .memory_usage(deep=True)
        .sum()
       )
```

```
[147]: 525336
```

```
[ ]:
```

```
[148]: def investigate_column(column_name):
           return (matches
            [cols]
            [column_name]
            .value_counts(dropna=False)
           )
```

Investigate team and opponent columns

```
[149]: investigate_column(['team', 'opponent'])
```

```
[149]: team                        opponent
       Wolverhampton Wanderers     West Ham          4
       West Ham United             Wolves            4
       Everton                     Southampton       4
       Leeds United                Aston Villa       4
       Crystal Palace              Chelsea           4
                                                    ..
       Bournemouth                 Fulham            1
       Nottingham Forest           Arsenal           1
       Bournemouth                 Leeds United      1
       Southampton                 Nott'ham Forest   1
       Everton                     Bournemouth       1
       Length: 484, dtype: int64
```

TODO : it looks like a categorical columns. We have to convert it to a categorical type

[150]:
```
#chaining
(matches
 [cols]
 .astype({'team':'category','opponent':'category'})
 .memory_usage(deep=True)
 .sum() #reduced memory
)
```

[150]:
```
376040
```

investigate round column

[151]:
```
investigate_column('round')
```

[151]:
```
Matchweek 1     40
Matchweek 11    40
Matchweek 2     40
Matchweek 19    40
Matchweek 18    40
Matchweek 17    40
Matchweek 16    40
Matchweek 15    40
Matchweek 14    40
Matchweek 13    40
Matchweek 20    40
Matchweek 3     40
Matchweek 5     40
Matchweek 9     40
Matchweek 10    40
Matchweek 4     40
Matchweek 6     40
Matchweek 12    38
Matchweek 8     34
Matchweek 7     26
Matchweek 29    20
Matchweek 36    20
Matchweek 35    20
Matchweek 37    20
Matchweek 34    20
Matchweek 33    20
Matchweek 32    20
Matchweek 31    20
Matchweek 25    20
Matchweek 30    20
Matchweek 28    20
Matchweek 27    20
Matchweek 26    20
Matchweek 23    20
Matchweek 24    20
Matchweek 22    20
Matchweek 21    20
Matchweek 38    20
Name: round, dtype: int64
```

TODO : it looks like a categorical columns. We have to convert it to a categorical type

[152]:
```
#chaining
(matches
 [cols]
 .astype({'team':'category','opponent':'category', 'round':'category'})
 .memory_usage(deep=True)
 .sum() #reduced memory
)
```

[152]:
```
302681
```

Investigate date column

[153]:
```
investigate_column('date')
```

[153]:
```
22/05/2022    20
03/09/2022    16
12/11/2022    16
20/11/2021    16
19/02/2022    16
              ..
14/03/2022     2
07/03/2022     2
01/03/2022     2
27/02/2022     2
19/01/2023     2
Name: date, Length: 179, dtype: int64
```

TODO : We have to convert it to a pandas datetime type

[154]:
```
#chaining
(matches
 [cols]
 .astype({'team':'category', 'opponent':'category', 'round':'category'})
 .assign(date = pd.to_datetime(matches.date, infer_datetime_format=True))
 .memory_usage(deep=True)
 .sum() #reduced memory
)
```

[154]:
```
235539
```

investigate venue column

[155]:
```
investigate_column('venue')
```

[155]:
```
Away    569
Home    569
Name: venue, dtype: int64
```

TODO : it looks like a categorical columns. We have to convert it to a categorical type

[156]:
```
#chaining
(matches
 [cols]
 .astype({'team':'category', 'opponent':'category' ,'round':'category', 'venue':
 ↪'category'})
 .assign(date = pd.to_datetime(matches.date, infer_datetime_format=True))
 .memory_usage(deep=True)
 .sum() #reduced memory
)
```

[156]:
```
167489
```

investigate time column

[157]:
```
investigate_column('time')
```

[157]:
```
15:00    366
14:00    184
20:00    122
17:30     98
12:30     88
16:30     84
19:45     72
19:30     60
20:15     24
16:00     20
14:15      8
12:00      6
18:00      2
19:00      2
16:15      2
Name: time, dtype: int64
```

TODO : Let's only take the hours of the time columns

```
[158]: #chaining
(matches
 [cols]
 .astype({'team':'category', 'opponent':'category' ,'round':'category', 'venue':
 ↪'category'})
 .assign(date = pd.to_datetime(matches.date, infer_datetime_format=True))
 .assign(hour = matches.time.str.replace(":.+", "", regex=True).astype("int"))
 .drop(columns = ['time'])
 .memory_usage(deep=True)
 .sum() #reduced memory
)
```

```
[158]: 106037
```

investigate result column

```
[172]: investigate_column('result')
```

```
[172]: L    439
W    439
D    260
Name: result, dtype: int64
```

TODO : it looks like a categorical columns. We have to convert it to a categorical type

```
[173]: #chaining
(matches
 [cols]
 .astype({'team':'category', 'opponent':'category' ,'round':'category', 'venue':
 ↪'category', 'result':'category'})
 .assign(date = pd.to_datetime(matches.date, infer_datetime_format=True))
 .assign(hour = matches.time.str.replace(":.+", "", regex=True).astype("int"))
 .drop(columns = ['time'])
 .memory_usage(deep=True)
 .sum() #reduced memory
)
```

```
[173]: 41453
```

```
[174]: def new_matches_df(matches):
    cols = ['team', 'opponent', 'round', 'date', 'venue','time', 'result']
    return (matches
 [cols]
 .astype({'team':'category', 'opponent':'category' ,'round':'category', 'venue':
 ↪'category', 'result':'category'})
 .assign(date = pd.to_datetime(matches.date, infer_datetime_format=True))
 .assign(hour = matches.time.str.replace(":.+", "", regex=True).astype("int"))
 .drop(columns = ['time'])
)
new_matches = new_matches_df(matches)
```

[175]:
```
new_matches
```

[175]:
```
                 team         opponent         round       date venue result  \
0             Arsenal        Brentford   Matchweek 1 2021-08-13  Away      L
0           Brentford          Arsenal   Matchweek 1 2021-08-13  Home      W
0           Liverpool      Norwich City  Matchweek 1 2021-08-14  Away      W
1             Chelsea   Crystal Palace   Matchweek 1 2021-08-14  Home      W
0   Manchester United     Leeds United   Matchweek 1 2021-08-14  Home      W
..                ...              ...           ...        ...   ...    ...
20     Crystal Palace          Chelsea  Matchweek 20 2023-01-15  Away      L
28  Manchester United   Crystal Palace   Matchweek 7 2023-01-18  Away      D
21     Crystal Palace   Manchester Utd   Matchweek 7 2023-01-18  Home      D
29    Manchester City        Tottenham   Matchweek 7 2023-01-19  Home      W
27  Tottenham Hotspur  Manchester City   Matchweek 7 2023-01-19  Away      L

    hour
0     20
0     20
0     17
1     15
0     12
..   ...
20    14
28    20
21    20
29    20
27    20

[1138 rows x 7 columns]
```

[176]:
```
from IPython.display import display
#create a variable to display a intermediate state
def get_var(df, var_name):
    print('get_var')
    globals()[var_name] = df
    return df
```

[177]:
```
def new_matches_df(matches):
    cols = ['team', 'opponent', 'round', 'date', 'venue','time', 'result']
    return (matches
 [cols]
 .pipe(get_var, 'original_df')
 .astype({'team':'category', 'opponent':'category' ,'round':'category', 'venue':
↪'category', 'result':'category'})
 .assign(date = pd.to_datetime(matches.date, infer_datetime_format=True))
 .assign(hour = matches.time.str.replace(":.+", "", regex=True).astype("int"))
 .drop(columns = ['time'])
#  .pipe(lambda df: display(df) or df)
)
new_matches = new_matches_df(matches)
```

get_var

[178]:
```
original_df
```

[178]:
```
                 team         opponent        round        date venue   time  \
0             Arsenal        Brentford  Matchweek 1  13/08/2021  Away  20:00
0           Brentford          Arsenal  Matchweek 1  13/08/2021  Home  20:00
0           Liverpool     Norwich City  Matchweek 1  14/08/2021  Away  17:30
1             Chelsea   Crystal Palace  Matchweek 1  14/08/2021  Home  15:00
0   Manchester United     Leeds United  Matchweek 1  14/08/2021  Home  12:30
..                ...              ...          ...         ...   ...    ...
20     Crystal Palace          Chelsea Matchweek 20  15/01/2023  Away  14:00
28  Manchester United   Crystal Palace  Matchweek 7  18/01/2023  Away  20:00
21     Crystal Palace  Manchester Utd  Matchweek 7  18/01/2023  Home  20:00
29    Manchester City        Tottenham  Matchweek 7  19/01/2023  Home  20:00
27  Tottenham Hotspur  Manchester City  Matchweek 7  19/01/2023  Away  20:00

   result
0       L
0       W
0       W
1       W
0       W
..    ...
20      L
28      D
21      D
29      W
27      L

[1138 rows x 7 columns]
```

[ ]:
```

```

[179]:
```
original_memory_usage = (matches
 [cols]
 .memory_usage(deep=True)
 .sum()
)
```

[180]:
```
new_memory_usage = (new_matches
 .memory_usage(deep=True)
 .sum()
)
```

[181]:
```
original_memory_usage
```

[181]:
```
525336
```

[182]:
```
new_memory_usage
```

[182]:
```
41453
```

[183]:
```
memory_gained = (original_memory_usage - new_memory_usage)/
 ↪original_memory_usage *100
```

[184]:
```
memory_gained
```

[184]:
```
92.10924056223064
```

[ ]:
```

```