Name:KAushik Kotian **Roll no no.:30** **Div:D15B** **Batch :B**

Experiment No:01

Aim: Data preparation using NumPy and Pandas

Theory:

Data Preprocessing:

Definition:
Data preprocessing is a crucial step in the data analysis and machine learning pipeline. It involves cleaning and transforming raw data into a format suitable for analysis or training machine learning models. The process includes handling missing values, removing duplicates, scaling, encoding categorical variables, and addressing outliers.
Purpose:
Data preprocessing enhances the quality and reliability of the data, making it suitable for analysis and model training. It helps to mitigate the impact of noise and inconsistencies in the data, leading to more accurate and robust results.

Numpy:

NumPy (Numerical Python) is a powerful library for numerical computing in Python. It provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays efficiently.
Key Features:

- Efficient array operations and mathematical functions.
- Broadcasting to perform operations on arrays of different shapes.
- Integration with other data analysis libraries like Pandas.

Use Cases:
NumPy is extensively used in scientific computing, machine learning, and data analysis for its array handling capabilities and numerical operations.

Pandas:

Pandas is a data manipulation and analysis library for Python. It provides data structures like DataFrame and Series, which are powerful tools for handling and analyzing structured data.
Key Features:

- DataFrame and Series for easy data manipulation.
- Built-in functions for data cleaning, exploration, and transformation.
- Integration with other libraries like NumPy and scikit-learn.

Use Cases:
Pandas is widely used for data cleaning, exploration, and preprocessing tasks in data science and analysis projects.

Normalization:

Normalization is the process of scaling numerical features to a standard range, often between 0 and 1. This ensures that all features contribute equally to the analysis or model training. In scikit-learn, the StandardScaler is commonly used for normalization.

Commands:

1.Importing Libraries and Packages :

```
[1]  import numpy as np
     import pandas as pd
```

2.Load dataset into Pandas:

pd.read_csv()

The following command will load the data in pandas and will show us some rows and columns from our dataset.

```
df=pd.read_csv("jobsdata.csv")
```

3.Description of the dataset.

df.info()

This method prints information about a DataFrame including the index data type and columns, non-null values and memory usage.

```
[23] df.info()

     <class 'pandas.core.frame.DataFrame'>
     RangeIndex: 9355 entries, 0 to 9354
     Data columns (total 12 columns):
      #   Column              Non-Null Count  Dtype 
     ---  ------              --------------  ----- 
      0   work_year           9355 non-null   int64 
      1   job_title           9355 non-null   object
      2   job_category        9355 non-null   object
      3   salary_currency     9355 non-null   object
      4   salary              9355 non-null   int64 
      5   salary_in_usd       9355 non-null   int64 
      6   employee_residence  9355 non-null   object
      7   experience_level    9355 non-null   object
      8   employment_type     9355 non-null   object
      9   work_setting        9355 non-null   object
      10  company_location    9355 non-null   object
      11  company_size        9355 non-null   object
     dtypes: int64(3), object(9)
     memory usage: 877.2+ KB
```

4.Drop column that are not useful:

df.drop():

The drop() function is used to drop specified labels from rows or columns. Remove rows or columns by specifying label names and corresponding axis, or by specifying directly index or column names.

labels >> Index or column labels to drop.

axis >> Whether to drop labels from the index (0 or ‘index’) or columns (1 or ‘columns’).

```
[25] cols = ['employee_residence','company_size','work_setting']
     df =df.drop(cols,axis=1)
```

```
[26] df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9355 entries, 0 to 9354
Data columns (total 9 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   work_year          9355 non-null   int64
 1   job_title          9355 non-null   object
 2   job_category       9355 non-null   object
 3   salary_currency    9355 non-null   object
 4   salary             9355 non-null   int64
 5   salary_in_usd      9355 non-null   int64
 6   experience_level   9355 non-null   object
 7   employment_type    9355 non-null   object
 8   company_location   9355 non-null   object
dtypes: int64(3), object(6)
memory usage: 657.9+ KB
```

5.Take Care of missing values:

Let’s compute a median or interpolate() the 'Código ISO del país'
and fill those missing values. Pandas has an interpolate() function that will replace all the missing NaNs to interpolated values.

```
[25] cols = ['employee_residence','company_size','work_setting']
     df =df.drop(cols,axis=1)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9355 entries, 0 to 9354
Data columns (total 9 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   work_year          9355 non-null   int64
 1   job_title          9355 non-null   object
 2   job_category       9355 non-null   object
 3   salary_currency    9355 non-null   object
 4   salary             9355 non-null   int64
 5   salary_in_usd      9355 non-null   int64
 6   experience_level   9355 non-null   object
 7   employment_type    9355 non-null   object
 8   company_location   9355 non-null   object
dtypes: int64(3), object(6)
memory usage: 657.9+ KB
```

6.Create Dummy Variables:

pandas.get_dummies() is used for data manipulation. It converts categorical data into dummy or indicator variables.

```
[30] dummies = []
     cols = ['experience_level', 'employment_type']
     for col in cols:
        dummies.append(pd.get_dummies(df[col]))
```

```
[31] temp =pd.concat(dummies,axis=1)
```

```
df = pd.concat((df,temp),axis=1)
```

```
[33] df = df.drop(['experience_level', 'employment_type'],axis=1)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9355 entries, 0 to 9354
Data columns (total 15 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   work_year         9355 non-null   int64
 1   job_title         9355 non-null   object
 2   job_category      9355 non-null   object
 3   salary_currency   9355 non-null   object
 4   salary            9355 non-null   int64
 5   salary_in_usd     9355 non-null   int64
 6   company_location  9355 non-null   object
 7   Entry-level       9355 non-null   uint8
 8   Executive         9355 non-null   uint8
 9   Mid-level         9355 non-null   uint8
 10  Senior            9355 non-null   uint8
 11  Contract          9355 non-null   uint8
 12  Freelance         9355 non-null   uint8
 13  Full-time         9355 non-null   uint8
 14  Part-time         9355 non-null   uint8
dtypes: int64(3), object(4), uint8(8)
memory usage: 584.8+ KB
```

7.Finding Outliers Manually:

In simple terms, an outlier is an extremely high or extremely low data point relative to the nearest data point and the rest of the neighboring co-existing values in a data graph or dataset you're working with. Outliers can give helpful insights into the data you're studying, and they can have an effect on statistical results. This can potentially help you discover inconsistencies and detect any errors in your statistical processes.

8.Normalization :

In statistics and machine learning, min-max normalization of data is a process of converting the original range of data to the range between 0 and 1. The resulting normalized values represent the original data on 0-1 scale. This will allow us to compare multiple features together and get more relevant information since now all the data will be on the same scale.

i) Using min-max :

```
[39]
    from sklearn.preprocessing import MinMaxScaler
```

```
[40] scaler = MinMaxScaler()
     for column in numerical_columns:
        df[column]= scaler.fit_transform(df[[column]])
```

```
[41] print(df.head())
```

```
   work_year           job_title                   job_category  \
0       2023  Data DevOps Engineer               Data Engineering
1       2023       Data Architect  Data Architecture and Modeling
2       2023       Data Architect  Data Architecture and Modeling
3       2023       Data Scientist       Data Science and Research
4       2023       Data Scientist       Data Science and Research

  salary_currency  salary  salary_in_usd company_location  Entry-level  \
0             EUR   88000       0.183936          Germany            0
1             USD  186000       0.393103    United States            0
2             USD   81800       0.153563    United States            0
3             USD  212000       0.452874    United States            0
4             USD   93300       0.180000    United States            0

   Executive  Mid-level  Senior  Contract  Freelance  Full-time  Part-time
0          0          1       0         0          0          1          0
1          0          0       1         0          0          1          0
2          0          0       1         0          0          1          0
3          0          0       1         0          0          1          0
4          0          0       1         0          0          1          0
```

ii) Using Mean:

```
[36] numerical_columns=['salary_in_usd']
```

```
[37] for column in numerical_columns:
       mean_value = df[column].mean()
       df[column] = (df[column]-mean_value) / df[column].std()
```

```
print(df.head())
```

```
   work_year           job_title                   job_category  \
0       2023  Data DevOps Engineer               Data Engineering
1       2023       Data Architect  Data Architecture and Modeling
2       2023       Data Architect  Data Architecture and Modeling
3       2023       Data Scientist       Data Science and Research
4       2023       Data Scientist       Data Science and Research

  salary_currency  salary  salary_in_usd company_location  Entry-level  \
0             EUR   88000      -0.875115          Germany            0
1             USD  186000       0.565084    United States            0
2             USD   81800      -1.084241    United States            0
3             USD  212000       0.976623    United States            0
4             USD   93300      -0.902214    United States            0

   Executive  Mid-level  Senior  Contract  Freelance  Full-time  Part-time
0          0          1       0         0          0          1          0
1          0          0       1         0          0          1          0
2          0          0       1         0          0          1          0
3          0          0       1         0          0          1          0
4          0          0       1         0          0          1          0
```

Standarization:

In statistics and machine learning, data standardization is a process of converting data to z-score values based on the mean and standard deviation of the data. The resulting standardized value shows the number of standard deviations the raw value is away from the mean. Basically each value of a given feature of a dataset will be converted to a representative number of standard deviations that it's away from the mean of the feature.

```
from sklearn.preprocessing import StandardScaler
import pandas as pd

numeric_cols = df.select_dtypes(include=['float64', 'int64']).columns


scaler = StandardScaler()
df[numeric_cols] = scaler.fit_transform(df[numeric_cols])

print(df)
```

```
      work_year                 job_title                    job_category  \
0      0.461170        Data DevOps Engineer              Data Engineering
1      0.461170              Data Architect  Data Architecture and Modeling
2      0.461170              Data Architect  Data Architecture and Modeling
3      0.461170              Data Scientist       Data Science and Research
4      0.461170              Data Scientist       Data Science and Research
...         ...                         ...                             ...
9350  -3.389115             Data Specialist    Data Management and Strategy
9351  -5.314258              Data Scientist       Data Science and Research
9352  -3.389115    Principal Data Scientist       Data Science and Research
9353  -5.314258              Data Scientist       Data Science and Research
9354  -5.314258       Business Data Analyst                   Data Analysis

     salary_currency    salary  salary_in_usd company_location  Entry-level  \
0                EUR -0.973627      -0.875162          Germany            0
1                USD  0.567122       0.565114    United States            0
2                USD -1.071103      -1.084299    United States            0
3                USD  0.975892       0.976676    United States            0
4                USD -0.890301      -0.902262    United States            0
...              ...       ...            ...              ...          ...
```

Conclusion: Thus we have implemented the concepts of data normalization on the dataset of job_data.