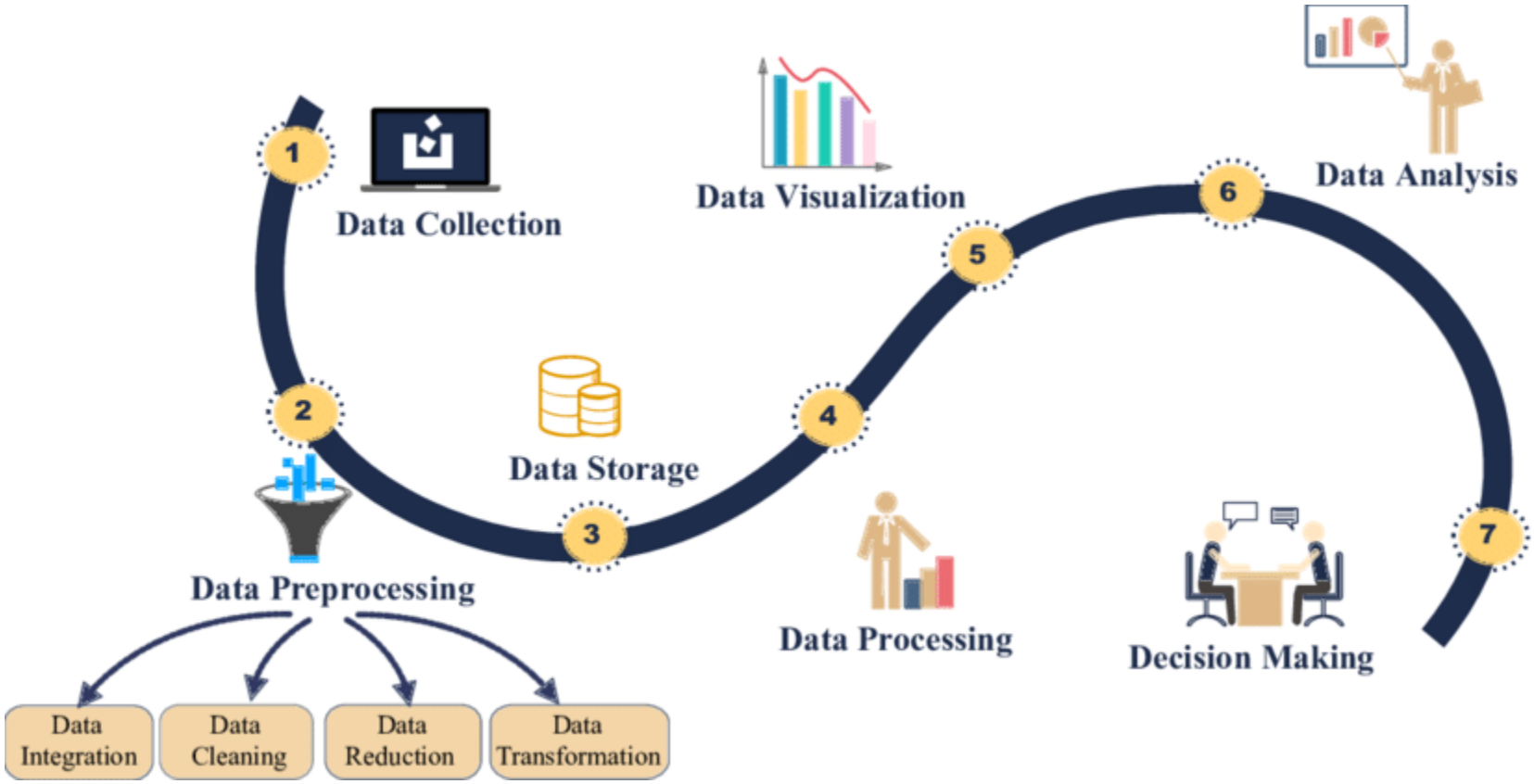
Practice 3:
Big Data System Preview

Big Data System Design

Big data processing

- Processing stages



Source: https://www.researchgate.net/figure/Big-data-processing-stages_fig1_351967613

Big data processing

- Data collection
  - Data is collected from various sources
    - Text
    - Image
    - Video
    - Audio
    - Sensor
    - Etc.

Big data processing

- Data preprocessing
  - Data is cleaned/transformed to be suitable for subsequent processes
    - Data cleaning
    - Data transformation
    - Data normalization
    - Feature selection
  - It is not a stage only for big data

Practice: data preprocessing

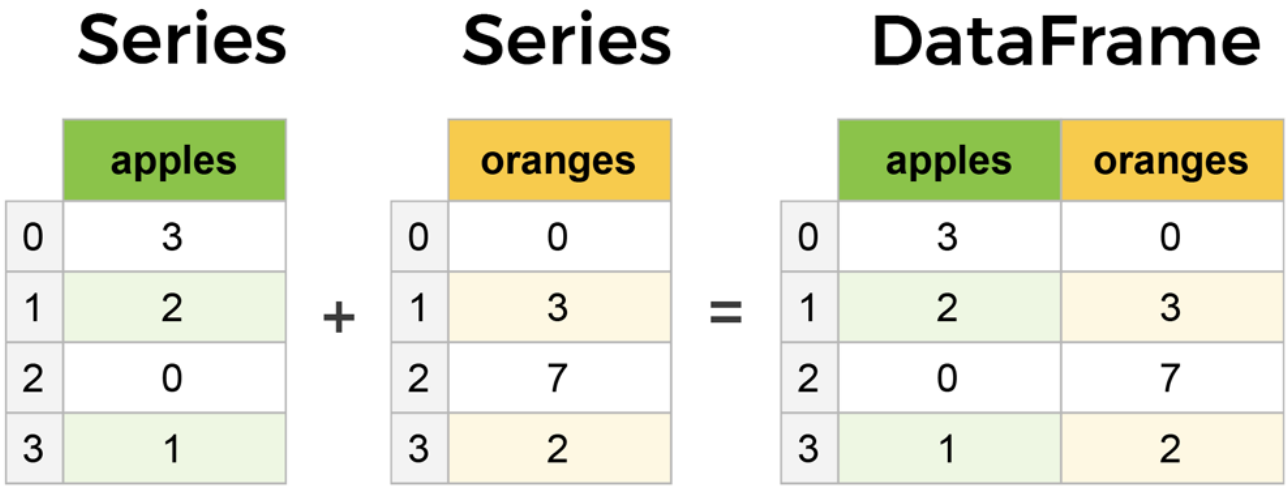
- Setup python libraries
  - Data processing
    - Pandas (pip install pandas)
    - Numpy
    - Scikit-learn
  - Data visualization
    - Seaborn (pip install seaborn)
    - Matplotlib

Practice: data preprocessing

- Appendix: Pandas
  - An open-source library
  - One of the most popular Python libraries for data science
    - Data read, data cleaning, data transforming, and data analysis

Practice: data preprocessing

- Appendix: Pandas
  - Data structure in Pandas
    - Series: a column
    - Data frame: multi-dimensional table (a collection of series)



Practice: data preprocessing

- Appendix: Pandas
  - Practice 1: create data frame

```
import pandas as pd

data = {'Artist': pd.Series(['Billie Holiday', 'Jimi Hendrix', 'Miles Davis', 'SIA']),
        'Genre': pd.Series(['Jazz', 'Rock', 'Jazz', 'Pop']),
        'Listeners': pd.Series([1300000, 2700000, 1500000, 2000000]),
        'Plays': pd.Series([27000000, 70000000, 48000000, 74000000])}

df = pd.DataFrame(data)
print(df)
```

Practice: data preprocessing

- Appendix: Pandas
  - Practice 1: create data frame

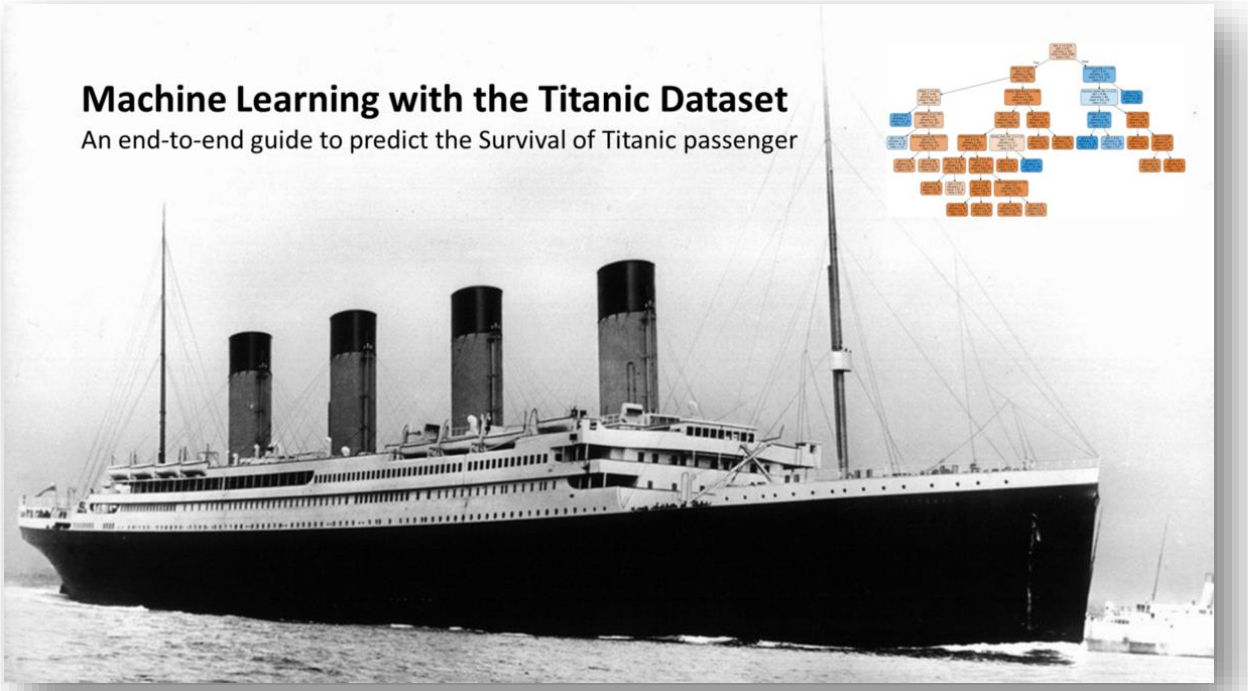
```
PS F:\PycharmProjects\bigdata_practice> python practice3.py
            Artist Genre  Listeners     Plays
0  Billie Holiday  Jazz     1300000  27000000
1     Jimi Hendrix  Rock     2700000  70000000
2     Miles Davis  Jazz     1500000  48000000
3              SIA   Pop     2000000  74000000
```

Practice: data preprocessing

- Dataset
  - titanic.csv

```
import pandas as pd

titanic_df = pd.read_csv("titanic.csv")
print(titanic_df)
```



Practice: data preprocessing

- Dataset
  - titanic.csv

```
PS F:\PycharmProjects\bigdata_practice> python practice3.py
     PassengerId  Survived  Pclass  ...     Fare Cabin  Embarked
0              1         0       3  ...   7.2500   NaN         S
1              2         1       1  ...  71.2833   C85         C
..           ...       ...     ...  ...      ...   ...       ...
890          891         0       3  ...   7.7500   NaN         Q
891          892         0       3  ...   7.7500   NaN         Q
892          893         0       3  ...   7.7500   NaN         Q
893          894         0       3  ...   7.7500   NaN         Q
894          895         0       3  ...   7.7500   NaN         Q
```

Practice: data preprocessing

- Dataset
  - titanic.csv

1	Passenger	Survived	Pclass	Name	Gender	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
2	1	0	3	Braund, M	male	22	1	0	A/5 21171			S
3	2	1	1	Cumings,	female	38	1	0	PC 17599	71.2833	C85	C
4	3	1	3	Heikkinen	female	26	0	0	STON/O2.	7.925		S
5	4	1	1	Futrelle, M	female	35	1	0	113803	53.1	C123	S
6	5	0	3	Allen, Mr.	male	35	0	0	373450	8.05		S
7	6	0	3	Moran, Mi	male		0	0	330877	8.4583		Q
8	7	0	1	McCarthy,	male	54	0	0	17463	51.8625	E46	S
9	8	0	3	Palsson, M	male	2	3	1	349909	21.075		S
10	9	1	3	Johnson, I	female	27	0	2	347742	11.1333		S
11	10	1	2	Nasser, M	female	14	1	0	237736	30.0708		C
12	11	1	3	Sandstron	female	4	1	1	PP 9549	16.7	G6	S
13	12	1	1	Bonnell, M	female	58	0	0	113783	26.55	C103	S
14	13	0	3	Saundercc	male	20	0	0	A/5. 2151	8.05		S
15	14	0	3	Andersson	male	39	1	5	347082	31.275		S
16	15	0	3	Vestrom,	female	14	0	0	350406	7.8542		S
17	16	1	2	Hewlett, M	female	55	0	0	248706	16		S
18	17	0	3	Rice, Mast	male	2	4	1	382652	29.125		Q
19	18	1	2	Williams,	male		0	0	244373	13		S

Practice: data preprocessing

- info()
  - Provides essential details about the target data frame
  - Features
    - Number of rows
    - Number of columns
    - Number of non-null values
    - Type of data in each column
    - How much memory the data frame is using

```
titanic_df.info()
```

Practice: data preprocessing

- info()
  - Provides essential details about the target data frame

```
PS F:\PycharmProjects\bigdata_practice> python practice3.py
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 895 entries, 0 to 894
 3   Name         895 non-null    object
 4   Gender       895 non-null    object
 5   Age          718 non-null    float64
 6   SibSp        895 non-null    int64
 7   Parch        895 non-null    int64
 8   Ticket       895 non-null    object
 9   Fare         895 non-null    float64
 10  Cabin        204 non-null    object
 11  Embarked     893 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 84.0+ KB
```

Practice: data preprocessing

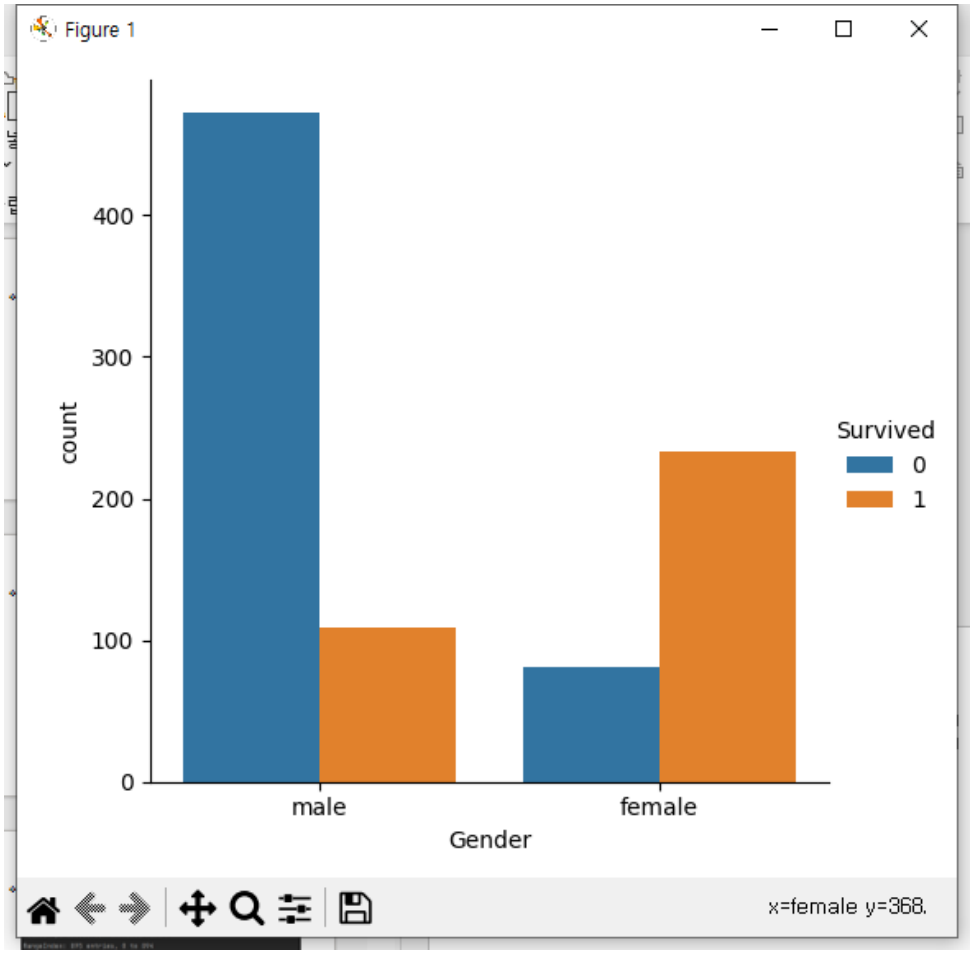
- Target variable
  - Gender, Survived

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

titanic_df = pd.read_csv("titanic.csv")
sns.catplot(x="Gender", hue="Survived", kind="count", data=titanic_df)
plt.show()
```

Practice: data preprocessing

- Target variable



Practice: data preprocessing

- head()
  - Print out a first five rows of the target data frame

```
print(titanic_df.head())
```

```
print(titanic_df.head(10))
```

Practice: data preprocessing

- head()
  - Print out a first five rows of the target data frame

```
PS F:\PycharmProjects\bigdata_practice> python practice3.py
   PassengerId  Survived  Pclass  ...     Fare Cabin  Embarked
0            1         0       3  ...   7.2500   NaN         S
1            2         1       1  ...  71.2833   C85         C
2            3         1       3  ...   7.9250   NaN         S
3            4         1       1  ...  53.1000  C123         S
4            5         0       3  ...   8.0500   NaN         S
```

```
   PassengerId  Survived  Pclass  ...     Fare Cabin  Embarked
0            1         0       3  ...   7.2500   NaN         S
1            2         1       1  ...  71.2833   C85         C
2            3         1       3  ...   7.9250   NaN         S
3            4         1       1  ...  53.1000  C123         S
4            5         0       3  ...   8.0500   NaN         S
5            6         0       3  ...   8.4583   NaN         Q
6            7         0       1  ...  51.8625   E46         S
7            8         0       3  ...  21.0750   NaN         S
8            9         1       3  ...  11.1333   NaN         S
9           10         1       2  ...  30.0708   NaN         C

[10 rows x 12 columns]
```

Practice: data preprocessing

- tail()
  - Print out the last five rows

```
print(titanic_df.tail())
```

```
print(titanic_df.tail(2))
```

Practice: data preprocessing

- tail()
  - Print out the last five rows

```
PS F:\PycharmProjects\bigdata_practice> python practice3.py
     PassengerId  Survived  Pclass                Name Gender  ...  Parch  Ticket  Fare Cabin  Embarked
890          891         0       3  Dooley, Mr. Patrick   male  ...      0  370376  7.75   NaN         Q
891          892         0       3  Dooley, Mr. Patrick   male  ...      0  370376  7.75   NaN         Q
892          893         0       3  Dooley, Mr. Patrick   male  ...      0  370376  7.75   NaN         Q
893          894         0       3  Dooley, Mr. Patrick   male  ...      0  370376  7.75   NaN         Q
894          895         0       3  Dooley, Mr. Patrick   male  ...      0  370376  7.75   NaN         Q
```

```
     PassengerId  Survived  Pclass                Name Gender  ...  Parch  Ticket  Fare Cabin  Embarked
893          894         0       3  Dooley, Mr. Patrick   male  ...      0  370376  7.75   NaN         Q
894          895         0       3  Dooley, Mr. Patrick   male  ...      0  370376  7.75   NaN         Q

[2 rows x 12 columns]
```

Practice: data preprocessing

- drop()
  - Used to delete columns and rows

```
titanic_df.drop(columns='Name', inplace=True)
titanic_df.info()
```

Practice: data preprocessing

- drop()
  - Used to delete columns and rows

```
PS F:\PycharmProjects\bigdata_practice> python practice3.py
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 895 entries, 0 to 894
Data columns (total 11 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  895 non-null    int64
 1   Survived     895 non-null    int64
 2   Pclass       895 non-null    int64
 3   Gender       895 non-null    object
 4   Age          718 non-null    float64
 5   SibSp        895 non-null    int64
 6   Parch        895 non-null    int64
 7   Ticket       895 non-null    object
 8   Fare         895 non-null    float64
 9   Cabin        204 non-null    object
 10  Embarked     893 non-null    object
dtypes: float64(2), int64(5), object(4)
memory usage: 77.0+ KB
```

Practice: data preprocessing

- Naïve data cleaning
  - Use drop()

```
titanic_df.drop(columns=['Cabin', 'Ticket', 'Embarked'], inplace=True)
print(titanic_df.head(5))
```

Practice: data preprocessing

- Naïve data cleaning
  - Use drop()

```
PS F:\PycharmProjects\bigdata_practice> python practice3.py
   PassengerId  Survived  Pclass                                               Name  Gender   Age  SibSp  Parch     Fare
0            1         0       3                            Braund, Mr. Owen Harris    male  22.0      1      0   7.2500
1            2         1       1  Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0      1      0  71.2833
2            3         1       3                             Heikkinen, Miss. Laina  female  26.0      0      0   7.9250
3            4         1       1       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1      0  53.1000
4            5         0       3                           Allen, Mr. William Henry    male  35.0      0      0   8.0500
```

Practice: data preprocessing

- Handling duplicate data
  - Data comes from different sources
    - When collecting and consolidating data from various sources, it's possible that data duplicates exist
  - Benefits of removing duplicate data?
    - Efficient storage allocation
    - Cost savings
    - Faster data analysis
    - Avoid misleading statistics and maintain high accuracy of analysis

Practice: data preprocessing

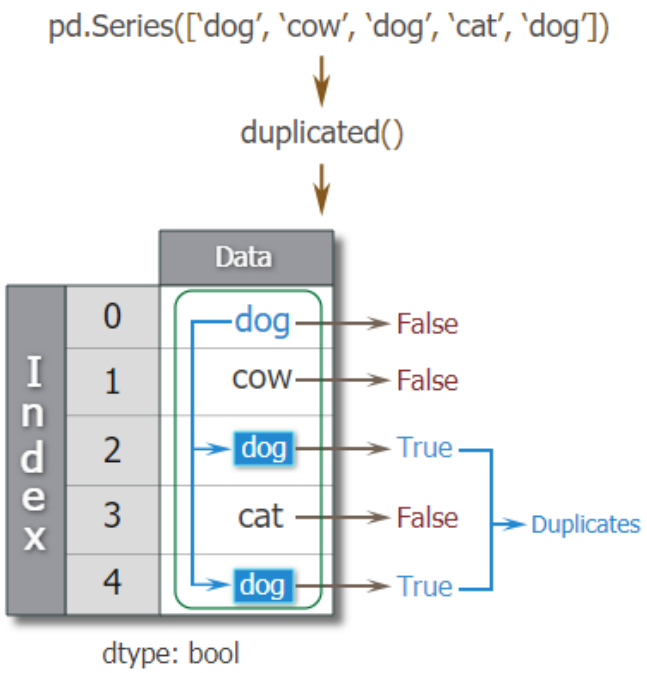
- Handling duplicate data

```
print(titanic_df.tail(10))
```

```
PS F:\PycharmProjects\bigdata_practice> python practice3.py
     PassengerId  Survived  Pclass                                          Name  Gender   Age  SibSp  Parch     Fare
885          886         0       3      Rice, Mrs. William (Margaret Norton)  female  39.0      0      5   29.125
886          887         0       2                     Montvila, Rev. Juozas    male  27.0      0      0   13.000
887          888         1       1              Graham, Miss. Margaret Edith  female  19.0      0      0   30.000
888          889         0       3  Johnston, Miss. Catherine Helen "Carrie"  female   NaN      1      2   23.450
889          890         1       1                     Behr, Mr. Karl Howell    male  26.0      0      0   30.000
890          891         0       3                       Dooley, Mr. Patrick    male  32.0      0      0    7.750
891          892         0       3                       Dooley, Mr. Patrick    male  32.0      0      0    7.750
892          893         0       3                       Dooley, Mr. Patrick    male  32.0      0      0    7.750
893          894         0       3                       Dooley, Mr. Patrick    male  32.0      0      0    7.750
894          895         0       3                       Dooley, Mr. Patrick    male  32.0      0      0    7.750
```

Practice: data preprocessing

- Handling duplicate data
  - duplicated()
    - Used to indicate duplicate values



Practice: data preprocessing

- Handling duplicate data
  - duplicated()
    - Used to indicate duplicate values

```
titanic_df.drop(columns=['Cabin', 'Ticket', 'Embarked', 'PassengerId'], inplace=True)
print(titanic_df[titanic_df.duplicated()])
```

Practice: data preprocessing

- Handling duplicate data
  - duplicated()
    - Used to indicate duplicate values

```
PS F:\PycharmProjects\bigdata_practice> python practice3.py
     Survived  Pclass                 Name Gender   Age  SibSp  Parch  Fare
891         0       3  Dooley, Mr. Patrick   male  32.0      0      0  7.75
892         0       3  Dooley, Mr. Patrick   male  32.0      0      0  7.75
893         0       3  Dooley, Mr. Patrick   male  32.0      0      0  7.75
894         0       3  Dooley, Mr. Patrick   male  32.0      0      0  7.75
```

Practice: data preprocessing

- Handling duplicate data
  - drop_duplicates()
    - Used to remove duplicate rows

```
titanic_df.drop(columns=['Cabin', 'Ticket', 'Embarked', 'PassengerId'], inplace=True)
titanic_df.drop_duplicates(inplace=True)
print(titanic_df[titanic_df.duplicated()])
print(titanic_df.tail(10))
```

Practice: data preprocessing

- Handling duplicate data
  - drop_duplicates()
    - Used to remove duplicate rows

```
PS F:\PycharmProjects\bigdata_practice> python practice3.py
Empty DataFrame
Columns: [Survived, Pclass, Name, Gender, Age, SibSp, Parch, Fare]
Index: []
     Survived  Pclass                                     Name  Gender   Age  SibSp  Parch     Fare
881         0       3                     Markun, Mr. Johann    male  33.0      0      0   7.8958
882         0       3              Dahlberg, Miss. Gerda Ulrika  female  22.0      0      0  10.5167
883         0       2             Banfield, Mr. Frederick James    male  28.0      0      0  10.5000
884         0       3                    Sutehall, Mr. Henry Jr    male  25.0      0      0   7.0500
885         0       3      Rice, Mrs. William (Margaret Norton)  female  39.0      0      5  29.1250
886         0       2                    Montvila, Rev. Juozas    male  27.0      0      0  13.0000
887         1       1              Graham, Miss. Margaret Edith  female  19.0      0      0  30.0000
888         0       3  Johnston, Miss. Catherine Helen "Carrie"  female   NaN      1      2  23.4500
889         1       1                     Behr, Mr. Karl Howell    male  26.0      0      0  30.0000
890         0       3                       Dooley, Mr. Patrick    male  32.0      0      0   7.7500
```

Practice: how to use text data?

- Setup scikit-learn
  - pip install scikit-learn

- Prepare an example dataset
  - Scikit-learn provides the 20 news groups dataset
    - News categories: 20
      - Atheism, graphics, ms-windows, pc.hardware, mac.hardware, windows.x, misc.forsale, rec.autos, rec.motorcycles, etc.
    - Number of news = 18,846

Practice: how to use text data?

- Prepare an example dataset
  - Fetch only eight categories

```
from sklearn.datasets import fetch_20newsgroups

cats = ['rec.motorcycles', 'rec.sport.baseball', 'comp.graphics',
        'comp.windows.x', 'talk.politics.mideast', 'sci.space',
        'sci.electronics', 'sci.med']

news_df = fetch_20newsgroups(subset='all', remove=('headers', 'footers', 'quotes'),
            categories=cats, random_state=15)

print(news_df["data"][0])
```

Practice: how to use text data?

- Prepare an example dataset
  - Fetch only eight categories

```
PS F:\PycharmProjects\bigdata_practice> python practice2.py



It works for me. I avoid obscenities, and try to remain calm cool and
collected, and try something like, "You almost just killed me, and I'm not
moving until you apologize." or something more or less benign like that. I
haven't been shot a single time, but I don't do it in Texas, and I do only
do it when there are plenty of witnesses around.
```

Practice: how to use text data?

- Dataset details

```
from sklearn.datasets import fetch_20newsgroups
import numpy as np  # pip install numpy

cats = ['rec.motorcycles', 'rec.sport.baseball', 'comp.graphics',
        'comp.windows.x', 'talk.politics.mideast', 'sci.space',
        'sci.electronics', 'sci.med']

news_df = fetch_20newsgroups(subset='all', remove=('headers', 'footers', 'quotes'),
                             categories=cats, random_state=15)

print(type(news_df))
print(news_df.keys())
print(type(news_df.data), type(news_df.target_names), type(news_df.target))

for i, val in zip(np.unique(news_df.target), news_df.target_names):
    print("index ({}): topic {} ".format(i, val))
```

Practice: how to use text data?

- Dataset details

```
PS F:\PycharmProjects\bigdata_practice> python practice2.py
<class 'sklearn.utils._bunch.Bunch'>
dict_keys(['data', 'filenames', 'target_names', 'target', 'DESCR'])
dict_keys(['data', 'filenames', 'target_names', 'target', 'DESCR'])
<class 'list'> <class 'list'> <class 'numpy.ndarray'>
index (0): topic comp.graphics
index (1): topic comp.windows.x
index (2): topic rec.motorcycles
index (3): topic rec.sport.baseball
index (4): topic sci.electronics
index (5): topic sci.med
index (6): topic sci.space
index (7): topic talk.politics.mideast
```

Practice: how to use text data?

- Dataset details

```
print(len(news_df.data), len(news_df.data[0]), len(news_df.data[1]))
print(len(news_df.target_names))
print(news_df.target.shape)
print(news_df.data[0][:100])
```

```
PS F:\PycharmProjects\bigdata_practice> python practice2.py
7852 350 1502
8
(7852,)



It works for me. I avoid obscenities, and try to remain calm cool and
collected, and try somethi
```

Practice: how to use text data?

- Data partitioning

```
cats = ['rec.motorcycles', 'rec.sport.baseball', 'comp.graphics',
        'comp.windows.x', 'talk.politics.mideast', 'sci.space',
        'sci.electronics', 'sci.med']

train_news = fetch_20newsgroups(subset='train', remove=('headers', 'footers', 'quotes'),
                                categories=cats, random_state=131)
test_news = fetch_20newsgroups(subset='test', remove=('headers', 'footers', 'quotes'),
                               categories=cats, random_state=131)
x_train = train_news.data
y_train = train_news.target
x_test = test_news.data
y_test = test_news.target
print('Train set size: {0}, Test set size: {1}'.format(len(train_news.data), len(test_news.data)))
```

Practice: how to use text data?

- Data partitioning

```
PS F:\PycharmProjects\bigdata_practice> python practice2.py
Train set size: 4714, Test set size: 3138
```

Practice: how to use text data?

- Vectorizer
  - CountVectorizer
    - Convert a collection of text documents to a matrix of token counts
  - TF-IDF Vectorizer
    - Convert a collection of raw documents to a matrix of TF-IDF features

Practice: how to use text data?

- Count vectorizer

```
from sklearn.feature_extraction.text import CountVectorizer
cnt_vect = CountVectorizer(max_df=0.95, max_features=1500,
                           min_df=2, stop_words='english',
                           ngram_range=(1, 2))
word_vect = cnt_vect.fit_transform(x_total)
np_vect = word_vect.todense()
print(np_vect[0])
print(cnt_vect.inverse_transform(word_vect[0]))
```

Practice: how to use text data?

- Count vectorizer

```
PS F:\PycharmProjects\bigdata_practice> python practice2.py
[[0 0 0 ... 0 0 0]]
[array(['separate', 'saw', 'middle', 'black', 'background', 'screen',
       'think', '500', 'feet'], dtype='<U19')]
```

Practice: how to use text data?

- Topic modeling using count vectorizer

```
from sklearn.decomposition import LatentDirichletAllocation
lda = LatentDirichletAllocation(n_components=8, random_state=42)
lda.fit(word_vect)
print(lda.components_.shape)
print(lda.components_)
```

Practice: how to use text data?

- Topic modeling using count vectorizer

```
PS F:\PycharmProjects\bigdata_practice> python practice2.py
(8, 1500)
[[1.25221325e-01 1.49870547e+02 1.95528685e+01 ... 8.36823791e+01
  4.31292153e+01 1.25077839e-01]
 [1.25102459e-01 6.72221321e-01 1.25087716e-01 ... 1.25053595e-01
  6.22477275e-01 2.19263348e+00]
 [3.23994680e-01 3.05135057e+02 1.25026746e-01 ... 1.39484450e+02
  3.46299260e+01 1.25000003e-01]
 ...
 [1.25090088e-01 1.25471005e-01 8.15700749e+00 ... 1.25068952e-01
  1.22051305e+01 1.25073027e-01]
 [1.67611828e+00 5.77774205e+01 1.25064246e-01 ... 1.33746461e-01
  9.60093075e+01 1.25110851e-01]
 [5.81133193e+01 1.11918128e+01 2.50930485e+00 ... 5.45190104e+00
  2.62359812e+00 1.36926247e+02]]
```

Practice: how to use text data?

- Topic modeling using count vectorizer

```
def display_topic_words(lda_model, feature_names, num_top_words):
    for topic_idx, topic in enumerate(lda_model.components_):
        print('\nTopic #', topic_idx+1)
        topic_word_idx = topic.argsort()[::-1]
        top_idx = topic_word_idx[:num_top_words]
        feature_concat = '+'.join([str(feature_names[i]) + '*' + str(round(topic[i], 1)) for i in top_idx])
        print(feature_concat)

feature_names = cnt_vect.get_feature_names_out()
display_topic_words(lda, feature_names, 15)
```

Practice: how to use text data?

- Topic modeling using count vectorizer

```
PS F:\PycharmProjects\bigdata_practice> python practice2.py

Topic # 1
10*517.1+medical*434.5+1993*420.5+health*377.3+12*323.6+disease*323.1+april*314.8+cancer*314.0+92*312.1+patients*303.1+20*295.3+11*276.2+research*258.4+study*250.3+hiv*245.1

Topic # 2
know*1165.9+like*1118.3+does*958.9+use*926.5+don*878.7+just*809.0+thanks*727.2+good*639.3+ve*633.2+need*606.1+used*589.0+want*514.3+help*481.4+think*468.0+make*462.3

Topic # 3
armenian*974.2+israel*808.6+people*748.1+armenians*724.6+turkish*683.1+jews*682.1+jewish*483.1+israeli*474.1+government*471.4+turkey*395.1+arab*386.1+war*369.6+armenia*350.8+said*343.0+russian*339.2

Topic # 4
space*1249.3+data*683.6+dos*638.5+nasa*481.5+software*464.7+dos dos*401.1+information*379.5+windows*360.7+program*338.7+systems*328.5+launch*319.2+available*290.3+support*283.8+pc*273.4+satellite*260.5

Topic # 5
year*586.8+game*479.1+team*378.2+entry*363.1+games*327.0+00*309.6+output*308.7+file*292.7+baseball*268.0+won*255.6+03*252.3+02*241.7+players*238.5+hit*228.6+win*226.5

Topic # 6
window*835.8+use*610.1+server*583.6+motif*483.3+display*449.7+widget*447.8+using*433.7+set*429.1+file*427.0+sun*395.6+application*375.5+program*362.2+x11*353.7+mit*317.4+xterm*304.1

Topic # 7
just*1265.2+don*1247.7+people*1139.1+like*1120.8+think*1021.1+time*967.2+know*856.3+said*811.1+didn*647.5+years*643.4+say*641.4+right*612.6+did*611.5+going*577.7+good*539.2

Topic # 8
edu*1712.1+image*1056.5+jpeg*803.1+file*768.2+ftp*764.3+graphics*755.1+com*686.6+pub*618.2+available*610.4+format*574.4+files*567.4+images*563.7+gif*543.7+mail*453.9+color*432.8
```

Practice: how to use text data?

- Topic probabilities for each document

```
doc_topics = lda.transform(word_vect)
print(doc_topics.shape)
print(doc_topics[:2])
import pandas as pd
def get_filename_list(newsdata):
    filename_lst = []
    for file in newsdata:
        filename_temp = file.split('\\')[-2:]
        filename = '.'.join(filename_temp)
        filename_lst.append(filename)
    return filename_lst
name_total = np.concatenate((train_news.filenames, test_news.filenames))
filename_lst = get_filename_list(name_total)
topic_names = ['Topic #' + str(i) for i in range(1, 9)]
topic_df = pd.DataFrame(data=doc_topics, columns=topic_names, index=filename_lst)
print(topic_df.head(20))
```

Practice: how to use text data?

- Topic probabilities for each document

	Topic #1	Topic #2	Topic #3	Topic #4	Topic #5	Topic #6	Topic #7	Topic #8
rec.sport.baseball.104862	0.010426	0.010436	0.010438	0.181947	0.290846	0.475042	0.010419	0.010445
rec.motorcycles.104468	0.002194	0.741931	0.064907	0.002197	0.002195	0.182184	0.002195	0.002196
talk.politics.mideast.76342	0.004171	0.442693	0.532264	0.004173	0.004178	0.004177	0.004174	0.004171
sci.electronics.53586	0.003574	0.845505	0.003580	0.003574	0.003575	0.003576	0.003574	0.133042
comp.windows.x.67220	0.002121	0.002122	0.002122	0.937402	0.049868	0.002123	0.002119	0.002123
sci.med.59090	0.062501	0.562280	0.062534	0.062523	0.062522	0.062628	0.062501	0.062511
talk.politics.mideast.76295	0.001738	0.236559	0.621531	0.001738	0.133218	0.001738	0.001739	0.001739
talk.politics.mideast.75954	0.004467	0.004473	0.968714	0.004467	0.004469	0.004472	0.004468	0.004470
talk.politics.mideast.75969	0.013890	0.595577	0.321025	0.013897	0.013906	0.013915	0.013900	0.013890
sci.space.60817	0.001625	0.642017	0.001625	0.001625	0.001625	0.001625	0.001626	0.348232
sci.electronics.53955	0.006599	0.953908	0.006582	0.006585	0.006582	0.006581	0.006582	0.006583
comp.windows.x.66899	0.011386	0.011394	0.011382	0.920307	0.011366	0.011385	0.011384	0.011396
comp.windows.x.66419	0.201842	0.005700	0.005687	0.764016	0.005687	0.005692	0.005684	0.005691
rec.motorcycles.104653	0.017877	0.426228	0.017864	0.183144	0.301224	0.017890	0.017887	0.017886
comp.windows.x.67175	0.913777	0.004174	0.004175	0.004176	0.004171	0.004170	0.061181	0.004176
rec.motorcycles.104537	0.025021	0.824827	0.025018	0.025012	0.025012	0.025056	0.025028	0.025026
sci.space.60982	0.002018	0.307617	0.002019	0.002018	0.002020	0.165425	0.002020	0.516863
rec.sport.baseball.104568	0.005212	0.005221	0.005214	0.005222	0.447249	0.521459	0.005211	0.005212

Practice: how to use text data?

- If you change the vectorizer settings, then the result is changed

```
cnt_vect = CountVectorizer(max_df=0.2, max_features=2000,
                           min_df=2, stop_words='english',
                           ngram_range=(1, 2))
```

Why?

Questions?

SEE YOU NEXT TIME!