Kmit

SONET

Data Science

(Level-I)

Ajeet K. Jain, M. Narsimlu

(ML TEAM)- SONET, KMIT, Hyderabad

Session – 27

This session deals with

Why data Preprocessing

Data Preprocessing

Types of Data

Handling Numerical and Categorical

Introduction

Why preprocessing ?

Real world data are generally

Incomplete: lacking attribute values, lacking certain attributes of interest, or containing only aggregate data

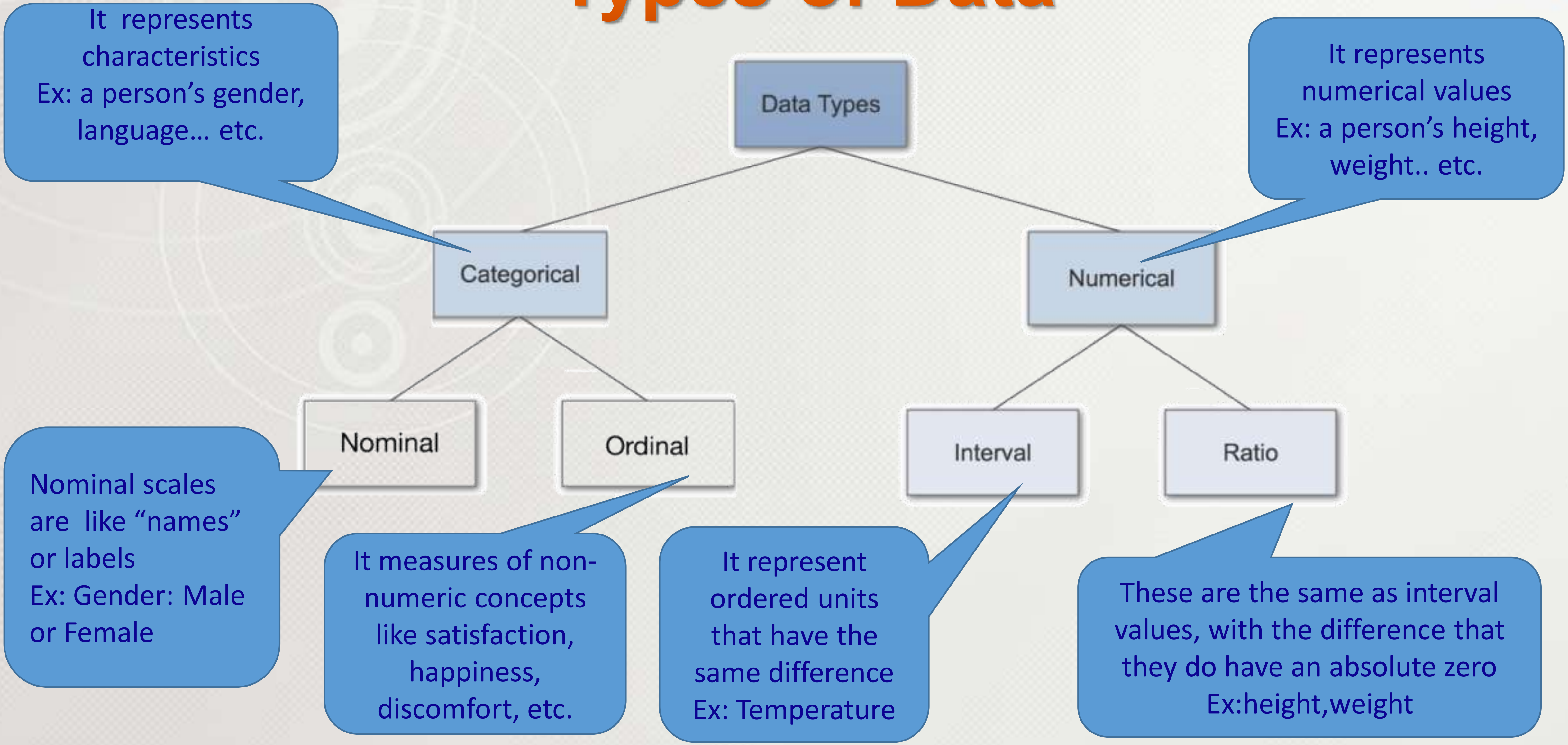
Noisy: containing errors or outliers

Inconsistent: containing discrepancies in codes or names Tasks in data preprocessing

Introduction

- Data pre-processing is an important step of solving every machine learning problem.
- Most of the datasets used with Machine Learning problems need to be processed / cleaned / transformed so that a Machine Learning algorithm can be trained on it.
- Most commonly used pre-processing techniques are very few like - missing value imputation, encoding categorical variables, scaling, etc.

Types of Data



Introduction

- It involves in every machine learning problem.
- Most of the datasets need to be processed / cleaned / transformed
- Pre-processing techniques - missing value imputation, encoding categorical variables, scaling, etc.

Handling Numeric data

Handling NAs

Outliers

Scaling / Normalization

SONET

Handling Numeric data

DATA SCIENCE

Handling NAs

- Removing
  - Too many Nulls
    - Drop Entire column
  - Few Nulls
    - Drop row using dropna()
- Filling
  - Filling with A value
  - Filling with Mean, Median, Mode

Handling Numerical data

1. Load Data Set and find the missing values

```
import pandas as pd
import matplotlib.pyplot as plt
data_loan=pd.read_csv("loan.csv")
print(data_loan.isnull().sum())
```

Output

```
Loan ID                               0
Customer ID                           0
Loan Status                           0
Current Loan Amount                   0
Term                                  0
Credit Score                      19154
Annual Income                     19154
Years in current job               4222
Home Ownership                        0
Purpose                               0
Monthly Debt                          0
Years of Credit History               0
Months since last delinquent      53141
Number of Open Accounts               0
Number of Credit Problems             0
Current Credit Balance                0
Maximum Open Credit                   2
Bankruptcies                        204
Tax Liens                            10
dtype: int64
```

Handling Numerical data

Dropping Entire column

```
import pandas as pd
import matplotlib.pyplot as plt
data_loan=pd.read_csv("loan.csv")
df_col=data_loan.drop(["Months since last delinquent"],axis=1)
print(df_col.head())
```

Handling Numerical data

Dropping all null values

```
import pandas as pd
import matplotlib.pyplot as plt
data_loan=pd.read_csv("loan.csv")
data_loan.dropna(inplace=True)
print(data_loan.isnull().sum())
```

Handling Numerical data

2. Check is there any null values

3. fillna() method –filling null values with ‘1’

```
import pandas as pd
import matplotlib.pyplot as plt
data_loan=pd.read_csv("loan.csv")
print(data_loan.isnull().sum())
data_loan.fillna(1,inplace=True)
print(data_loan.isnull().sum())
```

Handling Numerical data

Filling missing data with mean

```
import pandas as pd
import matplotlib.pyplot as plt
data_loan=pd.read_csv("loan.csv")
print(data_loan.isnull().sum())
data_loan_up=data_loan["Credit Score"].fillna(data_loan["Credit Score"].mean())
print(data_loan_up.isnull().sum())
```

Exercise-2

Read loan data set and perform following tasks

1.Fill Annual income with mean

2.Remove null values from Bankruptcies

4.Remove special characters from "Years in current job“

5.Convert the "Years in current job“ into float

6.Display top five records

Solution

```
import pandas as pd
data_loan=pd.read_csv("loan.csv")
print(data_loan.isnull().sum())
data_loan["Credit Score"].fillna(data_loan["Credit Score"].mean(),inplace=True)
data_loan["Annual Income"].fillna(data_loan["Annual Income"].mean(),inplace=True)
data_loan["Bankruptcies"].dropna(inplace=True)
print(data_loan.isnull().sum())
print(data_loan["Years in current job"].head())
data_loan["Years in current job"]=data_loan["Years in current job"].replace(
        '[+<a-z]','',regex=True)
data_loan["Years in current job"]=data_loan["Years in current job"].astype("float")
print(data_loan["Years in current job"].head())
```

Output

```
Loan ID                             0
Customer ID                         0
Loan Status                         0
Current Loan Amount                 0
Term                                0
Credit Score                    19154
Annual Income                   19154
Years in current job             4222
Home Ownership                      0
Purpose                             0
Monthly Debt                        0
Years of Credit History             0
Months since last delinquent    53141
Number of Open Accounts             0
Number of Credit Problems           0
Current Credit Balance              0
Maximum Open Credit                 2
Bankruptcies                      204
Tax Liens                          10
dtype: int64
```

Output

```
Loan ID                             0
Customer ID                         0
Loan Status                         0
Current Loan Amount                 0
Term                                0
Credit Score                        0
Annual Income                       0
Years in current job             4222
Home Ownership                      0
Purpose                             0
Monthly Debt                        0
Years of Credit History             0
Months since last delinquent    53141
Number of Open Accounts             0
Number of Credit Problems           0
```

```
Current Credit Balance          0
Maximum Open Credit             2
Bankruptcies                  204
Tax Liens                      10
dtype: int64
0      8 years
1    10+ years
2      8 years
3      3 years
4      5 years
Name: Years in current job, dtype: object
```

Output

```
0     8.0
1    10.0
2     8.0
3     3.0
4     5.0
Name: Years in current job, dtype: float64
```

Handling Numerical data

Handling Duplicated Values:

```
import pandas as pd
data=pd.read_csv("loan.csv")
dup_Anual_inc=data.duplicated(["Annual Income"])
print(dup_Anual_inc.sum())
```

```
data_pr
NPTEL_P
63825
```

Dropping Duplicated Values:

```
import pandas as pd
data=pd.read_csv("loan.csv")
dup1=data.drop_duplicates(["Annual Income"],keep="first")
dup_Anual_inc=dup1.duplicated(["Annual Income"])
print(dup_Anual_inc.sum())
```

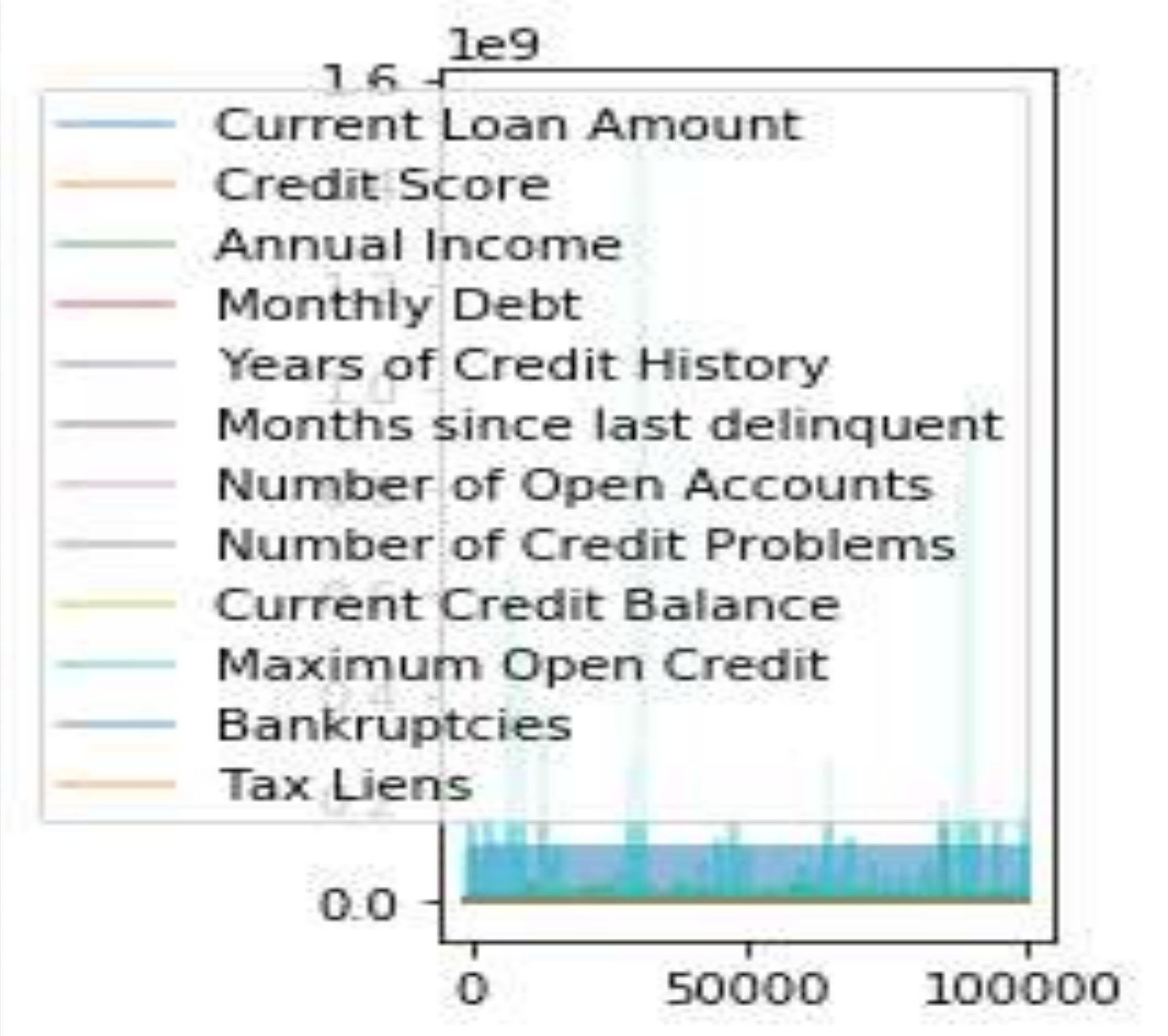
```
SO
NP
0

In
```

Handling Numerical data

Skewness : is a measure of the symmetry in a distribution. A symmetrical dataset will have a Skewness equal to 0. Skewness essentially measures the relative size of the two tails.

```
import pandas as pd
import matplotlib.pyplot as plt
data=pd.read_csv("loan.csv")
data_skew=data.skew()
data.plot(alpha=0.5,figsize=(2,4))
plt.show()
```

Output


1e9
1.6
Current Loan Amount
Credit Score
Annual Income
Monthly Debt
Years of Credit History
Months since last delinquent
Number of Open Accounts
Number of Credit Problems
Current Credit Balance
Maximum Open Credit
Bankruptcies
Tax Liens
0.0
0
50000
100000

Outliers

An *outlier* is an observation that lies an abnormal distance from other values in a random sample from a population.

To remove outliers we will use

- np.log() Natural logarithm, element-wise.
- np.sqrt() square root
- np.cbrt() cube root

Handling Outliers

Check the difference between minimum value and maximum value

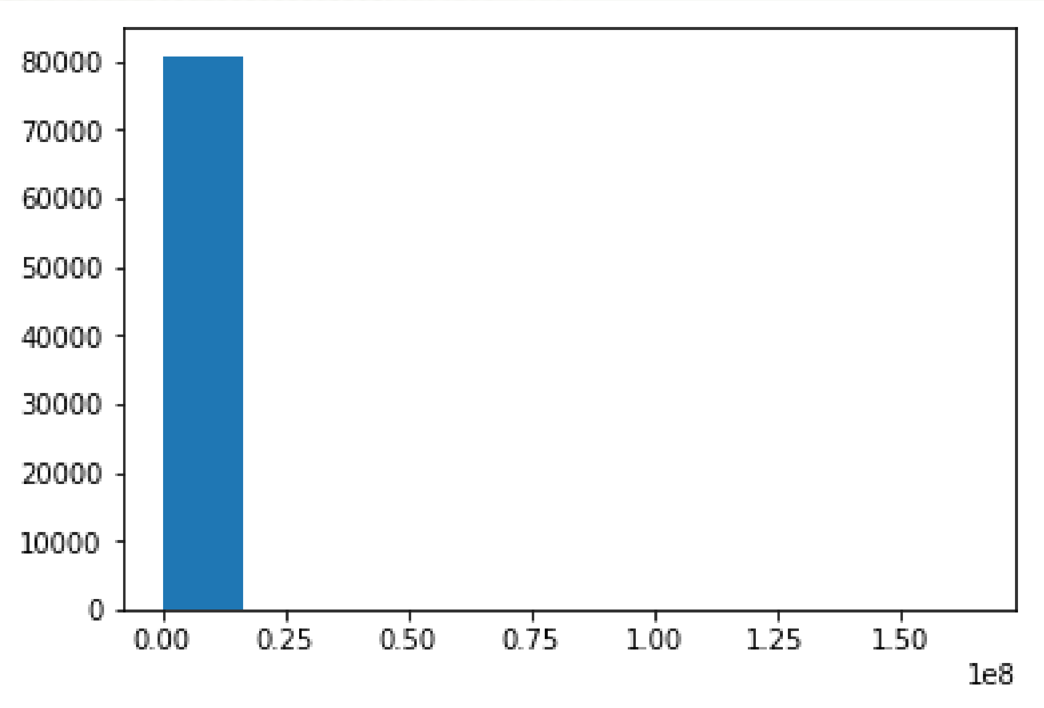
```
import pandas as pd
import matplotlib.pyplot as plt
data=pd.read_csv("loan.csv")
print(data["Annual Income"].min())
print(data["Annual Income"].max())
print(data["Annual Income"].mean())
```

```
NPTEL_Assignments/
76627.0
165557393.0
1378276.559842169

In [3]:
```

Handling Outliers

Histogram for original 'Annual Income' data

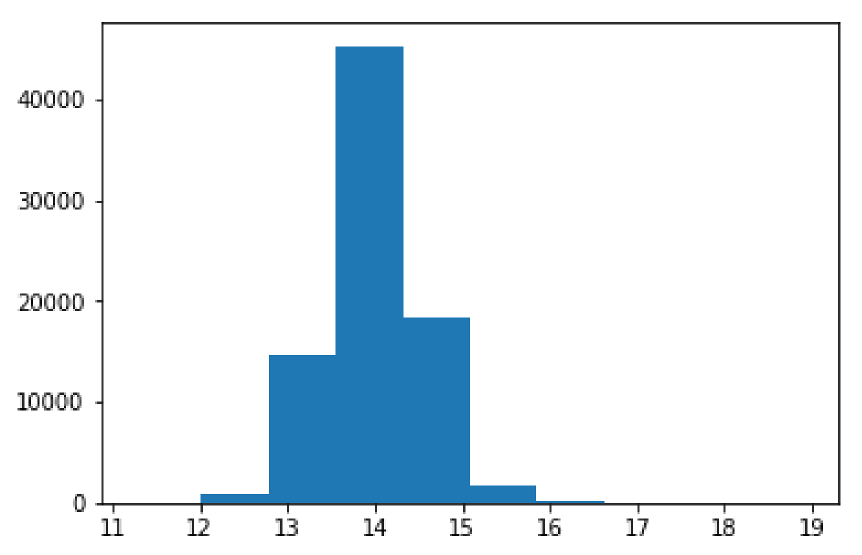
```python
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
data=pd.read_csv("loan.csv")
plt.hist(data["Annual Income"])
plt.show()
```



Handling Outliers

Handling outliers with numpy.log():
This mathematical function helps user to calculate **Natural logarithm of x** where x belongs to all the input elements.

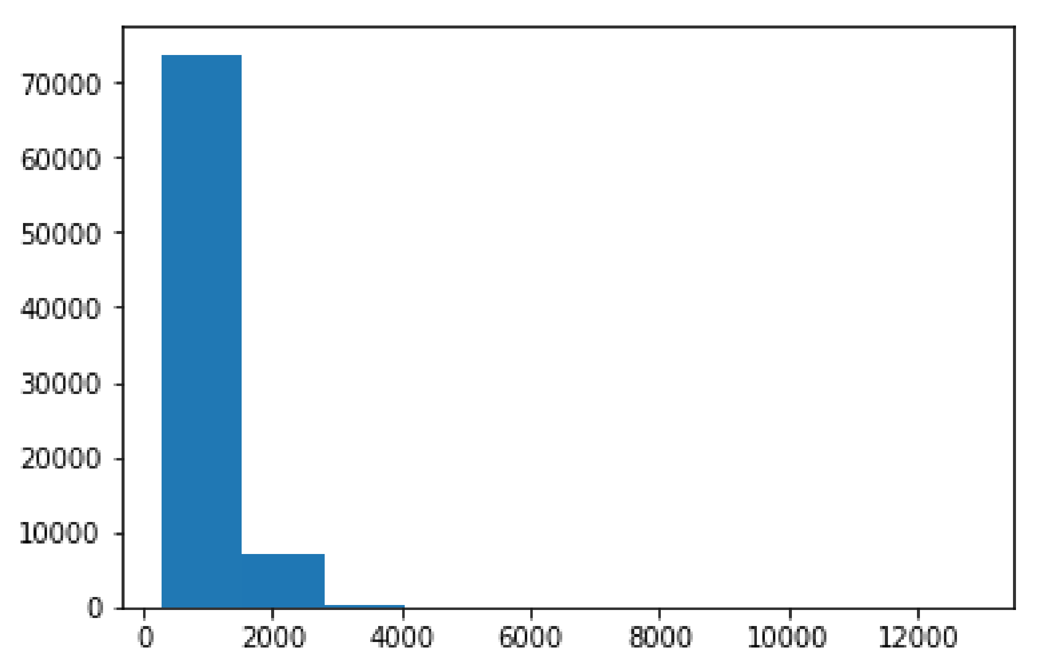
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
data=pd.read_csv("loan.csv")
data_log=np.log(data["Annual Income"])
data_log.plot.hist(alpha=2)
plt.show()
```



Handling Outliers

numpy.sqrt():
Return the positive square-root of an array(data), element-wise.

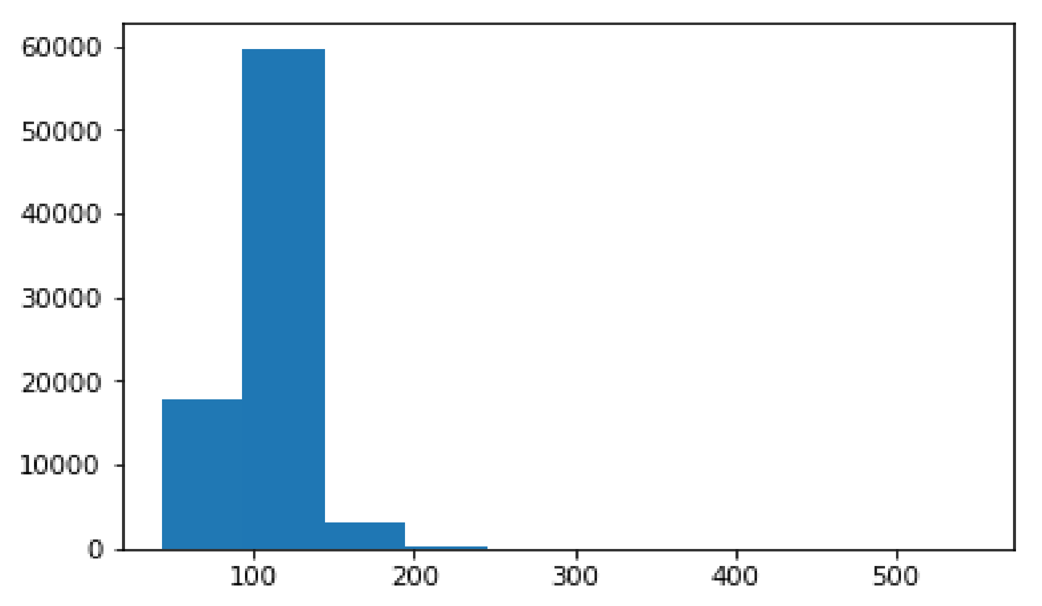
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
data=pd.read_csv("loan.csv")
data_log=np.sqrt(data["Annual Income"])
data_log.plot.hist(alpha=2)
plt.show()
```



Handling Outliers

numpy.cbrt():
This mathematical function helps user to calculate cube root of x for all x being the array elements.

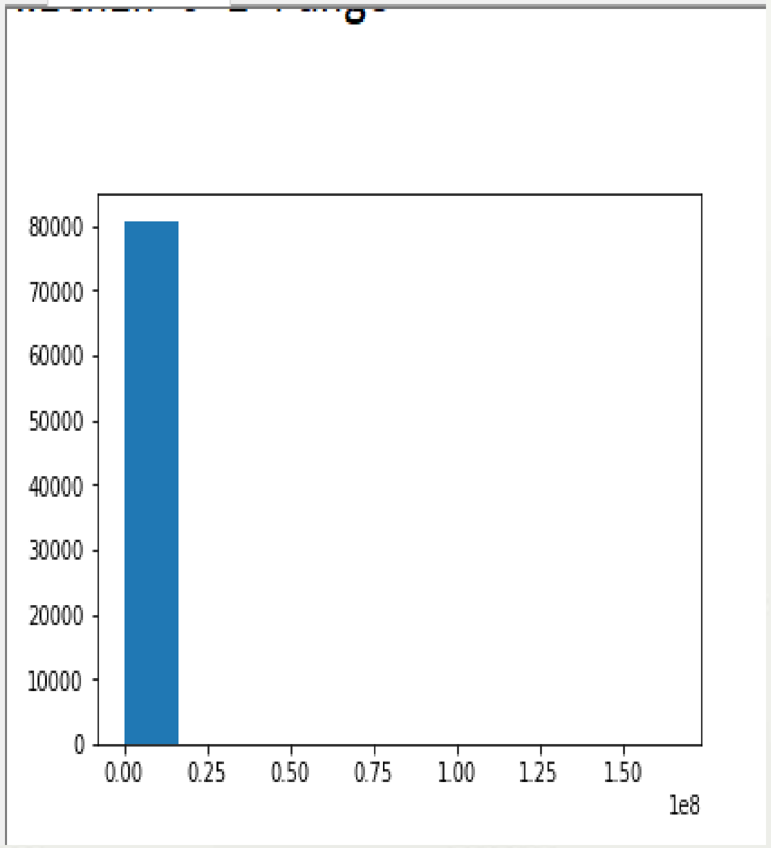
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
data=pd.read_csv("loan.csv")
data_cbrt=np.cbrt(data["Annual Income"])
data_cbrt.plot.hist(alpha=2)
plt.show()
```



Handling Outliers

Using Percentile:

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
data=pd.read_csv("loan.csv")
per=data["Annual Income"]
per1=np.percentile(per,0.25)
per.plot.hist(alpha=2)
plt.plot(per1)
plt.show()
```



Normalization

It is the process of reorganizing data in a dataset so that it meets two basic requirements:

(1)There is no redundancy of data (all data is stored in only one place), and

(2)data dependencies are logical (all related data items are stored together)

Normalizing in scikit-learn refers to rescaling each observation (row) to have a length of 1 (called a unit norm in linear algebra).

This preprocessing can be useful for sparse datasets (lots of zeros) with attributes of varying scales when using algorithms that weight input values such as neural networks and algorithms that use distance measures such as K-Nearest Neighbors.

Normalization

Normalizing in scikit-learn refers to rescaling each observation (row) to have a length of 0 to1

```
In [60]: import pandas as pd
         sweptarea=df['sweptarea']
         def normalize_list(sweptarea):
             max_value = max(sweptarea)
             min_value = min(sweptarea)
             for i in range(0, len(sweptarea)):
                 sweptarea[i] = (sweptarea[i] - min_value) / (max_value - min_value)
         for i in range(0, len(sweptarea)):
             print(sweptarea[i])

         0.17786849655999748
         0.20457497464201144
         0.0
         0.2281144668108021
         0.13335773765560585
         0.0
         0.1712660194000922
         0.17108696917202698
         0.19946643032036562
         0.17786849655999748
         0.0013115429205777952
         0.17842802852270134
```

Categorical data

Categorical Attributes –

- When the number unique values in a categorical column are too high, check the value counts of each of those values. Replace rarely occurring values together into a single value like ‘Other’ before encoding.
- When number of unique values is huge and even the values are equally distributed, try to find some related values and see if the multiple categorical values can be clubbed into single (grouping), thereby reducing the count of categorical values.

Related Attributes –

- If there multiple attributes with same information with different granularity, like city and state, it’s better to keep columns like state and delete city column. Additionally, keeping both columns and assessing feature importance might help in eliminating one column.

Handling Categorical data

1. Label encoding
2. Range encoding
3. one-hot encoding

Handling Categorical data

1.Label encoding

Sex	New_sex
Male	1
Female	0
Female	0
Male	1
Male	1

2.Range encoding

Height	Avg_Height	Low_Height	High_Height
100-110	105	100	100
110-120	115	110	110
120-130	125	120	120
130-140	135	130	130
140-150	145	140	150

Handling Categorical data

2.one-hot encoding

Hyderabad
Guntur
Hyderabad
Vizag
Vizag

city	New_city_hyd	New_city_gnt	New_city_viz
Hyderabad	1	0	0
Guntur	0	1	0
Hyderabad	1	0	0
Vizag	0	0	1
Vizag	0	0	1

Categorical data

Label Encoder	One Hot Encoder
Numeric representation, ordinals	Binary representation
Loses uniqueness of values, single dimension in vector space	Individual values expressed as a different dimension in orthogonal vector space
Suitable with categorical values that are ordinal in nature, like – fog_level (low, medium, high)	Suitable with non-ordinal types of categorical attributes, like – car_type (hatchback, sedan, SUV, etc.)
Label encoded categorical attributes don't pose any further challenges	One hot encoded categorical attributes might dramatically increase the feature space (curse of dimensionality). When One hot encoding is used, it's often followed by PCA to tackle high-dimensionality

Conclusion

You are aware of

Data Visualization

Data Interpretation

We will proceed with

Data Preprocessing

THANK YOU