Lab Seminar: 2022. 07. 05

# Importance of Data Science and Documentation & Paper Exploration

## Data Science from Scratch 2nd – Chapter 1, 2, 3

IDEALAB
Improving lives through learning

**Chan-Ki Kim**

**School of Computer Science/Department of AI Convergence Engineering**

**Gyeongsang National University (GNU)**

GNU 경상국립대학교 Gyeongsang National University

# Contents

# Introduction



- The Code in the Book
  - Recommendation System
  - Network Analysis
  - Simple Regression Analysis
  - Logistic Regression Analysis
  - Natural Language Processing

# Introduction

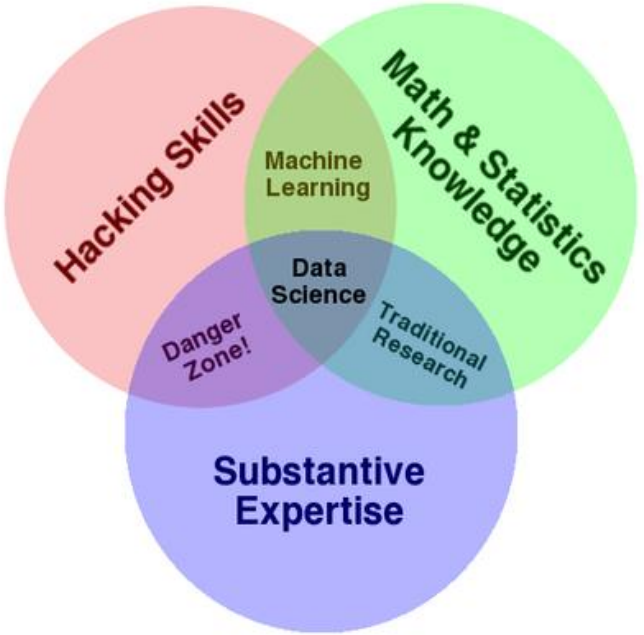


**What the author is trying to say through these codes?**

# The Data Science



- Data Science
  - Programming Skills
  - Math & Statistics Knowledge
  - Working Knowledge
    - An intersection of three elements



# Exploration of Python Documentation



- Docstring
  - 메소드, 모듈, 함수, 클래스 정의의 첫 번째 명령문으로 발생하는 String Literal
  - 작성한 docstring을 통해 프로그래밍의 속성으로 접근 가능

```
class ChanClass:
"""
클래스의 문서화 내용을 입력합니다.
"""

    def chanfunction(parameter):
        '''
        함수의 문서화 내용을 입력합니다.
        '''
```

# Exploration of Python Documentation



- Overload
  - 동일한 클래스 내에서, 매개 변수의 개수 또는 자료형이 다른 동일명의 메소드를 정의

```
from multipledispatch import dispatch

class OverloadKim():
    @dispatch(int, int)
    def chan(self, a, b):
        return a * b

    @dispatch(int, int, int)
    def chan(self, a, b, c):
        return a * b * c

kiki = OverloadKim()

print("Sample Overloading two parameмter : ", kiki.chan(2,2))

print("Sample Overloading three parameмter : ", kiki.chan(2,2,2))
```

# Exploration of Python Documentation



- Overload

```
Sample Overloading two paramemter :  4
Sample Overloading three paramemter :  8
```

# Exploration of Python Documentation



- Overriding
  - 부모 클래스의 메소드를 자식 클래스에서 재정의

```
class ParentKim():
    def __init__(self) :
        self.value = 20220705

    def changed_value(self) :
        return self.value

class ChildKim(ParentKim):
    def changed_value(self) :
        return self.value * 2022



testkim1 = ParentKim()
print("Before Overriding : ", testkim1.changed_value())

testkim2 = ChildKim()
print("After Overriding : ", testkim2.changed_value())
```

# Exploration of Python Documentation



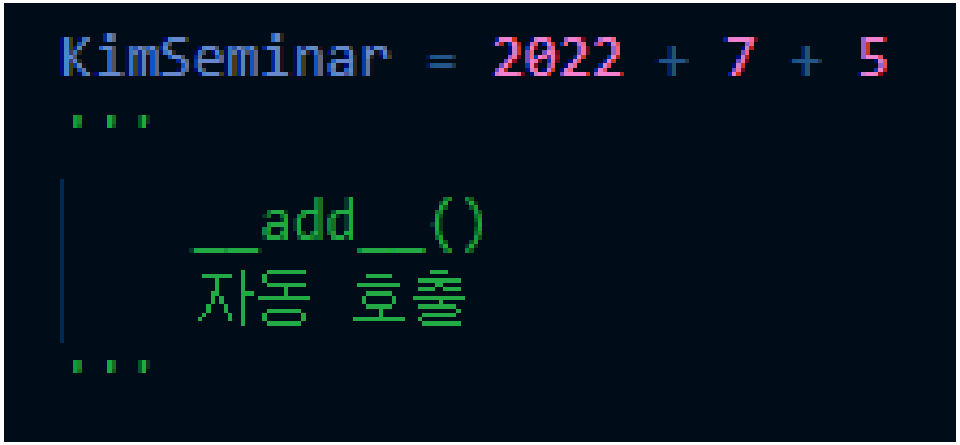
- Overriding

```
Before Overriding :  20220705
After Overriding :  40886265510
```

# Exploration of Python Documentation



- Magic Method
  - 파이썬 내에 정의되어 있고, 클래스 내부에서 Magic Method들을 Overriding해서 사용
  - 직접 호출해서 사용하지 않고, 정해진 규칙에 따라 알아서 호출
  - 대표적인 예로, + 연산자를 사용하여 두 개의 숫자를 더하면 내부적으로 __add__() 호출



# Exploration of Python Documentation



- Generator
  - 여러 개의 데이터를 미리 만들어 놓지 않고 필요할 때마다 즉석으로 하나씩 만들어낼 수 있는 객체 의미
  - 메모리에 한 번에 올리기 부담스러운 대용량 파일을 읽거나, 스트림 데이터를 처리할 때 유용
  - 한 번 호출되고 작업 수행이 완료되어도, 자기가 작업했던 일을 기억하면서 대기하고 재호출시 이전에 작업했던 일을 이어서 진행

# Exploration of Python Documentation



- Generator

```
from __future__ import division
import os
import psutil
import random
import time

input_data = ['KIM', 'CHAN', 'KI', '1ST', 'SUMMER', 'SEMINAR']
input_data2 = ['2022', '07', '05', '2001', '04']

process = psutil.Process(os.getpid())
mem_check_before = process.memory_info().rss / 1024 / 1024
'''

일반적인 함수와 제너레이터 정의 공간

'''
t1 = time.time()
#people = people_generator(100000000)
#people = people_list(100000000)
t2 = time.time()
mem_check_after = process.memory_info().rss / 1024 / 1024
total_time = t2 - t1

print('Memory usage before startup: {} MB'.format(mem_check_before))
print('Memory usage after process shutdown: {} MB'.format(mem_check_after))
print('총 소요된 시간: {:.7f} 초'.format(total_time))
```

# Exploration of Python Documentation



- Generator

```
def people_list(num_people):
    result = []
    for i in range(num_people):
        person = {
            'id': i,
            'name': random.choice(input_data),
            'major': random.choice(input_data2)
        }
        result.append(person)
    return result
```

# Exploration of Python Documentation



- Generator

```
Memory usage before startup: 13.83984375 MB
Memory usage after process shutdown: 9936.16796875 MB
총 소요된 시간: 106.6321659 초
```

# Exploration of Python Documentation



- Generator

```
def people_generator(num_people):
    for i in range(num_people):
        person = {
            'id': i,
            'name': random.choice(input_data),
            'major': random.choice(input_data2)
        }
        yield person
```

# Exploration of Python Documentation



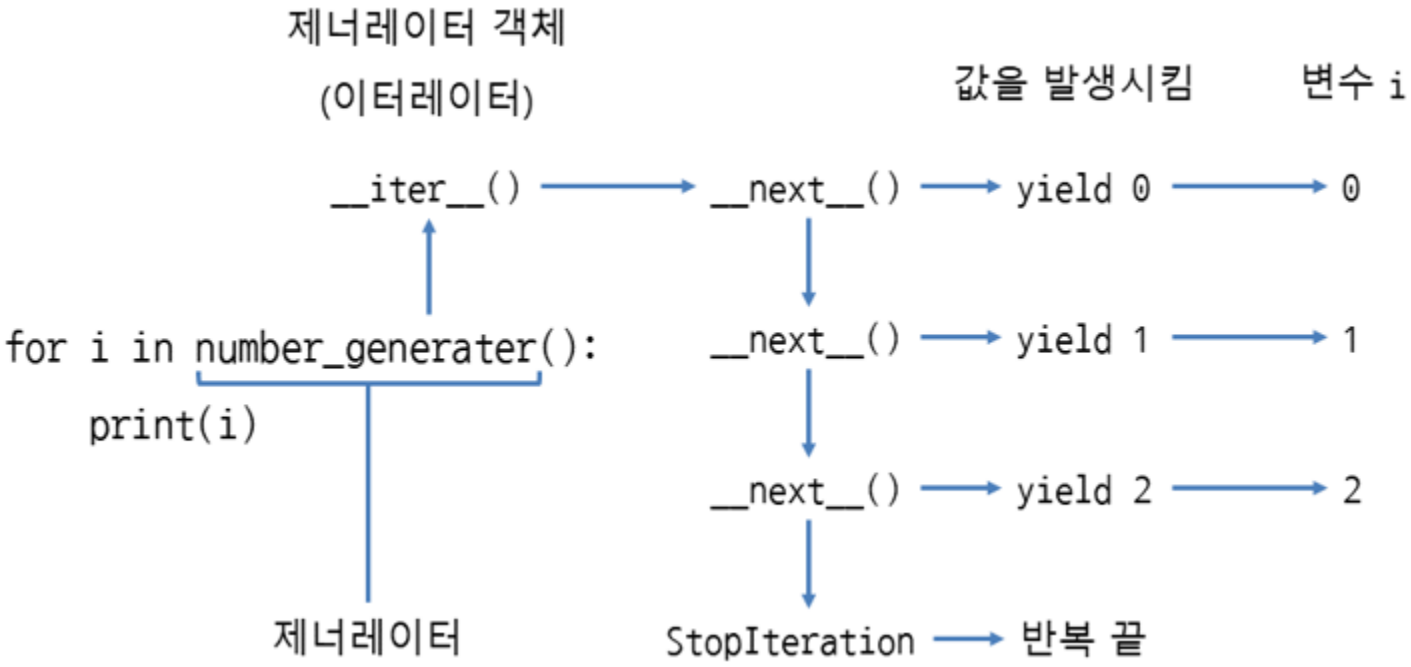
- Generator

```
Memory usage before startup: 13.7578125 MB
Memory usage after process shutdown: 13.76171875 MB
총 소요된 시간: 0.0000000 초
```

# Exploration of Python Documentation



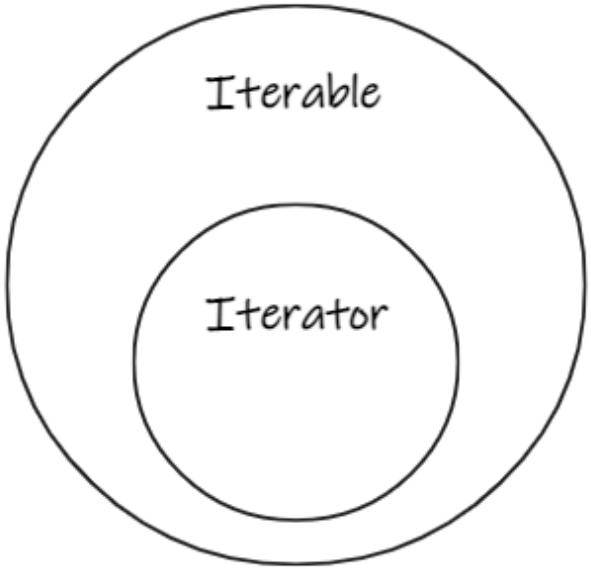
- yield
  - 잠시 함수 외부의 코드가 실행될 수 있도록 양보해 값을 가져가게 한 뒤 다시 Generator 안의 코드를 계속 실행
    - return
      - 반환 즉시 함수 종료



# Exploration of Python Documentation



- iterator
  - 값을 차례대로 꺼낼 수 있는 객체이며, 바로 다음에 처리해야 할 데이터를 기억
- iterable
  - 반복 가능한 객체, iterator 객체로 변환 가능한 객체
  - 한 번에 하나의 멤버를 반환 가능한 객체를 의미
    - Ex) list, dictionary



# Exploration of Python Documentation



- iterator

```
class ChanSeminar_iterator:

    def __init__(self, max = 10):
        if max >= 10:
            self.max = 10
        else:
            self.max = max
        self.n = 1

    def __iter__(self):
        return self

    def __next__(self):
        while self.n < self.max:
            self.n += 1
            return self.n - 1
        else:
            raise StopIteration
```

# Exploration of Python Documentation



- iterator

```
sample_kim_iter = iter([2022, 7, 5, 22])

print(next(sample_kim_iter))
print(next(sample_kim_iter))
print(next(sample_kim_iter))
print(next(sample_kim_iter))
```

```
2022
7
5
22
```

# Exploration of Python Documentation



- iterable

```python
class ChanSeminar_iterable:

    def __init__(self, max = 10):
        if max >= 10:
            self.max = 10
        else:
            self.max = max
        self.n = 1

    def __iter__(self):
        while self.n < self.max:
            yield self.n
            self.n += 1
```

# Exploration of Python Documentation



- iterable

```
sample_kim_list = [2022, 7, 5, 22]

for item in sample_kim_list:
    print(item)
```

```
2022
7
5
22
```

# Exploration of Python Documentation



- iterable

```
sample_kim_list = [2022, 7, 5, 22]

print(next(sample_kim_list))
```

```
TypeError: 'list' object is not an iterator
```

# Exploration of Python Documentation



- iterator & iterable

```python
chaniterableList = [2022, 7, 5]

print("This is iterable")

for item in chaniterableList:
    print(item)

for item in chaniterableList:
    print(item)


chaniterator = iter(chaniterableList)

print("This is iterator")

for item in chaniterator:
    print(item)

for item in chaniterator:
    print(item)
```

```
This is iterable
2022
7
5
2022
7
5
This is iterator
2022
7
5
```

# Matplotlib in Python (Niyazi Ari et al. ICECCO'14)



- About Matplotlib
  - Matplotlib은 Python programming language의 우수한 2D 및 3D 그래픽 플로팅 라이브러리 패키지
  - 사용자가 간단한 플롯들을 적은 수의 명령어들로, 또는 하나로 쉽게 제작 가능
  - Numpy와 SciPy 프레임워크로 만들어진 멀티-플랫폼 데이터 시각화 도구
  - 많은 작동 시스템들과 그래픽 백엔드들과 함께 원활한 작동

# Matplotlib in Python (Niyazi Ari et al. ICECCO'14)



- Advantages of Matplotlib
  - 친숙한 인터페이스
  - 일관된 비전 보유
  - STScI과 JPL의 천문학자들로부터 초기 제도적 지원을 입각해 제작
  - Python 세계 내에서 열정적으로 프로젝트를 홍보하는 Matplotlib의 개발자 Hunter의 존재

# Matplotlib in Python (Niyazi Ari et al. ICECCO'14)



- Matplotlib vs MATLAB
  - 시작 난이도가 상대적으로 쉬움
  - 그림 크기 및 DPI를 포함하여 그림의 모든 요소에 대한 뛰어난 제어
  - 대규모 소프트웨어 개발에 적합하며 완전한 기능을 갖춘 최신 객체 지향 프로그래밍 언어인 Python 기반
  - 무료, 오픈 소스, 라이선스 서버
  - 코딩이 사용자가 이해하고 확장하기에 충분한 난이도

# Expansion of Paper



- Seaborn

```
# Import seaborn objects
from .rcmod import *  # noqa: F401,F403
from .utils import *  # noqa: F401,F403
from .palettes import *  # noqa: F401,F403
from .relational import *  # noqa: F401,F403
from .regression import *  # noqa: F401,F403
from .categorical import *  # noqa: F401,F403
from .distributions import *  # noqa: F401,F403
from .matrix import *  # noqa: F401,F403
from .miscplot import *  # noqa: F401,F403
from .axisgrid import *  # noqa: F401,F403
from .widgets import *  # noqa: F401,F403
from .colors import xkcd_rgb, crayons  # noqa: F401
from . import cm  # noqa: F401

# Capture the original matplotlib rcParams
import matplotlib as mpl
_orig_rc_params = mpl.rcParams.copy()

# Define the seaborn version
__version__ = "0.11.2"
```

# Expansion of Paper



- Compare with Matplotlib and Seaborn – Boxplot Parameter

```
matplotlib.pyplot.boxplot(x, notch=None, sym=None, vert=None, whis=None, positions=None,
widths=None, patch_artist=None, bootstrap=None, usermedians=None, conf_intervals=None,
meanline=None, showmeans=None, showcaps=None, showbox=None, showfliers=None,
boxprops=None, labels=None, flierprops=None, medianprops=None, meanprops=None,
capprops=None, whiskerprops=None, manage_ticks=True, autorange=False, zorder=None, *,
data=None)                                                                    [source]
```

```
seaborn.boxplot(*, x=None, y=None, hue=None, data=None, order=None, hue_order=None, orient=None, color=None, palette=None,
saturation=0.75, width=0.8, dodge=True, fliersize=5, linewidth=None, whis=1.5, ax=None, **kwargs)
```

kwargs : *key, value mappings*

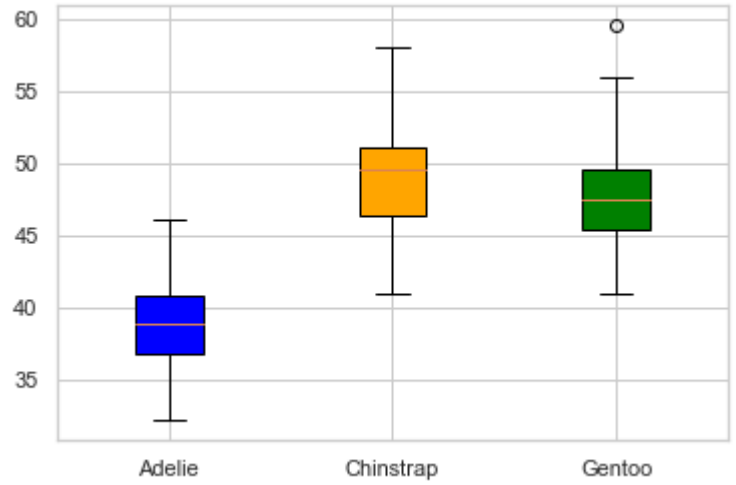
Other keyword arguments are passed through to `matplotlib.axes.Axes.boxplot()`.

# Expansion of Paper



- Compare with Matplotlib and Seaborn – Boxplot Parameter

```
import matplotlib.pyplot as plt
import seaborn as sns
penguin_data = sns.load_dataset('penguins')

category = penguin_data.species.unique()

species_a = penguin_data[penguin_data['species']==category[0]]
species_b = penguin_data[penguin_data['species']==category[1]]
species_c = penguin_data[penguin_data['species']==category[2]]

species_a_dropna = species_a.dropna()
species_b_dropna = species_b.dropna()
species_c_dropna = species_c.dropna()


fig, ax = plt.subplots()
define_boxplot = ax.boxplot([species_a_dropna[species_a_dropna.columns[2]],
            species_b_dropna[species_b_dropna.columns[2]],
            species_c_dropna[species_c_dropna.columns[2]]], patch_artist = True)
plt.xticks([1, 2, 3], [category[0], category[1], category[2]])

colors = ['blue', 'orange', 'green']

for patch, color in zip(define_boxplot['boxes'], colors):
    patch.set_facecolor(color)

plt.show()
```



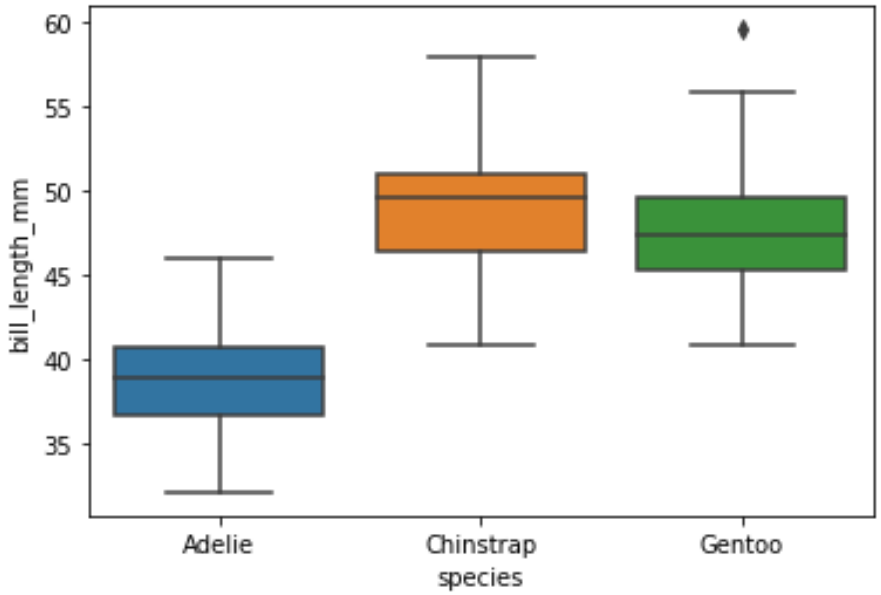
# Expansion of Paper



- Compare with Matplotlib and Seaborn – Boxplot Parameter

```
import seaborn as sns
penguin_data = sns.load_dataset('penguins')
penguin_dropna = penguin_data.dropna()
sns.boxplot(y='bill_length_mm', x='species', data=penguin_dropna)
```

```
<AxesSubplot:xlabel='species', ylabel='bill_length_mm'>
```

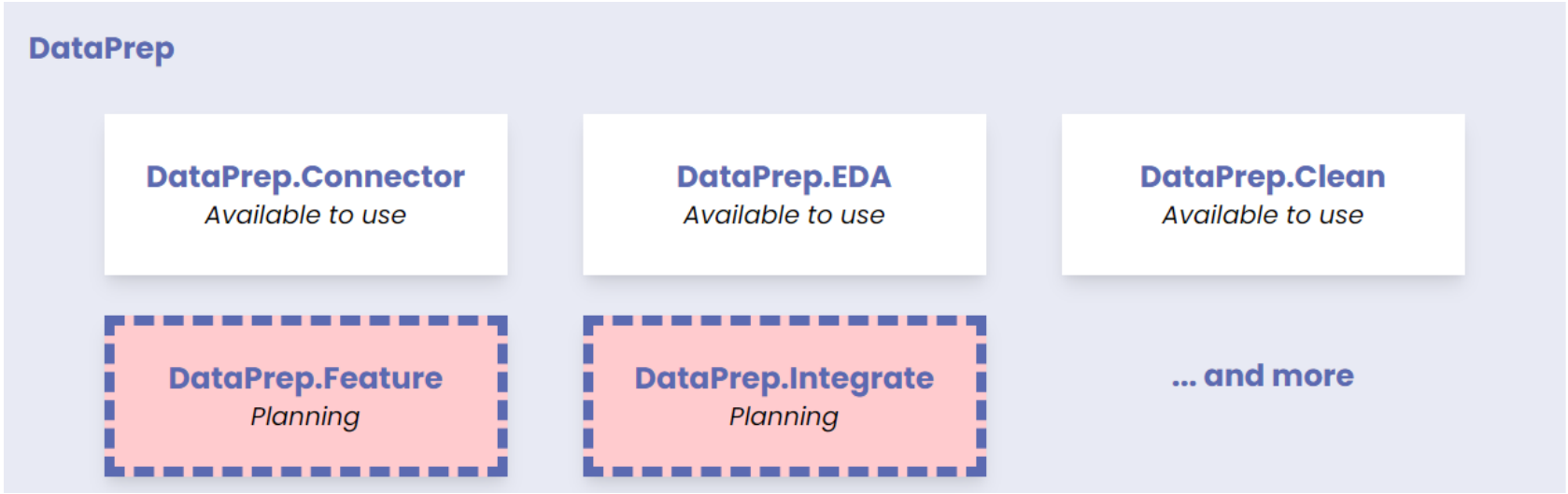


# Useful Library for Data Preparation & EDA



- DataPrep
  - 적은 수의 코드로 단일 라이브러리를 사용해 데이터를 준비할 수 있는 Python용 오픈 소스 라이브러리
  - 현재는 EDA, Connector, Clean, 이 세 가지 모듈만 존재

dataprep



# Useful Library for Data Preparation & EDA



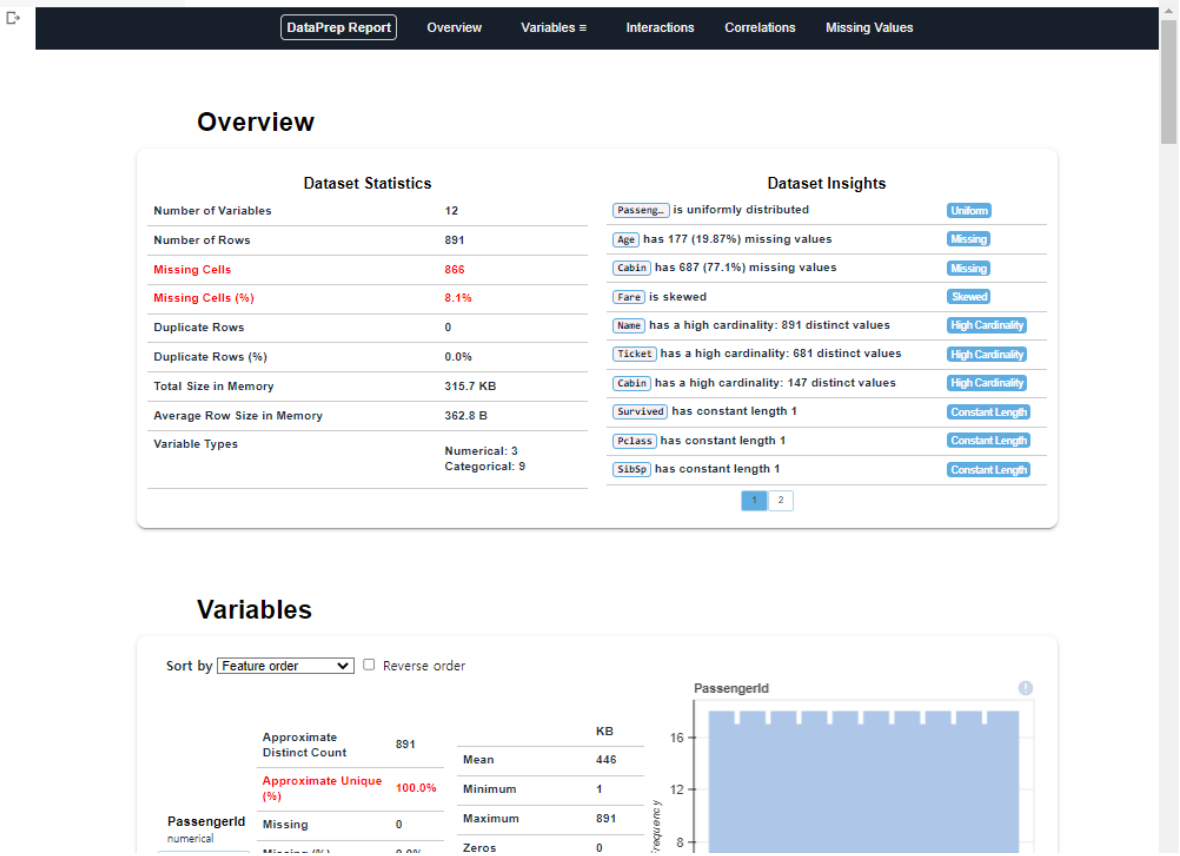
- DataPrep.EDA
  - Python에서 가장 빠르고 쉬운 EDA 도구
  - 빠른 html 형식의 프로필 보고서 생성
  - 빅데이터 지원
  - 대화형 시각화 in 보고서
  - 고도로 최적화된 Dask 기반 컴퓨팅 모듈 덕분에 Pandas 기반 프로파일링 도구보다 10배 빠른 속도 가능
    - Ex) Dask란, 빅데이터 분석에서 대용량 데이터를 다루기 위한 분산 컴퓨팅 도구

# Useful Library for Data Preparation & EDA



- DataPrep.EDA

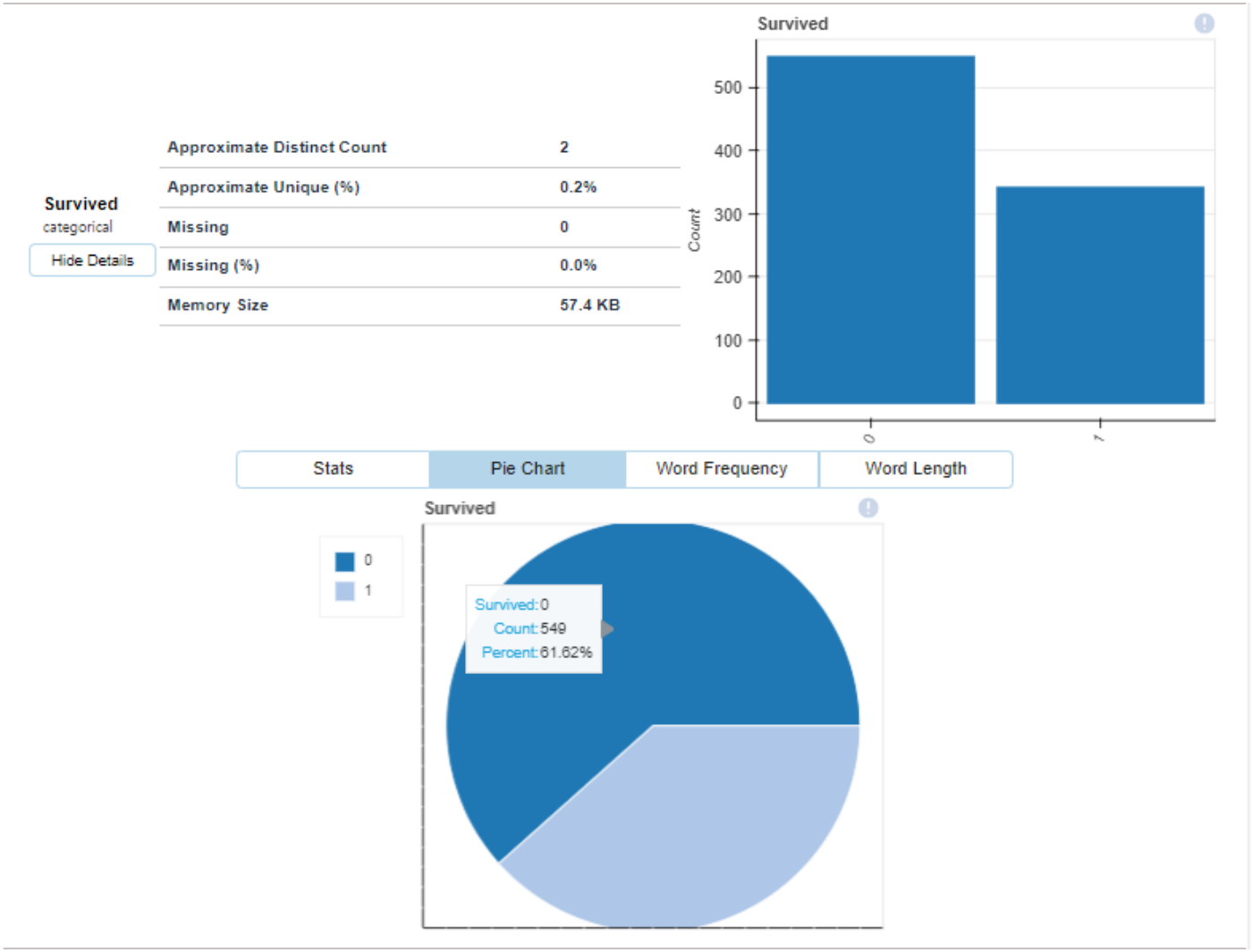
```
from dataprep.eda import *
import pandas as pd

train_df = pd.read_csv('Titanic_train_data.csv')
create_report(train_df)
```



# Useful Library for Data Preparation & EDA



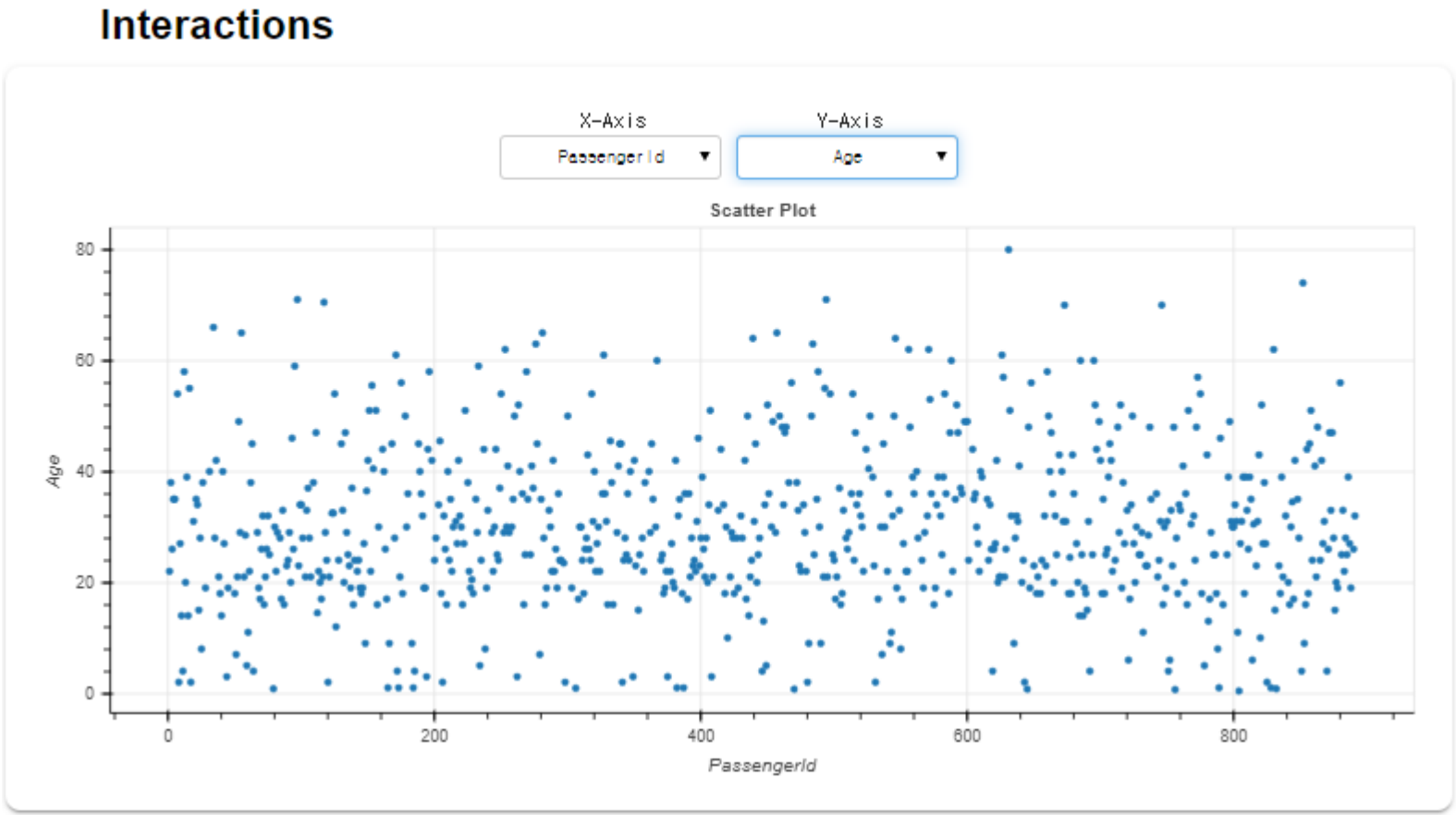
- DataPrep.EDA



# Useful Library for Data Preparation & EDA



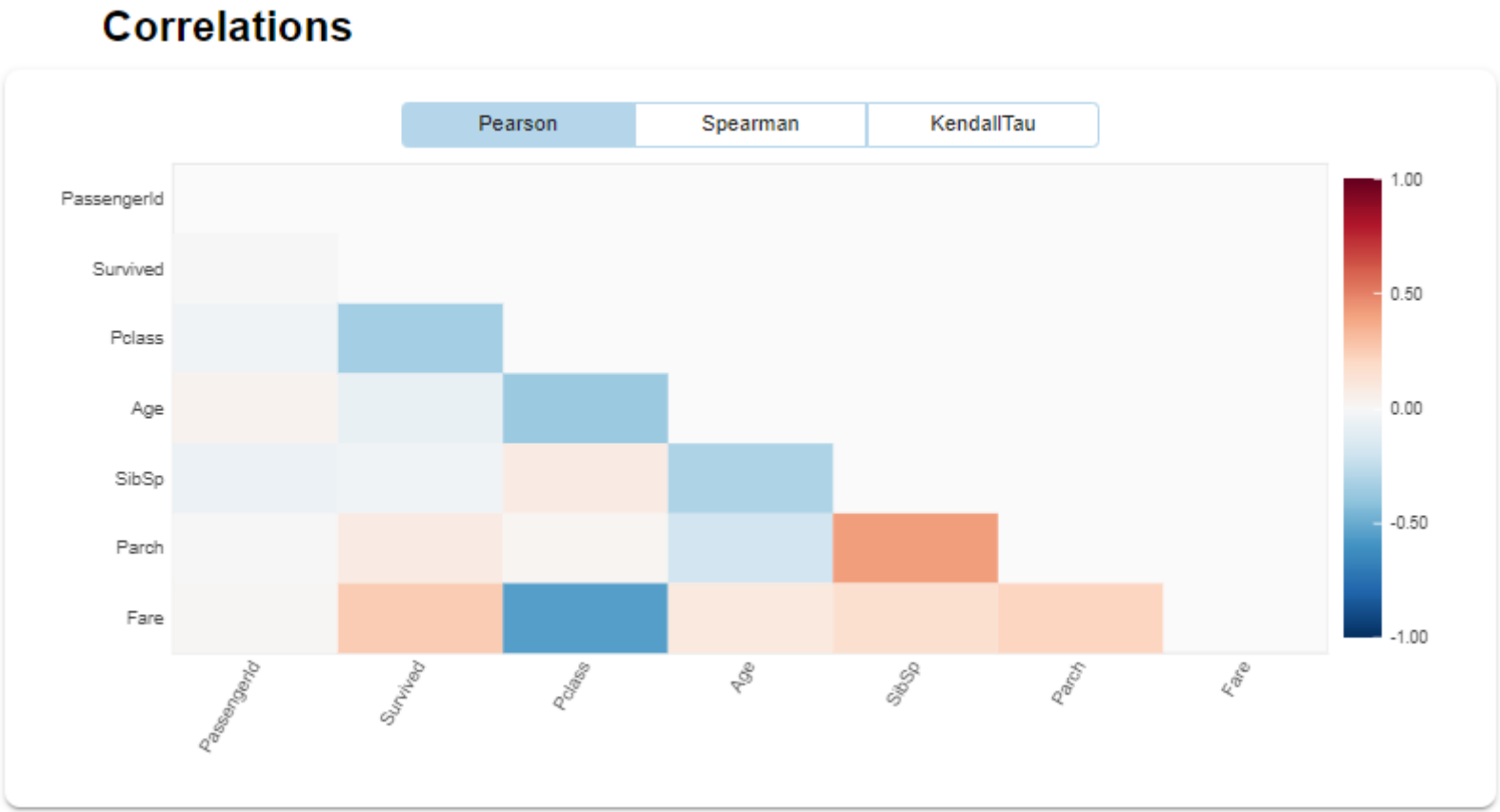
- DataPrep.EDA



# Useful Library for Data Preparation & EDA



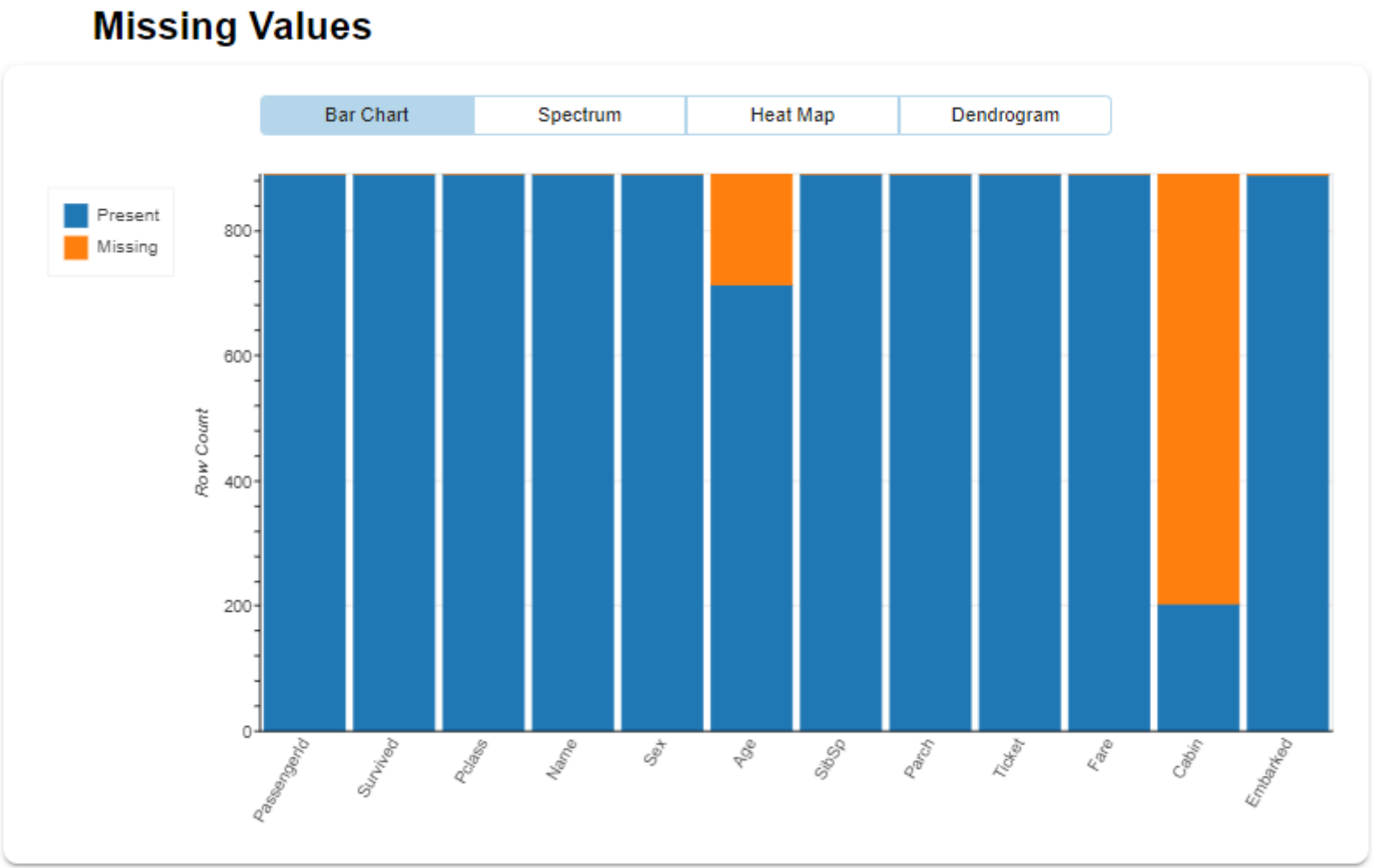
- DataPrep.EDA



# Useful Library for Data Preparation & EDA



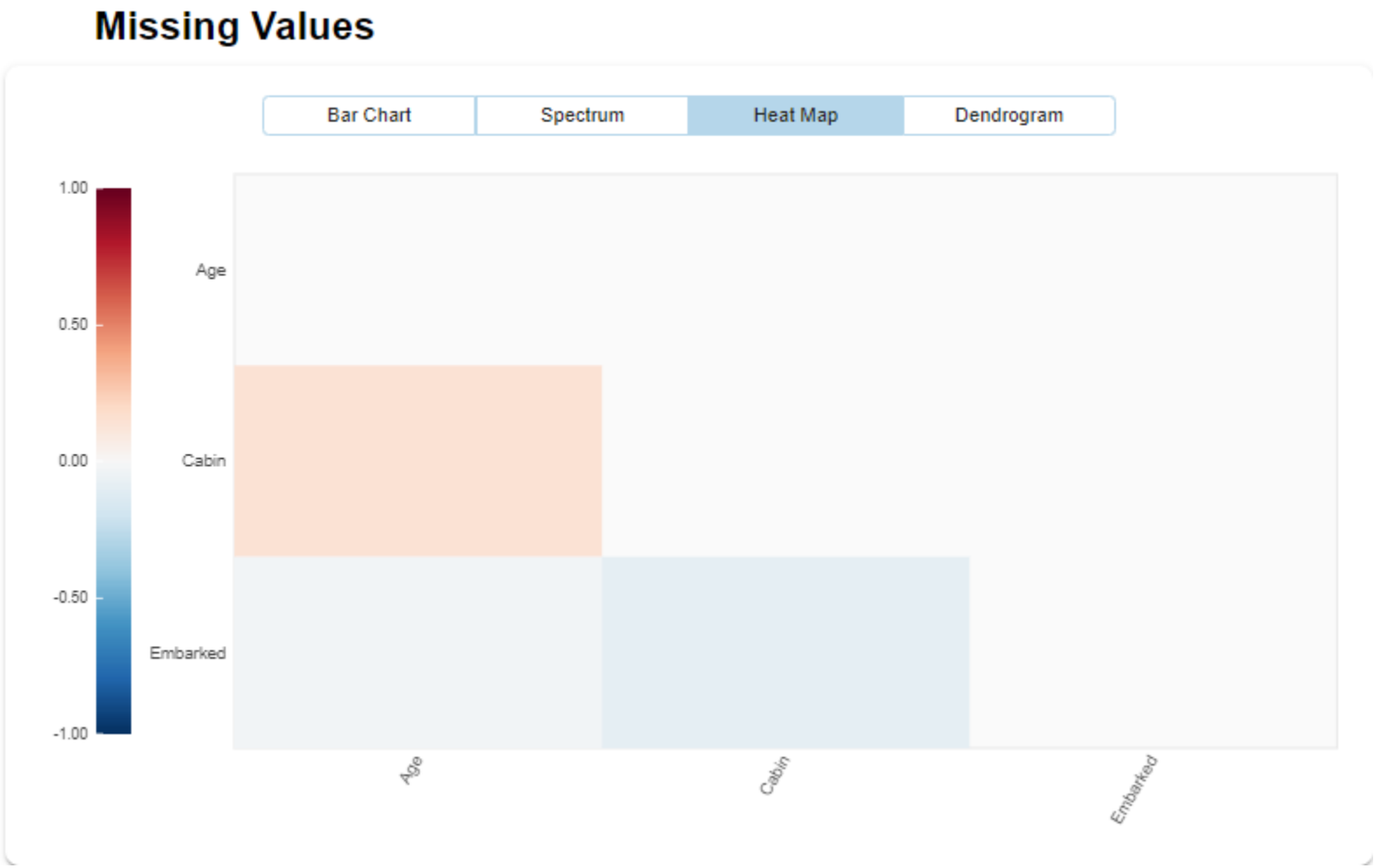
- DataPrep.EDA



# Useful Library for Data Preparation & EDA



- DataPrep.EDA



# Useful Library for Data Preparation & EDA



- DataPrep.Clean
  - DataFrame의 데이터를 정리하고, 유효성을 검사하도록 설계된 약 140개 이상의 함수 포함
  - 편리한 GUI
  - Dask를 사용하여 계산을 병렬 처리하여 빠른 속도로 데이터 정리
  - 정리 중에 발생한 데이터 변경 사항 요약 보고서 생성을 통해 투명성 보장

# Useful Library for Data Preparation & EDA



- DataPrep.Clean

```
from dataprep.clean import *
from dataprep.datasets import load_dataset

df = load_dataset('waste_hauler')
df.head()
```

| index | CREATED | PHONE | EMAIL |
|---|---|---|---|
| 0 | 04/03/2017 | (718) 326-0384 | NaN |
| 1 | 04/03/2017 | (516) 223-2010 | NaN |
| 2 | 03/01/2021 | 646-450-7057 | davidzuidema@aol.com |
| 3 | 04/03/2017 | (718) 356-3936 | NaN |
| 4 | 04/03/2017 | (516) 660-1343 | tom191@gmail.com |

Show 25 per page

Like what you see? Visit the data table notebook to learn more about interactive tables.

1 to 5 of 5 entries Filter

| LOCAL ADDRESS | city |
|---|---|
| 56-01 NURGE AVE., MASPETH, NY | NEW YORK |
| 228 MILLER AVENUE, FREEPORT, NY | New York |
| 1274 49TH STREET SUITE #13G, BROOKLYN, NY | new york |
| 67 EAST FIGUREA AVENUE, STATEN ISLAND, NY | New York |
| 855 E Broadway, Suite 5E, Long Beach, NY | New York |

# Useful Library for Data Preparation & EDA



- DataPrep.Clean

```
df = clean_headers(df)
df.columns
df.head()
```

```
Column Headers Cleaning Report:
```

| index | created | phone | email |
|---|---|---|---|
| 0 | 04/03/2017 | (718) 326-0384 | NaN |
| 1 | 04/03/2017 | (516) 223-2010 | NaN |
| 2 | 03/01/2021 | 646-450-7057 | davidzuidema@aol.com |
| 3 | 04/03/2017 | (718) 356-3936 | NaN |
| 4 | 04/03/2017 | (516) 660-1343 | tom191@gmail.com |

Show 25 per page

Like what you see? Visit the data table notebook to learn more about interactive tables.

1 to 5 of 5 entries Filter

| local_address | city |
|---|---|
| 56-01 NURGE AVE., MASPETH, NY | NEW YORK |
| 228 MILLER AVENUE, FREEPORT, NY | New York |
| 1274 49TH STREET SUITE #13G, BROOKLYN, NY | new york |
| 67 EAST FIGUREA AVENUE, STATEN ISLAND, NY | New York |
| 855 E Broadway, Suite 5E, Long Beach, NY | New York |

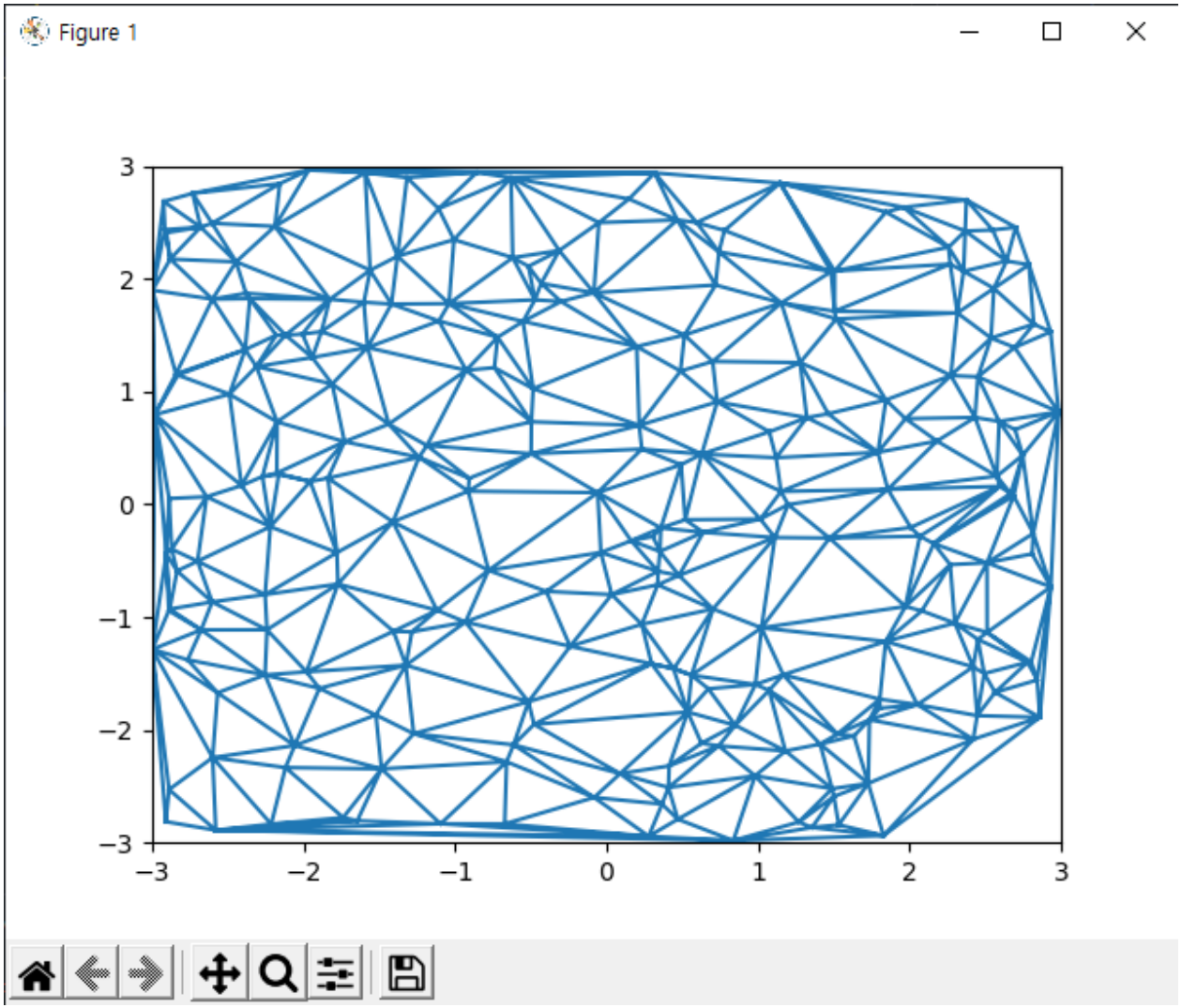
# Useful Library for Data Preparation & EDA



- DataPrep.Connector
  - 적은 줄 수의 코드를 사용해 데이터를 가져와 수십 개의 인기 웹 사이트에서 데이터 가져오기 가능
  - 규모가 큰 결과의 집합 반환이 가능하며, 자동 페이지 매김 수행
  - Web API에 동시 요청을 하여 더 빠르게 결과 가져오기 가능
  - 예시 코드 및 결과 화면
    - https://github.com/sfu-db/dataprep/blob/develop/examples/DataConnector_Youtube.ipynb

# Conclusion & Realization





# Thank you for your Attention!

경상국립대학교

Gyeongsang National University

**Improving lives through learning**

**IDEALAB**