Scraping and Cleaning

September 3, 2017

In [62]:
```
# Import all the necessary libraries

import numpy as np
import pandas as pd
from datetime import datetime as dt
import itertools

%matplotlib inline
```

In [63]:
```
# Read data from the CSV into a dataframe

loc = "E:/Data Science/Football Project/Datasets/"

raw_data_1 = pd.read_csv(loc + '2000-01.csv')
raw_data_2 = pd.read_csv(loc + '2001-02.csv')
raw_data_3 = pd.read_csv(loc + '2002-03.csv')
raw_data_4 = pd.read_csv(loc + '2003-04.csv')
raw_data_5 = pd.read_csv(loc + '2004-05.csv')
raw_data_6 = pd.read_csv(loc + '2005-06.csv')
raw_data_7 = pd.read_csv(loc + '2006-07.csv')
raw_data_8 = pd.read_csv(loc + '2007-08.csv')
raw_data_9 = pd.read_csv(loc + '2008-09.csv')
raw_data_10 = pd.read_csv(loc + '2009-10.csv')
raw_data_11 = pd.read_csv(loc + '2010-11.csv')
raw_data_12 = pd.read_csv(loc + '2011-12.csv')
raw_data_13 = pd.read_csv(loc + '2012-13.csv')
raw_data_14 = pd.read_csv(loc + '2013-14.csv')
raw_data_15 = pd.read_csv(loc + '2014-15.csv')
raw_data_16 = pd.read_csv(loc + '2015-16.csv')
```

In [64]:
```
# Parse data as time

def parse_date(date):
    if date == '':
        return None
    else:
        return dt.strptime(date, '%d/%m/%y').date()
```

```
def parse_date_other(date):
    if date == '':
        return None
    else:
        return dt.strptime(date, '%d/%m/%Y').date()

raw_data_1.Date = raw_data_1.Date.apply(parse_date)
raw_data_2.Date = raw_data_2.Date.apply(parse_date)
raw_data_3.Date = raw_data_3.Date.apply(parse_date_other)          # The da
raw_data_4.Date = raw_data_4.Date.apply(parse_date)
raw_data_5.Date = raw_data_5.Date.apply(parse_date)
raw_data_6.Date = raw_data_6.Date.apply(parse_date)
raw_data_7.Date = raw_data_7.Date.apply(parse_date)
raw_data_8.Date = raw_data_8.Date.apply(parse_date)
raw_data_9.Date = raw_data_9.Date.apply(parse_date)
raw_data_10.Date = raw_data_10.Date.apply(parse_date)
raw_data_11.Date = raw_data_11.Date.apply(parse_date)
raw_data_12.Date = raw_data_12.Date.apply(parse_date)
raw_data_13.Date = raw_data_13.Date.apply(parse_date)
raw_data_14.Date = raw_data_14.Date.apply(parse_date)
raw_data_15.Date = raw_data_15.Date.apply(parse_date)
raw_data_16.Date = raw_data_16.Date.apply(parse_date)



#Gets all the statistics related to gameplay

columns_req = ['Date','HomeTeam','AwayTeam','FTHG','FTAG','FTR']

playing_statistics_1 = raw_data_1[columns_req]
playing_statistics_2 = raw_data_2[columns_req]
playing_statistics_3 = raw_data_3[columns_req]
playing_statistics_4 = raw_data_4[columns_req]
playing_statistics_5 = raw_data_5[columns_req]
playing_statistics_6 = raw_data_6[columns_req]
playing_statistics_7 = raw_data_7[columns_req]
playing_statistics_8 = raw_data_8[columns_req]
playing_statistics_9 = raw_data_9[columns_req]
playing_statistics_10 = raw_data_10[columns_req]
playing_statistics_11 = raw_data_11[columns_req]
playing_statistics_12 = raw_data_12[columns_req]
playing_statistics_13 = raw_data_13[columns_req]
playing_statistics_14 = raw_data_14[columns_req]
playing_statistics_15 = raw_data_15[columns_req]
playing_statistics_16 = raw_data_16[columns_req]
```

** GOALS SCORED AND CONCEDED AT THE END OF MATCHWEEK, ARRANGED BY TEAMS AND MATCHWEEK **

In [65]:
```
# Gets the goals scored agg arranged by teams and matchweek
def get_goals_scored(playing_stat):
    # Create a dictionary with team names as keys
    teams = {}
    for i in playing_stat.groupby('HomeTeam').mean().T.columns:
        teams[i] = []

    # the value corresponding to keys is a list containing the match locat
    for i in range(len(playing_stat)):
        HTGS = playing_stat.iloc[i]['FTHG']
        ATGS = playing_stat.iloc[i]['FTAG']
        teams[playing_stat.iloc[i].HomeTeam].append(HTGS)
        teams[playing_stat.iloc[i].AwayTeam].append(ATGS)

    # Create a dataframe for goals scored where rows are teams and cols ar
    GoalsScored = pd.DataFrame(data=teams, index = [i for i in range(1,39)
    GoalsScored[0] = 0
    # Aggregate to get uptil that point
    for i in range(2,39):
        GoalsScored[i] = GoalsScored[i] + GoalsScored[i-1]
    return GoalsScored



# Gets the goals conceded agg arranged by teams and matchweek
def get_goals_conceded(playing_stat):
    # Create a dictionary with team names as keys
    teams = {}
    for i in playing_stat.groupby('HomeTeam').mean().T.columns:
        teams[i] = []

    # the value corresponding to keys is a list containing the match locat
    for i in range(len(playing_stat)):
        ATGC = playing_stat.iloc[i]['FTHG']
        HTGC = playing_stat.iloc[i]['FTAG']
        teams[playing_stat.iloc[i].HomeTeam].append(HTGC)
        teams[playing_stat.iloc[i].AwayTeam].append(ATGC)

    # Create a dataframe for goals scored where rows are teams and cols ar
    GoalsConceded = pd.DataFrame(data=teams, index = [i for i in range(1,3
    GoalsConceded[0] = 0
    # Aggregate to get uptil that point
    for i in range(2,39):
        GoalsConceded[i] = GoalsConceded[i] + GoalsConceded[i-1]
    return GoalsConceded
```

```
def get_gss(playing_stat):
    GC = get_goals_conceded(playing_stat)
    GS = get_goals_scored(playing_stat)

    j = 0
    HTGS = []
    ATGS = []
    HTGC = []
    ATGC = []

    for i in range(380):
        ht = playing_stat.iloc[i].HomeTeam
        at = playing_stat.iloc[i].AwayTeam
        HTGS.append(GS.loc[ht][j])
        ATGS.append(GS.loc[at][j])
        HTGC.append(GC.loc[ht][j])
        ATGC.append(GC.loc[at][j])

        if ((i + 1)% 10) == 0:
            j = j + 1

    playing_stat['HTGS'] = HTGS
    playing_stat['ATGS'] = ATGS
    playing_stat['HTGC'] = HTGC
    playing_stat['ATGC'] = ATGC

    return playing_stat


# Apply to each dataset
playing_statistics_1 = get_gss(playing_statistics_1)
playing_statistics_2 = get_gss(playing_statistics_2)
playing_statistics_3 = get_gss(playing_statistics_3)
playing_statistics_4 = get_gss(playing_statistics_4)
playing_statistics_5 = get_gss(playing_statistics_5)
playing_statistics_6 = get_gss(playing_statistics_6)
playing_statistics_7 = get_gss(playing_statistics_7)
playing_statistics_8 = get_gss(playing_statistics_8)
playing_statistics_9 = get_gss(playing_statistics_9)
playing_statistics_10 = get_gss(playing_statistics_10)
playing_statistics_11 = get_gss(playing_statistics_11)
playing_statistics_12 = get_gss(playing_statistics_12)
playing_statistics_13 = get_gss(playing_statistics_13)
playing_statistics_14 = get_gss(playing_statistics_14)
playing_statistics_15 = get_gss(playing_statistics_15)
playing_statistics_16 = get_gss(playing_statistics_16)
```

```
E:\Installed_Programs\Anaconda2\lib\site-packages\ipykernel\__main__.py:68: Setting
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/i
E:\Installed_Programs\Anaconda2\lib\site-packages\ipykernel\__main__.py:69: Setting
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/i
E:\Installed_Programs\Anaconda2\lib\site-packages\ipykernel\__main__.py:70: Setting
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/i
E:\Installed_Programs\Anaconda2\lib\site-packages\ipykernel\__main__.py:71: Setting
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/i
```

GET RESPECTIVE POINTS:

```
In [66]: def get_points(result):
             if result == 'W':
                 return 3
             elif result == 'D':
                 return 1
             else:
                 return 0


         def get_cuml_points(matchres):
             matchres_points = matchres.applymap(get_points)
             for i in range(2,39):
                 matchres_points[i] = matchres_points[i] + matchres_points[i-1]

             matchres_points.insert(column =0, loc = 0, value = [0*i for i in range
             return matchres_points


         def get_matchres(playing_stat):
             # Create a dictionary with team names as keys
             teams = {}
             for i in playing_stat.groupby('HomeTeam').mean().T.columns:
                 teams[i] = []

             # the value corresponding to keys is a list containing the match resul
```

```
    for i in range(len(playing_stat)):
        if playing_stat.iloc[i].FTR == 'H':
            teams[playing_stat.iloc[i].HomeTeam].append('W')
            teams[playing_stat.iloc[i].AwayTeam].append('L')
        elif playing_stat.iloc[i].FTR == 'A':
            teams[playing_stat.iloc[i].AwayTeam].append('W')
            teams[playing_stat.iloc[i].HomeTeam].append('L')
        else:
            teams[playing_stat.iloc[i].AwayTeam].append('D')
            teams[playing_stat.iloc[i].HomeTeam].append('D')

    return pd.DataFrame(data=teams, index = [i for i in range(1,39)]).T

def get_agg_points(playing_stat):
    matchres = get_matchres(playing_stat)
    cum_pts = get_cuml_points(matchres)
    HTP = []
    ATP = []
    j = 0
    for i in range(380):
        ht = playing_stat.iloc[i].HomeTeam
        at = playing_stat.iloc[i].AwayTeam
        HTP.append(cum_pts.loc[ht][j])
        ATP.append(cum_pts.loc[at][j])

        if ((i + 1)% 10) == 0:
            j = j + 1

    playing_stat['HTP'] = HTP
    playing_stat['ATP'] = ATP
    return playing_stat

# Apply to each dataset
playing_statistics_1 = get_agg_points(playing_statistics_1)
playing_statistics_2 = get_agg_points(playing_statistics_2)
playing_statistics_3 = get_agg_points(playing_statistics_3)
playing_statistics_4 = get_agg_points(playing_statistics_4)
playing_statistics_5 = get_agg_points(playing_statistics_5)
playing_statistics_6 = get_agg_points(playing_statistics_6)
playing_statistics_7 = get_agg_points(playing_statistics_7)
playing_statistics_8 = get_agg_points(playing_statistics_8)
playing_statistics_9 = get_agg_points(playing_statistics_9)
playing_statistics_10 = get_agg_points(playing_statistics_10)
playing_statistics_11 = get_agg_points(playing_statistics_11)
playing_statistics_12 = get_agg_points(playing_statistics_12)
playing_statistics_13 = get_agg_points(playing_statistics_13)
playing_statistics_14 = get_agg_points(playing_statistics_14)
playing_statistics_15 = get_agg_points(playing_statistics_15)
```

```
        playing_statistics_16 = get_agg_points(playing_statistics_16)
```

```
E:\Installed_Programs\Anaconda2\lib\site-packages\ipykernel\__main__.py:54: Setting
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/i
E:\Installed_Programs\Anaconda2\lib\site-packages\ipykernel\__main__.py:55: Setting
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/i
```

GET TEAM FORM:

```
In [67]: def get_form(playing_stat,num):
             form = get_matchres(playing_stat)
             form_final = form.copy()
             for i in range(num,39):
                 form_final[i] = ''
                 j = 0
                 while j < num:
                     form_final[i] += form[i-j]
                     j += 1
             return form_final

         def add_form(playing_stat,num):
             form = get_form(playing_stat,num)
             h = ['M' for i in range(num * 10)]  # since form is not available for
             a = ['M' for i in range(num * 10)]

             j = num
             for i in range((num*10),380):
                 ht = playing_stat.iloc[i].HomeTeam
                 at = playing_stat.iloc[i].AwayTeam

                 past = form.loc[ht][j]                # get past n results
                 h.append(past[num-1])                     # 0 index is most recent

                 past = form.loc[at][j]                # get past n results.
                 a.append(past[num-1])                   # 0 index is most recent

                 if ((i + 1)% 10) == 0:
                     j = j + 1

             playing_stat['HM' + str(num)] = h
             playing_stat['AM' + str(num)] = a
```

```
            return playing_stat


        def add_form_df(playing_statistics):
            playing_statistics = add_form(playing_statistics,1)
            playing_statistics = add_form(playing_statistics,2)
            playing_statistics = add_form(playing_statistics,3)
            playing_statistics = add_form(playing_statistics,4)
            playing_statistics = add_form(playing_statistics,5)
            return playing_statistics

        # Make changes to df
        playing_statistics_1 = add_form_df(playing_statistics_1)
        playing_statistics_2 = add_form_df(playing_statistics_2)
        playing_statistics_3 = add_form_df(playing_statistics_3)
        playing_statistics_4 = add_form_df(playing_statistics_4)
        playing_statistics_5 = add_form_df(playing_statistics_5)
        playing_statistics_6 = add_form_df(playing_statistics_6)
        playing_statistics_7 = add_form_df(playing_statistics_7)
        playing_statistics_8 = add_form_df(playing_statistics_8)
        playing_statistics_9 = add_form_df(playing_statistics_9)
        playing_statistics_10 = add_form_df(playing_statistics_10)
        playing_statistics_11 = add_form_df(playing_statistics_11)
        playing_statistics_12 = add_form_df(playing_statistics_12)
        playing_statistics_13 = add_form_df(playing_statistics_13)
        playing_statistics_14 = add_form_df(playing_statistics_14)
        playing_statistics_15 = add_form_df(playing_statistics_15)
        playing_statistics_16 = add_form_df(playing_statistics_16)
```

```
E:\Installed_Programs\Anaconda2\lib\site-packages\ipykernel\__main__.py:31: Setting
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/i
E:\Installed_Programs\Anaconda2\lib\site-packages\ipykernel\__main__.py:32: Setting
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/i
```

```
In [68]: # Rearranging columns
         cols = ['Date', 'HomeTeam', 'AwayTeam', 'FTHG', 'FTAG', 'FTR', 'HTGS', 'AT
                 'HM4', 'HM5', 'AM1', 'AM2', 'AM3', 'AM4', 'AM5' ]

         playing_statistics_1 = playing_statistics_1[cols]
         playing_statistics_2 = playing_statistics_2[cols]
```

```
playing_statistics_3 = playing_statistics_3[cols]
playing_statistics_4 = playing_statistics_4[cols]
playing_statistics_5 = playing_statistics_5[cols]
playing_statistics_6 = playing_statistics_6[cols]
playing_statistics_7 = playing_statistics_7[cols]
playing_statistics_8 = playing_statistics_8[cols]
playing_statistics_9 = playing_statistics_9[cols]
playing_statistics_10 = playing_statistics_10[cols]
playing_statistics_11 = playing_statistics_11[cols]
playing_statistics_12 = playing_statistics_12[cols]
playing_statistics_13 = playing_statistics_13[cols]
playing_statistics_14 = playing_statistics_14[cols]
playing_statistics_15 = playing_statistics_15[cols]
playing_statistics_16 = playing_statistics_16[cols]
```

Get Last Year's Position as also an independent variable:

```
In [69]: Standings = pd.read_csv(loc + "EPLStandings.csv")
         Standings.set_index(['Team'], inplace=True)
         Standings = Standings.fillna(18)

         def get_last(playing_stat, Standings, year):
             HomeTeamLP = []
             AwayTeamLP = []
             for i in range(380):
                 ht = playing_stat.iloc[i].HomeTeam
                 at = playing_stat.iloc[i].AwayTeam
                 HomeTeamLP.append(Standings.loc[ht][year])
                 AwayTeamLP.append(Standings.loc[at][year])
             playing_stat['HomeTeamLP'] = HomeTeamLP
             playing_stat['AwayTeamLP'] = AwayTeamLP
             return playing_stat

         playing_statistics_1 = get_last(playing_statistics_1, Standings, 0)
         playing_statistics_2 = get_last(playing_statistics_2, Standings, 1)
         playing_statistics_3 = get_last(playing_statistics_3, Standings, 2)
         playing_statistics_4 = get_last(playing_statistics_4, Standings, 3)
         playing_statistics_5 = get_last(playing_statistics_5, Standings, 4)
         playing_statistics_6 = get_last(playing_statistics_6, Standings, 5)
         playing_statistics_7 = get_last(playing_statistics_7, Standings, 6)
         playing_statistics_8 = get_last(playing_statistics_8, Standings, 7)
         playing_statistics_9 = get_last(playing_statistics_9, Standings, 8)
         playing_statistics_10 = get_last(playing_statistics_10, Standings, 9)
         playing_statistics_11 = get_last(playing_statistics_11, Standings, 10)
         playing_statistics_12 = get_last(playing_statistics_12, Standings, 11)
         playing_statistics_13 = get_last(playing_statistics_13, Standings, 12)
         playing_statistics_14 = get_last(playing_statistics_14, Standings, 13)
         playing_statistics_15 = get_last(playing_statistics_15, Standings, 14)
         playing_statistics_16 = get_last(playing_statistics_16, Standings, 15)
```

Get MatchWeek:

```
In [70]: def get_mw(playing_stat):
             j = 1
             MatchWeek = []
             for i in range(380):
                 MatchWeek.append(j)
                 if ((i + 1)% 10) == 0:
                     j = j + 1
             playing_stat['MW'] = MatchWeek
             return playing_stat

         playing_statistics_1 = get_mw(playing_statistics_1)
         playing_statistics_2 = get_mw(playing_statistics_2)
         playing_statistics_3 = get_mw(playing_statistics_3)
         playing_statistics_4 = get_mw(playing_statistics_4)
         playing_statistics_5 = get_mw(playing_statistics_5)
         playing_statistics_6 = get_mw(playing_statistics_6)
         playing_statistics_7 = get_mw(playing_statistics_7)
         playing_statistics_8 = get_mw(playing_statistics_8)
         playing_statistics_9 = get_mw(playing_statistics_9)
         playing_statistics_10 = get_mw(playing_statistics_10)
         playing_statistics_11 = get_mw(playing_statistics_11)
         playing_statistics_12 = get_mw(playing_statistics_12)
         playing_statistics_13 = get_mw(playing_statistics_13)
         playing_statistics_14 = get_mw(playing_statistics_14)
         playing_statistics_15 = get_mw(playing_statistics_15)
         playing_statistics_16 = get_mw(playing_statistics_16)
```

FINAL DATAFRAME

```
In [71]: playing_stat = pd.concat([playing_statistics_1,
                                   playing_statistics_2,
                                   playing_statistics_3,
                                   playing_statistics_4,
                                   playing_statistics_5,
                                   playing_statistics_6,
                                   playing_statistics_7,
                                   playing_statistics_8,
                                   playing_statistics_9,
                                   playing_statistics_10,
                                   playing_statistics_11,
                                   playing_statistics_12,
                                   playing_statistics_13,
                                   playing_statistics_14,
                                   playing_statistics_15,
                                   playing_statistics_16], ignore_index=True)
```

```
# Gets the form points.
def get_form_points(string):
    sum = 0
    for letter in string:
        sum += get_points(letter)
    return sum

playing_stat['HTFormPtsStr'] = playing_stat['HM1'] + playing_stat['HM2'] +
playing_stat['ATFormPtsStr'] = playing_stat['AM1'] + playing_stat['AM2'] +

playing_stat['HTFormPts'] = playing_stat['HTFormPtsStr'].apply(get_form_po
playing_stat['ATFormPts'] = playing_stat['ATFormPtsStr'].apply(get_form_po

# Identify Win/Loss Streaks if any.
def get_3game_ws(string):
    if string[-3:] == 'WWW':
        return 1
    else:
        return 0

def get_5game_ws(string):
    if string == 'WWWWW':
        return 1
    else:
        return 0

def get_3game_ls(string):
    if string[-3:] == 'LLL':
        return 1
    else:
        return 0

def get_5game_ls(string):
    if string == 'LLLLL':
        return 1
    else:
        return 0

playing_stat['HTWinStreak3'] = playing_stat['HTFormPtsStr'].apply(get_3gam
playing_stat['HTWinStreak5'] = playing_stat['HTFormPtsStr'].apply(get_5gam
playing_stat['HTLossStreak3'] = playing_stat['HTFormPtsStr'].apply(get_3ga
playing_stat['HTLossStreak5'] = playing_stat['HTFormPtsStr'].apply(get_5ga

playing_stat['ATWinStreak3'] = playing_stat['ATFormPtsStr'].apply(get_3gam
playing_stat['ATWinStreak5'] = playing_stat['ATFormPtsStr'].apply(get_5gam
playing_stat['ATLossStreak3'] = playing_stat['ATFormPtsStr'].apply(get_3ga
playing_stat['ATLossStreak5'] = playing_stat['ATFormPtsStr'].apply(get_5ga
```

```
playing_stat.keys()
```

```
Out[71]: Index([u'Date', u'HomeTeam', u'AwayTeam', u'FTHG', u'FTAG', u'FTR', u'HTGS
       u'ATGS', u'HTGC', u'ATGC', u'HTP', u'ATP', u'HM1', u'HM2', u'HM3',
       u'HM4', u'HM5', u'AM1', u'AM2', u'AM3', u'AM4', u'AM5', u'HomeTeamL
       u'AwayTeamLP', u'MW', u'HTFormPtsStr', u'ATFormPtsStr', u'HTFormPts
       u'ATFormPts', u'HTWinStreak3', u'HTWinStreak5', u'HTLossStreak3',
       u'HTLossStreak5', u'ATWinStreak3', u'ATWinStreak5', u'ATLossStreak3
       u'ATLossStreak5'],
      dtype='object')
```

```
In [72]: # Get Goal Difference
playing_stat['HTGD'] = playing_stat['HTGS'] - playing_stat['HTGC']
playing_stat['ATGD'] = playing_stat['ATGS'] - playing_stat['ATGC']

# Diff in points
playing_stat['DiffPts'] = playing_stat['HTP'] - playing_stat['ATP']
playing_stat['DiffFormPts'] = playing_stat['HTFormPts'] - playing_stat['AT

# Diff in last year positions
playing_stat['DiffLP'] = playing_stat['HomeTeamLP'] - playing_stat['AwayTe
```

```
In [73]: # Scale DiffPts , DiffFormPts, HTGD, ATGD by Matchweek.
cols = ['HTGD','ATGD','DiffPts','DiffFormPts','HTP','ATP']
playing_stat.MW = playing_stat.MW.astype(float)

for col in cols:
    playing_stat[col] = playing_stat[col] / playing_stat.MW
```

```
In [74]: def only_hw(string):
    if string == 'H':
        return 'H'
    else:
        return 'NH'

playing_stat['FTR'] = playing_stat.FTR.apply(only_hw)

# Testing set (2015-16 season)
playing_stat_test = playing_stat[5700:]
```

```
In [75]: playing_stat.to_csv(loc + "final_dataset.csv")
playing_stat_test.to_csv(loc+"test.csv")
```

```
In [ ]:
```