import pandas as pd import numpy as np

SubmissionDate starttime

0 2021-11-13 19:10:21 2021-11-13 13:12:59

endtime 2021-11-13 13:23:11

\

SubmissionDate starttime

0 2021-11-13 19:10:21 2021-11-13 13:12:59

1 2021-11-13 19:10:22 2021-11-13 13:24:01

endtime 2021-11-13 13:23:11

2021-11-13 13:35:51

\

df = pd.read\_csv('C:/Users/2023/Downloads/dataset/cleaned\_data\_data.csv')

df.head(1)

deviceid subscriberid simid devicephonenum

username \

0 ef443f91bed409a9 405864986514130 NaN 9.179798e+11 uchicago

duration interviewer\_name ... i\_spend\_entertainment i\_totalexp \

0 487 Gunjan Kumari ... 300.0 3250.0

i\_expenditure i\_savings i\_loan i\_loan\_amt i\_loan\_rate i\_highest i\_lowest \

0 11000.0 -98.0 Yes 6000.0 2.0 -98.0

3000.0

formdef\_version

0 2111090155

[1 rows x 65 columns]

*# unique values in the 'agree' column*

unique\_values = df['agree'].unique()

*# Display the unique values -100 and -1 i have alredy done 0 as no and*

*1 as yes*

print(unique\_values) ['Yes' '-100' 'No' '-1']

*# Replace '-100' with 'Error' and '-1' with 'Unknown'*

df['agree'] = df['agree'].replace({'-100': 'Did not consent', '-1': 'Refused to answer'})

*# Verify the change*

print(df['agree'].unique())

['Yes' 'Did not consent' 'No' 'Refused to answer'] df.head(10)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
| 2 | 2021-11-13 19:10:22 | 2021-11-13 13:39:04 | 2021-11-13 13:50:27 |
| 3 | 2021-11-13 19:10:23 | 2021-11-13 17:04:27 | 2021-11-13 17:14:15 |
| 4 | 2021-11-13 19:10:23 | 2021-11-13 17:15:16 | 2021-11-13 17:25:23 |
| 5 | 2021-11-13 19:10:24 | 2021-11-13 17:25:40 | 2021-11-13 17:36:30 |
| 6 | 2021-11-13 19:10:24 | 2021-11-13 17:38:00 | 2021-11-13 17:48:37 |
| 7 | 2021-11-13 19:10:25 | 2021-11-13 17:50:18 | 2021-11-13 17:56:49 |
| 8 | 2021-11-13 19:13:50 | 2021-11-11 15:16:04 | 2021-11-13 19:06:54 |
|  | 9 | 2021-11-13 19:13:50 | 2021-11-13 12:21:52 | 2021-11-13 19:05:57 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | deviceid | subscriberid | simid | devicephonenum | \ |
| 0 | ef443f91bed409a9 | 405864986514130 | NaN | 9.179798e+11 |  |
| 1 | ef443f91bed409a9 | 405864986514130 | NaN | 9.179798e+11 |  |
| 2 | ef443f91bed409a9 | 405864986514130 | NaN | 9.179798e+11 |  |
| 3 | ef443f91bed409a9 | 405864986514130 | NaN | 9.179798e+11 |  |
| 4 | ef443f91bed409a9 | 405864986514130 | NaN | 9.179798e+11 |  |
| 5 | ef443f91bed409a9 | 405864986514130 | NaN | 9.179798e+11 |  |
| 6 | ef443f91bed409a9 | 405864986514130 | NaN | 9.179798e+11 |  |
| 7 | ef443f91bed409a9 | 405864986514130 | NaN | 9.179798e+11 |  |
| 8 | 359475073412828 | 405864986514130 | NaN | NaN |  |
| 9 | 359475073412828 | 405864986514130 | NaN | NaN |  |

|  |  |  |
| --- | --- | --- |
| username | duration | interviewer\_name ... \ |
| 0 uchicago | 487 | Gunjan Kumari ... |
| 1 uchicago | 661 | Gunjan Kumari ... |
| 2 uchicago | 659 | Gunjan Kumari ... |
| 3 uchicago | 588 | Gunjan Kumari ... |
| 4 uchicago | 568 | Gunjan Kumari ... |
| 5 uchicago | 574 | Gunjan Kumari ... |
| 6 uchicago | 599 | Gunjan Kumari ... |
| 7 uchicago | 392 | Gunjan Kumari ... |
| 8 [contact@outlineindia.com](mailto:contact@outlineindia.com) | 2085 | Simaila ... |
| 9 [contact@outlineindia.com](mailto:contact@outlineindia.com) | 1170 | Simaila Kumari ... |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| i\_spend\_entertainment | i\_totalexp | i\_expenditure | i\_savings | i\_loan | \ |
| 0 300.0 | 3250.0 | 11000.0 | -98.0 | Yes |  |
| 1 250.0 | 3099.0 | 6000.0 | -98.0 | No |  |
| 2 1000.0 | 7000.0 | 20000.0 | -98.0 | No |  |
| 3 700.0 | 3200.0 | 16000.0 | 4000.0 | Yes |  |
| 4 500.0 | 2250.0 | 10000.0 | 3000.0 | No |  |
| 5 400.0 | 2150.0 | 12000.0 | -98.0 | Yes |  |
| 6 400.0 | 2700.0 | 8000.0 | 1000.0 | Yes |  |
| 7 800.0 | 4800.0 | 16000.0 | 5000.0 | No |  |
| 8 No | 5799.0 | 8000.0 | 15000.0 | No |  |
| 9 No | 7199.0 | 12000.0 | -99.0 | No |  |

i\_loan\_amt i\_loan\_rate i\_highest i\_lowest

formdef\_version

0 6000.000000 2.0 -98.0 3000.0

2111090155

1 58458.609756 0.07317073170731707 -98.0 -98.0

2111131313

0

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

Yes Yes No Yes No No Yes Yes No Yes Yes NaN No No Yes Yes No Yes Yes Yes Yes

2 58458.609756 0.07317073170731707 -98.0 -98.0

2111131313

3 50000.000000 5.0 25000.0 5000.0

2111131313

4 58458.609756 0.07317073170731707 -99.0 -98.0

2111131313

5 10000.000000 2.0 -98.0 -98.0

2111131313

6 20000.000000 3.0 -98.0 -98.0

2111131313

7 58458.609756 0.07317073170731707 -99.0 -99.0

2111131313

8 58458.609756 0.07317073170731707 -99.0 1000.0

2111111118

9 58458.609756 0.07317073170731707 10000.0 5000.0

2111111118

[10 rows x 65 columns]

*# View unique values in the 'hh\_BPL\_bin' column* unique\_values = df['hh\_BPL\_bin'].unique() print(unique\_values)

['Yes' 'No' nan '-999.0']

pd.set\_option('display.max\_rows', None) print(df['hh\_BPL\_bin'])

1. Yes
2. Yes
3. No
4. Yes
5. No
6. Yes
7. Yes
8. NaN
9. Yes
10. No
11. No
12. No
13. No
14. No
15. No
16. No
17. No
18. Yes
19. Yes
20. Yes
21. Yes
22. Yes
23. Yes
24. Yes
25. Yes
26. No
27. No
28. No
29. Yes
30. Yes
31. No
32. No
33. No
34. No
35. No
36. Yes
37. Yes
38. Yes
39. Yes
40. No
41. No
42. Yes
43. No
44. Yes
45. Yes
46. NaN
47. No
48. Yes
49. Yes
50. Yes
51. Yes
52. No
53. Yes
54. NaN
55. No
56. Yes
57. Yes
58. No
59. No
60. Yes
61. Yes
62. No
63. Yes
64. No
65. Yes

86 -999.0

1. No
2. Yes
3. Yes
4. No
5. Yes
6. Yes
7. NaN
8. NaN
9. Yes
10. NaN
11. NaN
12. Yes
13. Yes
14. NaN
15. Yes
16. Yes
17. Yes
18. Yes
19. No
20. No
21. NaN
22. No
23. No
24. Yes
25. NaN
26. Yes
27. Yes
28. Yes
29. No
30. Yes
31. Yes
32. Yes

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 119 | No |
| 120 | Yes |
| 121 | Yes |
| 122 | Yes |
| 123 | Yes |
| 124 | Yes |
| 125 | No |
| 126 | No |
| 127 | No |
| 128 | Yes |
| 129 | Yes |
| 130 | No |
| 131 | No |
| 132 | Yes |
| 133 | No |
| 134 | No |
| 135 | Yes |
| 136 | Yes |
| 137 | Yes |
| 138 | Yes |
| 139 | Yes |
| 140 | Yes |
| 141 | Yes |
| 142 | Yes |
| 143 | Yes |
| 144 | Yes |
| 145 | Yes |
|  | 146 | Yes |

Name: hh\_BPL\_bin, dtype: object print(df['hh\_BPL\_bin'].dtype) object

*# Replace '-999' and '-999.0' (as strings) with 'Don't know' in the 'hh\_BPL\_bin' column*

df['hh\_BPL\_bin'] = df['hh\_BPL\_bin'].replace({'-999': 'Don\'t know', '- 999.0': 'Don\'t know'})

*# Verify the change*

print(df['hh\_BPL\_bin'].unique()) ['Yes' 'No' nan "Don't know"]

*# Check unique values for specific columns*

columns\_to\_check = ['el\_source', 'el\_source\_1', 'el\_source\_2', 'el\_source\_3', 'el\_source\_4', 'el\_source\_5', 'el\_source 999', 'el\_source 1', 'el\_source 100']

for column in columns\_to\_check:

unique\_values = df[column].unique()

print(f"Unique values in {column}: {unique\_values}")

Unique values in el\_source: ['1' '1 5' nan '1 3'] Unique values in el\_source\_1: ['Yes' nan]

Unique values in el\_source\_2: ['No' nan] Unique values in el\_source\_3: ['No' nan 'Yes'] Unique values in el\_source\_4: ['No' nan] Unique values in el\_source\_5: ['No' 'Yes' nan] Unique values in el\_source 999: ['No' nan] Unique values in el\_source 1: ['No' nan] Unique values in el\_source 100: ['No' nan]

df['el\_source'] = df['el\_source'].astype(str).str.strip()

*# Replace 1 with 'Yes'*

df['el\_source'] = df['el\_source'].replace({'1': 'Yes'}) print(df['el\_source'].unique())

['Yes' '1 5' 'nan' '1 3']

columns\_to\_check = ['el\_bill\_bin', 'el\_bill\_date', 'el\_bill\_amt']

for column in columns\_to\_check: unique\_values = df[column].unique()

print(f"Unique values in {column}: {unique\_values}")

Unique values in el\_bill\_bin: ['Yes' nan 'No' '-1.0']

Unique values in el\_bill\_date: [ 8. 10. 9. 6. nan 11. -

100. 3. 5. -999. 7.]

Unique values in el\_bill\_amt: [ 1600. 3200. 2200. 400. 2500.

1200. 200. 300. nan 600.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 250. | 350. 99. 1642. | 875. | -99. 1350. | 100. 9000. | 1950. |
| 500. | 211. 1000. 2000. | 800. | 760. 700. | 5432. 1300. | 3000. |
| 4000. | 820. 16000. 1500. | 3500. | 150. 15000. | 1527. 1400. | 203. |
| 249. | 2146. 178. 501. | 780. | 98. 2020. | 4398. 1753. | 11049. |

1084. 5000. 2400. 894. 655. 6200.]

*# Strip any leading/trailing whitespaces before replacing*

df['el\_bill\_bin'] = df['el\_bill\_bin'].str.strip()

*# Replace '-1.0' with 'Refused to answer'*

df['el\_bill\_bin'] = df['el\_bill\_bin'].replace({'-1.0': 'Refused to answer'})

*# Verify the change*

print(df['el\_bill\_bin'].unique()) ['Yes' nan 'No' 'Refused to answer']

*# Replace '-100' with 'Did not consent' and '-999.0' with 'Don't know'* df['el\_bill\_date'] = df['el\_bill\_date'].replace({-100: 'Did not consent', -999.0: 'Don\'t know'})

print(df['el\_bill\_date'].unique())

[8.0 10.0 9.0 6.0 nan 11.0 'Did not consent' 3.0 5.0 "Don't know" 7.0] columns\_to\_check = ['el\_pay\_freq', 'el\_pay\_freq', 'el\_pay\_freq']

for column in columns\_to\_check: unique\_values = df[column].unique()

print(f"Unique values in {column}: {unique\_values}")

Unique values in el\_pay\_freq: ['3.0' '4.0' '2.0' 'Yes' nan '-999.0'

'5.0']

Unique values in el\_pay\_freq: ['3.0' '4.0' '2.0' 'Yes' nan '-999.0'

'5.0']

Unique values in el\_pay\_freq: ['3.0' '4.0' '2.0' 'Yes' nan '-999.0'

'5.0']

replacement\_dict = {

'3.0': 'Every 3 months',

'Yes': 'Monthly',

'2.0': 'Every two months',

'4.0': 'Less than once in 3 months', '5.0': 'Other',

'-999.0': 'Don\'t know'

}

df['el\_pay\_freq'] = df['el\_pay\_freq'].replace(replacement\_dict) print(df['el\_pay\_freq'].unique())

['Every 3 months' 'Less than once in 3 months' 'Every two months'

'Monthly' nan "Don't know" 'Other'] print(df['el\_pay\_freq'].unique())

['Every 3 months' 'Less than once in 3 months' 'Every two months'

'Monthly' nan "Don't know" 'Other']

columns = [

'el\_unpaid\_reason\_2', 'el\_unpaid\_reason\_3', 'el\_unpaid\_reason\_4', 'el\_unpaid\_reason\_5', 'el\_unpaid\_reason\_6', 'el\_unpaid\_reason\_7', 'el\_unpaid\_reason\_8', 'el\_unpaid\_reason 999',

'el\_unpaid\_reason 1', 'el\_unpaid\_reason 100'

]

*# Check the unique values in each of the specified columns*

for col in columns:

print(f"Unique values in {col}:") print(df[col].unique())

Unique values in el\_unpaid\_reason\_2:

['No' nan 'Yes']

Unique values in el\_unpaid\_reason\_3:

['No' 'Yes' nan]

Unique values in el\_unpaid\_reason\_4:

['No' 'Yes' nan]

Unique values in el\_unpaid\_reason\_5:

['Yes' 'No' nan]

Unique values in el\_unpaid\_reason\_6:

['Yes' 'No' nan]

Unique values in el\_unpaid\_reason\_7:

['No' 'Yes' nan]

Unique values in el\_unpaid\_reason\_8:

['No' 'Yes' nan]

Unique values in el\_unpaid\_reason 999:

['No' nan 'Yes']

Unique values in el\_unpaid\_reason 1:

['No' nan]

Unique values in el\_unpaid\_reason 100:

['No' nan]

columns = ['el\_right']

*# Check the unique values in each of the specified columns*

for col in columns:

print(f"Unique values in {col}:") print(df[col].unique())

Unique values in el\_right:

['No' "Don't know" 'Yes' 'Refused to answer' 'nan']

df['el\_right'] = df['el\_right'].replace({'-999.0':'Don\'t know', '- 1.0':'Refused to answer', 'Very unlikely': 'Yes'}) print(df['el\_right'].unique())

['No' "Don't know" 'Yes' 'Refused to answer' 'nan'] columns = ['el\_notice']

*# Check the unique values in each of the specified columns*

for col in columns:

print(f"Unique values in {col}:") print(df[col].unique())

Unique values in el\_notice:

['-999.0' '5.0' '4.0' '-1.0' nan '3.0' 'Yes' '2.0' '-100.0']

df['el\_notice'] = df['el\_notice'].astype(str)

*# these are specific values with corresponding labels*

df['el\_notice'] = df['el\_notice'].replace({ '-999.0': "Don't know",

'-1.0': 'Refused to answer',

'5.0': 'Very Likely',

'4.0': 'Somewhat likely',

'3.0': 'Neither unlikely or likely', '2.0': 'Somewhat unlikely',

'-100.0': 'Does not consent', 'Yes': 'Very unlikely',

})

print("Unique values in el\_notice after replacement:") print(df['el\_notice'].unique())

Unique values in el\_notice after replacement:

["Don't know" 'Very Likely' 'Somewhat likely' 'Refused to answer' 'nan'

'Neither unlikely or likely' 'Very unlikely' 'Somewhat unlikely'

'Does not consent'] columns = ['el\_unpaid\_nbr']

*# Check the unique values in each of the specified columns*

for col in columns:

print(f"Unique values in {col}:") print(df[col].unique())

Unique values in el\_unpaid\_nbr:

['4.0' '2.0' '3.0' '-999.0' 'Yes' nan '-100.0' '-1.0']

*# Replace values in 'el\_unpaid\_nbr' with labels*

df['el\_unpaid\_nbr'] = df['el\_unpaid\_nbr'].replace({ '4.0': 'Almost everybody pays on time and in full',

'3.0': 'More than 50 percent pay on time and in full',

'2.0': 'Less than 50 percent pay on time and in full', '-999.0': "Don\'t know",

'-1.0': 'Refused to answer',

'-100.0': 'Does not consent',

'Yes': 'Almost nobody pays on time in full',

})

print("Unique values in el\_unpaid\_nbr after replacement:") print(df['el\_unpaid\_nbr'].unique())

Unique values in el\_unpaid\_nbr after replacement: ['Almost everybody pays on time and in full'

'Less than 50 percent pay on time and in full'

'More than 50 percent pay on time and in full' "Don't know"

'Almost nobody pays on time in full' nan 'Does not consent'

'Refused to answer']

df['el\_disconnect'] = df['el\_disconnect'].replace({

'-999.0': "Don\'t know",

'-1.0': 'Refused to answer',

})

print(df['el\_disconnect'].unique())

["Don't know" 'Yes' 'No' nan 'Refused to answer'] columns = ['i\_source']

*# Check the unique values in each of the specified columns*

for col in columns:

print(f"Unique values in {col}:") print(df[col].unique())

Unique values in i\_source:

['7.0' 'Yes' '2.0' '6.0' '3.0' '5.0' '8.0' '9.0' '11.0' nan '4.0' '-

999.0']

*# Define the replacements for i\_source*

i\_source\_labels = {

'Yes': 'Agriculture and related activities', '2.0': 'Pension and Income',

'3.0': 'Small business',

'4.0': 'Professional',

'5.0': 'Small industry/mill/factory',

'6.0': 'Transportation',

'7.0': 'Construction',

'8.0': 'Income from remittances', '9.0': 'Rent/dividend',

'10.0': 'Forest based livelihood', '11.0': 'Other',

'-999.0': 'Don\'t know',

'-1.0': 'Refused to answer',

'-100.0': "Did not consent",

}

df['i\_source'] = df['i\_source'].astype(str)

df['i\_source'] = df['i\_source'].replace(i\_source\_labels)

print("Unique values in i\_source after replacement:") print(df['i\_source'].unique())

Unique values in i\_source after replacement:

['Construction' 'Agriculture and related activities' 'Pension and Income'

'Transportation' 'Small business' 'Small industry/mill/factory'

'Income from remittances' 'Rent/dividend' 'Other' 'nan' 'Professional'

"Don't know"] columns = ['i\_loan']

*# Check the unique values in each of the specified columns*

for col in columns:

print(f"Unique values in {col}:") print(df[col].unique())

Unique values in i\_loan:

['Yes' 'No' nan '-100.0' '-999.0']

i\_loan\_labels = {

'-999.0': "Don't know",

'-1.0': 'Refused to answer', '-100.0': "Did not consent"

}

df['i\_loan'] = df['i\_loan'].astype(str) df['i\_loan'] = df['i\_loan'].replace(i\_loan\_labels)

print("Unique values in i\_loan after replacement:") print(df['i\_loan'].unique())

Unique values in i\_loan after replacement:

['Yes' 'No' 'nan' 'Did not consent' "Don't know"]

*# Save DataFrame to a new CSV file* df.to\_csv('C:/Users/2023/Downloads/dataset/labeled\_data.csv', index=False)

print("CSV file saved successfully!") CSV file saved successfully!

import pandas as pd import numpy as np import seaborn as sns

import matplotlib.pyplot as plt

df = pd.read\_csv('C:/Users/2023/Downloads/dataset/labeled\_data.csv')

*#for cleaning further anlysis i will drop some of the irrelavent columns*

df = df.drop(columns=['SubmissionDate', 'starttime', 'endtime', 'deviceid', 'subscriberid', 'simid', 'devicephonenum', 'username'], errors='ignore')

*#no use of this data*

*#Likely related to internal mapping*

df= df.drop(columns=['list\_id', 'list\_id\_1'] ,errors= 'ignore') df['el\_disconnect'] = df['el\_disconnect'].replace({'Yes': 1})

import seaborn as sns

import matplotlib.pyplot as plt

*# the average loan amount by unpaid reason* loan\_by\_unpaid\_reason = df.groupby('el\_unpaid\_reason') ['i\_loan\_amt'].mean().sort\_values(ascending=False)

plt.figure(figsize=(8, 6)) sns.barplot(x=loan\_by\_unpaid\_reason.index, y=loan\_by\_unpaid\_reason.values, palette='Set2') plt.title('Average Loan Amount by Unpaid Reason', fontsize=16, fontweight='bold')

plt.xlabel('Reason for Unpaid Bill', fontsize=14) plt.ylabel('Average Loan Amount (in USD)', fontsize=14) plt.xticks(rotation=45, ha='right')

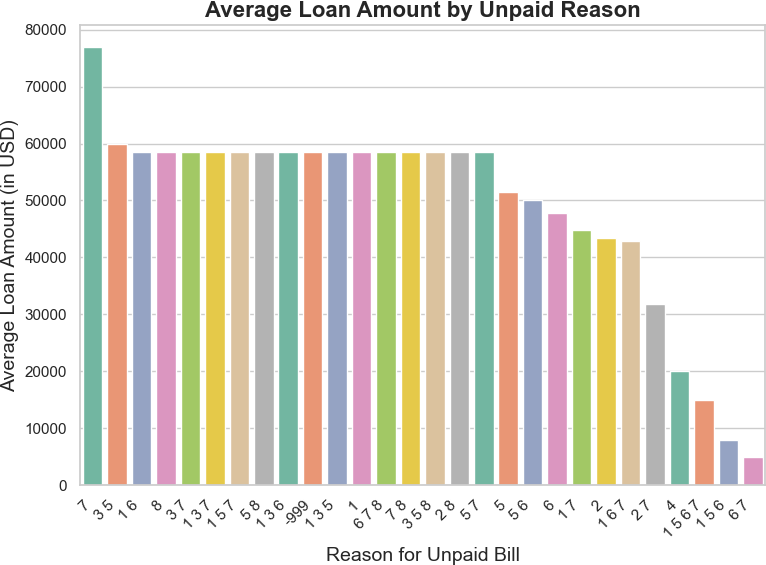
plt.tight\_layout() plt.show()

C:\Users\2023\AppData\Local\Temp\ipykernel\_13164\3172667531.py:9: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set

`legend=False` for the same effect.

sns.barplot(x=loan\_by\_unpaid\_reason.index, y=loan\_by\_unpaid\_reason.values, palette='Set2')



df['el\_unpaid\_reason'] = df['el\_unpaid\_reason'].replace({ 1: "Can't afford to pay",

2: "No punishment for late payments", 3: "Service quality is poor",

4: "Electricity is a right; shouldn't have to pay", 5: "Going to the payment location takes a long time", 6: "My bills are often incorrect",

7: "I always pay in full", 8: "Other",

-999: "Don't know",

-1: "Refused to answer",

-100: "Did not consent/didn't reach this question"

})

*# Calculate the average loan amount by unpaid reason*

loan\_by\_unpaid\_reason = ( df.groupby('el\_unpaid\_reason')['i\_loan\_amt']

.mean()

.sort\_values(ascending=False)

)

plt.figure(figsize=(10, 6))

sns.barplot(

x=loan\_by\_unpaid\_reason.index, y=loan\_by\_unpaid\_reason.values, hue=loan\_by\_unpaid\_reason.index, palette='Set2',

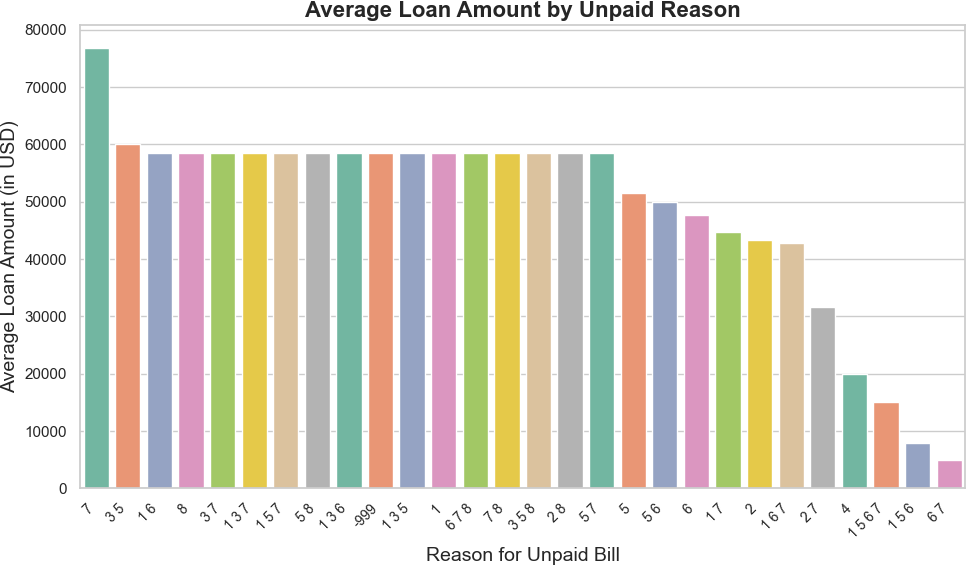
dodge=False

)

plt.title('Average Loan Amount by Unpaid Reason', fontsize=16, fontweight='bold')

plt.xlabel('Reason for Unpaid Bill', fontsize=14) plt.ylabel('Average Loan Amount (in USD)', fontsize=14) plt.xticks(rotation=45, ha='right', fontsize=10) plt.legend([], [], frameon=False)

plt.tight\_layout() plt.show()



*# Income Source vs Loan Amount*

income\_vs\_loan = df.groupby('i\_source')['i\_loan\_amt'].mean() print("\nAverage Loan Amount by Income Source:") print(income\_vs\_loan)

*# Bar plot for Income Source vs Loan Amount (fix for the FutureWarning)*

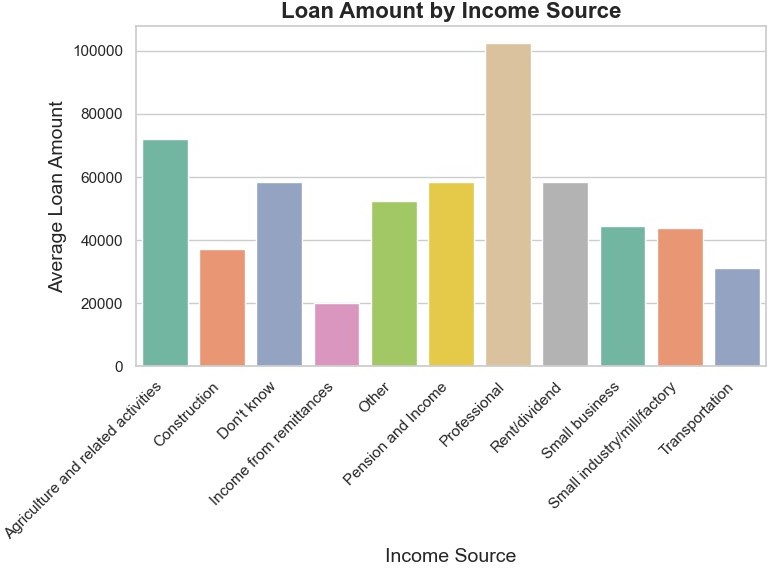
plt.figure(figsize=(8, 6)) sns.barplot(x=income\_vs\_loan.index, y=income\_vs\_loan.values,

hue=income\_vs\_loan.index, palette='Set2', legend=False) plt.title('Loan Amount by Income Source', fontsize=16, fontweight='bold')

plt.xlabel('Income Source', fontsize=14) plt.ylabel('Average Loan Amount', fontsize=14) plt.xticks(rotation=45, ha='right') plt.tight\_layout()

plt.show()

|  |  |  |
| --- | --- | --- |
|  | Average Loan Amount by Income Source: |  |
| i\_source |
| Agriculture and related activities | 72188.831053 |
| Construction | 37319.504878 |
| Don't know | 58458.609756 |
| Income from remittances | 20000.000000 |
| Other | 52465.508130 |
| Pension and Income | 58458.609756 |
| Professional | 102612.748780 |
| Rent/dividend | 58458.609756 |
| Small business | 44527.962060 |
| Small industry/mill/factory | 43899.006969 |
| Transportation | 31152.869919 |
|  | Name: i\_loan\_amt, dtype: float64 |  |



df['hh\_size'] = pd.to\_numeric(df['hh\_size'], errors='coerce') df['el\_hours'] = pd.to\_numeric(df['el\_hours'], errors='coerce')

*# Calculate the correlation between household size and electricity hours*

correlation = df[['hh\_size', 'el\_hours']].corr() print("\nCorrelation between household size and electricity hours:") print(correlation)

grouped\_by\_size = df.groupby('hh\_size') ['el\_hours'].mean().reset\_index()

plt.figure(figsize=(8, 5))

sns.lineplot(x='hh\_size', y='el\_hours', data=grouped\_by\_size, marker='o', color='blue')

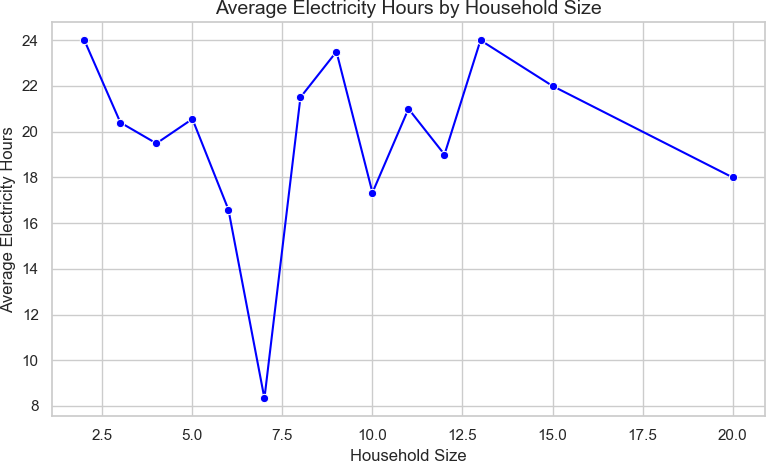
plt.title('Average Electricity Hours by Household Size', fontsize=14) plt.xlabel('Household Size', fontsize=12)

plt.ylabel('Average Electricity Hours', fontsize=12) plt.tight\_layout()

plt.show()

Correlation between household size and electricity hours:

|  |  |  |
| --- | --- | --- |
| hh\_size | hh\_size 1.000000 | el\_hours  -0.016449 |
| el\_hours | -0.016449 | 1.000000 |



df.to\_csv('C:/Users/2023/Downloads/dataset/labeled\_dataa.csv', index=False)

print("CSV file saved successfully!") CSV file saved successfully!

mean\_bill = df['el\_bill\_amt'].mean() min\_bill = df['el\_bill\_amt'].min() max\_bill = df['el\_bill\_amt'].max() median\_bill = df['el\_bill\_amt'].median() std\_dev = df['el\_bill\_amt'].std()

print(f"Mean Bill Amount: {mean\_bill}") print(f"Minimum Bill Amount: {min\_bill}") print(f"Maximum Bill Amount: {max\_bill}") print(f"Median Bill Amount: {median\_bill}") print(f"Standard Deviation: {std\_dev}")

Mean Bill Amount: 1312.1304347826087 Minimum Bill Amount: -99.0

Maximum Bill Amount: 16000.0 Median Bill Amount: 400.0

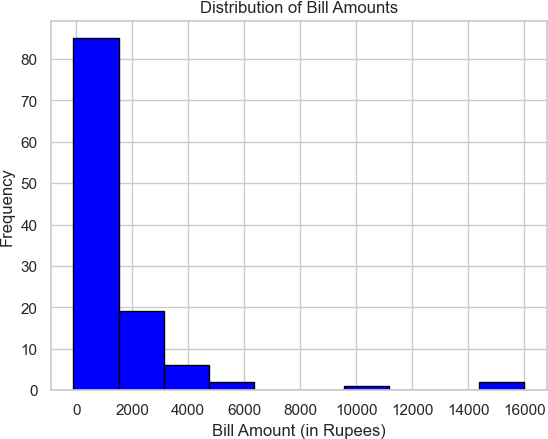
Standard Deviation: 2433.3943858781904

*# Plot histogram for bill amounts*

plt.hist(df['el\_bill\_amt'], bins=10, color='blue', edgecolor='black') plt.title('Distribution of Bill Amounts')

plt.xlabel('Bill Amount (in Rupees)') plt.ylabel('Frequency')

plt.show()



*# Group by household size and calculate mean bill amount* mean\_bill\_by\_hh\_size = df.groupby('hh\_size')['el\_bill\_amt'].mean() print(mean\_bill\_by\_hh\_size)

*# Group by BPL status and calculate mean savings* mean\_savings\_by\_bpl = df.groupby('hh\_BPL\_bin')['i\_savings'].mean() print(mean\_savings\_by\_bpl)

hh\_size

2.0 800.000000

dtype: float64

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 3.0 | 1640.000000 |
| 4.0 | 1006.171429 |
| 5.0 | 571.000000 |
| 6.0 | 1327.148148 |
| 7.0 | 3467.333333 |
| 8.0 | 1345.125000 |
| 9.0 | 1675.000000 |
| 10.0 | 1133.666667 |
| 11.0 | 2200.000000 |
| 12.0 | 1500.000000 |
| 13.0 | 1500.000000 |
| 15.0 | 3000.000000 |
| 20.0 | 300.000000 |
| Name: el\_bill\_amt, | | |

hh\_BPL\_bin

Don't know 500.000000

No 13300.046512

Yes 3538.760563

Name: i\_savings, dtype: float64

*# Average bill amount by payment frequency*

avg\_bill\_by\_pay\_freq = df.groupby('el\_pay\_freq')['el\_bill\_amt'].mean() print(avg\_bill\_by\_pay\_freq)

*# Group by interviewer and calculate average bill amount* avg\_bill\_by\_interviewer = df.groupby('interviewer\_name') ['el\_bill\_amt'].mean()

print(avg\_bill\_by\_interviewer) el\_pay\_freq

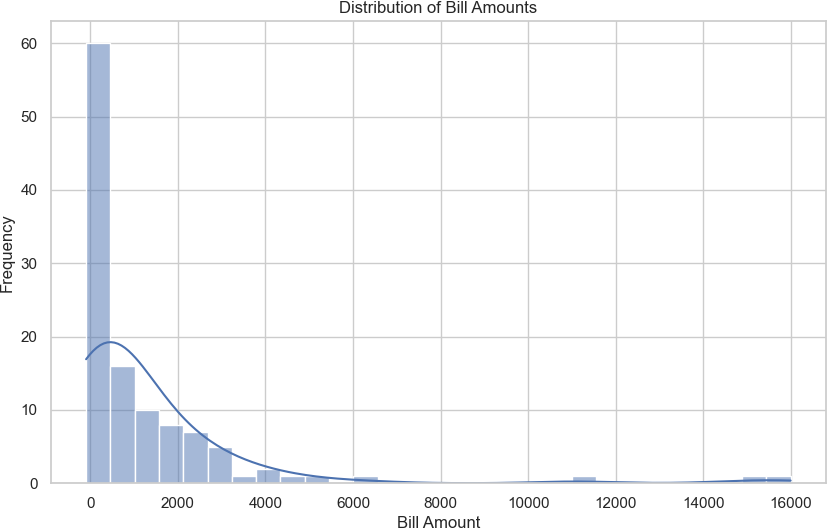
|  |  |  |
| --- | --- | --- |
| Don't know |  | -33.333333 |
| Every 3 months |  | 2448.375000 |
| Every two months |  | 639.578947 |
| Less than once in | 3 months | 2068.777778 |
| Monthly |  | 1026.446429 |
| Other |  | 1650.000000 |
| Name: el\_bill\_amt, dtype: float64 interviewer\_name | | |
| Gunjan Kumari | 1055.263158 | |
| KALPANA ratre | 2400.000000 | |
| Kalpana ratre | 400.000000 | |
| Navjeet kumar | 1349.200000 | |
| Rizwana | 1184.714286 | |
| Sangeeta Dahariya | 893.250000 | |
| Seema | -99.000000 | |
| Seema Bano | 775.400000 | |
| Simaila | 250.000000 | |
| Simaila Kumari | 1945.000000 | |
| Swati mishra | 4324.500000 | |
| Uma kumari | 1536.666667 | |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Umakumari | 3000.000000 |
| Urmila Bhagat | 150.000000 |
| Urmila Bhagat | 1655.000000 |
| Urmilla Bhagat | 1000.000000 |
| VARSHA Rajput | 100.000000 |
| Varsha Rajput | 100.666667 |
| Vikas Vikram | 690.250000 |
| navjeet kumar | 1084.000000 |
| seema Bano | 200.000000 |
|  | swati mishra | 1059.000000 |

Name: el\_bill\_amt, dtype: float64

*# Histogram for bill amount distribution* plt.figure(figsize=(10, 6)) sns.histplot(df['el\_bill\_amt'], kde=True) plt.title('Distribution of Bill Amounts') plt.xlabel('Bill Amount') plt.ylabel('Frequency')

plt.show()

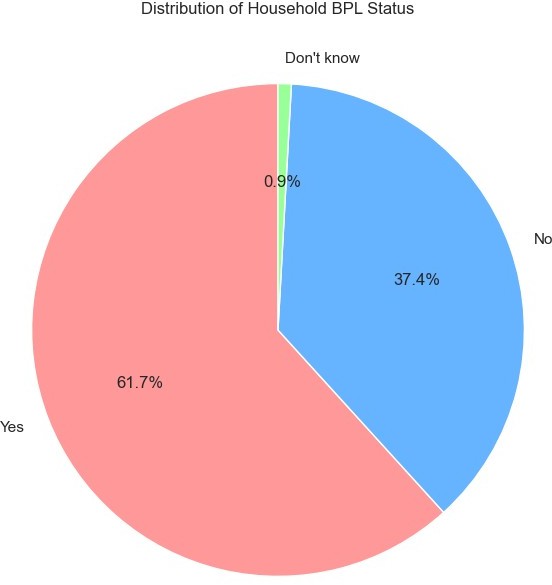


*# chart for household BPL statusie*

plt.figure(figsize=(8, 8)) df['hh\_BPL\_bin'].value\_counts().plot.pie(autopct='%1.1f%%', colors=['#ff9999','#66b3ff','#99ff99'], startangle=90)

plt.title('Distribution of Household BPL Status') plt.ylabel('')

plt.show()



from ydata\_profiling import ProfileReport

Report= ProfileReport(df, title ="Data Profiling Report", html={"style": {"full\_width": True}})

Report.to\_file(output\_file='analysis.html')

{"model\_id":"0da17c76da95443f996bed07cba4656f","version\_major":2,"vers ion\_minor":0}

{"model\_id":"515ac8cdf6634e78a972c0b64b227a16","version\_major":2,"vers ion\_minor":0}

{"model\_id":"ae3b184a29be407fa2747f0f0744008c","version\_major":2,"vers ion\_minor":0}

{"model\_id":"953ec8fa6406496cb64681f701139a27","version\_major":2,"vers ion\_minor":0}

df['el\_disconnect'] = df['el\_disdconnect'].replace({'1': 'Yes'})