Understanding Data Part 3

Extracting Information from Data

Information

Information is the collection of facts and patterns extracted from data. Data provide opportunities for identifying trends, making connections, and addressing problems.

The size of a data set affects the amount of information that can be extracted from it. A popular subfield of computer science is data science where large datasets are processed and analyzed.

Large data sets are difficult to process using a single computer and may require parallel or distributed systems. **Scalability** of systems is an important consideration when working with data sets, as the computational capacity of a system affects how data sets can be processed and stored.

Scalability is the ability of a system to handle a bigger load by adding resources to the system. For example, Amazon's payment system is scalable because as its user base grows, it can add more servers to handle payments.

Computational Models

Sequential computing is a computational model in which operations are performed in order one at a time.

Parallel computing is a computational model where the program is broken into multiple smaller sequential computing operations, some of which are performed simultaneously.

Parallel computing uses a single computer with multiple processors.

Distributed computing is a computational model in which multiple devices are used to run a program.

Computational Models

A sequential solution takes as long as the sum of all of its steps. A parallel computing solution takes as long as its sequential tasks plus the longest of its parallel tasks.

The **"speedup"** of a parallel solution is measured in the time it took to complete the task sequentially divided by the time it took to complete the task when done in parallel.

Solutions that use parallel computing can scale more effectively than solutions that use sequential computing.

Distributed computing allows problems to be solved that could not be solved on a single computer because of either the processing time or storage needs involved. Distributed computing allows much larger problems to be solved quicker than they could be solved using a single computer.

Information

Programs can be used to process data to acquire information.

Programs such as spreadsheets(for example, Excel) help efficiently organize and find trends in information. Excel can process tabular data or 2D data with mixed datatypes.

A powerful alternative is Python's pandas library. We will explore this library and see how it helps us:

- gain insight and knowledge that are obtained from translating and transforming digitally represented information.
- see patterns when data are transformed
- filter, modify, combine and compare data
- visualize data through charts, graphs, tables and other visualizations

We will use Jupyter Notebook(Lab) for all code in this lecture slides.

Data

Digitally processed data may show correlation between variables. A correlation found in data does not necessarily indicate that a causal relationship exists.

Additional research is needed to understand the exact nature of the relationship. Often, a single source does not contain the data needed to draw a conclusion. It may be necessary to combine data from a variety of sources to formulate a conclusion.

Combining data sources, clustering data, and classifying data are parts of the process of using programs to gain insight and knowledge from data.

Problems of bias are often created by the type or source of data being collected. Bias is not eliminated by simply collecting more data.

Cleaning Data

Data sets pose challenges regardless of size, such as:

- the need to clean data
- incomplete data
- invalid data
- not uniform(spellings, abbreviations, capitalizations)
- the need to combine data sources

Depending on how data were collected, they may not be uniform. For example, if users enter data into an open field, the way they choose to abbreviate, spell, or capitalize something may vary from user to user.

A simple step in the **data cleaning** process is to make the data uniform without changing their meaning (e.g., replacing all equivalent abbreviations, spellings, and capitalizations with the same word).

pandas

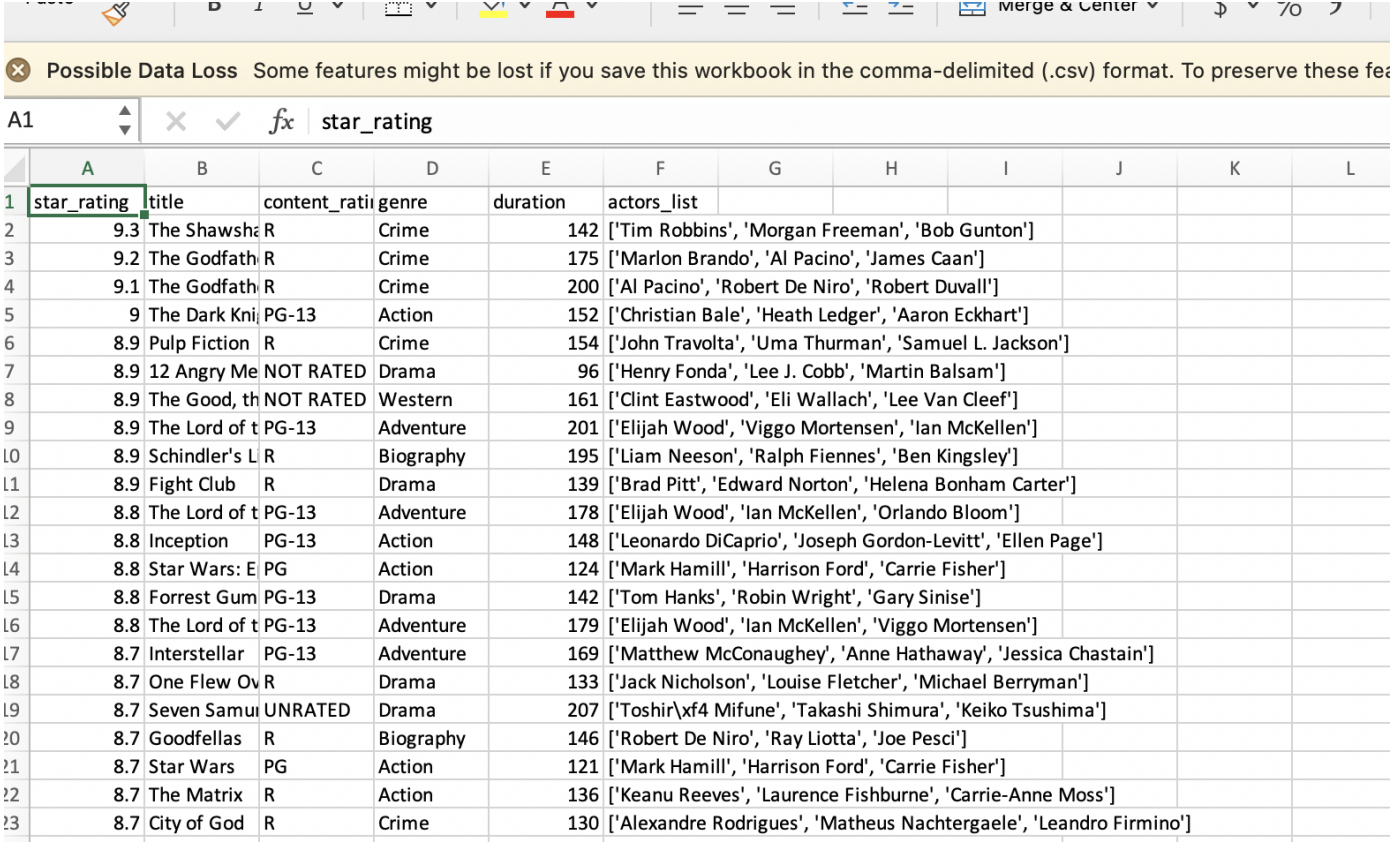
We'll explore pandas using Jupyter Lab by looking at a movies dataset: IMDB's Top 1000 movies. The "imdb_1000.csv" comma-separated values file is a text file that contains information about these movies.

imdb_1000.csv

```
star_rating,title,content_rating,genre,duration,actors_list
9.3,The Shawshank Redemption,R,Crime,142,"['Tim Robbins', 'Morgan Freeman', 'Bob Gunton']"
9.2,The Godfather,R,Crime,175,"['Marlon Brando', 'Al Pacino', 'James Caan']"
9.1,The Godfather: Part II,R,Crime,200,"['Al Pacino', 'Robert De Niro', 'Robert Duvall']"
9,The Dark Knight,PG-13,Action,152,"['Christian Bale', 'Heath Ledger', 'Aaron Eckhart']"
8.9,Pulp Fiction,R,Crime,154,"['John Travolta', 'Uma Thurman', 'Samuel L. Jackson']"
8.9,12 Angry Men,NOT RATED,Drama,96,"['Henry Fonda', 'Lee J. Cobb', 'Martin Balsam']"
8.9,"The Good, the Bad and the Ugly",NOT RATED,Western,161,"['Clint Eastwood', 'Eli Wallach', 'Lee Van Cleef']"
8.9,The Lord of the Rings: The Return of the King,PG-13,Adventure,201,"['Elijah Wood', 'Viggo Mortensen', 'Ian McKellen']"
8.9,Schindler's List,R,Biography,195,"['Liam Neeson', 'Ralph Fiennes', 'Ben Kingsley']"
8.9,Fight Club,R,Drama,139,"['Brad Pitt', 'Edward Norton', 'Helena Bonham Carter']"
8.8,The Lord of the Rings: The Fellowship of the Ring,PG-13,Adventure,178,"['Elijah Wood', 'Ian McKellen', 'Orlando Bloom']"
8.8,Inception,PG-13,Action,148,"['Leonardo DiCaprio', 'Joseph Gordon-Levitt', 'Ellen Page']"
8.8,Star Wars: Episode V - The Empire Strikes Back,PG,Action,124,"['Mark Hamill', 'Harrison Ford', 'Carrie Fisher']"
8.8,Forrest Gump,PG-13,Drama,142,"['Tom Hanks', 'Robin Wright', 'Gary Sinise']"
8.8,The Lord of the Rings: The Two Towers,PG-13,Adventure,179,"['Elijah Wood', 'Ian McKellen', 'Viggo Mortensen']"
8.7,Interstellar,PG-13,Adventure,169,"['Matthew McConaughey', 'Anne Hathaway', 'Jessica Chastain']"
8.7,One Flew Over the Cuckoo's Nest,R,Drama,133,"['Jack Nicholson', 'Louise Fletcher', 'Michael Berryman']"
8.7,Seven Samurai,UNRATED,Drama,207,"['Toshir\xf4 Mifune', 'Takashi Shimura', 'Keiko Tsushima']"
8.7,Goodfellas,R,Biography,146,"['Robert De Niro', 'Ray Liotta', 'Joe Pesci']"
8.7,Star Wars,PG,Action,121,"['Mark Hamill', 'Harrison Ford', 'Carrie Fisher']"
8.7,The Matrix,R,Action,136,"['Keanu Reeves', 'Laurence Fishburne', 'Carrie-Anne Moss']"
8.7,City of God,R,Crime,130,"['Alexandre Rodrigues', 'Matheus Nachtergaele', 'Leandro Firmino']"
```

pandas

Opening this text file using Excel:



	A	B	C	D	E	F	G	H	I	J	K	L
1	star_rating	title	content_rati	genre	duration	actors_list						
2	9.3	The Shawsha	R	Crime	142	['Tim Robbins', 'Morgan Freeman', 'Bob Gunton']						
3	9.2	The Godfath	R	Crime	175	['Marlon Brando', 'Al Pacino', 'James Caan']						
4	9.1	The Godfath	R	Crime	200	['Al Pacino', 'Robert De Niro', 'Robert Duvall']						
5	9	The Dark Kni	PG-13	Action	152	['Christian Bale', 'Heath Ledger', 'Aaron Eckhart']						
6	8.9	Pulp Fiction	R	Crime	154	['John Travolta', 'Uma Thurman', 'Samuel L. Jackson']						
7	8.9	12 Angry Me	NOT RATED	Drama	96	['Henry Fonda', 'Lee J. Cobb', 'Martin Balsam']						
8	8.9	The Good, th	NOT RATED	Western	161	['Clint Eastwood', 'Eli Wallach', 'Lee Van Cleef']						
9	8.9	The Lord of t	PG-13	Adventure	201	['Elijah Wood', 'Viggo Mortensen', 'Ian McKellen']						
10	8.9	Schindler's L	R	Biography	195	['Liam Neeson', 'Ralph Fiennes', 'Ben Kingsley']						
11	8.9	Fight Club	R	Drama	139	['Brad Pitt', 'Edward Norton', 'Helena Bonham Carter']						
12	8.8	The Lord of t	PG-13	Adventure	178	['Elijah Wood', 'Ian McKellen', 'Orlando Bloom']						
13	8.8	Inception	PG-13	Action	148	['Leonardo DiCaprio', 'Joseph Gordon-Levitt', 'Ellen Page']						
14	8.8	Star Wars: E	PG	Action	124	['Mark Hamill', 'Harrison Ford', 'Carrie Fisher']						
15	8.8	Forrest Gum	PG-13	Drama	142	['Tom Hanks', 'Robin Wright', 'Gary Sinise']						
16	8.8	The Lord of t	PG-13	Adventure	179	['Elijah Wood', 'Ian McKellen', 'Viggo Mortensen']						
17	8.7	Interstellar	PG-13	Adventure	169	['Matthew McConaughey', 'Anne Hathaway', 'Jessica Chastain']						
18	8.7	One Flew Ov	R	Drama	133	['Jack Nicholson', 'Louise Fletcher', 'Michael Berryman']						
19	8.7	Seven Samu	UNRATED	Drama	207	['Toshir\xf4 Mifune', 'Takashi Shimura', 'Keiko Tsushima']						
20	8.7	Goodfellas	R	Biography	146	['Robert De Niro', 'Ray Liotta', 'Joe Pesci']						
21	8.7	Star Wars	PG	Action	121	['Mark Hamill', 'Harrison Ford', 'Carrie Fisher']						
22	8.7	The Matrix	R	Action	136	['Keanu Reeves', 'Laurence Fishburne', 'Carrie-Anne Moss']						
23	8.7	City of God	R	Crime	130	['Alexandre Rodrigues', 'Matheus Nachtergaele', 'Leandro Firmino']						

pandas

Pandas can read in a csv file and store it as a **dataframe**, a 2D data structure.

```
In [1]: import numpy as np
        import pandas as pd

In [2]: movies = pd.read_csv("imdb_1000.csv")

In [3]: movies.head()
```

← Top five rows of the dataframe.

Out[3]:

	star_rating	title	content_rating	genre	duration	actors_list
0	9.3	The Shawshank Redemption	R	Crime	142	['Tim Robbins', 'Morgan Freeman', 'Bob Gunton']
1	9.2	The Godfather	R	Crime	175	['Marlon Brando', 'Al Pacino', 'James Caan']
2	9.1	The Godfather: Part II	R	Crime	200	['Al Pacino', 'Robert De Niro', 'Robert Duvall']
3	9.0	The Dark Knight	PG-13	Action	152	['Christian Bale', 'Heath Ledger', 'Aaron Eckh...
4	8.9	Pulp Fiction	R	Crime	154	['John Travolta', 'Uma Thurman', 'Samuel L. Ja...

Dataframe

The movies data structure in the previous slides is a **dataframe**. A dataframe is an enhanced two-dimensional data structure. You can think of it as a dictionary where the keys are the column names and the values are column values. Each column is called a **Series**.

Column names are like "keys" of a dictionary.

Column values are like "values" of a dictionary.

	star_rating	title	content_rating	genre	duration	actors_list
0	9.3	The Shawshank Redemption	R	Crime	142	['Tim Robbins', 'Morgan Freeman', 'Bob Gunton']
1	9.2	The Godfather	R	Crime	175	['Marlon Brando', 'Al Pacino', 'James Caan']
2	9.1	The Godfather: Part II	R	Crime	200	['Al Pacino', 'Robert De Niro', 'Robert Duvall']
3	9.0	The Dark Knight	PG-13	Action	152	['Christian Bale', 'Heath Ledger', 'Aaron Eckh...
4	8.9	Pulp Fiction	R	Crime	154	['John Travolta', 'Uma Thurman', 'Samuel L. Ja...

key

value

```
In [4]: movies["genre"]
```

Accessing Columns of a Dataframe

To access a column of a dataframe, we use the same syntax as a dictionary, specifying the column name as the "key".

```
In [4]: movies["genre"]

Out[4]: 0          Crime
        1          Crime
        2          Crime
        3         Action
        4          Crime
                  ...
        974       Comedy
        975    Adventure
        976       Action
        977       Horror
        978        Crime
        Name: genre, Length: 979, dtype: object
```

You can select a column by using [] notation and specifying the column name as a string.

This displays the "genre" Series.

Accessing Columns of a Dataframe

Here's another column in this dataframe called "title".

```
In [5]: movies["title"]
```

```
Out[5]: 0                                        The Shawshank Redemption
        1                                                   The Godfather
        2                                     The Godfather: Part II
        3                                              The Dark Knight
        4                                                 Pulp Fiction
                                          ...
        974                                                    Tootsie
        975                                 Back to the Future Part III
        976    Master and Commander: The Far Side of the World
        977                                                 Poltergeist
        978                                                 Wall Street
```

Analyzing Data

For categorical data like movies genre, the method value_counts() returns a columns(Series) containing counts of unique values.

For example, suppose I am interested in investigating to see which of the movies genre is most popular in the top movies rated on IMDB.

Patterns can emerge when data are transformed using programs.

1 genre is Drama!

```
In [5]: movies["genre"].value_counts()

Out[5]: Drama        278
        Comedy       156
        Action       136
        Crime        124
        Biography     77
        Adventure     75
        Animation     62
        Horror        29
        Mystery       16
        Western        9
        Sci-Fi         5
        Thriller       5
        Film-Noir      3
        Family         2
        History        1
        Fantasy        1
        Name: genre, dtype: int64
```

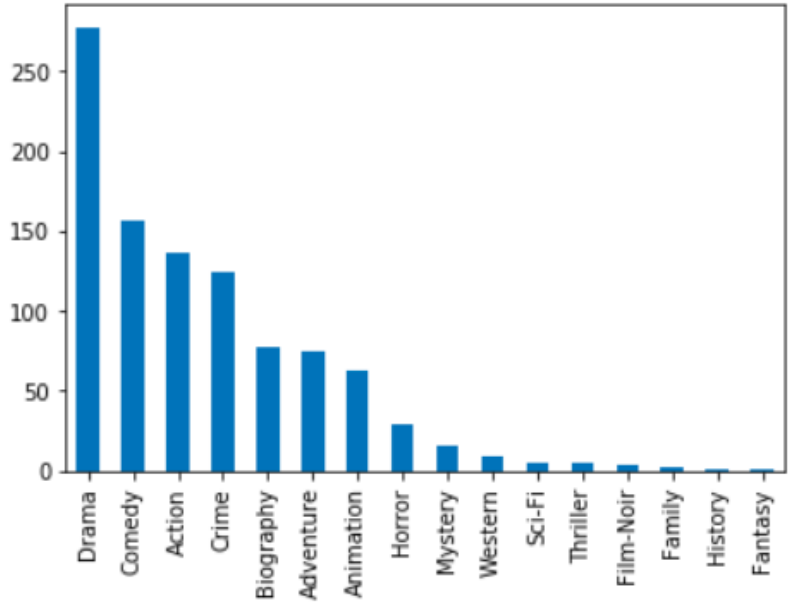
Plotting Data

Tables, charts, diagrams, text, and other visual tools can be used to communicate insight and knowledge gained from data.

The table of counts of unique values can be visually presented using pandas' plot function. We can specify the kind of graph with the "kind" optional parameter.

```
In [6]: movies["genre"].value_counts().plot(kind="bar")
```

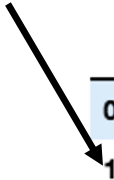
```
Out[6]: <matplotlib.axes._subplots.AxesSubplot at 0x110857978>
```



Positional Index

The columns of a dataframe share the same row "index". In this example, the index is **positional**.

Positional index



	star_rating	title	content_rating	genre	duration	actors_list
0	9.3	The Shawshank Redemption	R	Crime	142	['Tim Robbins', 'Morgan Freeman', 'Bob Gunton']
1	9.2	The Godfather	R	Crime	175	['Marlon Brando', 'Al Pacino', 'James Caan']
2	9.1	The Godfather: Part II	R	Crime	200	['Al Pacino', 'Robert De Niro', 'Robert Duvall']
3	9.0	The Dark Knight	PG-13	Action	152	['Christian Bale', 'Heath Ledger', 'Aaron Eckh...
4	8.9	Pulp Fiction	R	Crime	154	['John Travolta', 'Uma Thurman', 'Samuel L. Ja...

Accessing Rows with iloc

If the indexing is positional, `.iloc[]` notation allows us to access rows of a dataframe. For example, to get the top 4 rows of the dataframe:

slicing again!

When accessing rows, by default, pandas give us all columns

```
In [7]: movies.iloc[:4]
```

Out[7]:

	star_rating	title	content_rating	genre	duration	actors_list
0	9.3	The Shawshank Redemption	R	Crime	142	['Tim Robbins', 'Morgan Freeman', 'Bob Gunton']
1	9.2	The Godfather	R	Crime	175	['Marlon Brando', 'Al Pacino', 'James Caan']
2	9.1	The Godfather: Part II	R	Crime	200	['Al Pacino', 'Robert De Niro', 'Robert Duvall']
3	9.0	The Dark Knight	PG-13	Action	152	['Christian Bale', 'Heath Ledger', 'Aaron Eckh...

Labels Index

It is often to useful to index by **labels** instead of positional. For example, we may prefer to index the movies based the title of the movie instead of integers.

set_index returns a new dataframe with "title" as the row index

label index

```
In [6]: movies_modified = movies.set_index("title")
        movies_modified
```

Out[6]:

title	star_rating	content_rating	genre	duration	actors_list
The Shawshank Redemption	9.3	R	Crime	142	['Tim Robbins', 'Morgan Freeman', 'Bob Gunton']
The Godfather	9.2	R	Crime	175	['Marlon Brando', 'Al Pacino', 'James Caan']
The Godfather: Part II	9.1	R	Crime	200	['Al Pacino', 'Robert De Niro', 'Robert Duvall']
The Dark Knight	9.0	PG-13	Action	152	['Christian Bale', 'Heath Ledger', 'Aaron Eckh...

Accessing Rows with loc[]

If the indexing is by labels, `.loc[]` notation allows us to access rows of a dataframe.

For example:

slicing with labels

When accessing rows, by default, pandas give us all columns.

```
In [9]: movies_modified.loc["The Godfather":"The Dark Knight"]
```

Out[9]:

title	star_rating	content_rating	genre	duration	actors_list
The Godfather	9.2	R	Crime	175	['Marlon Brando', 'Al Pacino', 'James Caan']
The Godfather: Part II	9.1	R	Crime	200	['Al Pacino', 'Robert De Niro', 'Robert Duvall']
The Dark Knight	9.0	PG-13	Action	152	['Christian Bale', 'Heath Ledger', 'Aaron Eckh...

Note: Unlike slicing in general as well as slicing with integer index(iloc), slicing with loc includes the endpoint. Above, it includes the Dark Knight.

Filtering Data

Some processes can be used to extract information. For example, data filtering systems are important tools for finding information and recognizing patterns in data.

What if I want to filter my data and only want to look at movies with at least 9.0 rating? You can use the .loc[] notation and insert a filtering boolean condition.

returns all rows satisfying this boolean.

```
In [7]: movies.loc[movies["star_rating"] >= 9]
```

Out[7]:

	star_rating	title	content_rating	genre	duration	actors_list
0	9.3	The Shawshank Redemption	R	Crime	142	['Tim Robbins', 'Morgan Freeman', 'Bob Gunton']
1	9.2	The Godfather	R	Crime	175	['Marlon Brando', 'Al Pacino', 'James Caan']
2	9.1	The Godfather: Part II	R	Crime	200	['Al Pacino', 'Robert De Niro', 'Robert Duvall']
3	9.0	The Dark Knight	PG-13	Action	152	['Christian Bale', 'Heath Ledger', 'Aaron Eckh...

Searching

Search tools are useful for efficiently finding information.

For example, what if I want to watch a highly rated movies that starred Christian Bale? Series.str.contains("string") can be used here.

```
In [8]: movies.loc[movies["actors_list"].str.contains("Christian Bale")]
```

Out[8]:

	star_rating	title	content_rating	genre	duration	actors_list
3	9.0	The Dark Knight	PG-13	Action	152	['Christian Bale', 'Heath Ledger', 'Aaron Eckh...
43	8.5	The Dark Knight Rises	PG-13	Action	165	['Christian Bale', 'Tom Hardy', 'Anne Hathaway']
53	8.5	The Prestige	PG-13	Drama	130	['Christian Bale', 'Hugh Jackman', 'Scarlett J...
113	8.3	Batman Begins	PG-13	Action	140	['Christian Bale', 'Michael Caine', 'Ken Watan...
446	7.9	The Fighter	R	Biography	116	['Mark Wahlberg', 'Christian Bale', 'Amy Adams']
504	7.8	Empire of the Sun	PG	Drama	153	['Christian Bale', 'John Malkovich', 'Miranda ...
555	7.8	3:10 to Yuma	R	Adventure	122	['Russell Crowe', 'Christian Bale', 'Ben Foster']
589	7.7	The Machinist	R	Drama	101	['Christian Bale', 'Jennifer Jason Leigh', 'Ai...
702	7.6	Jin ling shi san chai	R	Drama	146	['Christian Bale', 'Ni Ni', 'Xinyi Zhang']
732	7.6	American Psycho	R	Crime	102	['Christian Bale', 'Justin Theroux', 'Josh Luc...
815	7.6	Equilibrium	R	Action	107	['Christian Bale', 'Sean Bean', 'Emily Watson']

Titanic Dataset Example

On April 15, 1912, during her maiden voyage, the Titanic sank after colliding with an iceberg, killing 1502 out of 2224(32% survival rate) passengers and crew.

In [1]:
```
import numpy as np
import pandas as pd
titanic = pd.read_csv("titanic_train.csv")
titanic.head()
```

Out[1]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

Titanic Dataset Example

Calling the info() method on the dataframe, we can see that this dataset contains incomplete data. For example, Age has only 714 entries(out of 891) and Cabin only has 204 entries(out of 891).

```
In [2]: titanic.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
PassengerId    891 non-null int64
Survived       891 non-null int64
Pclass         891 non-null int64
Name           891 non-null object
Sex            891 non-null object
Age            714 non-null float64
SibSp          891 non-null int64
Parch          891 non-null int64
Ticket         891 non-null object
Fare           891 non-null float64
Cabin          204 non-null object
Embarked       889 non-null object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

Cleaning Data

Data sets pose challenges regardless of size, such as:

- the need to clean data
- incomplete data
- invalid data
- not uniform(spellings, abbreviations, capitalizations)
- the need to combine data sources

Depending on how data were collected, they may not be uniform. For example, if users enter data into an open field, the way they choose to abbreviate, spell, or capitalize something may vary from user to user.

A simple step in the **data cleaning** process is to make the data uniform without changing their meaning (e.g., replacing all equivalent abbreviations, spellings, and capitalizations with the same word).

For modelling and predicting, data cleaning may involve other steps. See next slides for examples involving the Titanic dataset.

Cleaning Data

If we are trying to create a model that predicts which passengers survived the Titanic shipwreck. To get a good model, we may clean our data by:

- dropping features that contain too many blank or null values(e.g. Cabin)
- dropping features that may not have high correlation to our prediction(e.g. Name, PassengerId, Ticket)
- Completing features that contain null values but may have high correlation to prediction(e.g. Age contains null values but have high correlation to Surviving)
  a) For example, fill in Age null values with median value.
- Creating new features from existing features(create new column "Title" from "Name" to include title of a person, e.g. Mr, Mrs, Miss, Master, etc..)
- Models require numerical values; we may need to convert categorical value to numerical values. For example, "Male" to 0 and "Female" to 1, "Non-binary" to 2.

Cleaning Data

Pandas allow us to drop columns:

```
In [3]: titanic.drop(["Name", "Ticket", "PassengerId", "Cabin"], inplace=True, axis=1)
        titanic.head()
```

Out[3]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	male	22.0	1	0	7.2500	S
1	1	1	female	38.0	1	0	71.2833	C
2	1	3	female	26.0	0	0	7.9250	S
3	1	1	female	35.0	1	0	53.1000	S
4	0	3	male	35.0	0	0	8.0500	S

Note: We are down to 8 columns instead of 12.

Extracting Information

One of the reasons that the shipwreck led to such loss of life was that there were not enough lifeboats for the passengers and crew.

Although there was some element of luck involved in surviving the sinking, some groups of people were more likely to survive than others, such as women, children, and the upper-class.

For example, we can call the groupby() method to group the passengers based on the column Pclass(1 = upper-class, 2 = middle-class, 3 = lower-class) and look at their survivor rate.

```
In [4]: titanic.groupby("Pclass")["Survived"].mean()

Out[4]: Pclass
        1    0.629630
        2    0.472826
        3    0.242363
        Name: Survived, dtype: float64
```