Python & Spark
PTT18/19

Prof. Dr. Ralf Lämmel
Msc. Johannes Härtel
Msc. Marcel Heinz
The ‘Big Picture’

[Aggarwal15]
Plenty of Building Blocks are involved in this ‘Big Picture’
Back to the ‘Big Picture’

DATA COLLECTION → DATA PREPROCESSING

FEATURE EXTRACTION → CLEANING AND INTEGRATION

ANALYTICAL PROCESSING

BUILDING BLOCK 1 → BUILDING BLOCK 2

OUTPUT FOR ANALYST

FEEDBACK (OPTIONAL) → FEEDBACK (OPTIONAL)

[Aggarwal15]

(C) 2018, SoftLang Team, University of Koblenz-Landau
Foundations
Technologies and APIs
There are several technologies and APIs related to data-analysis in Python but the most convenient one is Pandas.

The following tutorial is inspired by the Book ‘Python for data Analysis’ [McKinney12].
What is contained in this CSV?

Some imports and configuration needed to read and print a CSV with Pandas.

```
import pandas as pd

# Some configurations needed to print Dataframes.
pd.options.display.max_rows = 4
pd.options.display.max_columns = 15
pd.options.display.width = 500
```
What is contained in this CSV?
Reading and printing CSV data with Pandas.

```python
In [2]:
data = pd.read_csv('data.csv')
data
```

<table>
<thead>
<tr>
<th>user_id</th>
<th>rating</th>
<th>gender</th>
<th>age</th>
<th>title</th>
<th>genres</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>F</td>
<td>1</td>
<td>One Flew Over the Cuckoo's Nest (1975)</td>
<td>Drama</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>M</td>
<td>56</td>
<td>One Flew Over the Cuckoo's Nest (1975)</td>
<td>Drama</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>1000207</td>
<td>5851</td>
<td>F</td>
<td>18</td>
<td>One Little Indian (1973)</td>
<td>Comedy</td>
</tr>
</tbody>
</table>

1000209 rows × 6 columns
What are the first 5 ratings in this CSV?
Selecting a range of rows returns another Dataframe.

In [30]:  data[:5]

<table>
<thead>
<tr>
<th>user_id</th>
<th>rating</th>
<th>gender</th>
<th>age</th>
<th>title</th>
<th>genres</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>5</td>
<td>F</td>
<td>1</td>
<td>One Flew Over the Cuckoo's Nest (1975)</td>
<td>Drama</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>M</td>
<td>56</td>
<td>One Flew Over the Cuckoo's Nest (1975)</td>
<td>Drama</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>M</td>
<td>25</td>
<td>One Flew Over the Cuckoo's Nest (1975)</td>
<td>Drama</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>M</td>
<td>50</td>
<td>One Flew Over the Cuckoo's Nest (1975)</td>
<td>Drama</td>
</tr>
</tbody>
</table>

5 rows x 6 columns
What is the title a rating refers to?
Selecting one column returns a Series (╯°□°)╯︵┻━┻

```
In [34]: data['title']

Out[34]:
    0                             One Flew Over the Cuckoo's Nest (1975)
    1                             One Flew Over the Cuckoo's Nest (1975)
                      ...                                           ...
  1000207                       One Little Indian (1973)
Name: title, Length: 1000209, dtype: object

In [44]: type(data['title'])

Out[44]: pandas.core.series.Series
```
What is the gender and the genre of a rating?

Selecting columns by passing a list returns a DataFrame:

```python
In [51]: data[['gender', 'genres']]`
What are ratings of female persons?

First we need a condition for filtering. Such condition can be stated as a Series of booleans.

```
In [58]: data['gender'] == 'F'

Out[58]:
     0    True
     1   False
     ... 
    1000207  True
    1000208   False
Name: gender, Length: 1000209, dtype: bool
```
What are ratings of female persons?
We can use this condition as a selection mechanism for rows.

```python
In [66]: data[data['gender'] == 'F']
```

<table>
<thead>
<tr>
<th>user_id</th>
<th>rating</th>
<th>gender</th>
<th>age</th>
<th>title</th>
<th>genres</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td>One Flew Over the Cuckoo's Nest (1975)</td>
<td>Drama</td>
</tr>
<tr>
<td>5</td>
<td>18</td>
<td>4</td>
<td>18</td>
<td>One Flew Over the Cuckoo's Nest (1975)</td>
<td>Drama</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>1000202</td>
<td>5494</td>
<td>4</td>
<td>35</td>
<td>Smoking/No Smoking (1993)</td>
<td>Comedy</td>
</tr>
<tr>
<td>1000207</td>
<td>5851</td>
<td>5</td>
<td>18</td>
<td>One Little Indian (1973)</td>
<td>Comedy</td>
</tr>
</tbody>
</table>

246440 rows × 6 columns
What is the amount of female and male ratings?
Let's try this!

```python
In [75]:
print('female users: ' + str(len(data[data['gender'] == 'F'])))
print('male users: ' + str(len(data[data['gender'] == 'M'])))
```

female users: 246440
male users: 753769
What is the amount of female and male ratings?

But we can also use dedicated Pandas functionality to create a Series that is indexed by the the distinct values.

```
In [85]: data['gender'].value_counts()

Out[85]:
M    753769
F    246440
Name: gender, dtype: int64
```
What is the amount of female and male ratings?

... and we can make python plot this.

```python
# Some new imports for plotting.
import matplotlib.pyplot as plt

# Plot the gender counts.
data['gender'].value_counts().plot(kind='bar')
```

Out[107]: `<matplotlib.axes._subplots.AxesSubplot at 0x27197018fd0>`
What is the average rating given by a user?

First we need to group the ratings of users. The following shows how to get all ratings of one user.

```
In [183]: data.groupby('user_id')

Out[183]: <pandas.core.groupby.DataFrameGroupBy object at 0x00000271D82186E0>
```

```
In [184]: (user_id, ratings) = list(data.groupby('user_id'))[0]

ratings
```

<table>
<thead>
<tr>
<th>user_id</th>
<th>rating</th>
<th>gender</th>
<th>age</th>
<th>title</th>
<th>genres</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>F</td>
<td>1</td>
<td>One Flew Over the Cuckoo's Nest (1975)</td>
<td>Drama</td>
</tr>
<tr>
<td>1725</td>
<td>3</td>
<td>F</td>
<td>1</td>
<td>James and the Giant Peach (1996)</td>
<td>Animation</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>56831</td>
<td>4</td>
<td>F</td>
<td>1</td>
<td>Fargo (1996)</td>
<td>Crime</td>
</tr>
<tr>
<td>59344</td>
<td>4</td>
<td>F</td>
<td>1</td>
<td>Dead Poets Society (1989)</td>
<td>Drama</td>
</tr>
</tbody>
</table>

53 rows x 6 columns
What is the average rating given by a user?
After grouping we can select the rating column and take the mean for each group.

```python
In [198]:
data.groupby('user_id')['rating'].mean()
```

```
Out[198]:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4.188679</td>
</tr>
<tr>
<td>2</td>
<td>3.713178</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>6039</td>
<td>3.878049</td>
</tr>
<tr>
<td>6040</td>
<td>3.577713</td>
</tr>
</tbody>
</table>
```

Name: rating, Length: 6040, dtype: float64
What is the average rating given by a user?
We can also create a summarization in terms of a boxplot.

```python
In [213]:
data.groupby('user_id')['rating'].mean().plot(kind='box')
```

```
Out[213]: <matplotlib.axes._subplots.AxesSubplot at 0x271ea34cdd8>
```
What is a gender’s average rating of a film?
A pivot table species rows and columns and aggregates the values using a passed function.

In [229]:

```python
data.pivot_table(values='rating', index=['title'], columns=['gender'], aggfunc='mean')
```

<table>
<thead>
<tr>
<th>gender</th>
<th>F</th>
<th>M</th>
</tr>
</thead>
<tbody>
<tr>
<td>title</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'81,000,000 Duck (1971)</td>
<td>3.375000</td>
<td>2.761905</td>
</tr>
<tr>
<td>'Night Mother (1986)</td>
<td>3.388889</td>
<td>3.352941</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Zeus and Roxanne (1997)</td>
<td>2.777778</td>
<td>2.357143</td>
</tr>
<tr>
<td>eXistenZ (1999)</td>
<td>3.098592</td>
<td>3.289086</td>
</tr>
</tbody>
</table>

3706 rows × 2 columns
What are the top female rated films?
i) We filter out films below a rating count of 250 to concentrate on the important candidates. ii) We increase the max rows since this is serious data! iii) We sort by column ‘F’ containing the average female ratings.

```python
# From previous slide.
film_mean_ratings = data.pivot_table(values='rating', index=['title'], columns=['gender'], aggfunc='mean')

# Some filtering of active films.
rating_counts = data.groupby('title').size()
active_titles = rating_counts.index[rating_counts >= 250]
film_mean_ratings = film_mean_ratings.loc[active_titles]

# Increase row count.
pd.options.display.max_rows = 15

# Sorting by female rating.
film_mean_ratings.sort_values(by='F', ascending=False)
```
What are the top female rated films?

<table>
<thead>
<tr>
<th>gender</th>
<th>F</th>
<th>M</th>
</tr>
</thead>
<tbody>
<tr>
<td>title</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Close Shave, A (1995)</td>
<td>4.644444</td>
<td>4.473795</td>
</tr>
<tr>
<td>Wrong Trousers, The (1993)</td>
<td>4.588235</td>
<td>4.478261</td>
</tr>
<tr>
<td>Sunset Blvd. (a.k.a. Sunset Boulevard) (1950)</td>
<td>4.572650</td>
<td>4.464589</td>
</tr>
<tr>
<td>Wallace &amp; Gromit: The Best of Aardman Animation (1996)</td>
<td>4.563107</td>
<td>4.385075</td>
</tr>
<tr>
<td>Schindler's List (1993)</td>
<td>4.562602</td>
<td>4.491415</td>
</tr>
<tr>
<td>Grand Day Out, A (1992)</td>
<td>4.537879</td>
<td>4.293255</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Howard the Duck (1986)</td>
<td>2.074627</td>
<td>2.103542</td>
</tr>
<tr>
<td>Anaconda (1997)</td>
<td>2.000000</td>
<td>2.248447</td>
</tr>
<tr>
<td>Avengers, The (1998)</td>
<td>1.915254</td>
<td>2.017467</td>
</tr>
<tr>
<td>Speed 2: Cruise Control (1997)</td>
<td>1.906667</td>
<td>1.863014</td>
</tr>
<tr>
<td>Rocky V (1990)</td>
<td>1.878788</td>
<td>2.132780</td>
</tr>
<tr>
<td>Barb Wire (1996)</td>
<td>1.585366</td>
<td>2.100386</td>
</tr>
<tr>
<td>Battlefield Earth (2000)</td>
<td>1.574468</td>
<td>1.616949</td>
</tr>
</tbody>
</table>
What is the film with the biggest disagreement in female and male rating?

We add a new column to the ‘film_mean_ratings’ Dataframe assigned to the difference between the female and male column.

```{python}
film_mean_ratings['diff'] = film_mean_ratings['M'] - film_mean_ratings['F']
film_mean_ratings.sort_values(by='diff')
```
What is the film with the biggest disagreement in female and male rating?

<table>
<thead>
<tr>
<th>title</th>
<th>F</th>
<th>M</th>
<th>diff</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dirty Dancing (1987)</td>
<td>3.790378</td>
<td>2.959596</td>
<td>-0.830782</td>
</tr>
<tr>
<td>Jumpin' Jack Flash (1986)</td>
<td>3.254717</td>
<td>2.578358</td>
<td>-0.676359</td>
</tr>
<tr>
<td>Grease (1978)</td>
<td>3.975265</td>
<td>3.367041</td>
<td>-0.608224</td>
</tr>
<tr>
<td>Little Women (1994)</td>
<td>3.870588</td>
<td>3.321739</td>
<td>-0.548849</td>
</tr>
<tr>
<td>Steel Magnolias (1989)</td>
<td>3.901734</td>
<td>3.365957</td>
<td>-0.535777</td>
</tr>
<tr>
<td>Anastasia (1997)</td>
<td>3.800000</td>
<td>3.281609</td>
<td>-0.518391</td>
</tr>
<tr>
<td>Rocky Horror Picture Show, The (1975)</td>
<td>3.673016</td>
<td>3.160131</td>
<td>-0.512885</td>
</tr>
<tr>
<td>Hidden, The (1987)</td>
<td>3.137931</td>
<td>3.745098</td>
<td>0.607167</td>
</tr>
<tr>
<td>Evil Dead II (Dead By Dawn) (1987)</td>
<td>3.297297</td>
<td>3.909283</td>
<td>0.611985</td>
</tr>
<tr>
<td>Cable Guy, The (1996)</td>
<td>2.250000</td>
<td>2.863787</td>
<td>0.613787</td>
</tr>
<tr>
<td>Longest Day, The (1962)</td>
<td>3.411765</td>
<td>4.031447</td>
<td>0.619682</td>
</tr>
<tr>
<td>Dumb &amp; Dumber (1994)</td>
<td>2.697987</td>
<td>3.336595</td>
<td>0.638608</td>
</tr>
<tr>
<td>Kentucky Fried Movie, The (1977)</td>
<td>2.878788</td>
<td>3.555147</td>
<td>0.676359</td>
</tr>
<tr>
<td>Good, The Bad and The Ugly, The (1966)</td>
<td>3.494949</td>
<td>4.221300</td>
<td>0.726351</td>
</tr>
</tbody>
</table>

(C) 2018, SoftLang Team, University of Koblenz-Landau
What is the movies with the most disagreement among all viewers?
The standard deviation can be used to describe such disagreement in ratings.

In [317]:
    rating_std_by_title = data.groupby('title')['rating'].std()
    # Filter down to active_titles.
    rating_std_by_title = rating_std_by_title.loc[active_titles]
    # Order Series by value in descending order.
    rating_std_by_title.sort_values(ascending=False)
What is the movie with the most disagreement among all viewers?

<table>
<thead>
<tr>
<th>title</th>
<th>rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dumb &amp; Dumber (1994)</td>
<td>1.321333</td>
</tr>
<tr>
<td>Blair Witch Project, The (1999)</td>
<td>1.316368</td>
</tr>
<tr>
<td>Natural Born Killers (1994)</td>
<td>1.307198</td>
</tr>
<tr>
<td>Tank Girl (1995)</td>
<td>1.277695</td>
</tr>
<tr>
<td>Rocky Horror Picture Show, The (1975)</td>
<td>1.260177</td>
</tr>
<tr>
<td>Eyes Wide Shut (1999)</td>
<td>1.259624</td>
</tr>
<tr>
<td>Evita (1996)</td>
<td>1.253631</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>North by Northwest (1959)</td>
<td>0.732515</td>
</tr>
<tr>
<td>Raiders of the Lost Ark (1981)</td>
<td>0.725647</td>
</tr>
<tr>
<td>Wrong Trousers, The (1993)</td>
<td>0.708666</td>
</tr>
<tr>
<td>Shawshank Redemption, The (1994)</td>
<td>0.700443</td>
</tr>
<tr>
<td>Great Escape, The (1963)</td>
<td>0.692585</td>
</tr>
<tr>
<td>Rear Window (1954)</td>
<td>0.688946</td>
</tr>
<tr>
<td>Close Shave, A (1995)</td>
<td>0.667143</td>
</tr>
</tbody>
</table>

Name: rating, Length: 1216, dtype: float64
Back to the ‘Big Picture’

DATA COLLECTION → DATA PREPROCESSING

- FEATURE EXTRACTION
- CLEANING AND INTEGRATION

ANALYTICAL PROCESSING

- BUILDING BLOCK 1
- BUILDING BLOCK 2

FEEDBACK (OPTIONAL)

OUTPUT FOR ANALYST

(C) 2018, SoftLang Team, University of Koblenz-Landau
Data
Data Integration (JSON)

JSON data can be loaded from a file and accessed comparable to dictionaries.

```python
import json

with open('office.json', 'r') as f:
    datastore = json.load(f)

datastore['parking']
```

Output:
```
{'location': 'premium', 'price': 750, 'style': 'covered'}
```
Data Integration (SQL)

An sqlite package provides, for instance, an in-memory database.

```python
import sqlite3

# Create a database in RAM
db = sqlite3.connect(':memory:)

# Get a cursor object
cursor = db.cursor()
cursor.execute(''
    CREATE TABLE users(id INTEGER PRIMARY KEY, name TEXT,''
        phone TEXT, email TEXT unique, password TEXT)
''
) db.commit()

db.close()
```

cf. [web_sql]
Data Integration (CSV)

Some CSV data needs to be combined before being processed.

```python
import pandas as pd

In [5]: pd.read_csv('ratings.csv')

<table>
<thead>
<tr>
<th>user_id</th>
<th>movie_id</th>
<th>rating</th>
<th>timestamp</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>5</td>
<td>978300760</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
<td>978302109</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

In [10]: pd.read_csv('movies.csv')

<table>
<thead>
<tr>
<th>movie_id</th>
<th>title</th>
<th>genres</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Toy Story (1995)</td>
<td>Animation, Children's, Comedy</td>
</tr>
<tr>
<td>1</td>
<td>Jumanji (1995)</td>
<td>Adventure, Children's, Fantasy</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
```

cf. [McKinney12]
Data Integration (CSV)

Comparable to joining tables in SQL, Pandas can merge different Dataframes.

```python
In [17]:
movies = pd.read_csv('movies.csv')
ratings = pd.read_csv('ratings.csv')
pd.merge(movies, ratings)
```

<table>
<thead>
<tr>
<th>movie_id</th>
<th>title</th>
<th>genres</th>
<th>user_id</th>
<th>rating</th>
<th>timestamp</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Toy Story (1995)</td>
<td>Animation, Children's, Comedy</td>
<td>1</td>
<td>5</td>
<td>978824268</td>
</tr>
<tr>
<td>1</td>
<td>Toy Story (1995)</td>
<td>Animation, Children's, Comedy</td>
<td>6</td>
<td>4</td>
<td>978237008</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>688259</td>
<td>Contender, The (2000)</td>
<td>Drama, Thriller</td>
<td>4085</td>
<td>3</td>
<td>987612960</td>
</tr>
</tbody>
</table>

688261 rows x 6 columns

cf. [McKinney12]
Feature Extraction (Java)
The ‘right’ features need to be extracted from artifacts for further processing.

```
package org.softlang.example;

public class SomeClassDoingNothing {
}
```

[AntonioliCCD00]
Feature Extraction (Java)

The ‘javalang’ package provides a parser for Java written in Python that can be installed from git.
Feature Extraction (Java)

The Java abstract syntax tree can be created from a file using 'javalang'.

```python
In[124]: import javalang

with open('SomeClassDoingNothing.java', 'r') as f:
    cu = javalang.parse.parse(f.read())

cu

Out[124]: CompilationUnit
```
Feature Extraction (Java)

Intuitively, the most relevant feature in this artifact is the classname.

```java
package org.softlang.example;

public class SomeClassDoingNothing {

}
```

In [125]: `cu.attrs`

Out[125]: `['package', 'imports', 'types']`

In [126]: `cu.types[0].attrs`

Out[126]: `['modifiers', 'annotations', 'documentation', 'name', 'body', 'type_parameters', 'extends', 'implements']`

In [127]: `cu.types[0].name`

Out[127]: `'SomeClassDoingNothing'`
Feature Extraction (Java)

Camel-case is split and strings are made lower-case.

```python
import re

def camel_case_split(identifier):
    matches = re.finditer('\b\w+:?\w+\b', identifier)
    return [m.group(0) for m in matches]

camel_case_split(cu.types[0].name)
```

```
Out[182]: ['Some', 'Class', 'Doing', 'Nothing']
```

```python
[x.lower() for x in camel_case_split(cu.types[0].name)]
```

```
Out[206]: ['some', 'class', 'doing', 'nothing']
```
Back to the ‘Big Picture’

DATA COLLECTION → DATA PREPROCESSING

FEATURE EXTRACTION → CLEANING AND INTEGRATION

FEEDBACK (OPTIONAL) → ANALYTICAL PROCESSING

BUILDING BLOCK 1 → BUILDING BLOCK 2 → OUTPUT FOR ANALYST

FEEDBACK (OPTIONAL)

(C) 2018, SoftLang Team, University of Koblenz-Landau
Analytical Processing
Classification

Support vector machines are provided by the ‘scikit-learn’ package as a supervised machine learning technique doing classification.

```
In [5]:
from sklearn import svm
   # The training data.
X = [[0, 0], [1, 1]]
y = [0, 1]
   # Fit on this data.
clf = svm.SVC()
clf.fit(X, y)
   # Use it to predict a classification.
clf.predict([[2., 2.]])
```

Out[5]: array([1])

cf. [scikit_cls]

[Aggarwal15]
Classification
Support vector machines in Python Spark.

Refer to the Python API docs for more details.

```python
from pyspark.ml.classification import LinearSVC

# Load training data
training = spark.read.format("libsvm").load("data/mllib/sample_libsvm_data.txt")

lsvc = LinearSVC(maxIter=10, regParam=0.1)

# Fit the model
lsvcModel = lsvc.fit(training)

# Print the coefficients and intercept for linearSVC
print("Coefficients: "+ str(lsvcModel.coefficients))
print("Intercept: "+ str(lsvcModel.intercept))
```

[spark]

(C) 2018, SoftLang Team, University of Koblenz-Landau
Clustering

The ‘scipy’ package provides hierarchical clustering as a unsupervised machine learning technique used to group this two-dimensional data.

```python
In [71]:
import matplotlib.pyplot as plt
from scipy.cluster.hierarchy import dendrogram, linkage
import numpy as np
```

cf. [web_cluster]
Clustering

Hierarchical clustering outputs a linkage array that can be depicted as a dendrogram.

```python
In [73]:
dendrogram(linkage(data, 'centroid'), leaf_rotation=90.)
plt.show()
```

cf. [web_cluster]
Clustering

K-means clustering in Python Spark.

```python
# Loads data.
dataset = spark.read.format("libsvm").load("data/mllib/sample_kmeans_data.txt")

# Trains a k-means model.
kmeans = KMeans().setK(2).setSeed(1)
model = kmeans.fit(dataset)

# Evaluate clustering by computing Within Set Sum of Squared Errors.
wssse = model.computeCost(dataset)
print("Within Set Sum of Squared Errors = " + str(wssse))

# Shows the result.
centers = model.clusterCenters()
print("Cluster Centers: ")
for center in centers:
    print(center)
```

(C) 2018, SoftLang Team, University of Koblenz-Landau
Back to the ‘Big Picture’

DATA COLLECTION → DATA PREPROCESSING → ANALYTICAL PROCESSING → OUTPUT FOR ANALYST

- FEATURE EXTRACTION
- CLEANING AND INTEGRATION
- BUILDING BLOCK 1
- BUILDING BLOCK 2

[Aggarwal15]

Feedback (optional)
Plot Types (Boxplot)

Gives a summary of distribution of numeric variables.

Package:
- Matplotlib
- Seaborn

cf. [seaborn]
Plot Types (Line chart)
Depicts the evolution of one or many columns.

Package:
● Matplotlib
Plot Types (Bar chart)
Depicts the ranking present in one column.

Package:
- Matplotlib
Plot Types (Scatter plot)
Depicts the correlation of two columns.

Package:
- Matplotlib
- Seaborn
Plot Types (Pie plot)
Depicts the part-whole relation.

Package:
- Matplotlib

cf. [py_pie]
Scaling and Axis

The table shows metrics on, e.g., the contributed code of Developers (column ‘DCon_PE_d’). While a few developers share very high contribution values most developer’s contributions is very low for one project.
Scaling and Axis

Axis can have different scales to correctly depict the data.

```python
In [28]: fig, ax = plt.subplots(figsize=(4, 4))
    ...: data.plot(kind='scatter', x='DConc_PE_d', y='DConc_AR_d', ax=ax)
    ...: plt.plot()
```

Out[28]: []
Scaling and Axis

Setting the axis on log does not work due to the 0 entries.

```python
In [40]:
fig, ax = plt.subplots(figsize=(4, 4))
ax.set_xscale("log")
ax.set_yscale("log")
data.plot(kind='scatter', x='DConc_PE_d', y='DConc_AR_d', ax=ax)
plt.plot()
```

```
Out[40]: []
```
Scaling and Axis

However, symlog works as it starts to scale linear under a given threshold.

In [41]:
   fig, ax = plt.subplots(figsize=(4, 4))
   ax.set_xscale("symlog", linthreshx=0.00001)
   ax.set_yscale("symlog", linthreshy=0.00001)
   data.plot(kind='scatter', x='DConc_PE_d', y='DConc_AR_d', ax=ax)
   plt.plot()

Out[41]: []
Subplots

Supplots can be used to group multiple plots that optionally share axis.

In [53]: plt.subplots(2, 2)

Out[53]:

{(matplotlib.figure.Figure at 0x13a78f9dce0),
 array([[<matplotlib.axes._subplots.AxesSubplot object at 0x0000013a78b7f400>,
         <matplotlib.axes._subplots.AxesSubplot object at 0x0000013a77a79588>],
        [<matplotlib.axes._subplots.AxesSubplot object at 0x0000013a779c10f0>,
         <matplotlib.axes._subplots.AxesSubplot object at 0x0000013a78f0c588>]], dtype=object)}
Subplots

Some sample of subplots showing the relation between API usage and lines of code for individual APIs.
Subplots
Some other sample of different kinds of subplots sharing axis.
Back to the ‘Big Picture’

DATA COLLECTION → DATA PREPROCESSING

FEATURE EXTRACTION → CLEANING AND INTEGRATION

ANALYTICAL PROCESSING

BUILDING BLOCK 1 → BUILDING BLOCK 2

OUTPUT FOR ANALYST

FEEDBACK (OPTIONAL)

[Aggarwal15]

(C) 2018, SoftLang Team, University of Koblenz-Landau
References

- [web_sql] https://www.pythoncentral.io/introduction-to-sqlite-in-python/
- [webGG] https://python-graph-gallery.com/
- [web_jl] https://github.com/c2nes/javalang
- [NL_reuters] https://github.com/fergiemcdowall/reuters-21578-json.git
- [seborn] https://seaborn.pydata.org/
- [py_pie] https://pythonspot.com/matplotlib-pie-chart/
- [spark] https://spark.apache.org/docs/latest/
- [spark_bp] https://umbertogriffo.gitbooks.io/apache-spark-best-practices-and-tuning/content/avoiding_shuffle_less_stage,_more_fast.html