DATA SCIENCE-SSCA3021

Module-II Data Collection and Management

Created by: Mr. Anurag Anand

MODULE-II

2.	**Data collection and management** Introduction, Sources of data, Data collection and APIs, Exploring and fixing data, Data storage and management,Using multiple data sources	07

1. Introduction

This part covers the **basics of data collection and management**, its **importance**, and its **applications** in real-world scenarios.

- **Data Collection**: The process of gathering and measuring information on variables of interest.
- **Data Management**: Involves storing, organizing, maintaining, and ensuring the quality of collected data.
- **Why it's important**:
  - Helps make data-driven decisions
  - Enhances the accuracy of analysis and predictions
  - Improves business processes and scientific research

2. Sources of Data

Data can come from a wide variety of sources. These are typically divided into **primary** and **secondary** sources.

◆ Primary Sources:

Data collected firsthand for a specific purpose.

- Surveys and questionnaires
- Interviews
- Observations
- Experiments
- Sensors and IoT devices

2. Sources of Data

🔷 Secondary Sources:

- Data collected by someone else, used for another purpose.
- Government databases (e.g., Census)
- Public APIs
- Research papers
- Websites and social media
- Open data platforms (e.g., Kaggle, Data.gov)

3. Data Collection and APIs

This section explores how data is collected practically, especially using **APIs** (Application Programming Interfaces).

🔷 **Manual Methods:**

- Surveys
- Paper or digital forms (Example- in .xlsx, .csv)
- Mobile data collection apps

Opening a CSV File

1. Importing pandas

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

2. Opening a local csv file

```
In [34]: df = pd.read_csv('aug_train.csv')
         df
```

Out[34]:

	enrollee_id	city	city_development_index	gender	relevent_experience	enrolled_university	education_level	major_discipline	experience	company_s
0	8949	city_103	0.920	Male	Has relevent experience	no_enrollment	Graduate	STEM	>20	N
1	29725	city_40	0.776	Male	No relevent experience	no_enrollment	Graduate	STEM	15	50-
2	11561	city_21	0.624	NaN	No relevent experience	Full time course	Graduate	STEM	5	N
3	33241	city_115	0.789	NaN	No relevent experience	NaN	Graduate	Business Degree	<1	N

Opening a CSV file from URl

```
import requests
from io import StringIO

url = ""
headers = {"User-Agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10.14; rv:66.0) Gecko/20100101 Firefox/66.0"}
req = requests.get(url, headers=headers)
data = StringIO(req.text)

pd.read_csv(data)
```

Opening a JSON file from URl

jupyter working with JSON Last Checkpoint: Yesterday at 10:05 AM (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Run Code

```
In [11]: import pandas as pd
```

Woking with JSON

```
In [12]: pd.read_json('train.json')
```

Out[12]:

	id	cuisine	ingredients
0	10259	greek	[romaine lettuce, black olives, grape tomatoes...
1	25693	southern_us	[plain flour, ground pepper, salt, tomatoes, g...
2	20130	filipino	[eggs, pepper, salt, mayonaise, cooking oil, g...
3	22213	indian	[water, vegetable oil, wheat, salt]
4	13162	indian	[black pepper, shallots, cornflour, cayenne pe...
...	...	...	...
39769	29109	irish	[light brown sugar, granulated sugar, butter, ...
39770	11462	italian	[KRAFT Zesty Italian Dressing, purple onion, b...
39771	2238	irish	[eggs, citrus fruit, raisins, sourdough starte...
39772	41882	chinese	[boneless chicken skinless thigh, minced garli...
39773	2362	mexican	[green chile, jalapeno chilies, onions, ground...

39774 rows × 3 columns

Working with SQL

Working with SQL

```
In [17]: !pip install mysql.connector
```

```
Processing c:\users\91842\appdata\local\pip\cache\wheels\57\e4\98\5feafb5c393dd2540e44b064a
ector-2.2.9-cp38-cp38-win_amd64.whl
Installing collected packages: mysql.connector
Successfully installed mysql.connector
```

```
In [18]: import mysql.connector
```

```
In [20]: conn = mysql.connector.connect(host='localhost',user='root',password='',database='world')
```

```
In [27]: df = pd.read_sql_query("SELECT * FROM countrylanguage",conn)
```

```
In [28]: df
```

Out[28]:

	CountryCode	Language	IsOfficial	Percentage
0	ABW	Dutch	T	5.3
1	ABW	English	F	9.5
2	ABW	Papiamento	F	76.7
3	ABW	Spanish	F	7.4
4	AFG	Balochi	F	0.9
...	...	...	...	...
979	ZMB	Tongan	F	11.0
980	ZWE	English	T	2.2
981	ZWE	Ndebele	F	16.2
982	ZWE	Nyanja	F	2.2
983	ZWE	Shona	F	72.1

984 rows × 4 columns

3. Data Collection and APIs

🔷 Automated Methods:

- **Web scraping**: Extracting data from websites
- **APIs**: Most modern applications provide APIs to access their data

🔷 API Example:

Using Python's requests library to call a weather API:

APIs are reliable, structured, and faster than scraping data.

3. Data Collection and APIs

jupyter API TO DATAFRAME Last Checkpoint: Last Thursday at 10:05 AM (autosaved)

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

Run Code

In [1]:
```
import pandas as pd
import requests
```

In [6]:
```
response = requests.get('https://api.themoviedb.org/3/movie/top_rated?api_key=8265bd1679663a7ea12ac168da84d2e8&language=en-US&pag
```

In [7]:
```
temp_df = pd.DataFrame(response.json()['results'])[['id','title','overview','release_date','popularity','vote_average','vote_cour
```

In [34]:
```
df.head()
```

Out[34]:

	id	title	overview	release_date	popularity	vote_average	vote_count
0	19404	Dilwale Dulhania Le Jayenge	Raj is a rich, carefree, happy-go-lucky second...	1995-10-20	18.433	8.7	2763
1	724089	Gabriel's Inferno Part II	Professor Gabriel Emerson finally learns the t...	2020-07-31	8.439	8.7	1223
2	278	The Shawshank Redemption	Framed in the 1940s for the double murder of h...	1994-09-23	65.570	8.7	18637
3	238	The Godfather	Spanning the years 1945 to 1955, a chronicle o...	1972-03-14	63.277	8.7	14052
4	761053	Gabriel's Inferno Part III	The final part of the film adaption of the ero...	2020-11-19	26.691	8.7	773

3. Data Collection and APIs

In [19]:
```
for i in range(1,429):
    response = requests.get('https://api.themoviedb.org/3/movie/top_rated?api_key=8265bd1679663a7ea12ac168da84d2e8&language=en-US
    temp_df = pd.DataFrame(response.json()['results'])[['id','title','overview','release_date','popularity','vote_average','vote_
    df = df.append(temp_df,ignore_index=True)
```

In [21]:
```
df
```

Out[21]:

	id	title	overview	release_date	popularity	vote_average	vote_count
0	19404	Dilwale Dulhania Le Jayenge	Raj is a rich, carefree, happy-go-lucky second...	1995-10-20	18.433	8.7	2763
1	724089	Gabriel's Inferno Part II	Professor Gabriel Emerson finally learns the t...	2020-07-31	8.439	8.7	1223
2	278	The Shawshank Redemption	Framed in the 1940s for the double murder of h...	1994-09-23	65.570	8.7	18637
3	238	The Godfather	Spanning the years 1945 to 1955, a chronicle o...	1972-03-14	63.277	8.7	14052
4	761053	Gabriel's Inferno Part III	The final part of the film adaption of the ero...	2020-11-19	26.691	8.7	773
...	...	...	...	...	...	...	...
8546	13805	Disaster Movie	The filmmaking team behind the hits "Scary Mov...	2008-08-29	14.630	3.2	714
8547	5491	Battlefield Earth	In the year 3000, man is no match for the Psyc...	2000-05-12	10.647	3.0	543
8548	14164	Dragonball Evolution	The young warrior Son Goku sets out on a quest...	2009-03-12	32.244	2.8	1447
8549	11059	House of the Dead	Set on an island off the coast, a techno rave ...	2003-04-11	14.502	2.8	238
8550	40016	Birdemic: Shock and Terror	A platoon of eagles and vultures attacks the r...	2010-02-27	9.824	2.2	215

8551 rows × 7 columns

3. Data Collection and APIs

Out[18]: —

In [19]:
```
):
sts.get('https://api.themoviedb.org/3/movie/top_rated?api_key=8265bd1679663a7ea12ac168da84d2e8&language=en-US&page={}'.format(i))
aFrame(response.json()['results'])[['id','title','overview','release_date','popularity','vote_average','vote_count']]
emp_df,ignore_index=True)
```

In [21]:
```
df
```

Out[21]:

	id	title	overview	release_date	popularity	vote_average	vote_count
0	19404	Dilwale Dulhania Le Jayenge	Raj is a rich, carefree, happy-go-lucky second...	1995-10-20	18.433	8.7	2763
1	724089	Gabriel's Inferno Part II	Professor Gabriel Emerson finally learns the t...	2020-07-31	8.439	8.7	1223
2	278	The Shawshank Redemption	Framed in the 1940s for the double murder of h...	1994-09-23	65.570	8.7	18637
3	238	The Godfather	Spanning the years 1945 to 1955, a chronicle o...	1972-03-14	63.277	8.7	14052
4	761053	Gabriel's Inferno Part III	The final part of the film adaption of the ero...	2020-11-19	26.691	8.7	773
...	...	...	...	...	...	...	...
8546	13805	Disaster Movie	The filmmaking team behind the hits "Scary Mov...	2008-08-29	14.630	3.2	714
8547	5491	Battlefield Earth	In the year 3000, man is no match for the Psyc...	2000-05-12	10.647	3.0	543
8548	14164	Dragonball Evolution	The young warrior Son Goku sets out on a quest...	2009-03-12	32.244	2.8	1447
8549	11059	House of the Dead	Set on an island off the coast, a techno rave ...	2003-04-11	14.502	2.8	238
8550	40016	Birdemic: Shock and Terror	A platoon of eagles and vultures attacks the r...	2010-02-27	9.824	2.2	215

8551 rows × 7 columns

4. Exploring

Once data is collected, it's often **messy or incomplete**. This step involves understanding and cleaning the data.

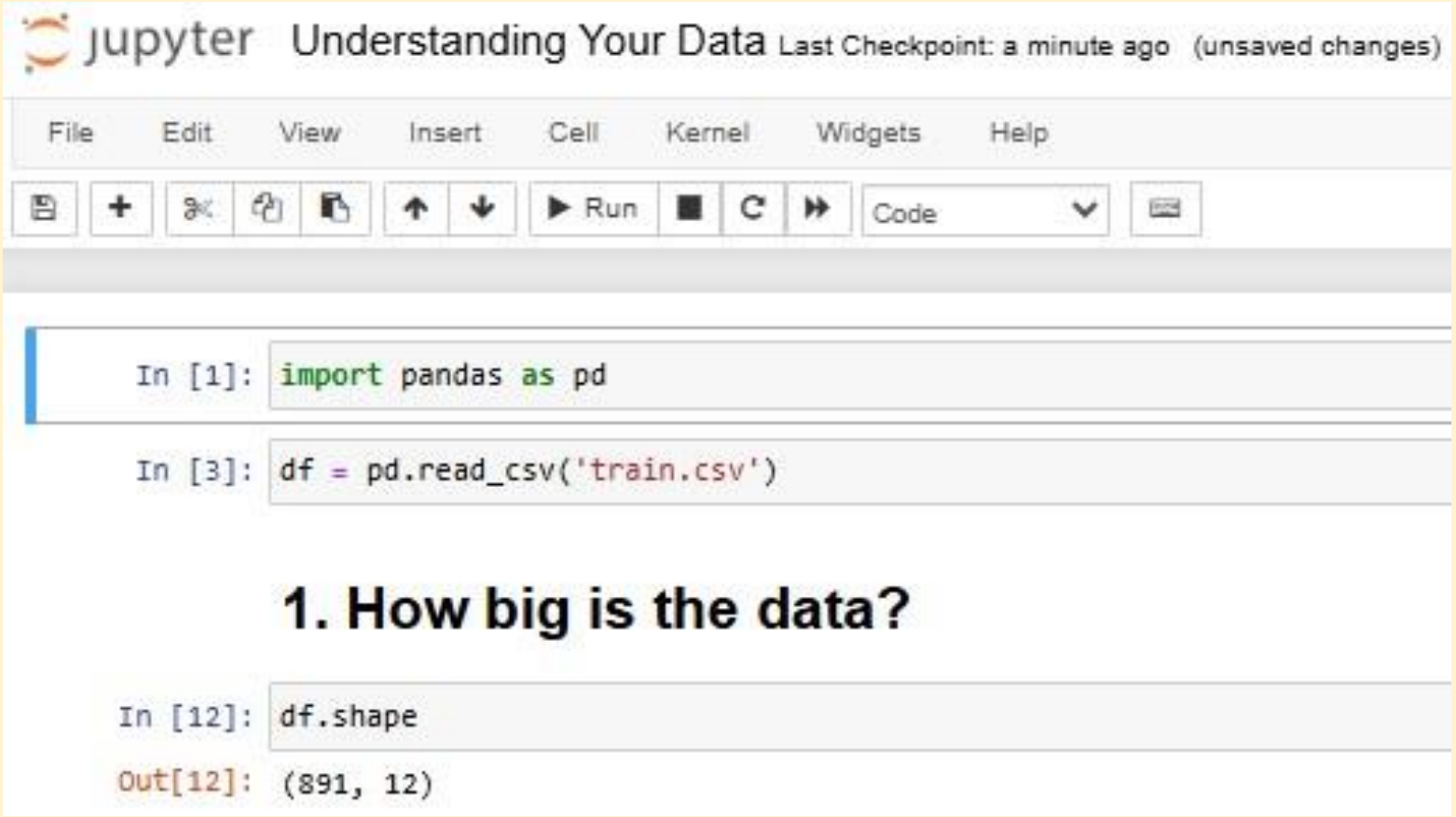
🔷 **Data Exploration:**

•Summary statistics (mean, median, mode)
•Visualizations (histograms, scatter plots)
•Understanding distributions and correlations

🔷 **Tools Used:**

•**Pandas** in Python
•**Excel**
•**Power BI / Tableau**

Exploring the data



Exploring the data

2. How does the data look like?

```
In [14]: df.sample(5)
```

Out[14]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
498	499	0	1	Allison, Mrs. Hudson J C (Bessie Waldo Daniels)	female	25.0	1	2	113781	151.5500	C22 C26	S
125	126	1	3	Nicola-Yarred, Master. Elias	male	12.0	1	0	2651	11.2417	NaN	C
604	605	1	1	Homer, Mr. Harry ("Mr E Haven")	male	35.0	0	0	111426	26.5500	NaN	C
751	752	1	3	Moor, Master. Meier	male	6.0	0	1	392096	12.4750	E121	S
172	173	1	3	Johnson, Miss. Eleanor Ileen	female	1.0	1	1	347742	11.1333	NaN	S

Exploring the data

3. What is the data type of cols?

```
In [15]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64
 7   Parch        891 non-null    int64
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object
 11  Embarked     889 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

Exploring the data

4. Are there any missing values?

```
In [16]: df.isnull().sum()

Out[16]: PassengerId      0
         Survived         0
         Pclass           0
         Name             0
         Sex              0
         Age            177
         SibSp            0
         Parch            0
         Ticket           0
         Fare             0
         Cabin          687
         Embarked         2
         dtype: int64
```

Exploring the data

5. How does the data look mathematically?

```
In [17]: df.describe()
```

Out[17]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

Exploring the data

6. Are there duplicate values?

```
In [18]: df.duplicated().sum()
Out[18]: 0
```

Exploring the data

7. How is the correlation between cols?

```
In [20]: df.corr()['Survived']

Out[20]: PassengerId    -0.005007
         Survived        1.000000
         Pclass         -0.338481
         Age            -0.077221
         SibSp          -0.035322
         Parch           0.081629
         Fare            0.257307
         Name: Survived, dtype: float64
```

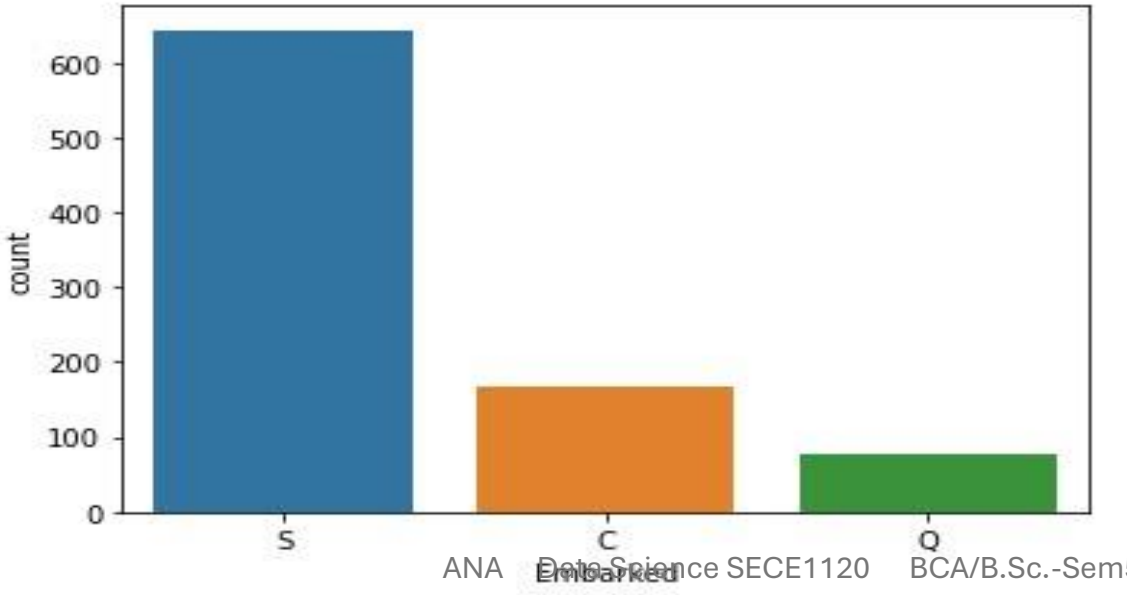
Exploring the data

1. Categorical Data

a. Countplot

In [12]:
```
sns.countplot(df['Embarked'])
#df['Survived'].value_counts().plot(kind='bar')
```

Out[12]: <matplotlib.axes._subplots.AxesSubplot at 0x1cc48b021f0>

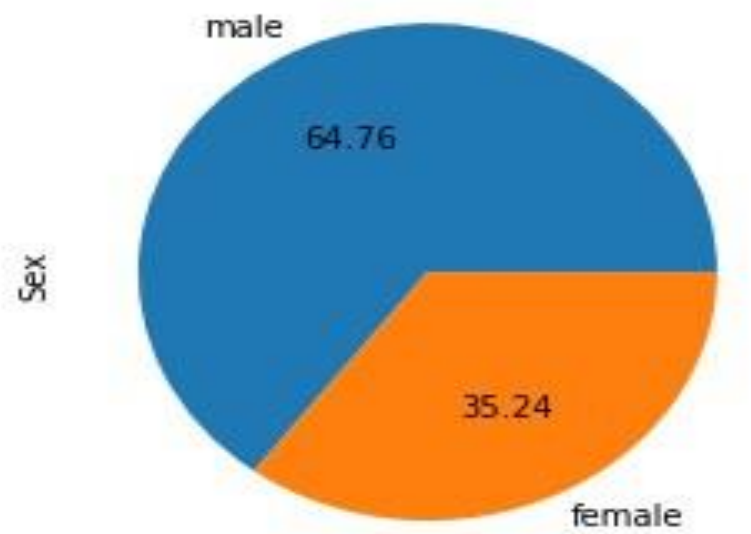


Exploring the data

b. PieChart

```
In [16]: df['Sex'].value_counts().plot(kind='pie',autopct='%.2f')
```

```
Out[16]: <matplotlib.axes._subplots.AxesSubplot at 0x1cc48b142e0>
```



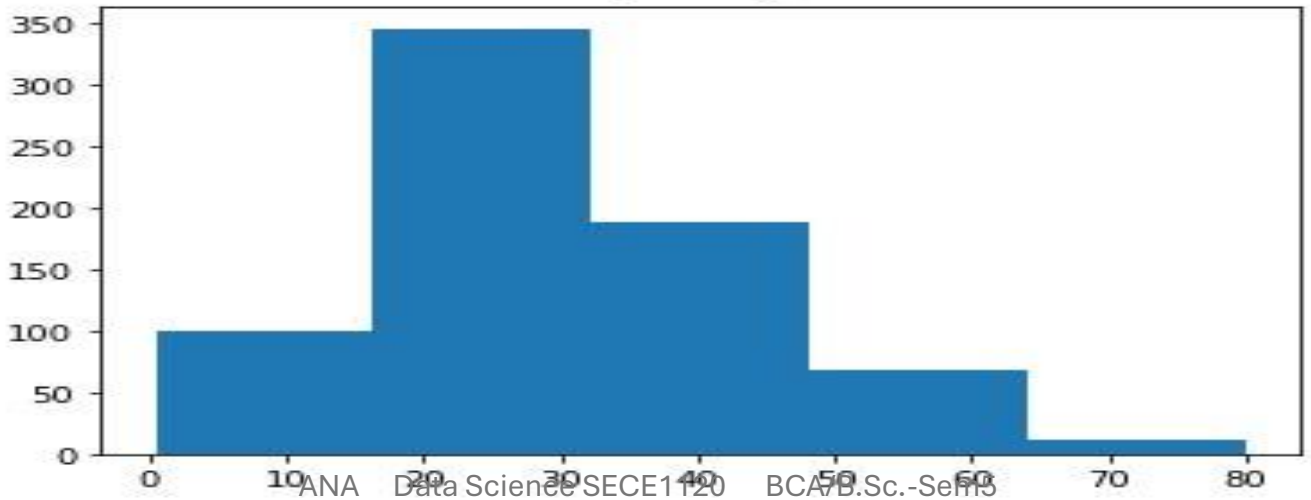
Exploring the data

2. Numerical Data

a. Histogram

In [25]:
```
import matplotlib.pyplot as plt
plt.hist(df['Age'],bins=5)
```

Out[25]:
```
(array([100., 346., 188.,  69.,  11.]),
 array([ 0.42 , 16.336, 32.252, 48.168, 64.084, 80.   ]),
 <a list of 5 Patch objects>)
```

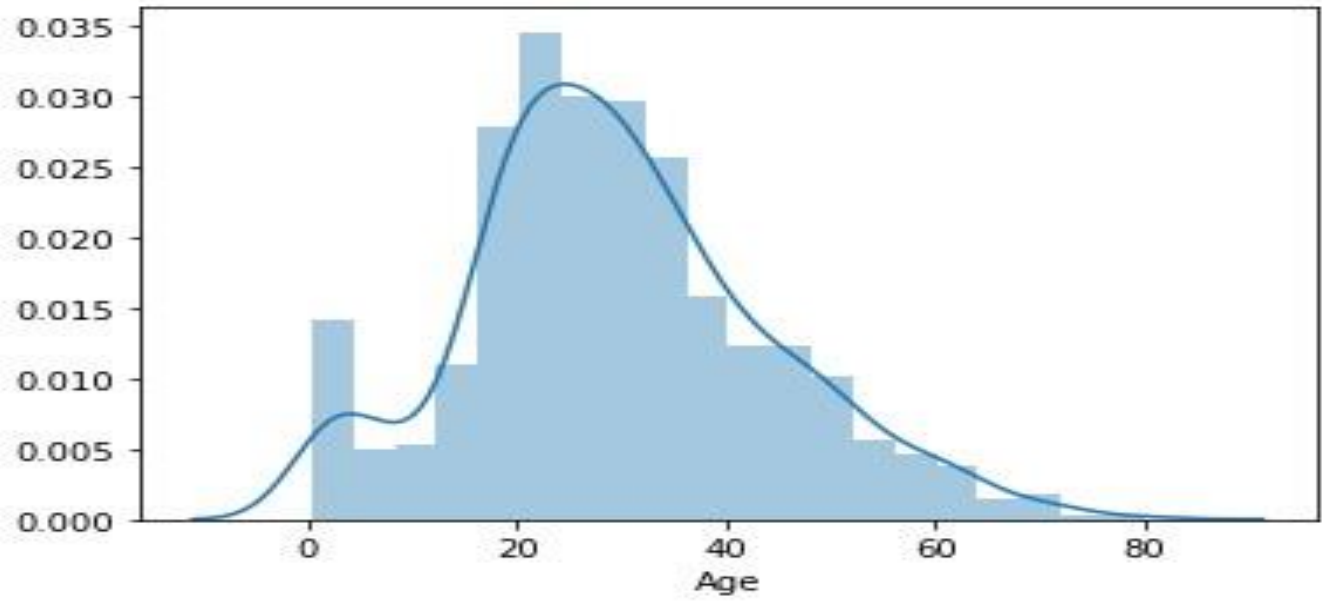


Exploring the data

b. Distplot

```
In [26]: sns.distplot(df['Age'])
```

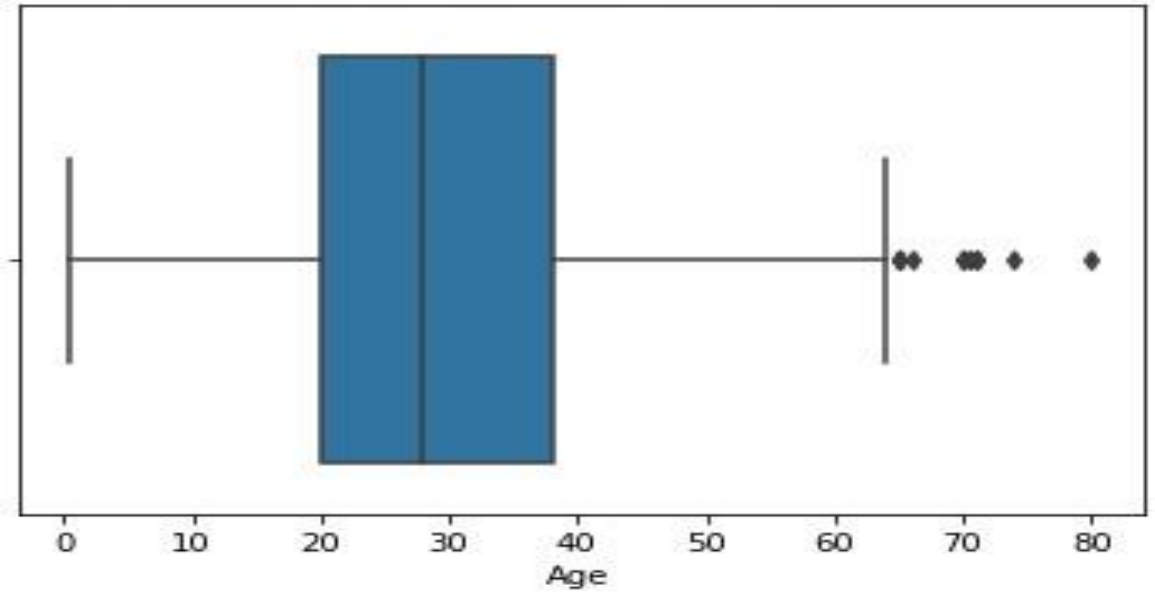
```
Out[26]: <matplotlib.axes._subplots.AxesSubplot at 0x1cc4914c4f0>
```



Exploring the data

c. Boxplot

```
In [28]: sns.boxplot(df['Age'])

Out[28]: <matplotlib.axes._subplots.AxesSubplot at 0x1cc48ee1520>
```



Exploring the data

```
In [29]: df['Age'].min()

Out[29]: 0.42

In [30]: df['Age'].max()

Out[30]: 80.0

In [31]: df['Age'].mean()

Out[31]: 29.69911764705882

In [32]: df['Age'].skew()

Out[32]: 0.3891077823008270 4

In [ ]:
```

Fixing the Data

◆ Data Cleaning:

- Removing missing or duplicate values
- Handling nulls (NaN)
- Imputing the missing data
- Correcting incorrect data types (e.g., date stored as text)
- Removing outliers

Removing the Missing Data

In [34]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [35]:
```
df = pd.read_csv('data_science_job.csv')
```

In [36]:
```
df.head()
```

Out[36]:

	enrollee_id	city	city_development_index	gender	relevent_experience	enrolled_university	education_level	major_discipline
0	8949	city_103	0.920	Male	Has relevent experience	no_enrollment	Graduate	STEM
1	29725	city_40	0.776	Male	No relevent experience	no_enrollment	Graduate	STEM
2	11561	city_21	0.624	NaN	No relevent experience	Full time course	Graduate	STEM
3	33241	city_115	0.789	NaN	No relevent experience	NaN	Graduate	Business Degree
4	666	city_162	0.767	Male	Has relevent experience	no_enrollment	Masters	STEM

Removing the Missing Data

```
In [37]: df.isnull().mean()*100

Out[37]: enrollee_id              0.000000
         city                     0.000000
         city_development_index   2.500261
         gender                  23.530640
         relevent_experience      0.000000
         enrolled_university      2.014824
         education_level          2.401086
         major_discipline        14.683161
         experience               0.339284
         company_size            30.994885
         company_type            32.049274
         training_hours           3.998330
         target                   0.000000
         dtype: float64
```

Removing the Missing Data

Check for the Columns with missing values 0 to 5%

In [38]:
```
df.shape
```

Out[38]:
```
(19158, 13)
```

In [39]:
```
cols = [var for var in df.columns if df[var].isnull().mean() < 0.05 and df[var].isnull().mean() > 0]
cols
```

Out[39]:
```
['city_development_index',
 'enrolled_university',
 'education_level',
 'experience',
 'training_hours']
```

In [40]:
```
df[cols].sample(5)
```

Removing the Missing Data

In [40]:
```
df[cols].sample(5)
```

Out[40]:

	city_development_index	enrolled_university	education_level	experience	training_hours
15056	0.939	no_enrollment	Graduate	7.0	7.0
11474	0.887	no_enrollment	Masters	5.0	107.0
7940	0.920	no_enrollment	Graduate	7.0	11.0
11338	0.689	NaN	High School	3.0	99.0
552	0.698	no_enrollment	Graduate	14.0	24.0

Removing the Missing Data

Check for the value counts for the colum ‘education_level’.

```
In [50]: df['education_level'].value_counts()

Out[50]: Graduate          11598
         Masters            4361
         High School        2017
         Phd                 414
         Primary School      308
         Name: education_level, dtype: int64
```

Removing the Missing Data

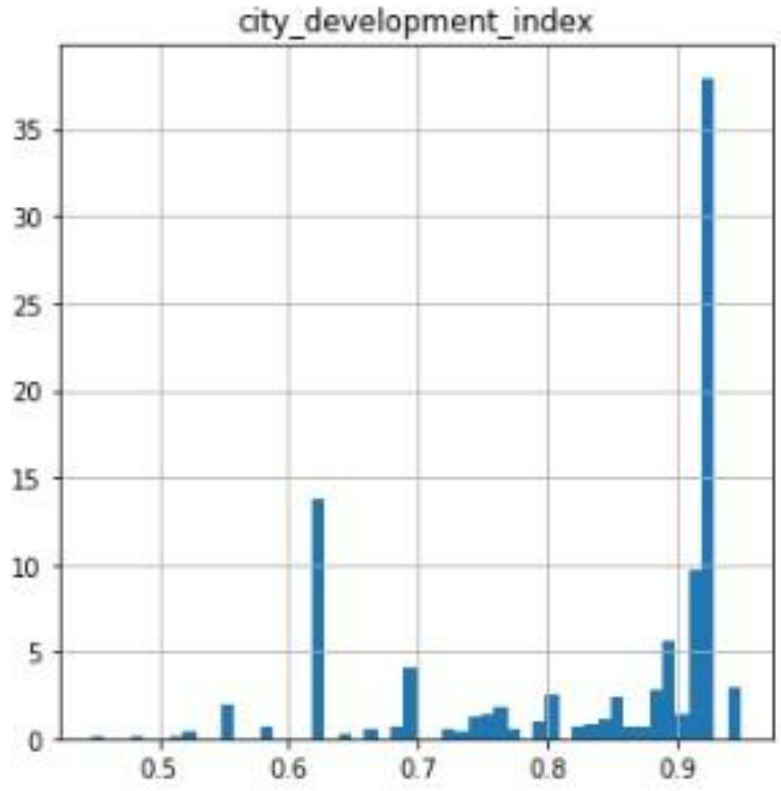
Creating a new_df after dropping the null or Nan Values

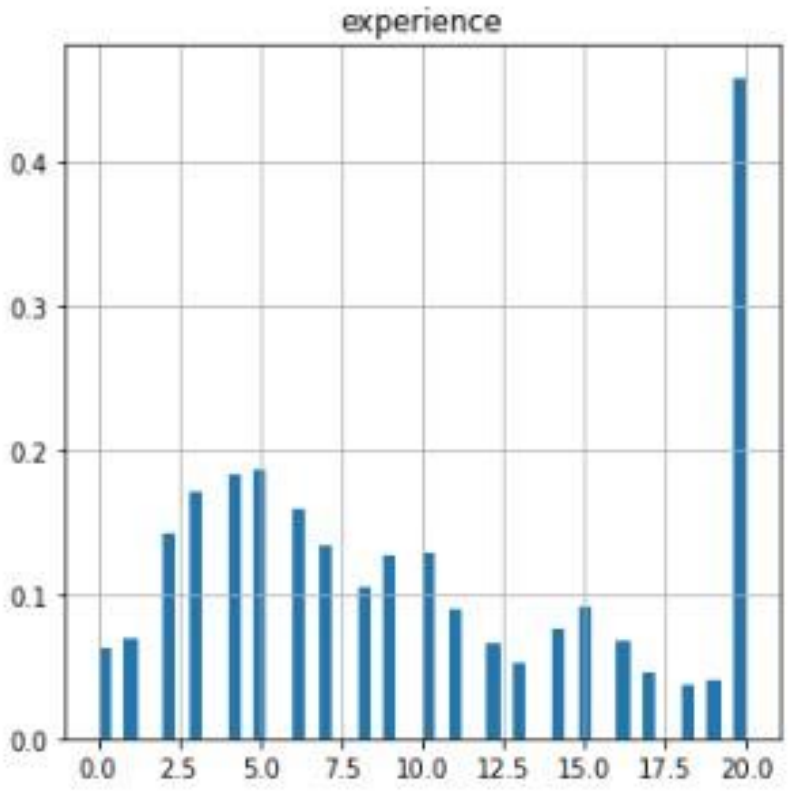
```
In [13]: len(df[cols].dropna()) / len(df)

Out[13]: 0.8968577095730244

In [41]: new_df = df[cols].dropna()
         df.shape, new_df.shape

Out[41]: ((19158, 13), (17182, 5))
```

Removing the Missing Data

Histogram plot for the new dataframe

In [42]:
```
new_df.hist(bins=50, density=True, figsize=(12, 12))
plt.show()
```



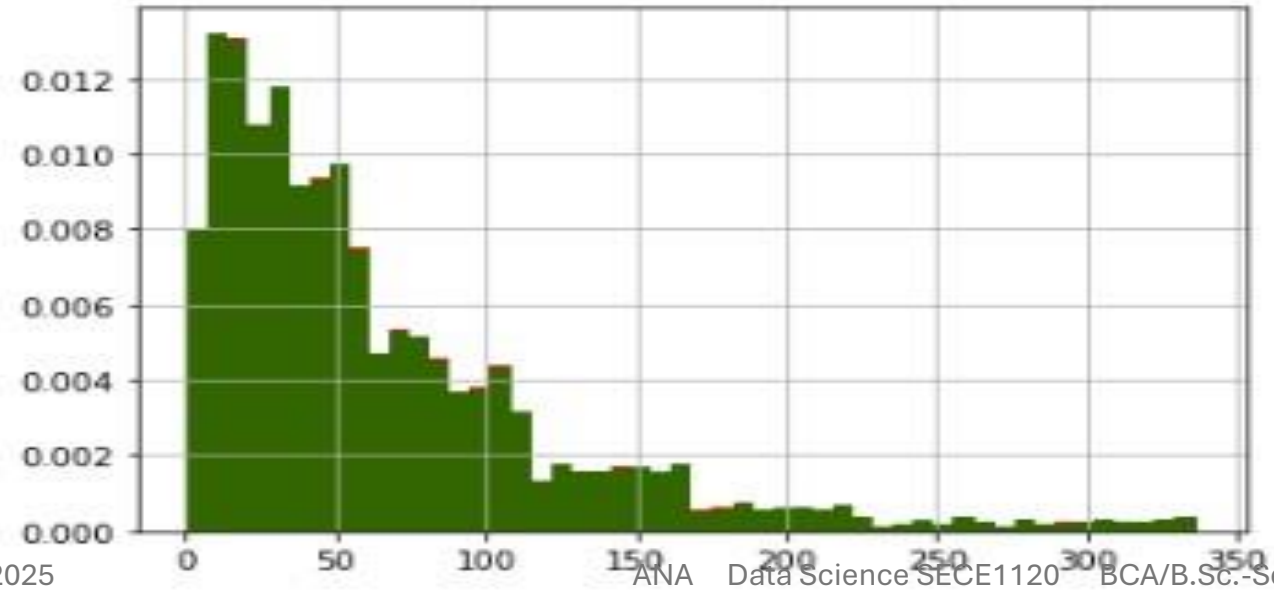


Removing the Missing Data

Overlapping the histogram plot of new_df with the df

In [43]:
```
fig = plt.figure()
ax = fig.add_subplot(111)

# original data
df['training_hours'].hist(bins=50, ax=ax, density=True, color='red')

# data after cca, the argument alpha makes the color transparent, so we can
# see the overlay of the 2 distributions
new_df['training_hours'].hist(bins=50, ax=ax, color='green', density=True, alpha=0.8)
```

Out[43]: <matplotlib.axes._subplots.AxesSubplot at 0x2dfc2344dc0>

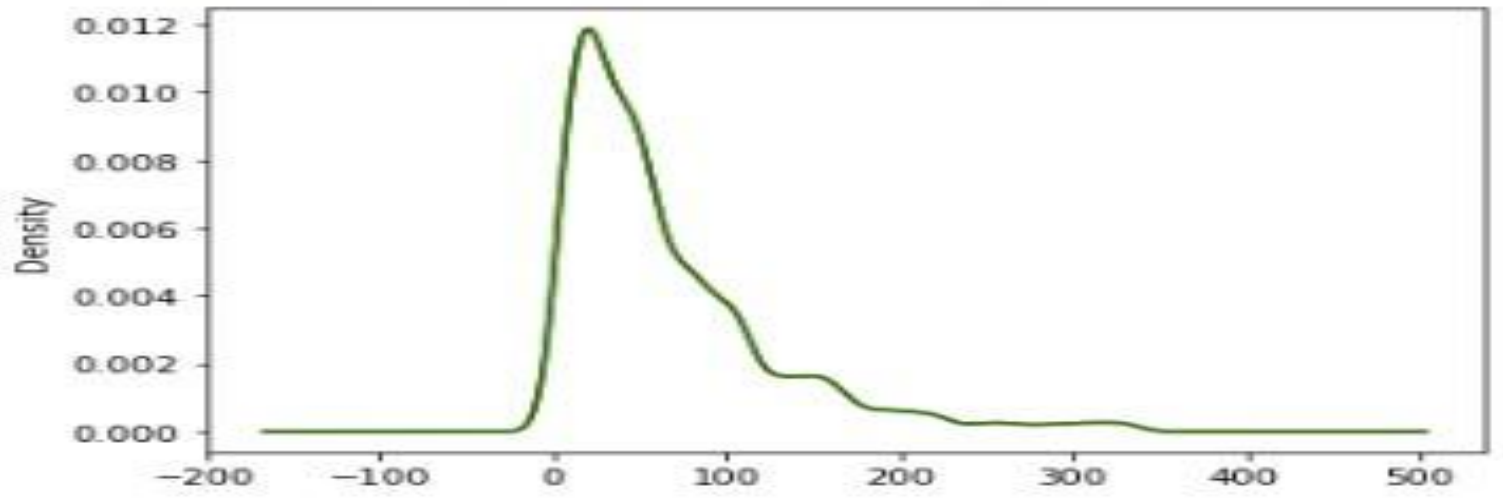


Removing the Missing Data

Overlapping the density plot of new_df with the df

In [44]:
```
fig = plt.figure()
ax = fig.add_subplot(111)

# original data
df['training_hours'].plot.density(color='red')

# data after cca
new_df['training_hours'].plot.density(color='green')
```

Out[44]: <matplotlib.axes._subplots.AxesSubplot at 0x2dfc24f13d0>

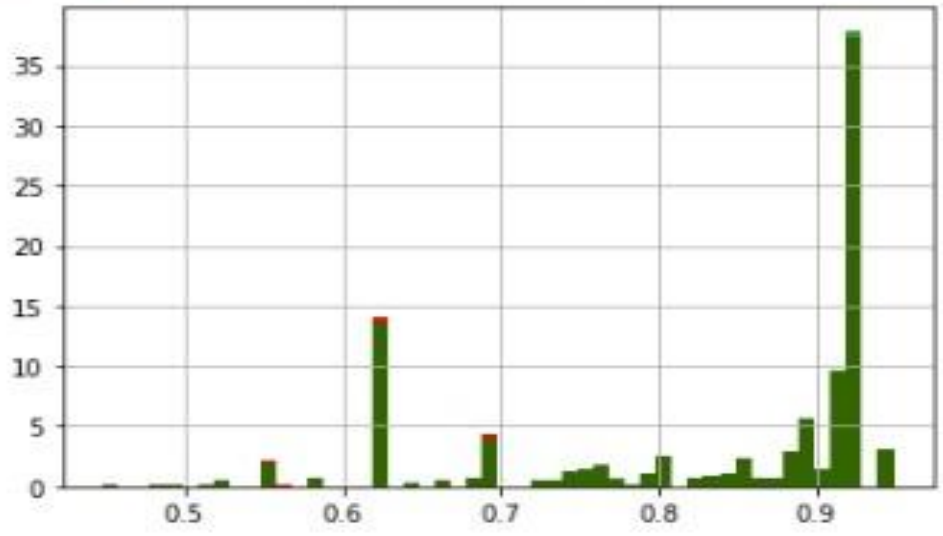


Removing the Missing Data

Overlapping the Histogram plot of new_df with the df

In [45]:
```
fig = plt.figure()
ax = fig.add_subplot(111)

# original data
df['city_development_index'].hist(bins=50, ax=ax, density=True, color='red')

# data after cca, the argument alpha makes the color transparent, so we can
# see the overlay of the 2 distributions
new_df['city_development_index'].hist(bins=50, ax=ax, color='green', density=True, alpha=0.8)
```

Out[45]: <matplotlib.axes._subplots.AxesSubplot at 0x2dfc218dca0>

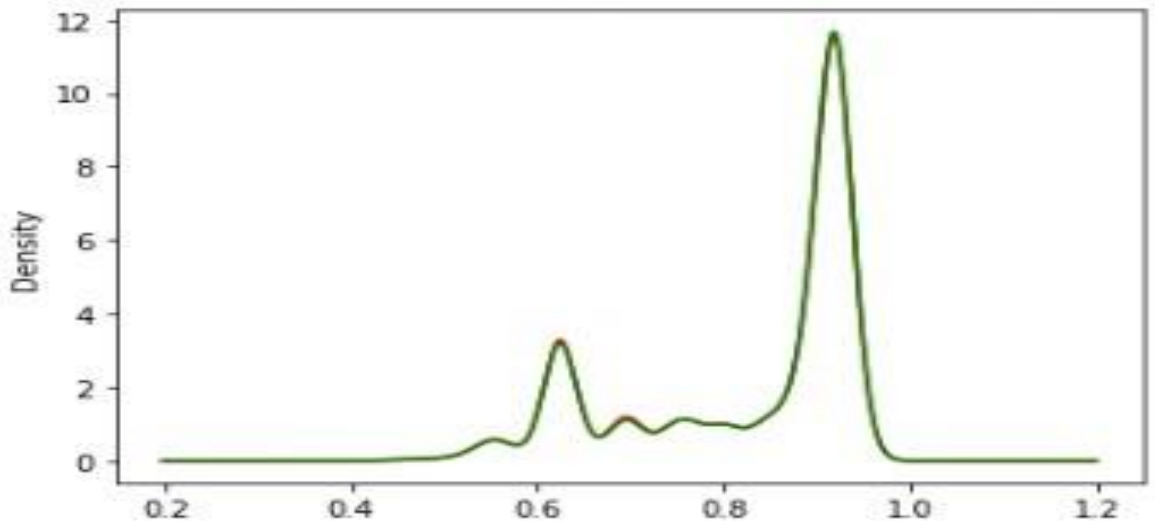


Removing the Missing Data

Overlapping the Density plot of new_df with the df

In [46]:
```
fig = plt.figure()
ax = fig.add_subplot(111)

# original data
df['city_development_index'].plot.density(color='red')

# data after cca
new_df['city_development_index'].plot.density(color='green')
```

Out[46]: <matplotlib.axes._subplots.AxesSubplot at 0x2dfc218dd90>

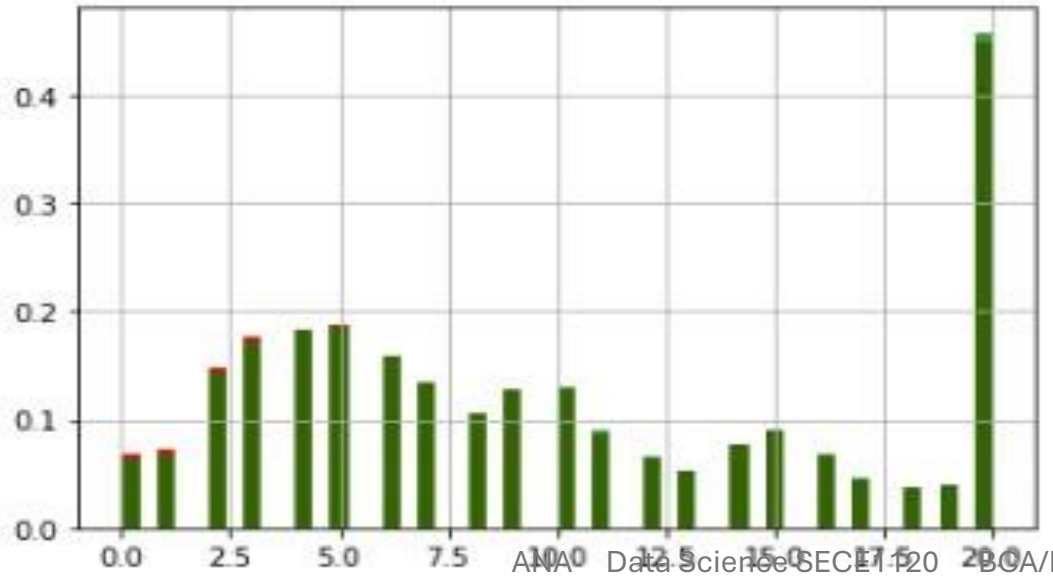


Removing the Missing Data

Overlapping the Histogram plot of new_df with the df

In [47]:
```
fig = plt.figure()
ax = fig.add_subplot(111)

# original data
df['experience'].hist(bins=50, ax=ax, density=True, color='red')

# data after cca, the argument alpha makes the color transparent, so we can
# see the overlay of the 2 distributions
new_df['experience'].hist(bins=50, ax=ax, color='green', density=True, alpha=0.8)
```

Out[47]: <matplotlib.axes._subplots.AxesSubplot at 0x2dfc23c38e0>

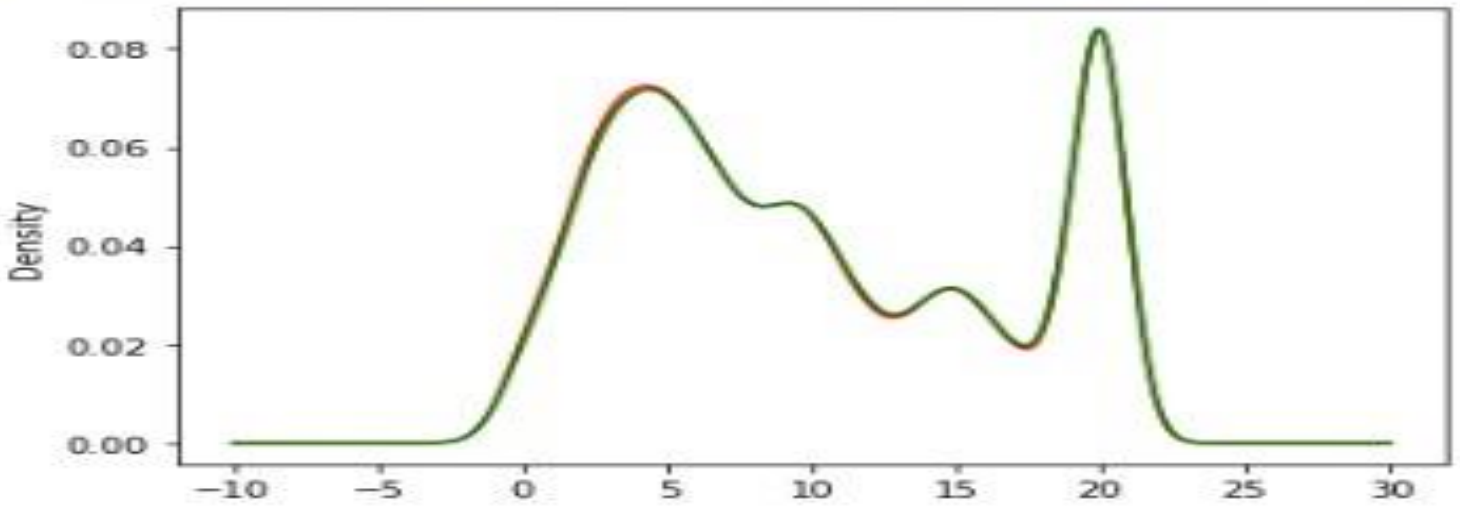


Removing the Missing Data

Overlapping the Density plot of new_df with the df

```
In [48]: fig = plt.figure()
         ax = fig.add_subplot(111)

         # original data
         df['experience'].plot.density(color='red')

         # data after cca
         new_df['experience'].plot.density(color='green')
```

```
Out[48]: <matplotlib.axes._subplots.AxesSubplot at 0x2dfc29474c0>
```



Removing the Missing Data

Checking the value counts for Categorical Data

In [51]:

```
temp = pd.concat([
            # percentage of observations per category, original data
            df['enrolled_university'].value_counts() / len(df),

            # percentage of observations per category, cca data
            new_df['enrolled_university'].value_counts() / len(new_df)
        ],
        axis=1)

# add column names
temp.columns = ['original', 'cca']

temp
```

Out[51]:

	original	cca
no_enrollment	0.721213	0.735188
Full time course	0.196106	0.200733
Part time course	0.062533	0.064079

Removing the Missing Data

Checking the value counts for Categorical Data

In [52]:
```
temp = pd.concat([
            # percentage of observations per category, original data
            df['education_level'].value_counts() / len(df),

            # percentage of observations per category, cca data
            new_df['education_level'].value_counts() / len(new_df)
        ],
        axis=1)

# add column names
temp.columns = ['original', 'cca']

temp
```

Out[52]:

	original	cca
Graduate	0.605387	0.619835
Masters	0.227633	0.234082
High School	0.105282	0.107380
Phd	0.021610	0.022116
Primary School	0.016077	0.016587