Welcome to ML | Day 10

Import Libraries

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Import Dataset

In [2]:
```
df = pd.read_csv('data_science_job.csv')
```

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

Out[3]:

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

Finding Missing Data Column Wise

In [4]:
```
df.isnull().mean()*100
```

Out[4]:
```
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
```

In [5]:
```
df.shape
```

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

Selected column name (Below 5%)

In [6]:
```
cols = [var for var in df.columns if df[var].isnull().mean() < 0.05 and df[var].isnul
cols
```

Out[6]:
```
['city_development_index',
 'enrolled_university',
 'education_level',
 'experience',
 'training_hours']
```

In [7]:
```
df[cols].sample(5)
```

Out[7]:

	city_development_index	enrolled_university	education_level	experience	training_hours
9468	0.624	no_enrollment	Graduate	7.0	25.0
5762	0.920	Part time course	High School	2.0	47.0
6185	0.920	no_enrollment	Graduate	1.0	8.0
18266	NaN	no_enrollment	Masters	20.0	33.0
10011	0.920	no_enrollment	Masters	2.0	80.0

Calculated drop rows

In [8]:
```
len(df[cols].dropna()) / len(df)
```

Out[8]:
```
0.8968577095730244
```

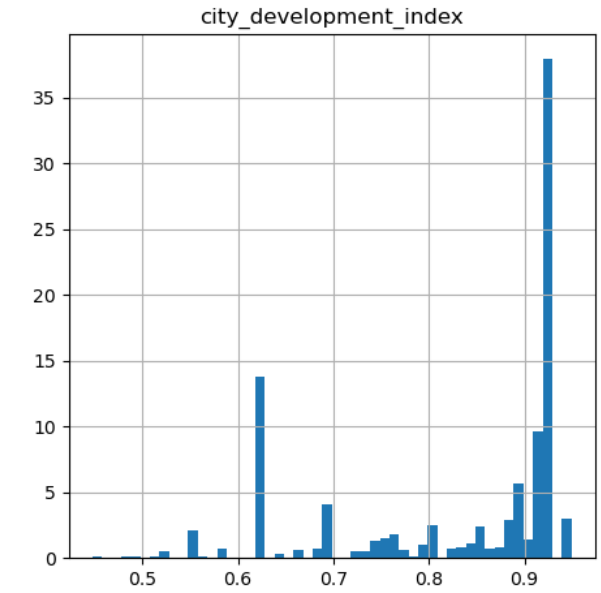
Create: New Data Frame

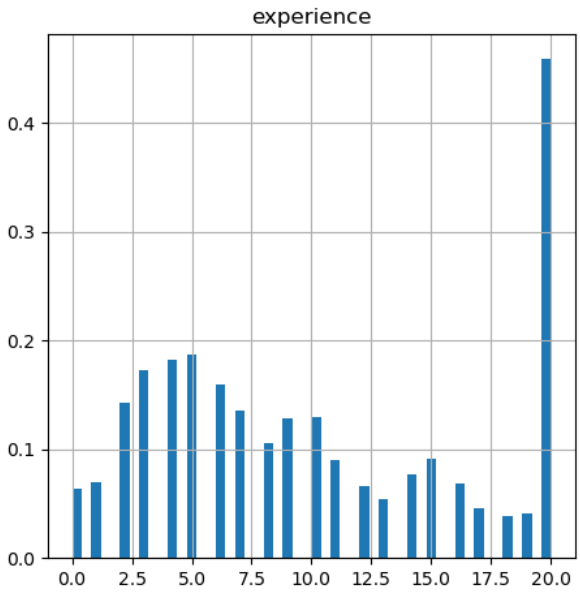
In [9]:
```
new_df = df[cols].dropna()
df.shape, new_df.shape
```

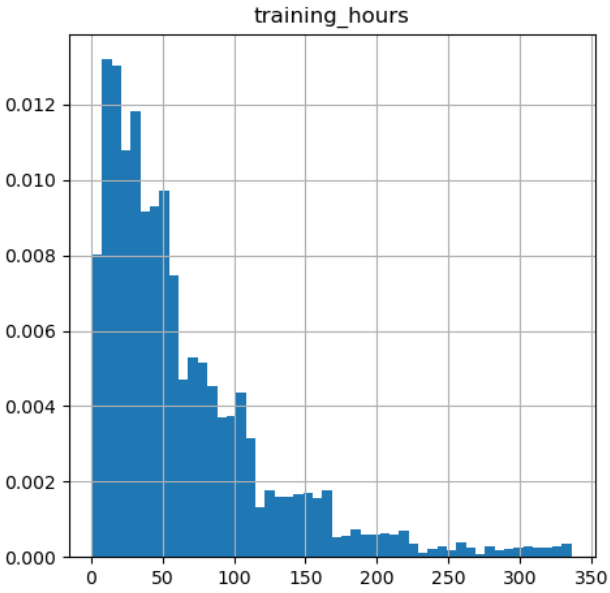
Out[9]:
```
((19158, 13), (17182, 5))
```

Plot Histogram (Applying CCA: Numerical Data)

In [10]:
```
new_df.hist(bins=50, density=True, figsize=(12, 12))
plt.show()
```



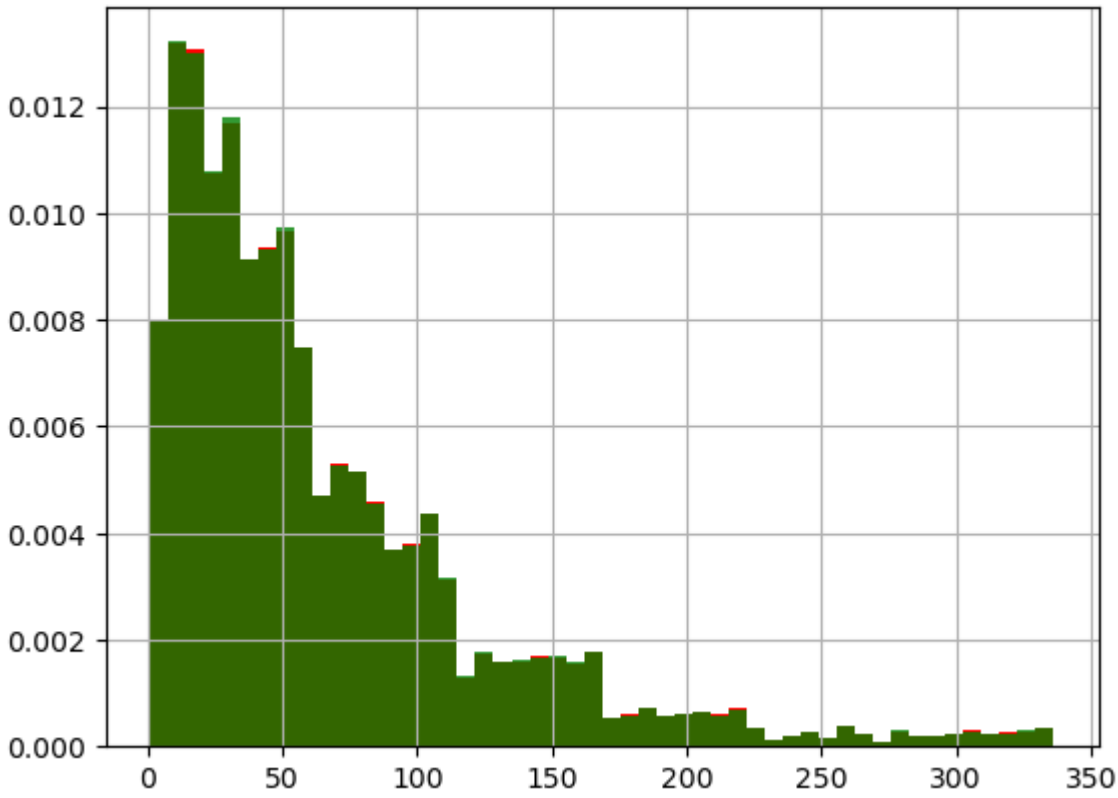




Plot Histogram: Training Hours

In [12]:
```
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)
```

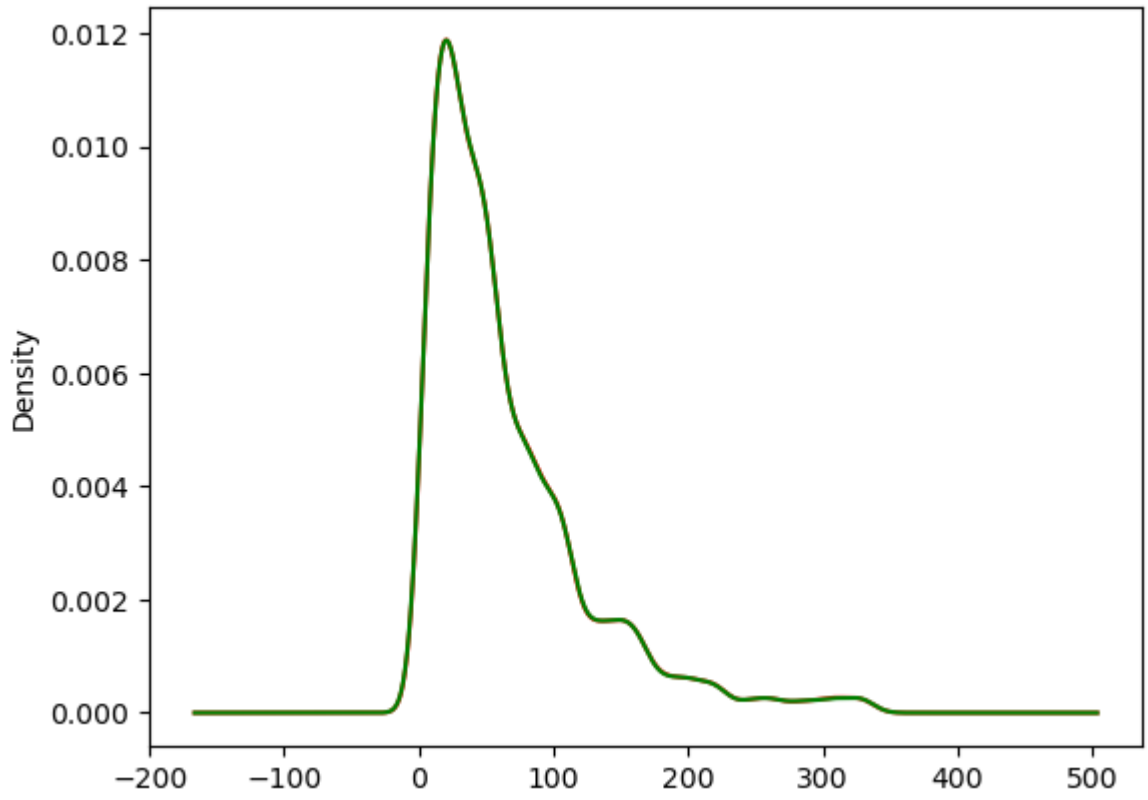
Out[12]: <Axes: >



Plot Probability Density Function (PDF): Training Hours

In [14]:
```
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')
```

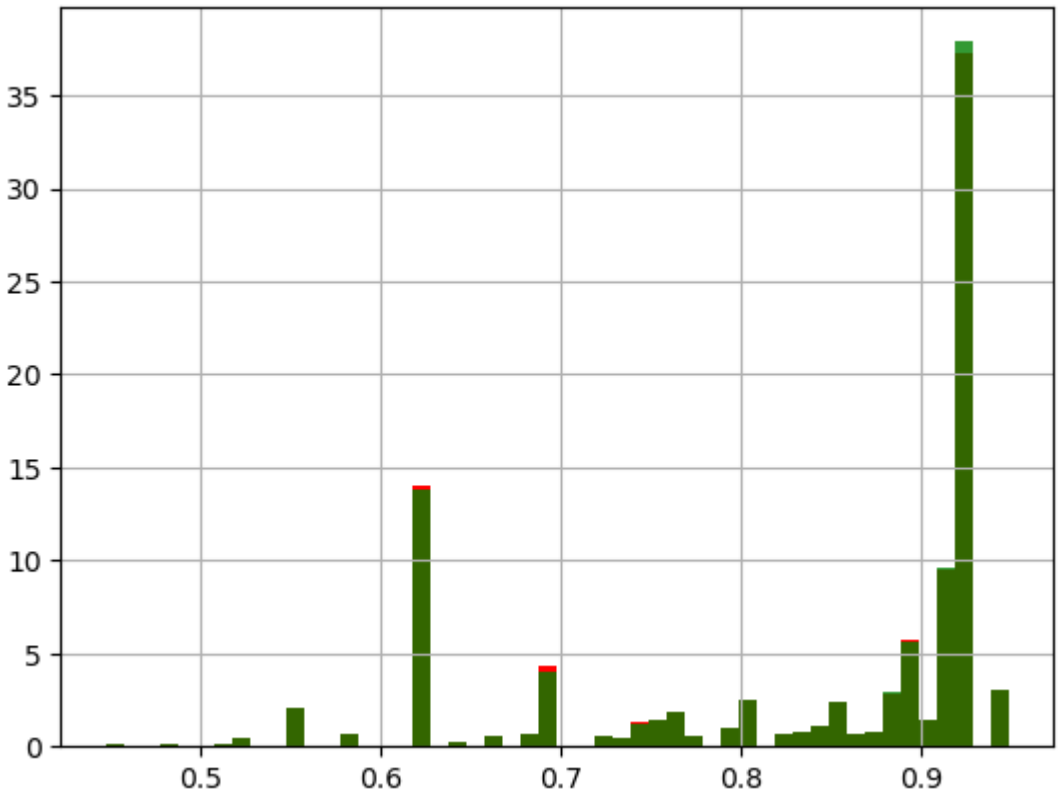
Out[14]: <Axes: ylabel='Density'>



Plot Histogram: City Development

In [15]:
```
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, al
```

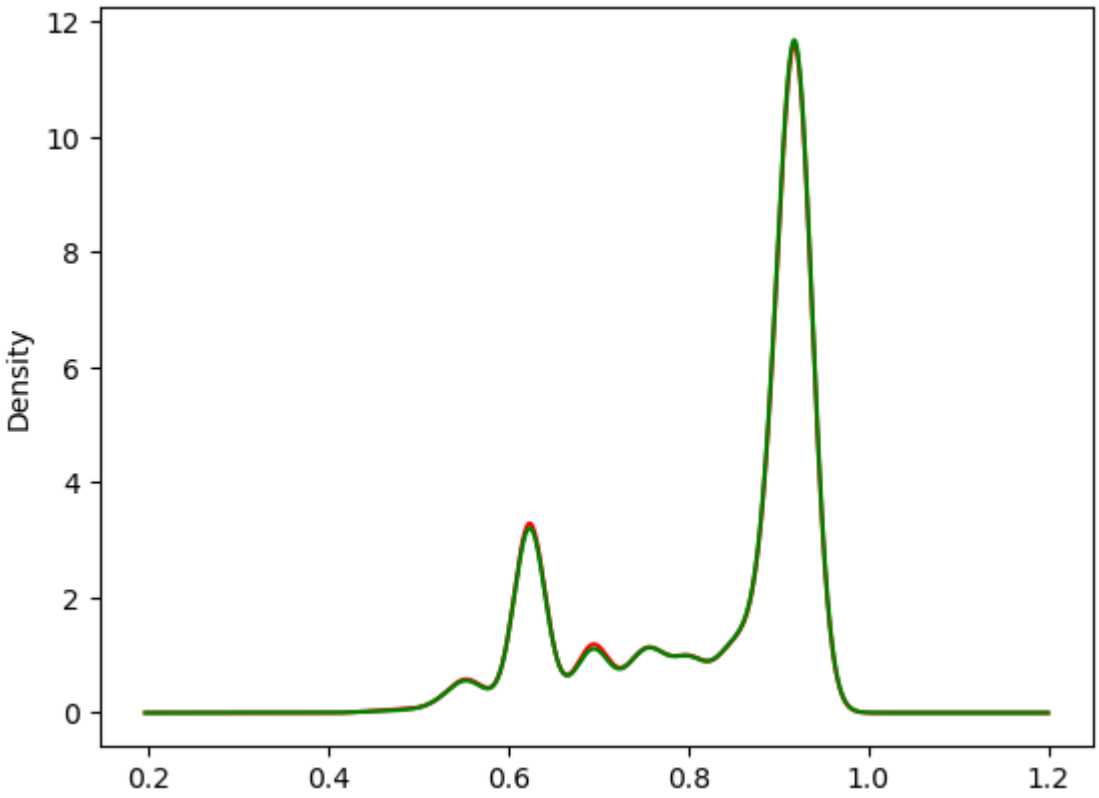
Out[15]: <Axes: >



Plot PDF: City Development

In [16]:
```
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')
```

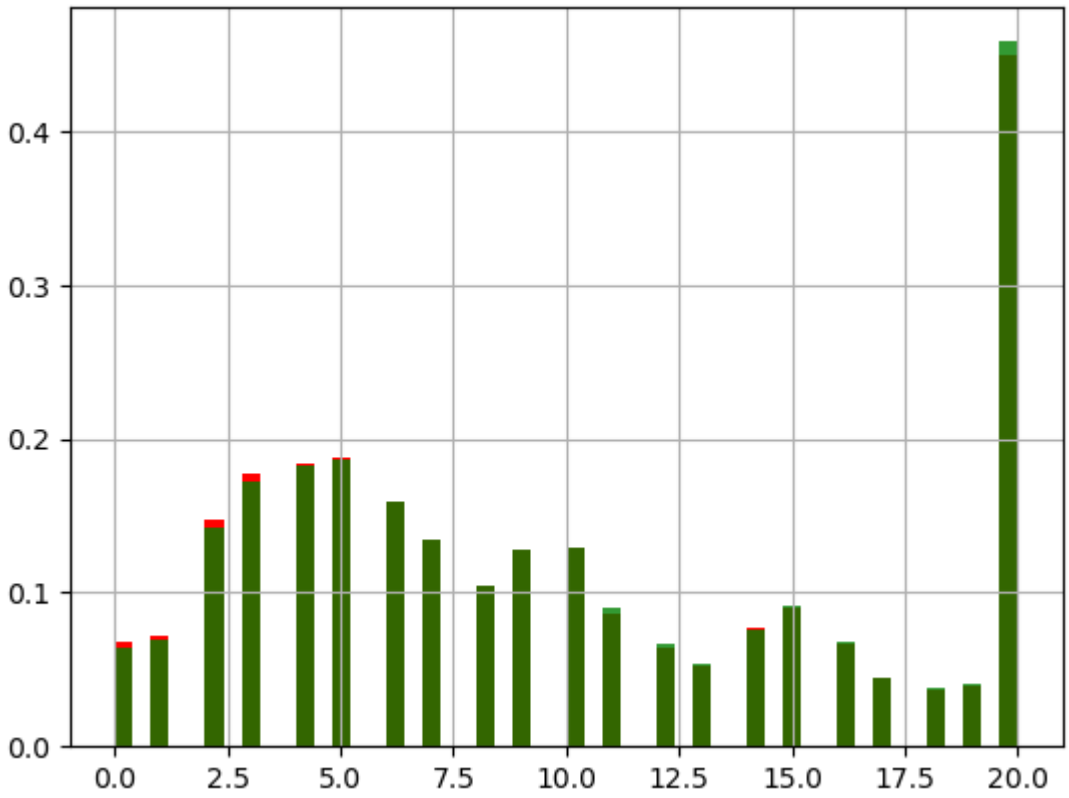
Out[16]: <Axes: ylabel='Density'>



Plot Histogram: Experience

In [17]:
```
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)
```

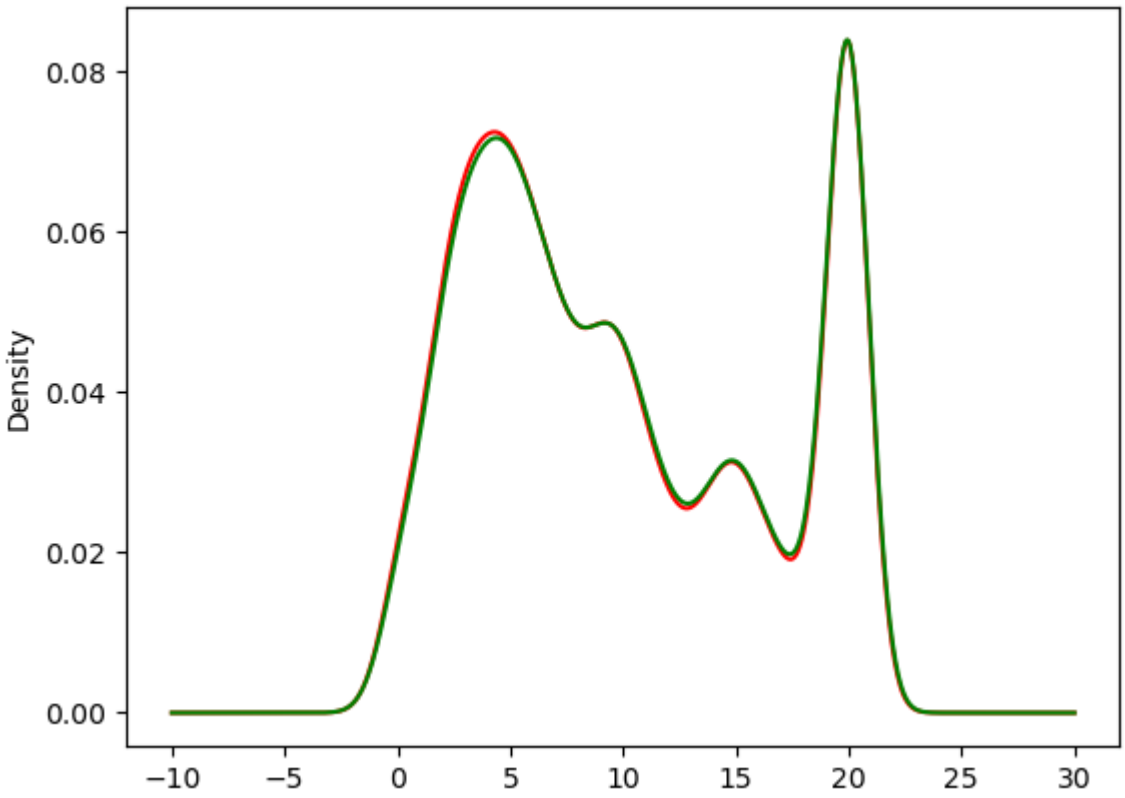
Out[17]: <Axes: >



Plot PDF: Experience

In [18]:
```
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[18]: <Axes: ylabel='Density'>



CCA in Categorical Data

In [19]:
```
df['education_level'].value_counts()
```

Out[19]:
```
education_level
Graduate          11598
Masters            4361
High School        2017
Phd                 414
Primary School      308
Name: count, dtype: int64
```

In [20]:
```
df['enrolled_university'].value_counts()
```

Out[20]:
```
enrolled_university
no_enrollment       13817
Full time course     3757
Part time course     1198
Name: count, dtype: int64
```

CCA Apply in Enrolled University

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

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

CCA Apply in Education Level

In [22]:
```
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[22]:

education_level	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

In [ ]:
```
```