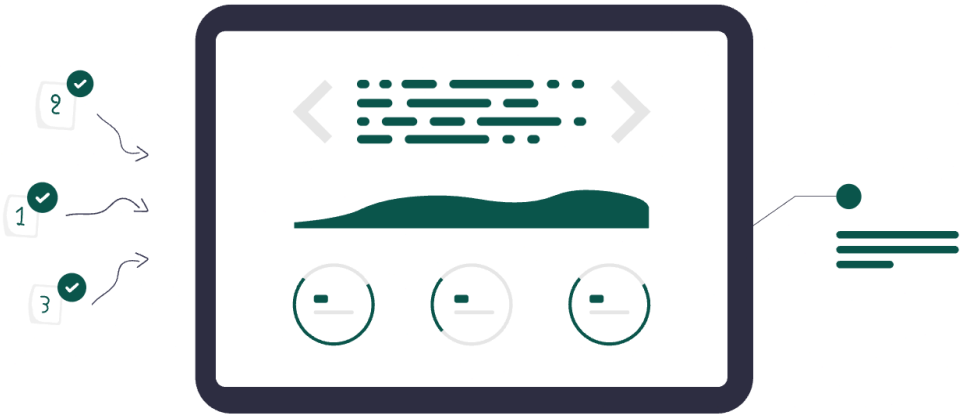
Assignment of CSE303

Analyzing Financial Factors in Loan Approvals



Submitted By:

Aklhak Hossain
2022-3-60-057
https://akhlak.dev
Group: I

Submitted To:

Puja Chakraborty
Lecturer
Department of CSE
East West University

Types of data?

```
# What are the data type?
print(df.dtypes)
# gives us input type
```

Column Name	Python Data Type	Qualitative / Quantitative	Nominal / Ordinal	Discrete / Continuous	Explanation
age	float64	Quantitative	–	Continuous	Age in years
employ	int64	Quantitative	–	Discrete	Years of employment
address	int64	Quantitative	–	Discrete	Years at current address
income	float64	Quantitative	–	Continuous	Annual income in currency
debtinc	float64	Quantitative	–	Continuous	Debt-to-income ratio
creddebt	float64	Quantitative	–	Continuous	Credit card debt
othdebt	float64	Quantitative	–	Continuous	Other debts
ed	float64	Quantitative	Ordinal	Discrete	Education level (assumed scale: 1 to 6, etc.)
default	int64	Qualitative	Nominal	Discrete	0 = No default, 1 = Default

Balanced & imbalanced dataset & the impact of them in the performance of a model

Balanced Dataset: A balanced dataset is a type of dataset that has roughly equal numbers of samples in each class.

For example: if we are looking into a dataset where 40% to 60% of the responders are male and the rest are female, meaning the ratio between males and female responders are 50:50 or 60:40, we can call it a balanced dataset.

Imbalanced Dataset: A balanced dataset is a type of dataset that has uneven class distribution.

For example: Examples include fraud detection (where fraud is a minority class) or medical diagnosis (where a rare disease might be the minority class). Even in the large set of data these will always be minor class.

Effect of data balance in prediction model: Imbalanced datasets can lead to models that are highly accurate on the majority class but perform poorly on the minority class. It also leads to a biased model toward the majority class.

A short description of your task and dataset, with columns in the dataset, type, range, and categories of the values

Task: To develop a model to predict what type of people are more likely to default. This is a binary classification task in which the target variable is default (0 = no default, 1 = default).

Dataset Description: The dataset consists of 850 rows and 9 columns. It contains continuous and discrete numeric variables such as age, work experience, debt, income and education. The response variable default refers to approval or denial status.

We can use `print(df.describe())` code to get more information about the dataset, Which gives me a output like:

```
              age      employ     address     income     debtinc    creddebt
\
count  680.000000  700.000000  700.000000  663.00000  700.000000  700.000000
mean    34.750000    8.388571    8.268571   45.74359   10.260571    1.553553
std      7.973215    6.658039    6.821609   37.44108    6.827234    2.117197
min     20.000000    0.000000    0.000000   14.00000    0.400000    0.011696
25%     28.000000    3.000000    3.000000   24.00000    5.000000    0.369059
50%     34.000000    7.000000    7.000000   34.00000    8.600000    0.854869
75%     40.000000   12.000000   12.000000   54.50000   14.125000    1.901955
max     56.000000   31.000000   34.000000  446.00000   41.300000   20.561310

          othdebt          ed     default
count  700.000000  680.000000  700.000000
mean     3.058209    1.717647    0.261429
std      3.287555    0.925652    0.439727
min      0.045584    1.000000    0.000000
25%      1.044178    1.000000    0.000000
50%      1.987567    1.000000    0.000000
75%      3.923065    2.000000    1.000000
max     27.033600    5.000000    1.000000
```

Which I can make a table like below:

Column	Type	Range (Min–Max)	Description
age	float64	20 – 56	Age of the applicant
employ	int64	0 – 31	Years of employment
address	int64	0 – 34	Years at current address
income	float64	14 – 446	Annual income in thousands
debtinc	float64	0.4 – 41.3	Debt-to-income ratio
creddebt	float64	0.01 – 20.56	Credit card debt
othdebt	float64	0.04 – 27.03	Other debts
ed	float64	1 – 5	Education level (ordinal)
default	int64	0 or 1	Loan status: 0 = No default, 1 = Defaulted

Dataset Balance Analysis

For the **Loans and Liability** Dataset, if we count the ratio of the "**default**" class we can get this via python script:

```
counts = df['default'].value_counts()
percentages = df['default'].value_counts(normalize=True) * 100

print("Class Distribution (Count):\n", counts)
print("\nClass Distribution (Percentage):\n", percentages)
```

We get the ratio like this:

```
Class Distribution (Count):
 default
0    517
1    183
Name: count, dtype: int64

Class Distribution (Percentage):
 default
0    73.857143
1    26.142857
Name: proportion, dtype: float64
```

This indicates that the data set is skewed. There is a strong imbalance, with the majority class (0) representing over three quarters of the data and the minority class (1) less than one quarter. This imbalance may cause the machine learning model to be skewed in favor of predicting the majority class and thus lower its ability to accurately predict defaults.

Statistical calculations over the dataset

I could write a code in python to get mean, median, variance and standard deviation of the data continuous numerical values:

```
continuous_cols = ['age', 'income', 'debtinc', 'creddebt', 'othdebt']

for col in continuous_cols:
    print(f"\n Statistics for {col}:")
    print("Mean:", df[col].mean())
    print("Median:", df[col].median())
    print("Variance:", df[col].var())
    print("Standard Deviation:", df[col].std())
```

And frequency and percentage of each type of values for categorical or discrete values:

```
discrete_cols = ['employ', 'address', 'ed', 'default']
print(discrete_cols)

for col in discrete_cols:
    print(f"\n Frequency and Percentage for {col}:")
    counts = df[col].value_counts()
    percentages = df[col].value_counts(normalize=True) * 100
    summary = pd.concat([counts, percentages], axis=1)
    summary.columns = ['Count', 'Percentage']
    print(summary)
```

We get the output like:

```
Continuous Columns:
['age', 'income', 'debtinc', 'creddebt', 'othdebt']

 Statistics for age:
Mean: 34.75
Median: 34.0
Variance: 63.57216494845361
Standard Deviation: 7.973215471091548

 Statistics for income:
Mean: 45.743589743589745
Median: 34.0
Variance: 1401.83445658068
Standard Deviation: 37.441079799875965
```

```
 Statistics for debtinc:
Mean: 10.260571428571428
Median: 8.6
Variance: 46.611118414060904
Standard Deviation: 6.827233584260971

 Statistics for creddebt:
Mean: 1.553552817142857
Median: 0.8548695
Variance: 4.482523082205469
Standard Deviation: 2.117196987104759

 Statistics for othdebt:
Mean: 3.0582086114285723
Median: 1.9875675
Variance: 10.808014786001815
Standard Deviation: 3.287554529738148

--------------------------------------------------------------------------------
Discrete Columns:
['employ', 'address', 'ed', 'default']

 Frequency and Percentage for employ:
        Count  Percentage
employ
0          62    8.857143
1          49    7.000000
4          47    6.714286
6          46    6.571429
9          45    6.428571
2          44    6.285714
3          42    6.000000
7          38    5.428571
5          36    5.142857
8          31    4.428571
12         30    4.285714
10         30    4.285714
13         27    3.857143
11         26    3.714286
16         25    3.571429
15         19    2.714286
18         17    2.428571
14         14    2.000000
22         13    1.857143
17         12    1.714286
19         12    1.714286
21          8    1.142857
```

```
20          5    0.714286
23          5    0.714286
24          4    0.571429
25          3    0.428571
31          3    0.428571
27          2    0.285714
30          2    0.285714
26          1    0.142857
29          1    0.142857
28          1    0.142857

 Frequency and Percentage for address:
         Count  Percentage
address
2           59    8.428571
1           57    8.142857
0           50    7.142857
4           49    7.000000
3           48    6.857143
6           43    6.142857
8           40    5.714286
9           39    5.571429
5           34    4.857143
7           34    4.857143
10          32    4.571429
11          27    3.857143
14          21    3.000000
12          20    2.857143
13          18    2.571429
16          18    2.571429
17          17    2.428571
15          16    2.285714
19          13    1.857143
21           9    1.285714
23           9    1.285714
18           9    1.285714
26           7    1.000000
25           7    1.000000
20           7    1.000000
22           7    1.000000
27           3    0.428571
24           3    0.428571
31           2    0.285714
29           1    0.142857
34           1    0.142857

 Frequency and Percentage for ed:
```

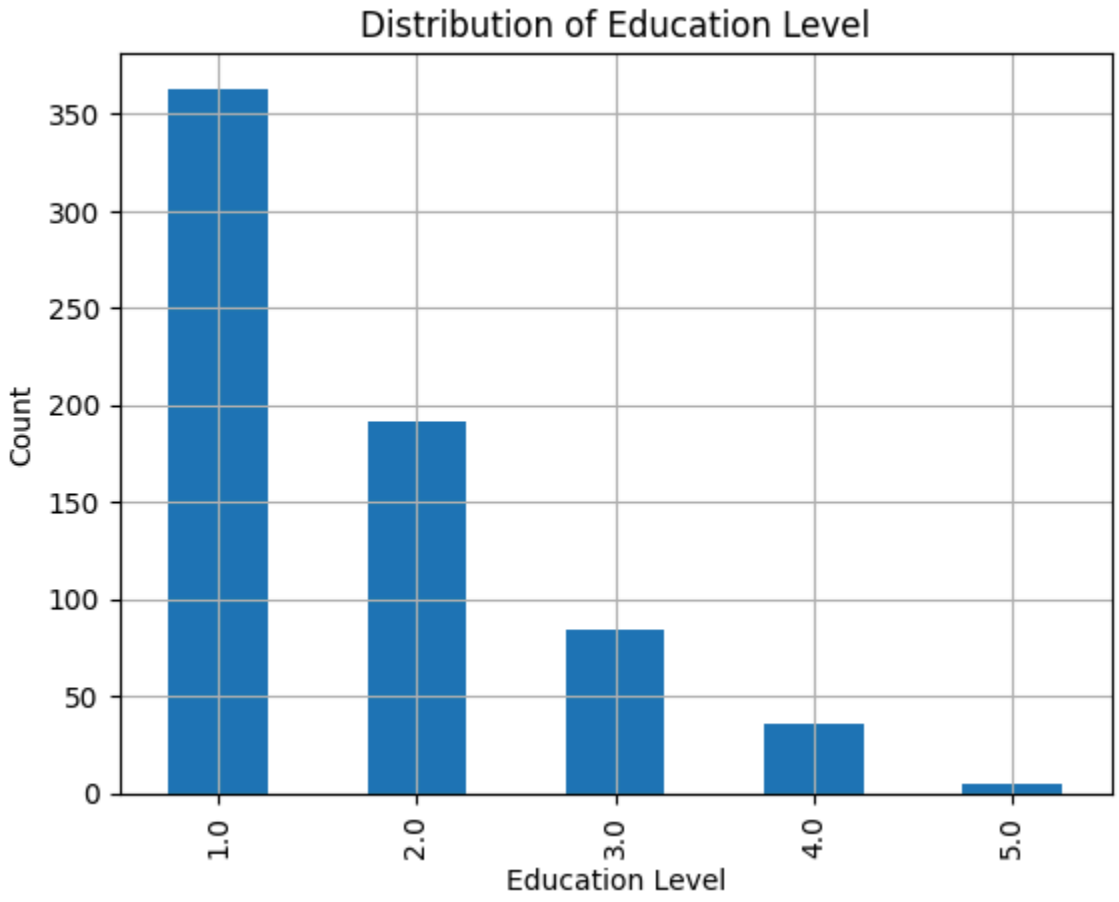
```
     Count  Percentage
ed
1.0    363   53.382353
2.0    192   28.235294
3.0     84   12.352941
4.0     36    5.294118
5.0      5    0.735294

 Frequency and Percentage for default:
         Count  Percentage
default
0          517   73.857143
1          183   26.142857
```

Data visualization by columns

Bar chart for Column ***ed***:

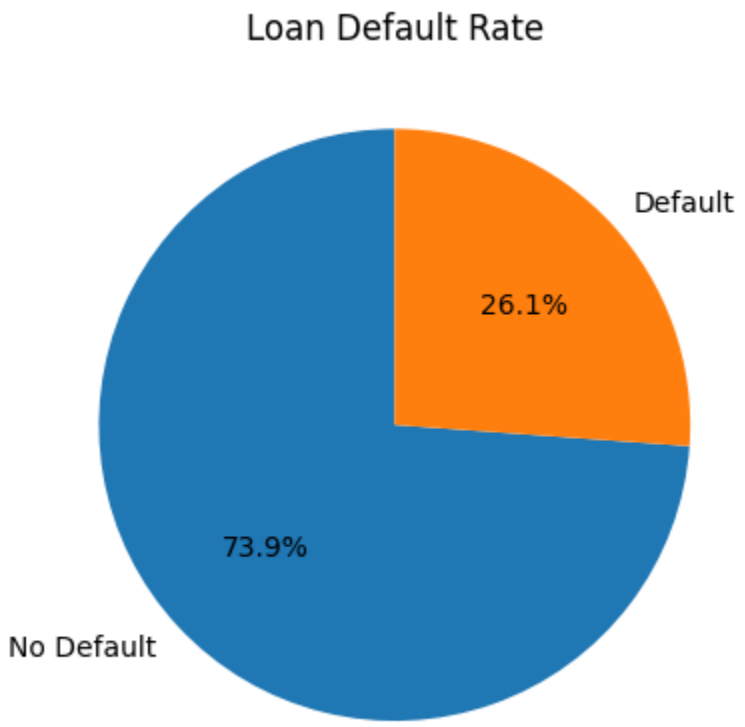
```
df['ed'].value_counts().sort_index().plot(kind='bar')
plt.title('Distribution of Education Level')
plt.xlabel('Education Level')
plt.ylabel('Count')
plt.grid(True)
plt.show()
```

Which gives a bar chart



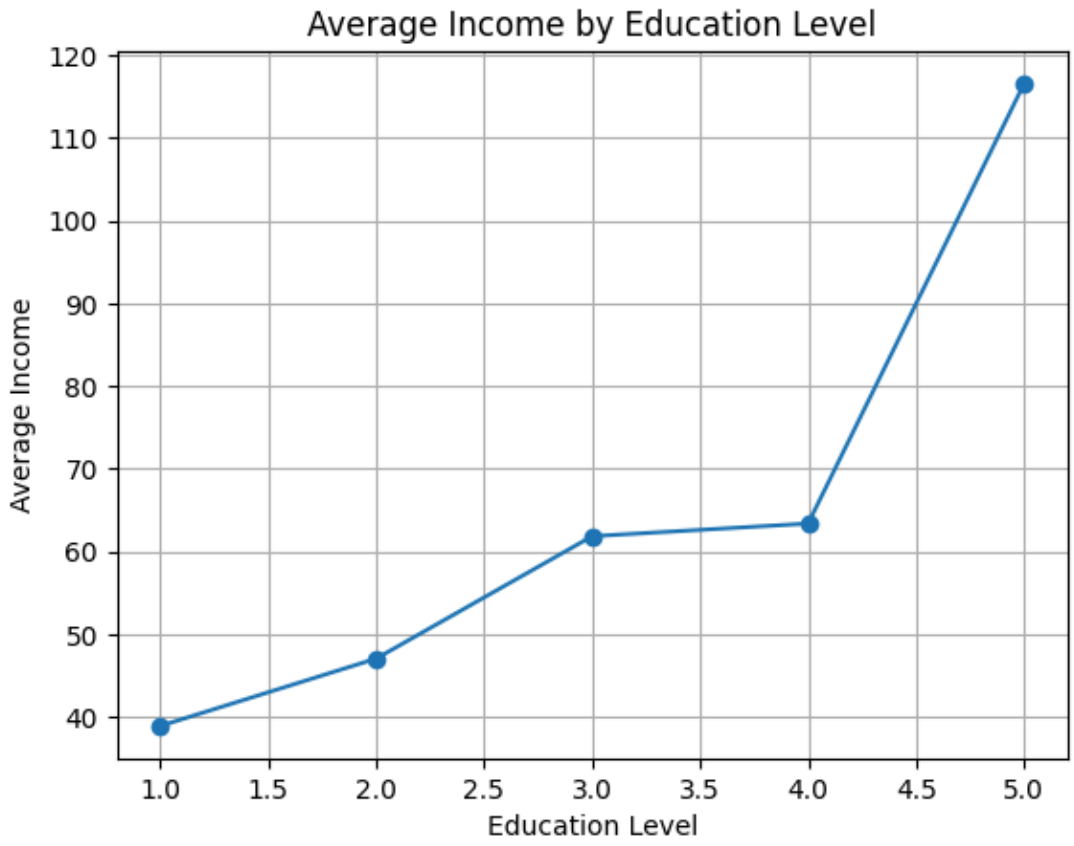
Pie Chart: Loan Default Distribution

```
df['default'].value_counts().plot(kind='pie', autopct='%1.1f%%', labels=['No
Default', 'Default'], startangle=90)
plt.title('Loan Default Rate')
plt.ylabel('')
plt.show()
```



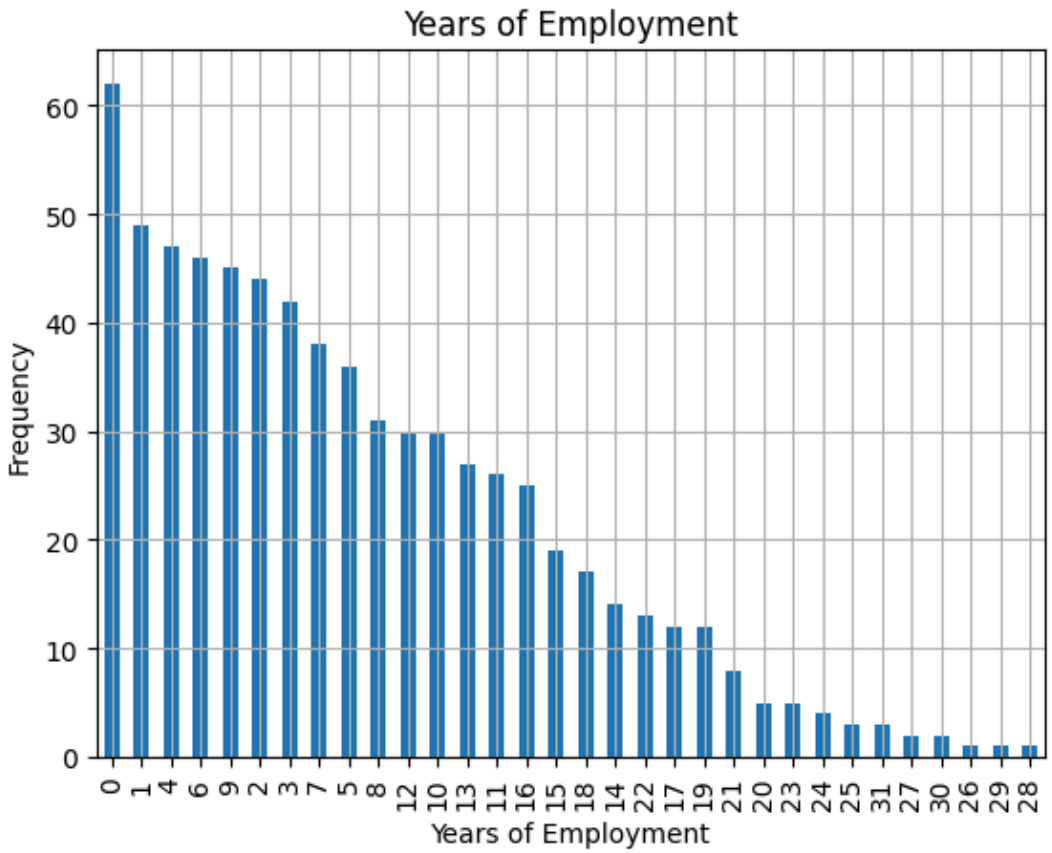
Line Graph: Average Income by Education Level

```
df.groupby('ed')['income'].mean().plot(kind='line', marker='o')
plt.title('Average Income by Education Level')
plt.xlabel('Education Level')
plt.ylabel('Average Income')
plt.grid(True)
plt.show()
```



Bar Chart: Employment Years

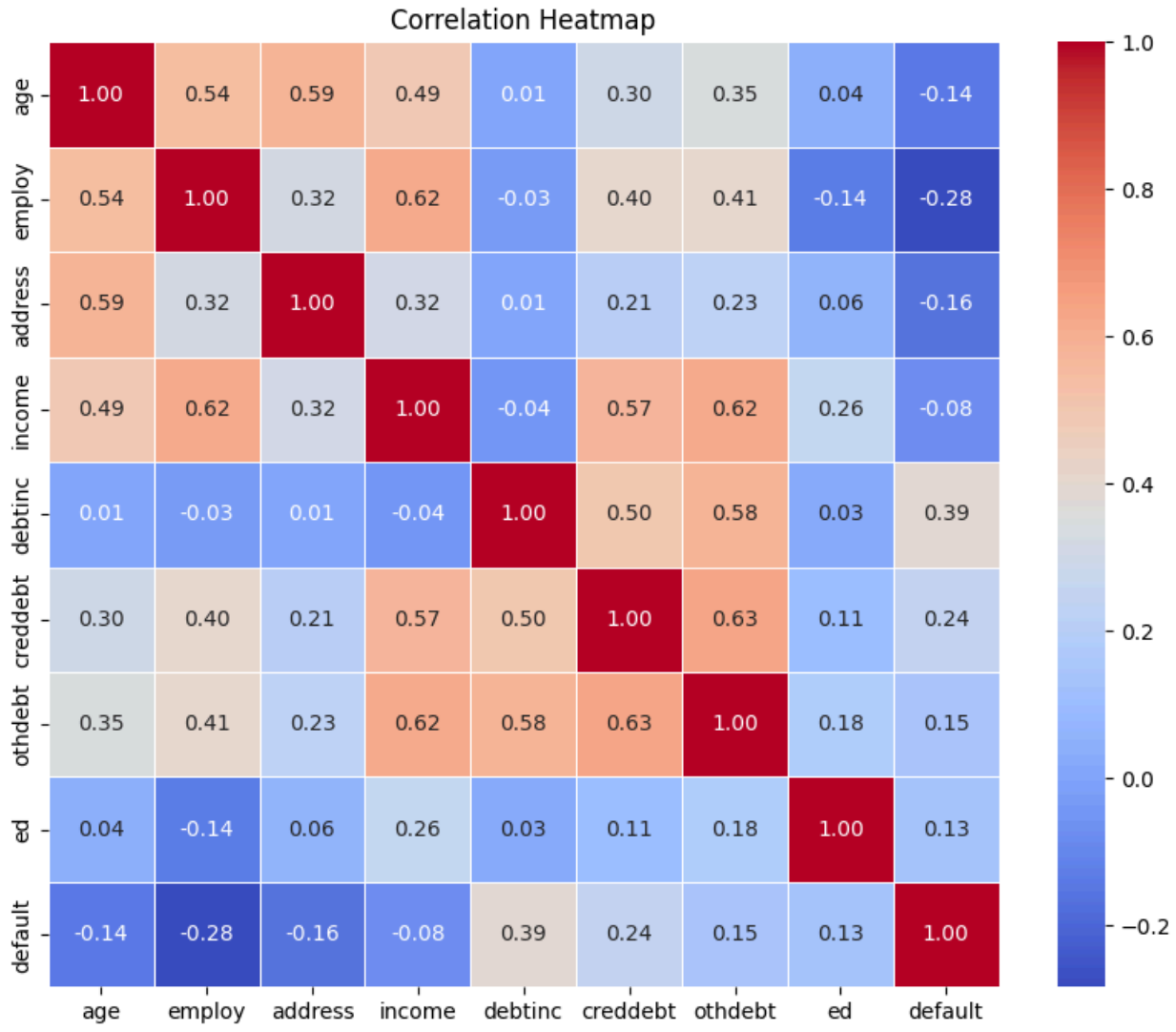
```
df['employ'].value_counts().sort_values(ascending=False).plot(kind='bar')
plt.title('Years of Employment')
plt.xlabel('Years of Employment')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()
```


Years of Employment
Frequency
60
50
40
30
20
10
0
0
1
4
6
9
2
3
7
5
8
12
10
13
11
16
15
18
14
22
17
19
21
20
23
24
25
31
27
30
26
29
28
Years of Employment

Correlation heat-map

Can be derived by:

```
import seaborn as sns

corr = df.corr()

plt.figure(figsize=(10, 8))
sns.heatmap(corr, annot=True, cmap='coolwarm', fmt=".2f", linewidths=0.5)
plt.title('Correlation Heatmap')
plt.show()
```



Conversion of categorical values to numerical values

Numerical inputs are required by machine learning models. Therefore, categorical data must be represented as numerical values before training models. two commonly used approaches:

Label Encoding: method that assigns a unique integer to each category. Ideal for ordinal variables where categories have a logical order (ed: 1 to 5).
Example: Low 0, Medium 1, High 2

One-Hot Encoding: method has an alternate binary column for each category and is used for nominal variables with no ordering.
Example: Red, Blue, Green are three columns with values 0 or 1.

The majority of the features in this dataset are already numeric. If there were any categorical text columns, unordered categories would have one-hot encoding, and ordered ones would have label encoding applied to them.

Handling missing values

To check for missing values we can run

```
print(df.isnull().sum())
```

Which gives:

```
age         20
employ       0
address      0
income      37
debtinc      0
creddebt     0
othdebt      0
ed          20
default      0
dtype: int64
```

For **age** & **income** we can replace them with the **mean** cause it's less sensitive to the outliers, and for **ed** we can take a **mode** as it is an ordinal value

Code to handle the changes:

```
df['age'] = df['age'].fillna(df['age'].median())
df['income'] = df['income'].fillna(df['income'].median())
df['ed'] = df['ed'].fillna(df['ed'].mode()[0])
```

Conclusion

This project was meant to analyze financial and personal factors impacting loan default choices. The data set had 700 instances and 9 numerical features such as age, income, work record, debt burden, and education. The target variable default was imbalanced with 73.86% non-default instances and 26.14% default instances.

Statistical visualization and modeling revealed income, credit debt, and debt-to-income ratio to be the relevant variables. Income, education, and age had missing values that were handled using median and mode imputation in order to retain data. There are no missing values in the dataset now, and it is fully numeric and therefore ready to be used in machine learning.

Feature encoding and correlation analysis were also performed to render the data compatible and discover correlations. The dataset is now well prepared for creating predictive models to assess loan risks.

Reference Links

Github Url:
https://github.com/Akhlak-Hossain-Jim/Learning-CSE-at-EWU/tree/main/Semester-8/CSE303/Lab/Section%204/Asssignment%201