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.

Parallel computing uses a single computer with multiple processors. **Distributed computing** uses multiple computing devices to process those tasks.

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.

pandas

We'll explore pandas using Jupyter Lab by looking at a movies dataset: IMDB's Top 1000 movies.

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

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

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

In Jupyter Lab, instead of printing an object, running a cell(SHIFT ENTER) on the object outputs that object in the next "Out" cell.

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.

```
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.

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
```

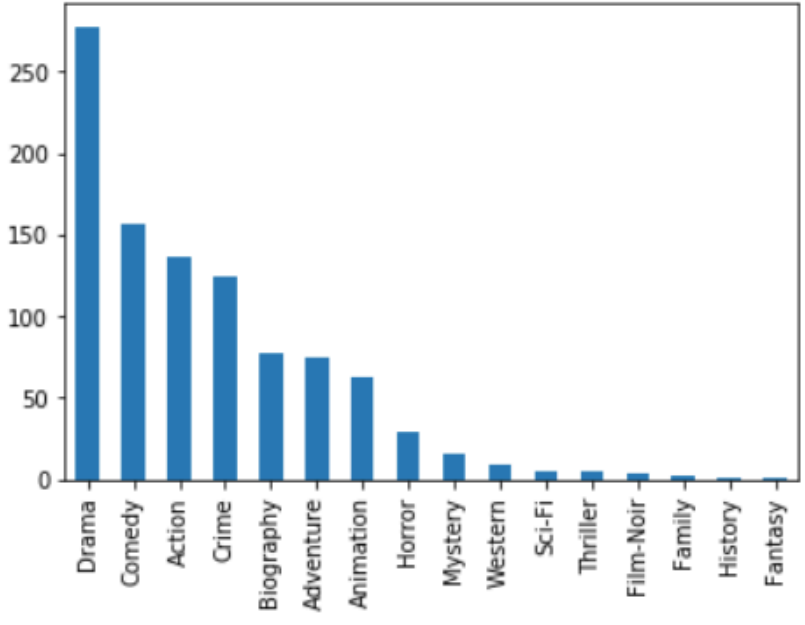
Plotting Data

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>
```



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.

```
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...

pandas

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']

Cleaning Data

Data sets pose challenges regardless of size, such as:

- the need to clean data
- incomplete data
- invalid data
- 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.

Cleaning data is a process that makes the data uniform without changing their meaning (e.g., replacing all equivalent abbreviations, spellings, and capitalizations with the same word).

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

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

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(Cabin)
- dropping features that may not have high correlation to our prediction(Name, PassengerId, Ticket)

```
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 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
```

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.