How to train your Pandas

Ilias Xenogiannis

Ioannis Karakasis

Pandas

University of Patras
IEEE Student Branch

ieeesbupatras

IEEE SB UPatras

ieeeupatras

ieeesb@upatras.gr

ieee.upatras.gr

How to train your Pandas

Motivation

Pandas

Pandas Birth-Certificate

2009

AQR Capital Management

Father

Wes Mckinney

BSc in Mathematics (MIT,2006)

Data Science

Experimental Sciences

Economics

"Apache Arrow"

University of Patras
IEEE Student Branch

Pandas Little Secrets

- Improved tool for visualizing data
- Efficient Coding
- Powerful package

  [Great Variety of commands,

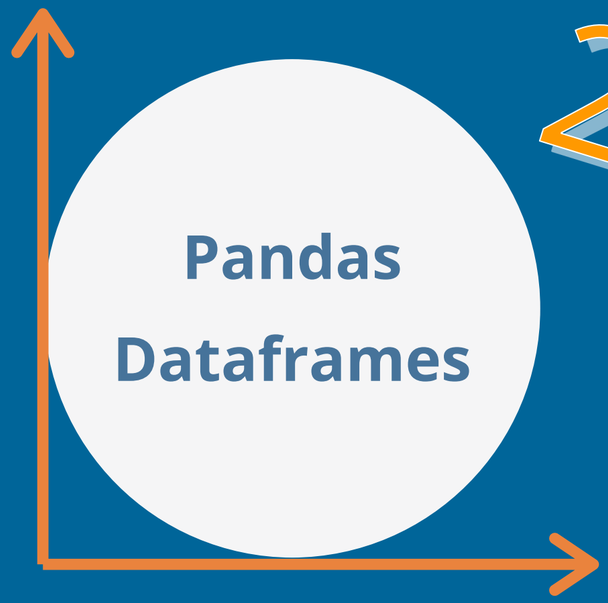
  Analysing, Filtering & Managing

  large datasets]

- Steep learning curve
- 3-D Matrices Compatibility

University of Patras
IEEE Student Branch

Pandas Data Structures



(list, dict)

(np.array,list, dict)

University of Patras
IEEE Student Branch

How to train your Pandas

Importing Data

Pandas

University of Patras
IEEE Student Branch

ieeesbupatras
IEEE SB UPatras
ieee.upatras.gr
ieeeupatras
ieeesb@upatras.gr

Importing Data

.csv files

```
import pandas as pd
data1 = pd.read_csv('filename.csv')
data2 = pd.read_csv(url)
```

.txt files

```
file = open('filename.txt', mode='r')
text=file.read()
text=file.close()
```

Excel files

```
import pandas as pd
data = pd.ExcelFile('filename.xlsx')
```

University of Patras
IEEE Student Branch

Importing Data

.json files

```
import json
with open('filename.json', 'r') as json_file:
  json_data=json.load(json_file)
```

→ [type of json_data: dictionary]

Other files

- MATLAB files
- Stata files
- SAS files
- Pickled files

University of Patras
IEEE Student Branch

Dataset

```
artist = ['Queen', 'ACDC', 'Rolling Stones', 'ACDC', 'The Who', 'Kansas']
song = ['Bohemian Rhapsody', 'Thunderstruck', 'Gimme Shelter', 'Demon Fire',
        'Baba O Riley', 'Carry on Wayward Son']

artist_series = pd.Series(artist)
song_series = pd.Series(song)

frame = {'Artist': artist_series, 'Song': song_series}

df = pd.DataFrame(frame)
year = [1975, 1990, 1969, 2020, 1971, 1976]

df['Year'] = pd.Series(year)
df
```

	Artist	Song	Year
0	Queen	Bohemian Rhapsody	1975
1	ACDC	Thunderstruck	1990
2	Rolling Stones	Gimme Shelter	1969
3	ACDC	Demon Fire	2020
4	The Who	Baba O Riley	1971
5	Kansas	Carry on Wayward Son	1976

University of Patras
IEEE Student Branch

Getting to know your Dataset

df.columns

→ list of column names

df.info()

→ Summary of datsaset

df.values()

→ Array of dataframe's values

```
1  df.info()
```

University of Patras
IEEE Student Branch

Getting to know your Dataset

df.shape

→ (rows, columns)=(6, 3)

df.describe()

→ Dataframe's Statistics

df.index

→ Range of indexing (start, step, stop)

```
1  df.describe()
```

University of Patras
IEEE Student Branch

How to train your Pandas

Slicing

&

Indexing

Pandas

University of Patras
IEEE Student Branch

ieeesbupatras

IEEE SB UPatras

ieeeupatras

ieeesb@upatras.gr

ieee.upatras.gr

Slicing & Indexing

```
1 df_indx = df.set_index("Year")
2 print(df_indx)
```

```
                Artist                    Song
Year
1975            Queen      Bohemian Rhapsody
1990            ACDC           Thunderstruck
1969  Rolling Stones           Gimme Shelter
2020            ACDC              Demon Fire
1971         The Who            Baba O Riley
1976          Kansas  Carry on Wayward Son
```

```
1 df_indx.reset_index()
```

Starting Dataset

```
1 df[df['Artist'].isin(['Metallica'])]
```

```
  Artist  Song  Year
```

?

University of Patras
IEEE Student Branch

Slicing & Indexing

df_indx.loc[categorical]

⟶ displays only the row of the item in [..]

df_indx.iloc[index range]

⟶ displays only the rows in the given range

```
1  df_indx2 = df.set_index("Artist")
2  df_indx2.loc[['ACDC', "The Who", "Metallica"]]
```

```
1 df_indx2 = df.set_index("Artist")
2 df_indx2.loc[['ACDC', "The Who"]]
```

Artist	Song	Year	Album	Release Date
ACDC	Thunderstruck	1990	The Razor's Edge	1990-09-21
ACDC	Demon Fire	2020	Power Up	2020-11-13
The Who	Baba O Riley	1971	Who's Next	1971-10-23

```
1 df_indx2.iloc[1:3]
```

Artist	Song	Year
ACDC	Thunderstruck	1990
Rolling Stones	Gimme Shelter	1969

Changed Dataset

```
df['Album'] = ['A Night At the Opera', "The Razor's Edge", 'Let It Bleed',
               'Power Up', "Who's Next", 'Leftoverture']
df['Release Date']= ['1975-10-31', '1990-09-21', '1969-12-05', '2020-11-13',
                     '1971-10-23', '1976-11-19']
df_indx3 = df.set_index(['Artist', 'Year'])
df_indx3
```

Artist	Year	Song	Album	Release Date
Queen	**1975**	Bohemian Rhapsody	A Night At the Opera	1975-10-31
ACDC	**1990**	Thunderstruck	The Razor's Edge	1990-09-21
Rolling Stones	**1969**	Gimme Shelter	Let It Bleed	1969-12-05
ACDC	**2020**	Demon Fire	Power Up	2020-11-13
The Who	**1971**	Baba O Riley	Who's Next	1971-10-23
Kansas	**1976**	Carry on Wayward Son	Leftoverture	1976-11-19

Multi-indexing ?

University of Patras
IEEE Student Branch

Slicing & Indexing

df.sort_index(level, ascending)

→ level= [list]

→ ascending= [list of booleans]

(alphabetically/numerically)

```
1  df_indx3.sort_index(level=['Artist', 'Year'], ascending=[True, True])
```

University of Patras
IEEE Student Branch

Multi-level Indexing

```
[15]  1 df_sorted=df_indx3.sort_index(level=['Artist', 'Year'], ascending=[True, True])
      2 df_sorted.loc[('ACDC', 2020):('The Who', 1971), 'Album':'Release Date']
```

		Album	Release Date
Artist	**Year**		
ACDC	**2020**	Power Up	2020-11-13
Kansas	**1976**	Leftoverture	1976-11-19
Queen	**1975**	A Night At the Opera	1975-10-31
Rolling Stones	**1969**	Let It Bleed	1969-12-05
The Who	**1971**	Who's Next	1971-10-23

University of Patras
IEEE Student Branch

Locating by date

```
1 df_indx4= df.set_index('Release Date').sort_index()
2 df_indx4
```

Release Date	Artist	Song	Year	Album
1969-12-05	Rolling Stones	Gimme Shelter	1969	Let It Bleed
1971-10-23	The Who	Baba O Riley	1971	Who's Next
1975-10-31	Queen	Bohemian Rhapsody	1975	A Night At the Opera
1976-11-19	Kansas	Carry on Wayward Son	1976	Leftoverture
1990-09-21	ACDC	Thunderstruck	1990	The Razor's Edge
2020-11-13	ACDC	Demon Fire	2020	Power Up

```
1 df_indx4.loc['1975-10-31':'1990-09-21']
```

partial locating (only by year)

```
1 df_indx4.loc['1975':'1991']
```

University of Patras
IEEE Student Branch

How to train your Pandas

Filtering Data

Pandas

University of Patras
IEEE Student Branch

ieeesbupatras
IEEE SB UPatras
ieee.upatras.gr
ieeeupatras
ieeesb@upatras.gr

Filtering Data

is to get to keep only the part of your Dataframe
that you need to find specific answers.

We can achieve that by different techniques using:

- lists of objects inside the dataframe
- boolean logic (True/False)
- strings

University of Patras
IEEE Student Branch

Filtering Data

We need to create a condition that will separate the data.

For this we use symbols:

>=, <=, == , !=

Finally, the data we want are
marked as True.

```
1   after_80 = df['Year'] >= 1980
2   print(after_80)
```

```
1   df_after_80 = df[after_80]
2   print(df_after_80)
```

Filtering Data

```
1 df[(df['Year'] <= 1980) & (df['Artist'] != 'Rolling Stones')]
```

	Artist	Song	Year	Album	Release Date
0	Queen	Bohemian Rhapsody	1975	A Night At the Opera	1975-10-31
4	The Who	Baba O Riley	1971	Who's Next	1971-10-23
5	Kansas	Carry on Wayward Son	1976	Leftoverture	1976-11-19

In case we want to use multi-conditional states, we have to use Boolean logic
(&, |)

```
1 df[(df['Year'] <= 1980) | (df['Artist'] != 'Rolling Stones')]
```

	Artist	Song	Year	Album	Release Date
0	Queen	Bohemian Rhapsody	1975	A Night At the Opera	1975-10-31
1	ACDC	Thunderstruck	1990	The Razor's Edge	1990-09-21
2	Rolling Stones	Gimme Shelter	1969	Let It Bleed	1969-12-05
3	ACDC	Demon Fire	2020	Power Up	2020-11-13
4	The Who	Baba O Riley	1971	Who's Next	1971-10-23
5	Kansas	Carry on Wayward Son	1976	Leftoverture	1976-11-19

University of Patras
IEEE Student Branch

Filtering Data

.mask(cond, other, inplace, axis, ...)

cond=condition

other= scalar, df/series,..

inplace=Boolean

(default=False)

axis={0,1}

Replaces values where condition is True

str

```
1 df.mask(df['Year'] > 1975)
```

	Artist	Song	Year	Album	Release Date
0	Queen	Bohemian Rhapsody	1975.0	A Night At the Opera	1975-10-31
1	NaN	NaN	NaN	NaN	NaN
2	Rolling Stones	Gimme Shelter	1969.0	Let It Bleed	1969-12-05
3	NaN	NaN	NaN	NaN	NaN
4	The Who	Baba O Riley	1971.0	Who's Next	1971-10-23
5	NaN	NaN	NaN	NaN	NaN

	A	B	C	D
0	2	7	5	0
1	6	81	17	6
2	14	13	12	21
3	5	9	4	34
4	32	10	0	1

```
1 df2.mask(df2['B']> 10, other=0.2)
```

	A	B	C	D
0	2.0	7.0	5.0	0.0
1	0.2	0.2	0.2	0.2
2	0.2	0.2	0.2	0.2
3	5.0	9.0	4.0	34.0
4	32.0	10.0	0.0	1.0

numerical

University of Patras
IEEE Student Branch

Filtering Data

.filter(items, like,regex, axis)

→ items = list, like = str

regex = str,

axis={0,1}

→ regex = 'str$'

'^str'

```
1 df.filter(items= ['Artist', 'Album'])
```

	Artist	Album
0	Queen	A Night At the Opera
1	NaN	NaN
2	Rolling Stones	Let It Bleed
3	NaN	NaN
4	The Who	Who's Next
5	NaN	NaN

```
1 df.filter(regex='e$', axis=1)
```

	Release Date
0	1975-10-31
1	NaN
2	1969-12-05
3	NaN
4	1971-10-23
5	NaN

```
1 df.set_index('Album').filter(like='Edge', axis=0)
```

Album	Artist	Song	Year	Release Date
The Razor's Edge	ACDC	Thunderstruck	1990	1990-09-21

University of Patras
IEEE Student Branch

Filtering Data

.query(expr, inplace)

→ expr = str

inlace = Boolean

→ checks a boolean expression between the columns of the DataFrame

```
[56] 1 df.query('Artist == "ACDC" and Song == "Thunderstruck"', inplace=False)
```

	Artist	Song	Year	Album	Release Date
1	ACDC	Thunderstruck	1990	The Razor's Edge	1990-09-21

```
1 df.query('(Artist == "The Who") and (Year == 1971)')
```

	Artist	Song	Year	Album	Release Date
4	The Who	Baba O Riley	1971	Who's Next	1971-10-23

```
1 df.query('(Artist == "The Who") or (Year == 1976)')
```

	Artist	Song	Year	Album	Release Date
4	The Who	Baba O Riley	1971	Who's Next	1971-10-23
5	Kansas	Carry on Wayward Son	1976	Leftoverture	1976-11-19

```
1 Artist_array= ['Queen', 'ACDC']
2 df.query('Artist in @Artist_array')
```

	Artist	Song	Year	Album	Release Date
0	Queen	Bohemian Rhapsody	1975	A Night At the Opera	1975-10-31
1	ACDC	Thunderstruck	1990	The Razor's Edge	1990-09-21
3	ACDC	Demon Fire	2020	Power Up	2020-11-13

Filtering Data

.where(cond, other, inplace, axis, ..)

cond =bool Series/df, callable, array-like

other= scalar, Series/df, callable

inlace = Boolean

Replaces the values for a False condition.

```
1 df2.where(df2['B']> 10, other=0.2)
```

	A	B	C	D
0	0.2	0.2	0.2	0.2
1	6.0	81.0	17.0	6.0
2	14.0	13.0	12.0	21.0
3	0.2	0.2	0.2	0.2
4	0.2	0.2	0.2	0.2

```
1 n = (df2 - 10) >= 0
2 df2.where(n, 0)
```

	A	B	C	D
0	0	0	0	0
1	0	81	17	0
2	14	13	12	21
3	0	0	0	34
4	32	10	0	0

```
[29] 1 n = (df2 - 10) >= 0
     2 df2.where(n, 'negative')
```

	A	B	C	D
0	negative	negative	negative	negative
1	negative	81	17	negative
2	14	13	12	21
3	negative	negative	negative	34
4	32	10	negative	negative

How to train your Pandas

Missing Values

Pandas

University of Patras
IEEE Student Branch

ieeesbupatras
IEEE SB UPatras
ieee.upatras.gr
ieeeupatras
ieeesb@upatras.gr

Handling Missing Values

⚠️

In real life problems, missing values can affect our analysis.

Missing values = unusable

Detecting Missing Values

.isna(obj)

→ Detects missing values in an array-like object

.notna(obj)

→ Detects non-missing values in an array-like object

```
1 masked=df.mask(df['Year'] > 1975)
2 masked
```

	Artist	Song	Year
0	Queen	Bohemian Rhapsody	1975.0
1	NaN	NaN	NaN
2	Rolling Stones	Gimme Shelter	1969.0
3	NaN	NaN	NaN
4	The Who	Baba O Riley	1971.0
5	NaN	NaN	NaN

```
1 masked.isna()
```

	Artist	Song	Year
0	False	False	False
1	True	True	False
2	False	False	False
3	True	True	False
4	False	False	False
5	True	True	False

Detecting Missing Values

.fillna(value, method, inplace, axis,..)

→ method {‘bfill’, ‘ffill’, None}

.dropna(axis, how, thresh, subset, inplace)

→ axis ={0,1}, how={any, all},
thresh=int, subset= array-like

```
1 masked['Year'].fillna(1980,inplace=True)
2 masked.fillna('Unkown')
```

	Artist	Song	Year
0	Queen	Bohemian Rhapsody	1975.0
1	Unkown	Unkown	1980.0
2	Rolling Stones	Gimme Shelter	1969.0
3	Unkown	Unkown	1980.0
4	The Who	Baba O Riley	1971.0
5	Unkown	Unkown	1980.0

How to train your Pandas

Functions

Pandas

University of Patras
IEEE Student Branch

ieeesbupatras
IEEE SB UPatras
ieee.upatras.gr
ieeeupatras
ieeesb@upatras.gr

Dataset

→ Random numerical data

→ Useful approach for "stress testing" Pandas

More on that later

```
1  df = pd.DataFrame(np.random.randint(0,47,size=(100, 2)),
2                    columns=["A","B"])
3  df
```

University of Patras
IEEE Student Branch

Builtin Functions

Can be applied both on dataframes and series

Built-in Functions available for most functionalities

min, max,
sum, median, mean,
std, var, etc.

```
1  df['A'].mean()
```

```
[65]  1  df['A'].std()
```

```
[60]  1  df['A'].median()
```

```
[61]  1  df['A'].min()
```

```
[54]  1  df['A'].max()
```

```
[55]  1  df['A'].sum()
```

```
[ ]  1
```

```
[ ]  1
```

```
[ ]  1
```

```
[ ]  1
```

Apply

It is used when you want
→ to apply a function along
the row or column.

→ If you must loop, just apply

```
1  df['Cat'] = df['B'].apply(grade)
2  df
```

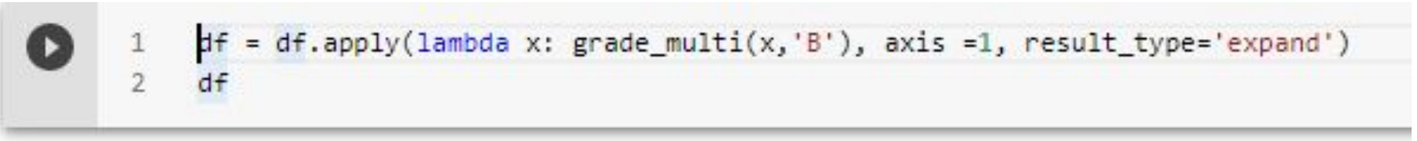
```
def grade(x):
    if x > 23:
        return 'High'
    else:
        return 'Low'
```

University of Patras
IEEE Student Branch

Apply with multi returns

Expand list-like results to columns of a Dataframe using expand

```
def grade_multi(x,column):

    if x[column] > 23:

        if x[column] % 2 == 0:

            return {**x, "Cat_1":'Even', "Cat_2": 'High' }

        else:

            return {**x, "Cat_1":'Odd', "Cat_2": 'High' }

    else:

        if x[column] % 2 == 0:

            return {**x, "Cat_1":'Even', "Cat_2": 'Low' }

        else:

            return {**x, "Cat_1":'Odd', "Cat_2": 'Low' }
```



Apply with multi inputs

⟶ Different approaches

```python
def square_difference_b(x):
    return (x['A']-x['B']) ** 2
```

```python
def square_difference_a(x,y):
    return (x-y) ** 2
```

```python
df['square_difference'] = df.apply(lambda x : square_difference_a(x['A'], x['B']), axis=1)
df['square_difference'] = df.apply(lambda x : square_difference_b(x), axis=1)
```

⟶ Same results!

University of Patras
IEEE Student Branch

Pandas Speed

Pandas Native

```
%timeit df['C'] = df['A'].mul(df['B'])
```

Python Native

```
%timeit df['C'] = df['A'] * df['B']
```

Apply

```
%timeit df['C'] = df.apply(lambda x: x['A'] * x['B'], axis=1)
```

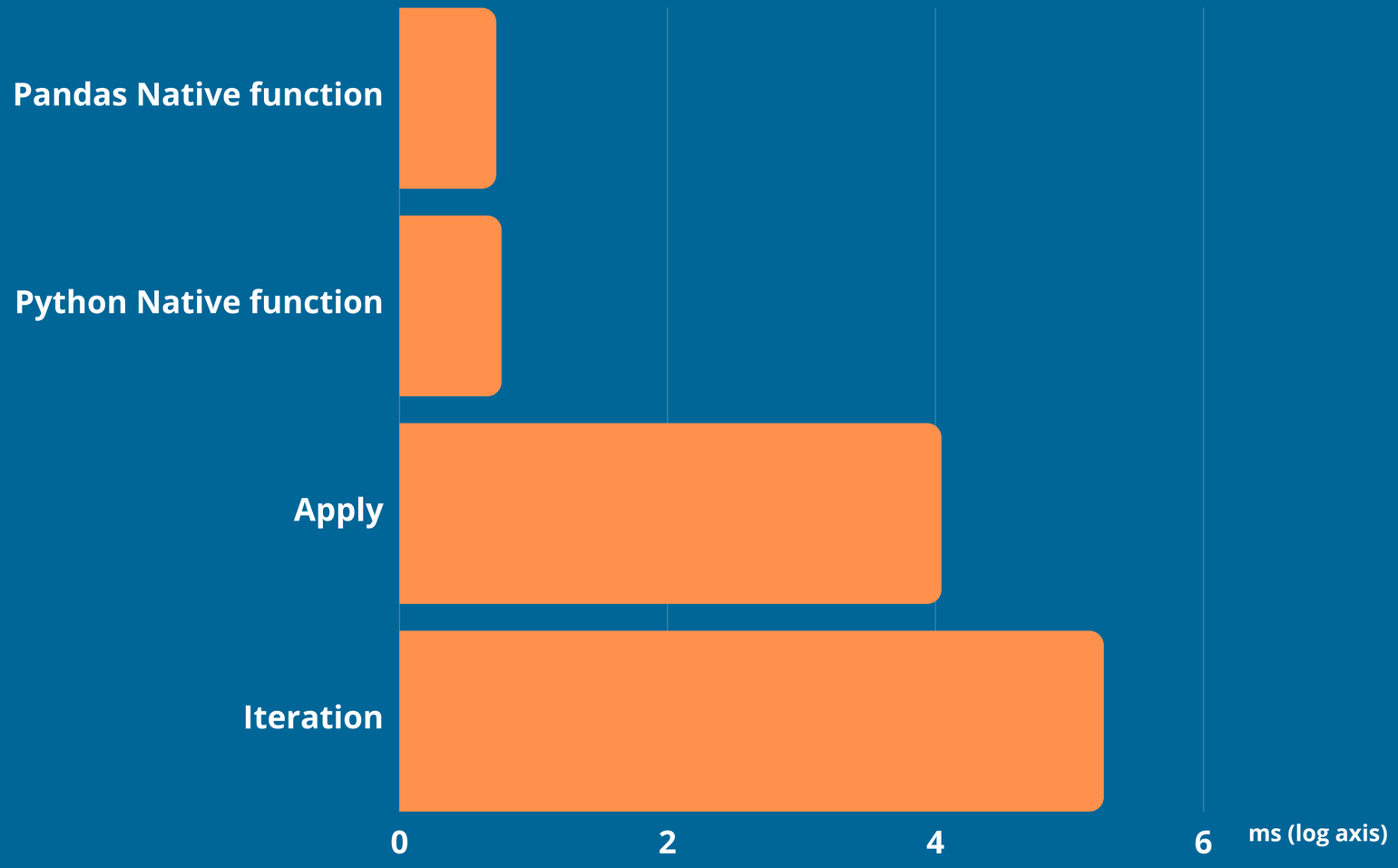
Iteration

```
%%timeit
for i, row in df.iterrows():

    df.loc[i, 'C'] = row['A'] * row['B']
```

This experiment used multiplication,
but can be replicated by other functions
and have the same results

University of Patras
IEEE Student Branch

Pandas Speed



WANNA GO FASTER?

GO NUMPY

How to train your Pandas

Aggregating

Pandas

University of Patras
IEEE Student Branch

ieeesbupatras
IEEE SB UPatras
ieee.upatras.gr
ieeeupatras
ieeesb@upatras.gr

Dataset

	Name	Year	Production Companies	Certification	Runtime	Rating	Cumulative Worldwide Gross
0	Jurassic World: Fallen Kingdom	2018	Universal Pictures	PG-13	128	6.2	1310464680
1	Star Wars: Episode IX - The Rise of Skywalker	2019	Walt Disney Pictures	PG-13	141	6.6	1074144248
2	Jason Bourne	2016	Universal Pictures	PG-13	123	6.6	415484914
3	Ocean's Thirteen	2007	Warner Bros.	PG-13	122	6.9	311312624
4	Mission: Impossible III	2006	Paramount Pictures	PG-13	126	6.9	398479497
5	Star Wars: Episode VIII - The Last Jedi	2017	Walt Disney Pictures	PG-13	152	7.0	1332697499
6	Insidious	2010	FilmDistrict	PG-13	103	6.8	99557032
7	Ant-Man and the Wasp	2018	Marvel Studios	PG-13	118	7.1	622674139
8	Wonder Woman	2017	Warner Bros.	PG-13	141	7.4	821983601
9	Iron Man	2008	Paramount Pictures	PG-13	126	7.9	585796247
10	Frozen	2013	Walt Disney Pictures	PG	102	7.4	1281019275
11	Brave	2012	Walt Disney Pictures	PG	93	7.1	538983207
12	Tangled	2010	Walt Disney Pictures	PG	100	7.7	592461959
13	Zootopia	2016	Walt Disney Pictures	PG	108	8.0	1023792209
14	The Favourite	2018	Fox Searchlight Pictures	R	119	7.5	95918706
15	The Lobster	2015	Film4	R	119	7.2	15456717

Aggregating

Split → Splitting into groups determined by "keys"

Apply → Applying a custom function or one of the dozens builtin

Combine → Combining the data to a single datatype

https://bit.ly/3arUm5F

Groupby

Groupby **doesn't perform** any operations on the table

Groupby returns ***DataframeGroupBy***

DataframeGroupBy need a function applied to it

```
1  df.groupby(by=['Certification']).mean()
```

```
1  df.groupby(by=['Production Companies','Certification']).count()
```

Apply

Column-specific aggregation, with control over the output column names

The function names can also be strings.

Check if a function is implemented before using apply

```
df.groupby('Production Companies').agg(
    Count=pd.NamedAgg(column='Year', aggfunc='count'),
    min_Year=pd.NamedAgg(column='Year', aggfunc=min),
    max_Year=pd.NamedAgg(column='Year', aggfunc=max),
    Avg_Revenue=pd.NamedAgg(column='Cumulative Worldwide Gross',aggfunc='mean')
)
```

Thank you for your Attention!

"*Ένα ξέρω, ότι ξέρω τα Pandas* "

Socrates the Panda -2020

University of Patras
IEEE Student Branch

ABOUT US

John Karakasis

linkedin.com/in/john-karakasis-157b03175/

Ilias Xenogiannis

linkedin.com/in/ilias-xenogiannis/

medium.com/@ixeno

Github link:
github.com/ilias1111/pandas-introduction-seminar

PANDAS DOCUMENTATION:
https://pandas.pydata.org/docs/

University of Patras
IEEE Student Branch

See you at the Workshop in 10'!

bit.ly/3awAwGm

University of Patras
IEEE Student Branch