Data Mining (Week 1)

BSc - Data Mining 1

Topic 02: Motivating Example

Part 01: Top X pandas commands



Prediction

Autumn Semester, 2021

Outline

- Reading data formats
- Computing descriptive statistics
- Processing data by filtering and grouping

Wrap up

Part I

Introduction

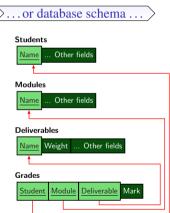
Before we start....

- The slides today are based on those prepared by my colleague Dr Kieran Murphy.
- Thanks Kieran!

Minimal Dataset

To better understand the various pandas operations we are going use a tiny* dataset based on (fictional) student results. (marks.csv)





...like to know ...

- Student performance weighted mark on each module, missing deliverables etc.
- Module performance number of attempts and average mark.
- Deliverable performance number of attempts and average mark, predictor of overall module grade, etc.

^{*}Dataset is small enough that you can verify operation results by hand.

Terminology

df.head(1000)

Student		Module	Deliverable	Weight	Mark
0	Α	Maths	Practical	40	60
1	Α	Maths	Exam	60	80
2	Α	Programming	Project 1	10	50
3	Α	Programming	Project 2	30	60
4	Α	Networking	Lab Work	100	80
5	В	Maths	Practical	40	60
6	В	Maths	Exam	60	80
7	В	Programming	Project 1	10	50
8	В	Programming	Project 2	30	60
9	В	Programming	Project 3	60	75
10	В	Networking	Project	100	80
11	C	Maths	Exam	60	80
12	С	Programming	Project 1	10	50
13	C	Programming	Project 2	30	60
14	С	Programming	Project 3	60	75
15	С	Networking	Lab Work	100	80

- A DataFrame is a table of data values.
 - df = pd.read_csv("marks.csv")
- A Series is a list of data values typically columns in a dataframe. We can access an individual column using
 - df.Deliverable (dot notation
 - df["Deliverable"] (dict notation
 - df.iloc[:,2] (numpy, index notation
- The index is a special column whose values can be used to access rows — rather using row number.
 - The default index is equal to the row number.

Terminology

df.head(1000)						
	Student	Module	Deliverable	Weight	Mark	
0	Α	Maths	Practical	40	60	
1	Α	Maths	Exam	60	80	
2	Α	Programming	Project 1	10	50	
3	Α	Programming	Project 2	30	60	
4	Α	Networking	Lab Work	100	80	
5	В	Maths	Practical	40	60	
6	В	Maths	Exam	60	80	
7	В	Programming	Project 1	10	50	
8	В	Programming	Project 2	30	60	
9	В	Programming	Project 3	60	75	
10	В	Networking	Project	100	80	
11	С	Maths	Exam	60	80	
12	С	Programming	Project 1	10	50	
13	С	Programming	Project 2	30	60	
14	С	Programming	Project 3	60	75	
15 C Netwo		Networking	Lab Work	100	80	

- A DataFrame is a table of data values.
 - df = pd.read_csv("marks.csv")
- A Series is a list of data values typically columns in a dataframe. We can access an individual column using

```
    df.Deliverable (dot notation)
    df["Deliverable"] (dict notation)
    df.iloc[:.2] (numpy, index notation)
```

- The index is a special column whose values can be used to access rows — rather using row number.
 - The default index is equal to the row number.

Terminology

df.head(1000)							
	Student	Module	Deliverable	Weight	Mark		
0	А	Maths	Practical	40	60		
1	Α	Maths	Exam	60	80		
2	Α	Programming	Project 1	10	50		
3	Α	Programming	Project 2	30	60		
4	Α	Networking	Lab Work	100	80		
5	В	Maths	Practical	40	60		
6	В	Maths	Exam	60	80		
7	В	Programming	Project 1	10	50		
8	В	Programming	Project 2	30	60		
9	В	Programming	Project 3	60	75		
10	В	Networking	Project	100	80		
11	С	Maths	Exam	60	80		
12	С	Programming	Project 1	10	50		
13	С	Programming	Project 2	30	60		
14	С	Programming	Project 3	60	75		
15	С	Networking	Lab Work	100	80		

- A DataFrame is a table of data values.
 - df = pd.read_csv("marks.csv")
- A Series is a list of data values typically columns in a dataframe. We can access an individual column using

```
    df.Deliverable (dot notation)
    df["Deliverable"] (dict notation)
    df.iloc[:,2] (numpy, index notation)
```

- The index is a special column whose values can be used to access rows rather using row number.
 - The default index is equal to the row number.

Part II

Input and Output

Setup

Minimal

We begin every data mining project with importing the three core data science packages:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
plt.style.use('seaborn-darkgrid')
```

numpy — fast array operations pandas — data manipulation matplotlib — visualisation

• We give modules nicknames (np, pd, ...) to simplify their later use, and we access properties/functions of a package using the dot notation (np.max, pd.DataFrame, ...).

Extra

```
import statsmodels.api as sm
pd.set_option('display.max_columns', 500)
pd.set_option('display.width', 1000)
```

seaborn — statistical visualisation statsmodels — statistical data exploration pandas options to show all columns for wider datasets

Setup

Minimal

We begin every data mining project with importing the three core data science packages:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
plt.style.use('seaborn-darkgrid')
```

numpy — fast array operations pandas — data manipulation matplotlib — visualisation

• We give modules nicknames (np, pd, ...) to simplify their later use, and we access properties/functions of a package using the dot notation (np.max, pd.DataFrame, ...).

Extra

```
import seaborn as sns
import statsmodels.api as sm

pd.set_option('display.max_columns', 500)
pd.set_option('display.width', 1000)
```

seaborn — statistical visualisation statsmodels — statistical data exploration pandas options to show all columns for wider datasets

Reading data from a CSV file

Pandas supports a huge variety of input/output formats so best approach is to focus on what is needed to process the given data and verify input. Our marks dataset is in CSV format so we start with

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
plt.style.use('seaborn-darkgrid')
```

and input using

```
df = pd.read_csv('marks.csv', sep=',')
print(df.shape)
df.head()
```

Reading data from a CSV file

Pandas supports a huge variety of input/output formats so best approach is to focus on what is needed to process the given data and verify input. Our marks dataset is in CSV format so we start with

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
plt.style.use('seaborn-darkgrid')
```

and input using

```
df = pd.read_csv('marks.csv', sep=',')
print(df.shape)
df.head()
```

Reading data from a CSV file

Pandas supports a huge variety of input/output formats so best approach is to focus on what is needed to process the given data and verify input. Our marks dataset is in CSV format so we start with

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
plt.style.use('seaborn-darkgrid')

and input using

df = pd.read_csv('marks.csv', sep=',')
print(df.shape)
df.head()

(16, 5)

[21] Student Module Deliverable Weight Mark

0 A Maths Practical 40 60

1 A Maths Exam 60 80

2 A Programming Project 1 10 50

3 A Programming Project 2 30 60

4 A Networking Lab Work 100 80
```

Always verify input by checking dataset dimensions and looking at some rows!!!

Datatypes

Pandas data types:

- object used for text or mixed numeric and non-numeric values.
- int64 integer values,
 - Does not support missing values, so an int column containing at least one missing value will automatically be converted to float.
- float64 floating point numbers.
- bool True/False values
- datetime64 date and time values
- category Finite (typically small) list of text values

Student	
Module	
Deliverable	
Weight	int64
Mark	int64
dtype: objec	

- Operations differ based on datatype, eg, '+' concatenate strings but adds numerical values.
- Datatype can change based on results, eg, int converts to float due to missing values.

Datatypes

Pandas data types:

- object used for text or mixed numeric and non-numeric values.
- int64 integer values,
 - Does not support missing values, so an int column containing at least one missing value will automatically
 be converted to float.
- float64 floating point numbers.
- bool True/False values
- datetime64 date and time values
- category Finite (typically small) list of text values

Student object
Module object
Deliverable object
Weight int64
Mark int64
dtype: object

df.dtypes

Regularly verifying datatypes is vital[†]

- Operations differ based on datatype, eg, '+' concatenate strings but adds numerical values.
- Datatype can change based on results, eg, int converts to float due to missing values.

[†]Google "Detecting Excel's gene auto-conversions."

Datatypes

Pandas data types:

- object used for text or mixed numeric and non-numeric values.
- int64 integer values,
 - Does not support missing values, so an int column containing at least one missing value will automatically be converted to float.
- float64 floating point numbers.
- bool True/False values
- datetime64 date and time values
- category Finite (typically small) list of text values

Student object object Module Deliverable object Weight int64 Mark int64 dtype: object

df.dtvpes

Regularly verifying datatypes is vital † :

- Operations differ based on datatype, eg, '+' concatenate strings but adds numerical values.
- Datatype can change based on results, eg, int converts to float due to missing values.

[†]Google "Detecting Excel's gene auto-conversions."

Datatypes — Converting

We will deal with modifying and creating new columns later, but while we are on datatypes, we will look at changing datatype ...

Using the Series function astype

```
df["Weight"] = df["Weight"].astype('float')
df["Weight"].dtype
dtype('float64')
```

- New datatype is required argument 'int', 'float', 'str', 'object', 'category', etc.
- Simple, but fragile if data conversion is possible.

or using pandas function to_numeric

```
df["Weight"] = pd.to_numeric(df["Weight"])
df["Weight"].dtype
dtype('float64')
```

- More powerful, can specify what to do in cases where the conversion fails etc
- Have functions to_numeric, to_datetime, and to_timedelta

Datatypes — Converting

We will deal with modifying and creating new columns later, but while we are on datatypes, we will look at changing datatype ...

Using the Series function astype

```
df["Weight"] = df["Weight"].astype('float')
df["Weight"].dtype
dtype('float64')
```

- New datatype is required argument 'int', 'float', 'str', 'object', 'category', etc.
- Simple, but fragile if data conversion is possible.

or using pandas function to_numeric

```
df["Weight"] = pd.to_numeric(df["Weight"])
df["Weight"].dtype
dtype('float64')
```

- More powerful, can specify what to do in cases where the conversion fails etc
- Have functions to_numeric, to_datetime, and to_timedelta

Datatypes — Converting

We will deal with modifying and creating new columns later, but while we are on datatypes, we will look at changing datatype ...

Using the Series function astype

```
df["Weight"] = df["Weight"].astype('float')
df["Weight"].dtype
dtype('float64')
```

- New datatype is required argument 'int', 'float', 'str', 'object', 'category', etc.
- Simple, but fragile if data conversion is possible.

or using pandas function to_numeric

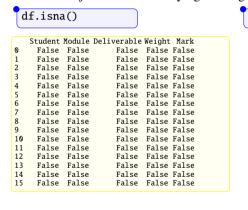
```
df["Weight"] = pd.to_numeric(df["Weight"])
df["Weight"].dtype
dtype('float64')
```

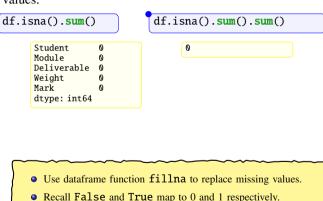
- More powerful, can specify what to do in cases where the conversion fails etc
- Have functions to_numeric, to_datetime, and to_timedelta.

Missing Values

Identifying and dealing with missing values is critical step in data preparation. What should you do? delete rows containing missing values? or impute then?

Here we will just look at identifying missing values.





• Use df.isna().sum(axis=1) to sum along rows.

Output

Saving dataframe to CSV is straightforward (I rarely include the (default) index when saving datasets).

df.to_csv('marks_2.csv', index=False)

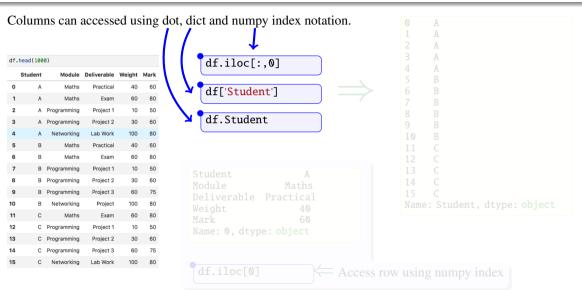
- CSV has become the default file format in Data Mining application especially for 'informal' datasets.
 - ✓ human readable, easy to generate / parse (if correct).
 - X Can be highly redundant, slow to input/output.
 - X No meta information.
- Other formats are better for speed and resulting file size and for saving meta data not supported by CSV (such as columns datatypes, category information, etc).

towards data science: The Best Format to Save Pandas Data

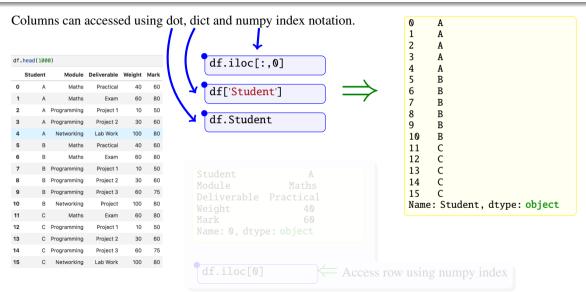
Part III

Filtering

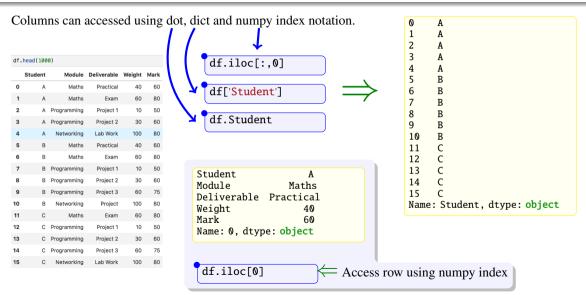
Selecting individual rows/columns results in a series



Selecting individual rows/columns results in a series



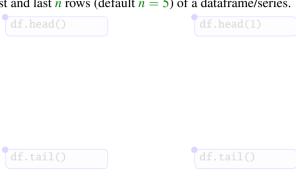
Selecting individual rows/columns results in a series



Head and Tail

Commands head and tail return the first and last n rows (default n = 5) of a dataframe/series.

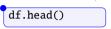
_	Student	Module	Deliverable	Weight	Mark
0	A	Maths	Practical	40	60
1	A	Maths	Exam	60	80
2	A	Programming	Project 1	10	50
3	A	Programming	Project 2	30	60
4	A	Networking	Lab Work	100	80
5	В	Maths	Practical	40	60
6	В	Maths	Exam	60	80
7	В	Programming	Project 1	10	50
8	В	Programming	Project 2	30	60
9	В	Programming	Project 3	60	75
10	В	Networking	Project	100	80
11	C	Maths	Exam	60	80
12	С	Programming	Project 1	10	50
13	C	Programming	Project 2	30	60
14	С	Programming	Project 3	60	75
15	C	Networking	Lab Work	100	80



Head and Tail

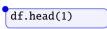
Commands head and tail return the first and last n rows (default n = 5) of a dataframe/series.

	Student	Module	Deliverable	Weight	Mark
0	A	Maths	Practical	40	60
1	A	Maths	Exam	60	80
2	A	Programming	Project 1	10	50
3	A	Programming	Project 2	30	60
4	A	Networking	Lab Work	100	80
5	В	Maths	Practical	40	60
6	В	Maths	Exam	60	80
7	В	Programming	Project 1	10	50
8	В	Programming	Project 2	30	60
9	В	Programming	Project 3	60	75
10	В	Networking	Project	100	80
11	С	Maths	Exam	60	80
12	С	Programming	Project 1	10	50
13	C	Programming	Project 2	30	60
14	C	Programming	Project 3	60	75
15	C	Networking	Lab Work	100	80



	Student	Module	Deliverable	Weight	Mark
0	A	Maths	Practical	40	60
	A	Maths	Exam	60	80
2	A	Programming	Project 1	10	50
3	A	Programming	Project 2	30	60
4	A	Networking	Lab Work	100	80





	Student	Module	Deliverable	Weight	Mark
() A	Maths	Practical	40	60



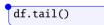
Head and Tail

Commands head and tail return the first and last n rows (default n = 5) of a dataframe/series.

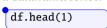
	Student	Module	Deliverable	Weight	Mark
0	A	Maths	Practical	40	60
1	A	Maths	Exam	60	80
2	A	Programming	Project 1	10	50
3	A	Programming	Project 2	30	60
4	A	Networking	Lab Work	100	80
5	В	Maths	Practical	40	60
6	В	Maths	Exam	60	80
7	В	Programming	Project 1	10	50
8	В	Programming	Project 2	30	60
9	В	Programming	Project 3	60	75
10	В	Networking	Project	100	80
11	C	Maths	Exam	60	80
12	С	Programming	Project 1	10	50
13	C	Programming	Project 2	30	60
14	С	Programming	Project 3	60	75
15	C	Networking	Lab Work	100	80



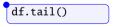
Studen	t Module	Deliverable	Weight	Marl
0 A	Maths	Practical	40	60
1 A	Maths	Exam	60	80
2 A	Programming	Project 1	10	50
3 A	Programming	Project 2	30	60
4 A	Networking	Lab Work	100	80



	Student	Module	Deliverable	Weight	Mark
11	С	Maths	Exam	60	80
12	C	Programming	Project 1	10	50
13	С	Programming	Project 2	30	60
14	С	Programming	Project 3	60	75
15	С	Networking	Lab Work	100	80



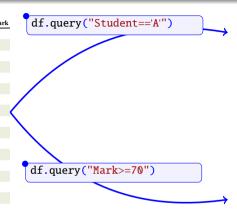
	Student	Module	Deliverable	Weight	Mark
0	Α	Maths	Practical	40	60



	Student	Module	Deliverable	Weight	Mark
15	С	Networking	Lab Work	100	80

Query — on a single-column criteria

	Student	Module	Deliverable	Weight	Mar
0	A	Maths	Practical	40	60
1	A	Maths	Exam	60	80
2	A	Programming	Project 1	10	50
3	A	Programming	Project 2	30	60
4	A	Networking	Lab Work	100	80
5	В	Maths	Practical	40	60
6	В	Maths	Exam	60	80
7	В	Programming	Project 1	10	50
8	В	Programming	Project 2	30	60
9	В	Programming	Project 3	60	75
10	В	Networking	Project	100	80
11	С	Maths	Exam	60	80
12	С	Programming	Project 1	10	50
13	C	Programming	Project 2	30	60
14	C	Programming	Project 3	60	75
15	C	Networking	Lab Work	100	80



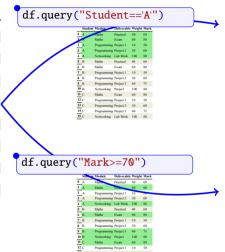
Query — on a single-column criteria

	Student	Module	Deliverable	Weight	Mai
0	A	Maths	Practical	40	60
1	A	Maths	Exam	60	80
2	A	Programming	Project 1	10	50
3	A	Programming	Project 2	30	60
4	A	Networking	Lab Work	100	80
5	В	Maths	Practical	40	60
6	В	Maths	Exam	60	80
7	В	Programming	Project 1	10	50
8	В	Programming	Project 2	30	60
9	В	Programming	Project 3	60	75
10	В	Networking	Project	100	80
11	С	Maths	Exam	60	80
12	С	Programming	Project 1	10	50
13	С	Programming	Project 2	30	60
14	С	Programming	Project 3	60	75
15	C	Networking	Lab Work	100	80



Query — on a single-column criteria

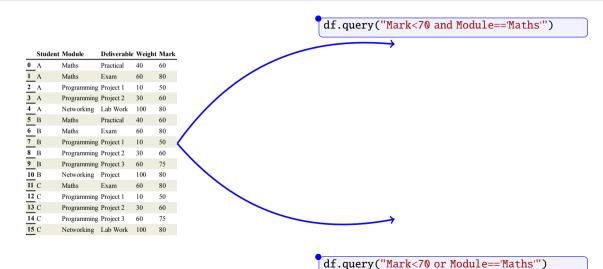
	Student	Module	Deliverable	Weight	Mar
0	A	Maths	Practical	40	60
1	A	Maths	Exam	60	80
2	A	Programming	Project 1	10	50
3	A	Programming	Project 2	30	60
4	A	Networking	Lab Work	100	80
5	В	Maths	Practical	40	60
6	В	Maths	Exam	60	80
7	В	Programming	Project 1	10	50
8	В	Programming	Project 2	30	60
9	В	Programming	Project 3	60	75
10	В	Networking	Project	100	80
11	C	Maths	Exam	60	80
12	С	Programming	Project 1	10	50
13	C	Programming	Project 2	30	60
14	C	Programming	Project 3	60	75
15	C	Networking	Lab Work	100	80



Stude	ent Module	Deliverable	Weight	Mark
0 A	Maths	Practical	40	60
1 A	Maths	Exam	60	80
2 A	Programming	Project 1	10	50
3 A	Programming	Project 2	30	60
4 A	Networking	Lab Work	100	80

	Student	Module	Deliverable	Weight	Mark
1	Α	Maths	Exam	60	80
4	Α	Networking	Lab Work	100	80
6	В	Maths	Exam	60	80
9	В	Programming	Project 3	60	75
10	В	Networking	Project	100	80
11	C	Maths	Exam	60	80
14	С	Programming	Project 3	60	75
15	С	Networking	Lab Work	100	80

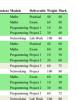
Query — on multiple columns (using python ogical operators)



Query — on multiple columns (using python logical operators)

	Student	Module	Deliverable	Weight	Mark
0	A	Maths	Practical	40	60
1	A	Maths	Exam	60	80
2	A	Programming	Project 1	10	50
3	A	Programming	Project 2	30	60
4	A	Networking	Lab Work	100	80
5	В	Maths	Practical	40	60
6	В	Maths	Exam	60	80
7	В	Programming	Project 1	10	50
8	В	Programming	Project 2	30	60
9	В	Programming	Project 3	60	75
10	В	Networking	Project	100	80
11	С	Maths	Exam	60	80
12	С	Programming	Project 1	10	50
13	С	Programming	Project 2	30	60
14	С	Programming	Project 3	60	75
15	C	Networking	Lab Work	100	80





df.query("Mark<70 and Module=='Maths'") df.query("Mark<70 or Module=='Maths'")

Query — on multiple columns (using python ogical operators)

	Student	Module	Deliverable	Weight	Mark
0	A	Maths	Practical	40	60
1	A	Maths	Exam	60	80
2	A	Programming	Project 1	10	50
3	A	Programming	Project 2	30	60
4	A	Networking	Lab Work	100	80
5	В	Maths	Practical	40	60
6	В	Maths	Exam	60	80
7	В	Programming	Project 1	10	50
8	В	Programming	Project 2	30	60
9	В	Programming	Project 3	60	75
10	В	Networking	Project	100	80
11	C	Maths	Exam	60	80
12	С	Programming	Project 1	10	50
13	C	Programming	Project 2	30	60
14	C	Programming	Project 3	60	75
15	C	Networking	Lab Work	100	80

	Student	Module	Deliverable	Weight	Mark
	A	Maths	Practical	40	60
1	A	Maths	Exam	60	80
2	A	Programming	Project I	10	50
3	A	Programming	Project 2	30	60
4	A	Networking	Lab Work	100	80
5	В	Maths	Practical	40	60
6	В	Maths	Exam	60	80
7	В	Programming	Project I	10	50
8	В	Programming	Project 2	30	60
9	В	Programming	Project 3	60	75
10	В	Networking	Project	100	80
Ш	C	Maths	Exam	60	80
12		Programming	Project I	10	50
13	C	Programming	Project 2	30	60
14	C	Programming	Project 3	60	75
15	C	Networking	Lab Work	100	80

	Student	Module	Deliverable	Weight	Mark
•	A	Maths	Practical	40	60
ı	A	Maths	Exam	60	80
2	A	Programming	Project I	10	50
3	A	Programming	Project 2	30	60
4	A	Networking	Lab Work	100	80
5	В	Maths	Practical	40	60
6	В	Maths	Exam	60	80
7	В	Programming	Project 1	10	50
8	В	Programming	Project 2	30	60
,	В	Programming	Project 3	60	75
10	В	Networking	Project	100	80
11		Maths	Exam	60	80
12	C	Programming	Project 1	10	50
13	C	Programming	Project 2	30	60
14	C	Programming	Project 3	60	75
15	C	Networking	Lab Work	100	80

df.query("Mark<70 and Module=='Maths'")</pre>

St	udent	Module	Deliverable	Weight	Mark
0 A		Maths	Practical	40	60
5 B		Maths	Practical	40	60

	Student	Module	Deliverable	Weight	Mark
0	A	Maths	Practical	40	60
1	A	Maths	Exam	60	80
2	A	Programming	Project 1	10	50
3	A	Programming	Project 2	30	60
5	В	Maths	Practical	40	60
6	В	Maths	Exam	60	80
7	В	Programming	Project 1	10	50
8	В	Programming	Project 2	30	60
11	С	Maths	Exam	60	80
12	С	Programming	Project 1	10	50
13	C	Programming	Project 2	30	60

df.query("Mark<70 or Module=='Maths'")

Query — on multiple columns (using pandas ogical operators)

	Student	Module	Deliverable	Weight	Mark
0	A	Maths	Practical	40	60
1	A	Maths	Exam	60	80
2	A	Programming	Project 1	10	50
3	A	Programming	Project 2	30	60
4	A	Networking	Lab Work	100	80
5	В	Maths	Practical	40	60
6	В	Maths	Exam	60	80
7	В	Programming	Project 1	10	50
8	В	Programming	Project 2	30	60
9	В	Programming	Project 3	60	75
10	В	Networking	Project	100	80
11	C	Maths	Exam	60	80
12	С	Programming	Project 1	10	50
13	C	Programming	Project 2	30	60
14	С	Programming	Project 3	60	75
15	C	Networking	Lab Work	100	80

	Student	Module	Deliverable	Weight	Mark
•	A	Maths	Practical	40	60
1	A	Maths	Exam	60	80
2	A	Programming	Project I	10	50
3	A	Programming	Project 2	30	60
4	A	Networking	Lab Work	100	80
5	В	Maths	Practical	40	60
6	В	Maths	Exam	60	80
7	В	Programming	Project I	10	50
8	В	Programming	Project 2	30	60
,	В	Programming	Project 3	60	75
10	В	Networking	Project	100	80
ш	C	Maths	Exam	60	80
12	C	Programming	Project I	10	50
13	C	Programming	Project 2	30	60
14	C	Programming	Project 3	60	75
15	C	Networking	Lab Work	100	80

Student	Module	Deliverable	Weight	Mark
0 A	Maths	Practical	40	60
1 A	Maths	Exam	60	80
2 A	Programming	Project I	10	50
3 A	Programming	Project 2	30	60
4 A	Networking	Lab Work	100	80
5 B	Maths	Practical	40	60
6 B	Maths	Exam	60	80
7 B	Programming	Project 1	10	50
8 B	Programming	Project 2	30	60
9 B	Programming	Project 3	60	75
10 B	Networking	Project	100	80
II C	Maths	Exam	60	80
12 C	Programming	Project 1	10	50
13 C	Programming	Project 2	30	60
14 C	Programming	Project 3	60	75
15 _. C	Networking	Lab Work	100	80

df.query("(Mark<70) & (Module=='Maths')")</pre>

St	tudent	Module	Deliverable	Weight	Mark
0 A		Maths	Practical	40	60
5 B		Maths	Practical	40	60

Studen	Module	Deliverable	Weight	Mark
0 A	Maths	Practical	40	60
1 A	Maths	Exam	60	80
2 A	Programming	Project 1	10	50
3 A	Programming	Project 2	30	60
5 B	Maths	Practical	40	60
6 B	Maths	Exam	60	80
7 B	Programming	Project 1	10	50
8 B	Programming	Project 2	30	60
11 C	Maths	Exam	60	80
12 C	Programming	Project 1	10	50
13 C	Programming	Project 2	30	60

df.query("(Mark<70) | (Module=='Maths')")

Filtering using 10c

	Student	Module	Deliverable	Weight	Mar
0	A	Maths	Practical	40	60
1	A	Maths	Exam	60	80
2	A	Programming	Project 1	10	50
3	A	Programming	Project 2	30	60
4	A	Networking	Lab Work	100	80
5	В	Maths	Practical	40	60
6	В	Maths	Exam	60	80
7	В	Programming	Project 1	10	50
8	В	Programming	Project 2	30	60
9	В	Programming	Project 3	60	75
10	В	Networking	Project	100	80
11	C	Maths	Exam	60	80
12	C	Programming	Project 1	10	50
13	С	Programming	Project 2	30	60
14	С	Programming	Project 3	60	75
15	C	Networking	Lab Work	100	80

Note the square (not round) brackets — think of loc as array indexing not a function call.

df.loc[ROW_SELECTION, COL_SELECTION]
where row and columns selection can be

- Single values: row number or column name
- An integer list for rows or list of column names
- A boolean list for logical indexing of rows
- A colon to indicate every row/column

df.loc[df.Module=="Maths", ["Student", "Mark"]]

Filtering using 10c

	Student	Module	Deliverable	Weight	Mar
0	A	Maths	Practical	40	60
1	A	Maths	Exam	60	80
2	A	Programming	Project 1	10	50
3	A	Programming	Project 2	30	60
4	A	Networking	Lab Work	100	80
5	В	Maths	Practical	40	60
6	В	Maths	Exam	60	80
7	В	Programming	Project 1	10	50
8	В	Programming	Project 2	30	60
9	В	Programming	Project 3	60	75
10	В	Networking	Project	100	80
11	C	Maths	Exam	60	80
12	С	Programming	Project 1	10	50
13	C	Programming	Project 2	30	60
14	C	Programming	Project 3	60	75
15	C	Networking	Lab Work	100	80

df.loc[ROW_SELECTION, COL_SELECTION]
where row and columns selection can be

- Single values: row number or column name
- An integer list for rows or list of column names
- A boolean list for logical indexing of rows
- A colon to indicate every row/column

	Student	Module	Deliverable	Weight	Mark
	Α	Maths	Practical	40	60
ī	Α	Maths	Exam	60	80
2	Α	Programming	Project I	10	50
3	Α	Programming	Project 2	30	60
4	Α	Networking	Lab Work	100	80
5	В	Maths	Practical	40	60
6	В	Maths	Exam	60	80
7	В	Programming	Project 1	10	50
8	В	Programming	Project 2	30	60
9	В	Programming	Project 3	60	75
10	В	Networking	Project	100	80
Ш	C	Maths	Exam	60	80
12	C	Programming	Project 1	10	50
13	C	Programming	Project 2	30	60
14	C	Programming	Project 3	60	75

df.loc[df.Module=="Maths", ["Student", "Mark"]]

19 of 29

Filtering using 10c

Note the square (not round) brackets — think of loc as array indexing not a function call.

	Student	Module	Deliverable	Weight	Mark
0	A	Maths	Practical	40	60
1	A	Maths	Exam	60	80
2	A	Programming	Project 1	10	50
3	A	Programming	Project 2	30	60
4	A	Networking	Lab Work	100	80
5	В	Maths	Practical	40	60
6	В	Maths	Exam	60	80
7	В	Programming	Project 1	10	50
8	В	Programming	Project 2	30	60
9	В	Programming	Project 3	60	75
10	В	Networking	Project	100	80
11	C	Maths	Exam	60	80
12	C	Programming	Project 1	10	50
13	C	Programming	Project 2	30	60
14	C	Programming	Project 3	60	75
15	C	Networking	Lab Work	100	80

df.loc[df.Module=="Maths", ["Student", "Mark"]]

df.loc[ROW_SELECTION, COL_SELECTION]
where row and columns selection can be

- Single values: row number or column name
- An integer list for rows or list of column names
- A boolean list for logical indexing of rows
- A colon to indicate every row/column



	Student	Mark
0	A	60
1	A	80
5	В	60
6	В	80
11	С	80

More complicated example

I prefer to define define row selection criteria, and the column list and order, separately to the loc statement.

	Student	Module	Deliverable	Weight	Mark
0	A	Maths	Practical	40	60
1	A	Maths	Exam	60	80
2	A	Programming	Project 1	10	50
3	A	Programming	Project 2	30	60
4	A	Networking	Lab Work	100	80
5	В	Maths	Practical	40	60
6	В	Maths	Exam	60	80
7	В	Programming	Project 1	10	50
8	В	Programming	Project 2	30	60
9	В	Programming	Project 3	60	75
10	В	Networking	Project	100	80

```
criteria = ((df.Mark<50) & (df.Module=='Maths')) | ((df.Mark<70) & (df.Module!='Maths'))
columns = ['Module', 'Student', 'Mark']
df.loc[criteria, columns]</pre>
```

More complicated example

I prefer to define define row selection criteria, and the column list and order, separately to the loc statement.

	Student	Module	Deliverable	Weight	Mark
0	A	Maths	Practical	40	60
1	A	Maths	Exam	60	80
2	A	Programming	Project 1	10	50
3	A	Programming	Project 2	30	60
4	A	Networking	Lab Work	100	80
5	В	Maths	Practical	40	60
6	В	Maths	Exam	60	80
7	В	Programming	Project 1	10	50
8	В	Programming	Project 2	30	60
9	В	Programming	Project 3	60	75
10	В	Networking	Project	100	80

	Student	Module	Deliverable	Weight	Mark
0	A	Maths	Practical	40	60
ı	A	Maths	Exam	60	80
2	A	Programming	Project I	10	50
	A	Programming	Project 2	30	60
4	A	Networking	Lab Work	100	80
5	В	Maths	Practical	40	60
	В	Maths	Exam	60	80
7	В	Programming	Project I	10	50
8	В	Programming	Project 2	30	60
9	В	Programming	Project 3	60	75
10	В	Networking	Project	100	80
11	C	Maths	Exam	60	80
12	C	Programming	Project I	10	50
13	C	Programming	Project 2	30	60
14	C	Programming	Project 3	60	75
15	C	Networking	Lab Work	100	80

```
criteria = ((df.Mark<50) & (df.Module=='Maths')) | ((df.Mark<70) & (df.Module!='Maths'))
columns = ['Module', 'Student', 'Mark']
df.loc[criteria, columns]</pre>
```

More complicated example

I prefer to define define row selection criteria, and the column list and order, separately to the loc statement.

	Student	Module	Deliverable	Weight	Mark
0	A	Maths	Practical	40	60
1	A	Maths	Exam	60	80
2	A	Programming	Project 1	10	50
3	A	Programming	Project 2	30	60
4	A	Networking	Lab Work	100	80
5	В	Maths	Practical	40	60
6	В	Maths	Exam	60	80
7	В	Programming	Project 1	10	50
8	В	Programming	Project 2	30	60
9	В	Programming	Project 3	60	75
10	В	Networking	Project	100	80

	Student	Module	Deliverable	Weight	Mark
0	A	Maths	Practical	40	60
ı	A	Maths	Exam	60	80
2	A	Programming	Project I	10	50
3	A	Programming	Project 2	30	60
	A	Networking	Lab Work	100	80
5	В	Maths	Practical	40	60
6	В	Maths	Exam	60	80
7	В	Programming	Project I	10	50
8	В	Programming	Project 2	30	60
9	В	Programming	Project 3	60	75
10	В	Networking	Project	100	80
11	C	Maths	Exam	60	80
12	C	Programming	Project I	10	50
13	C	Programming	Project 2	30	60
14	C	Programming	Project 3	60	75
	C	Networking	Lab Work	100	80

	Student	Module	Deliverable	Weight	Mark
0	Α	Maths	Practical	40	60
1	A	Maths	Exam	60	80
2	Α	Programming	Project 1	10	50
3	Α	Programming	Project 2	30	60
4	Α	Networking	Lab Work	100	80
5	В	Maths	Practical	40	60
6	В	Maths	Exam	60	80
7	В	Programming	Project 1	10	50
8	В	Programming	Project 2	30	60
9	В	Programming	Project 3	60	75
10	В	Networking	Project	100	80
11	C	Maths	Exam	60	80
12	С	Programming	Project 1	10	50
13	С	Programming	Project 2	30	60
14	С	Programming	Project 3	60	75

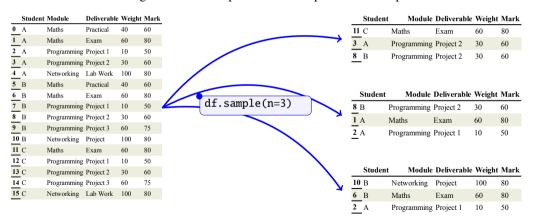
```
criteria = ((df.Mark<50) & (df.Module=='Maths')) | ((df.Mark<70) & (df.Module!='Maths'))
columns = ['Module', 'Student', 'Mark']</pre>
```

df.loc[criteria, columns]

Sampling

The sample function selects a random subset of the dataframe rows.

- Either specify the number of rows (as an integer) or fraction of the data (as a float).
- Can set the seed using random_state parameter for reproducible samples.



Part IV

Sorting

Sorting

A pandas dataframe has two sorting operations:

- sort_index() orders rows based on current index.
- sort_values(COLUMNS) orders rows based on single column or list of columns.

Two important modifications:

- By default, the sort order is in ascending. Set parameter ascending=False to reverse this.
- By default, a new dataframe is returned with desired sort order, set parameter inplace=True to update current dataframe instead (then no output is generated).

	Student	Module	Deliverable	Weight	Mark
0	Α	Maths	Practical	40	60
1	A	Maths	Exam	60	80
	A	Programming	Project 1	10	50
3	A	Programming	Project 2	df.	sor
_	A	Networking	Lab Work		
5	В	Maths	Practical	40	60
6	В	Maths	Exam	60	80
7	В	Programming	Project 1	10	50
8	В	Programming	Project 2	30	60
9	В	Programming	Project 3	60	75
	В	Networking	Project	100	80
11	C	Maths	Exam	60	80
12	С	Programming	Project 1	10	50
13	С	Programming	Project 2	30	60
	С	Programming	Project 3	60	75

Sorting

A pandas dataframe has two sorting operations:

- sort_index() orders rows based on current index.
- sort_values(COLUMNS) orders rows based on single column or list of columns.

Two important modifications:

- By default, the sort order is in ascending. Set parameter ascending=False to reverse this.
- By default, a new dataframe is returned with desired sort order, set parameter inplace=True to update current dataframe instead (then no output is generated).

	Student	Module	Deliverable	Weight	Mark		Student	Module	Deliverable	Weight	Mark
0	A	Maths	Practical	40	60	1	A	Maths	Exam	60	80
1	A	Maths	Exam	60	80	6	В	Maths	Exam	60	80
2	A	Programming	Project 1	10	50	11	С	Maths	Exam	60	80
3	A	Programming	Project 2	df.	sort_values(['Module','Deliverable'])	\mathcal{L}	A	Maths	Practical	40	60
4	A	Networking	Lab Work	<u> </u>	bolt_talues([nodale, beliverable])	厂	В	Maths	Practical	40	60
5	В	Maths	Practical	40	60	4	A	Networking	Lab Work	100	80
6	В	Maths	Exam	60	80	15	С	Networking	Lab Work	100	80
7	В	Programming	Project 1	10	50	10	В	Networking	Project	100	80
8	В	Programming	Project 2	30	60	2	A	Programming	Project 1	10	50
9	В	Programming	Project 3	60	75	7	В	Programming	Project 1	10	50
10	В	Networking	Project	100	80	12	С	Programming	Project 1	10	50
11	С	Maths	Exam	60	80	3	A	Programming	Project 2	30	60
12	C	Programming	Project 1	10	50	8	В	Programming	Project 2	30	60
13	С	Programming	Project 2	30	60	13	C	Programming	Project 2	30	60
14	С	Programming	Project 3	60	75	9	В	Programming	Project 3	60	75

23 of 29

Part V

Defining New Columns

Defining new columns — row-wise operation

We want to compute the weighted mark for each module for each student. Two steps:

- Create column, W_Mark, to store the weighted mark for each deliverable. This is a row by row calculation only need data in current row to compute the result.
 Create column, M_Mark, to store the module mark for each student. This is a group calculation.
- Create column, M_Mark, to store the module mark for each student. This is a group calculation need all rows for that student and module to compute the result.

	Student	Module	Deliverable	Weight	Mark
1	A	Maths	Exam	60	80
6	В	Maths	Exam	60	80
11	C	Maths	Exam	60	80
0	A	Maths	Practical	40	60
5	В	Maths	Practical	40	60
4	A	Networking	Lab Work	100 C	√]1£
15	C	Networking	Lab Work	100	00
10	В	Networking	Project	100	80
2	A	Programming	Project 1	10	50
7	В	Programming	Project 1	10	50
12	C	Programming	Project 1	10	50
3	A	Programming	Project 2	30	60
8	В	Programming	Project 2	30	60
13	C	Programming	Project 2	30	60
9	В	Programming	Project 3	60	75
14	C	Programming	Project 3	60	75

Defining new columns — row-wise operation

We want to compute the weighted mark for each module for each student. Two steps:

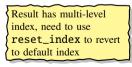
- Create column, W_Mark, to store the weighted mark for each deliverable. This is a row by row calculation only need data in current row to compute the result.
 Create column, M_Mark, to store the module mark for each student. This is a group calculation.
- Create column, M_Mark, to store the module mark for each student. This is a group calculation need all rows for that student and module to compute the result.

Studen	t Module	Deliverable	Weigh	t Mark				Student	Module	Deliverable	Weight	Mark	W_Mark
1 A	Maths	Exam	60	80			1	A	Maths	Exam	60	80	48
6 B	Maths	Exam	60	80			6	В	Maths	Exam	60	80	48
11 C	Maths	Exam	60	80			11	С	Maths	Exam	60	80	48
0 A	Maths	Practical	40	60			0	A	Maths	Practical	40	60	24
5 B	Maths	Practical	40	60			5	В	Maths	Practical	40	60	24
4 A	Networking	Lab Work	100	df['V	N_Mar	'k'] = df.Weight * df.Mark	// 1	00	etworking	Lab Work	100	80	80
15 C	Networking	Lab Work	100	-					Networking	Lab Work	100	80	80
10 B	Networking	Project	100	80			1 10	В	Networking	Project	100	80	80
2 A	Programming	Project 1	10	50			2	A	Programming	Project 1	10	50	5
7 B	Programming	Project 1	10	50			7	В	Programming	Project 1	10	50	5
12 C	Programming	Project 1	10	50	Г	Nacidate de la constitución de l	1	2 C	Programming	Project 1	10	50	5
3 A	Programming	Project 2	30	60	\ \	Need to use dict notation (not do	ot $\frac{1}{3}$	A	Programming	Project 2	30	60	18
8 B	Programming	Project 2	30	60	1	notation) when defining a new	8	В	Programming	Project 2	30	60	18
13 C	Programming	Project 2	30	60	. l	column.	1.	3 C	Programming	Project 2	30	60	18
9 B	Programming	Project 3	60	75			${9}$	В	Programming	Project 3	60	75	45
14 C	Programming	Project 3	60	75			1-	C	Programming	Project 3	60	75	45

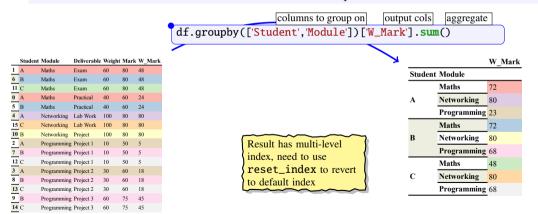
- Create column, W_Mark, to store the weighted mark for each deliverable. This is a row by row calculation — only need data in current row to compute the result.
- Create column, M_Mark, to store the module mark for each student. This is a group calculation need all rows for that student and module to compute the result.

columns to group on output cols aggregate df.groupby(['Student', 'Module'])['W_Mark'].sum()

	Student Module		Deliverable	Weight	Mark	W_Mark
1	A	Maths	Exam	60	80	48
6	В	Maths	Exam	60	80	48
11	С	Maths	Exam	60	80	48
0	A	Maths	Practical	40	60	24
5	В	Maths	Practical	40	60	24
4	A	Networking	Lab Work	100	80	80
15	С	Networking	Lab Work	100	80	80
10	В	Networking	Project	100	80	80
2	A	Programming	Project 1	10	50	5
7	В	Programming	Project 1	10	50	5
12	C	Programming	Project 1	10	50	5
3	A	Programming	Project 2	30	60	18
8	В	Programming	Project 2	30	60	18
13	С	Programming	Project 2	30	60	18
9	В	Programming	Project 3	60	75	45
14	С	Programming	Project 3	60	75	45



- Create column, W_Mark, to store the weighted mark for each deliverable. This is a row by row calculation — only need data in current row to compute the result.
- Create column, M_Mark, to store the module mark for each student. This is a group calculation need all rows for that student and module to compute the result.



```
columns to group on output cols aggregate df.groupby(['Student','Module'])[['W_Mark']].sum().reset_index()
```

	Student	Module	Deliverable	Weight	Mark	W_Mark
1	A	Maths	Exam	60	80	48
6	В	Maths	Exam	60	80	48
11	C	Maths	Exam	60	80	48
0	A	Maths	Practical	40	60	24
5	В	Maths	Practical	40	60	24
4	A	Networking	Lab Work	100	80	80
15	C	Networking	Lab Work	100	80	80
10	В	Networking	Project	100	80	80
2	A	Programming	Project 1	10	50	5
7	В	Programming	Project 1	10	50	5
12	C	Programming	Project 1	10	50	5
3	A	Programming	Project 2	30	60	18
8	В	Programming	Project 2	30	60	18
13	C	Programming	Project 2	30	60	18
9	В	Programming	Project 3	60	75	45
14	C	Programming	Project 3	60	75	45

Defining new columns — group aggregate result

Programming Project 3

Programming Project 3

14 C

75 45

60

60 75 45

output cols columns to group on aggregate df.groupby(['Student','Module'])[['W_Mark']].sum().reset_index() Student Module Deliverable Weight Mark W Mark Student Module W Mark 1 A Maths Exam 60 80 48 0 A Maths 72. 6 B
11 C
0 A
5 B
4 A
15 C
10 B
2 A
7 B
12 C
3 A
8 B
13 C Maths Exam 60 80 48 1 A Maths 80 48 Networking 80 Exam 60 This is the required result and Maths Practical 40 60 24 2 A Programming 23 we can save this to a new Maths Practical 40 60 24 3 B Maths 72 Networking Lab Work 100 80 80 dataframe. However, we often Networking Lab Work 100 80 80 want to put this into to out 4 B Networking 80 Networking Project 100 80 80 original dataframe as an extra 5 B Programming 68 Programming Project 1 50 5 10 column. Only problem we Programming Project 1 10 50 5 6 C Maths 48 have different rows so can't Programming Project 1 50 5 10 7 C just assign to a new column Networking 80 Programming Project 2 30 60 18 — need to use transform 8 C Programming Project 2 60 18 30 Programming 68 function. Programming Project 2 30 60 18

Defining new columns — group aggregate result

III

columns to group on output cols aggregate

df['M_Mark'] = df.groupby(['Student','Module'])[['W_Mark']].transform(sum)

	Student	Module	Deliverable	Weight	Mark	W_Mark
1	A	Maths	Exam	60	80	48
6	В	Maths	Exam	60	80	48
11	С	Maths	Exam	60	80	48
0	A	Maths	Practical	40	60	24
5	В	Maths	Practical	40	60	24
4	A	Networking	Lab Work	100	80	80
15	C	Networking	Lab Work	100	80	80
10	В	Networking	Project	100	80	80
2	A	Programming	Project 1	10	50	5
7	В	Programming	Project 1	10	50	5
12	С	Programming	Project 1	10	50	5
3	A	Programming	Project 2	30	60	18
8	В	Programming	Project 2	30	60	18
13	C	Programming	Project 2	30	60	18
9	В	Programming	Project 3	60	75	45
14	C	Programming	Project 3	60	75	45

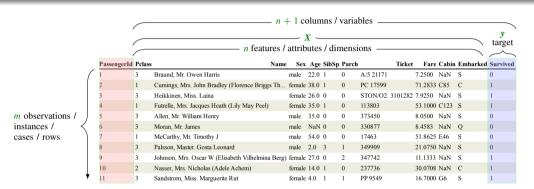
-			
		``	
		4	
	Student	Module	W_Mark
0	A	Maths	72
1	A A B	Networking	80
2	A	Programming	23
3	В	Maths	72
4	B B	Networking	80
5	В	Programming	68
6	C	Maths	48
7	C C	Networking	80
8	C	Programming	68

	Student	Module	Deliverable	Woight	Mork	W Mork	M Mork
1	A	Maths	Exam	60	80	48	72
6	В	Maths	Exam	60	80	48	72
-	C	Maths	Exam	60	80	48	48
0	A	Maths	Practical	40	60	24	72
5	В	Maths	Practical	40	60	24	72
4	A	Networking	Lab Work	100	80	80	80
15	C	Networking			80-	80	80
10	В	Networking	Project	100	80		80
2	A.	Programming	Project 1	10	50	5	23
7	В	Programming	Project 1	10	50	5	68
12	C	Programming	Project 1	10	50	5	68
3	A	Programming	Project 2	30	60	18	23
8	В	Programming	Project 2	30	60	18	68
13	C	Programming	Project 2	30	60	18	68
9	В	Programming	Project 3	60	75	45	68
14	C	Programming	Project 3	60	75	45	68

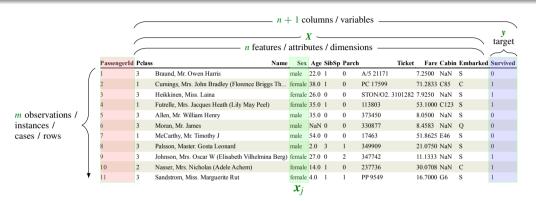
The transform broadcasts the result for each group over every row in that group.

PassengerId	Pclass	Nam	e Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Survived
1	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S	0
2	1	Cumings, Mrs. John Bradley (Florence Briggs Th.	. female	38.0	1	0	PC 17599	71.2833	C85	C	1
3	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S	1
4	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S	1
5	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S	0
6	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	NaN	Q	0
7	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46	S	0
8	3	Palsson, Master. Gosta Leonard	male	2.0	3	1	349909	21.0750	NaN	S	0
9	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg	g) female	27.0	0	2	347742	11.1333	NaN	S	1
10	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0	237736	30.0708	NaN	C	1
11	3	Sandstrom, Miss. Marguerite Rut	female	4.0	1	1	PP 9549	16.7000	G6	S	1

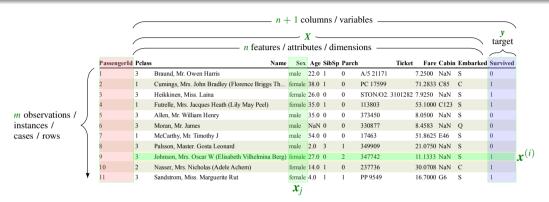
- A labeled dataset consists of m rows \times (n + 1) columns / variables.
- Use bold to represent vectors and matrices
- Use subscripts to indicate particular feature / attribute / column
- Use superscript in parenthesis to indicate particular observation / instance/ case / row
- So $x_i^{(i)}$ (or $x_{i,j}$) is the *i*-th observation in the *j*-th feature



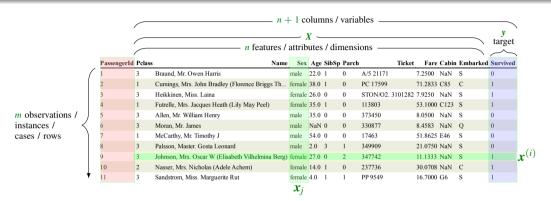
- A labeled dataset consists of m rows \times (n+1) columns / variables.
- Use bold to represent vectors and matrices.
- Use subscripts to indicate particular feature / attribute / column
- Use superscript in parenthesis to indicate particular observation / instance/ case / row
- So $x_i^{(i)}$ (or $x_{i,j}$) is the *i*-th observation in the *j*-th feature



- A labeled dataset consists of m rows \times (n+1) columns / variables.
- Use bold to represent vectors and matrices.
- Use subscripts to indicate particular feature / attribute / column
- Use superscript in parenthesis to indicate particular observation / instance/ case / row
- So $x_i^{(i)}$ (or $x_{i,j}$) is the *i*-th observation in the *j*-th feature



- A labeled dataset consists of m rows \times (n+1) columns / variables.
- Use bold to represent vectors and matrices.
- Use subscripts to indicate particular feature / attribute / column
- Use superscript in parenthesis to indicate particular observation / instance/ case / row
- So $x_i^{(i)}$ (or $x_{i,j}$) is the *i*-th observation in the *j*-th feature



- A labeled dataset consists of m rows \times (n+1) columns / variables.
- Use bold to represent vectors and matrices.
- Use subscripts to indicate particular feature / attribute / column
- Use superscript in parenthesis to indicate particular observation / instance/ case / row
- So $x_j^{(i)}$ (or $x_{i,j}$) is the *i*-th observation in the *j*-th feature