EXPERIMENT NO:1

Aim: Introduction to Data science and Data preparation using Pandas steps.

Theory: Data Science is an interdisciplinary field that uses scientific methods, processes, algorithms, and systems to extract knowledge and insights from structured and unstructured data. It combines expertise from domains like mathematics, statistics, computer science, and business to analyze and interpret data for decision-making.

Key Components of Data Science:

What is Data Science?

- Data Collection: Gathering data from various sources like databases, APIs, web scraping, or surveys.
- Data Preparation: Cleaning, transforming, and organizing raw data to make it suitable for analysis.
- Exploratory Data Analysis (EDA): Visualizing and summarizing data to understand patterns, relationships, and distributions.
- Model Building: Using statistical or machine learning algorithms to solve problems or make predictions.
- Model Deployment: Integrating models into real-world applications for practical use.
- Insights and Reporting: Interpreting results to drive decision-making.

What is Data Preparation?

Data preparation is the process of cleaning, transforming, and organizing raw data into a format that can be used for analysis or modeling. It is a crucial step in the data science workflow as raw data is often messy, incomplete, or inconsistent.

Steps in Data Preparation Using Pandas

1)Load Data: Data can be loaded into Pandas from various formats like CSV, Excel, or JSON using:

import pandas as pd

df = pd.read csv('data.csv') # Load data from a CSV file

2)Description of the Dataset: Use Pandas to get an overview of the dataset: print(df.info()) # Dataset structure, data types, and memory usage print(df.describe()) # Statistical summary of numeric columns

3)Drop Columns: Remove irrelevant or unnecessary columns that do not contribute to the analysis.

df = df.drop(['column name'], axis=1) # Drop specific columns

4) Handle Missing Data:

Drop rows or columns with excessive missing values:

df = df.dropna() # Drop rows with missing values Fill missing data with meaningful values:

df['column_name'].fillna(df['column_name'].mean(), inplace=True) # Replace missing values with mean

5)Create Dummy Variables: Convert categorical data into numerical format using one-hot encoding:

df = pd.get dummies(df, columns=['categorical column'], drop first=True)

6)Detect and Handle Outliers: Outliers can be identified manually by visualizing distributions (e.g., boxplots) or using statistical methods like the Interquartile Range (IQR):

```
Q1 = df['column_name'].quantile(0.25)
Q3 = df['column_name'].quantile(0.75)
IQR = Q3 - Q1
outlier_mask = (df['column_name'] < (Q1 - 1.5 * IQR)) | (df['column_name'] > (Q3 + 1.5 * IQR))
df_no_outliers = df[~outlier_mask]
Standardization and Normalization: Scale data to ensure features are on a similar range:
```

Standardization: Transform data to have a mean of 0 and standard deviation of 1.

Normalization: Scale values between 0 and 1.

Examples of both methods are provided in the earlier section.

Importance of Data Preparation

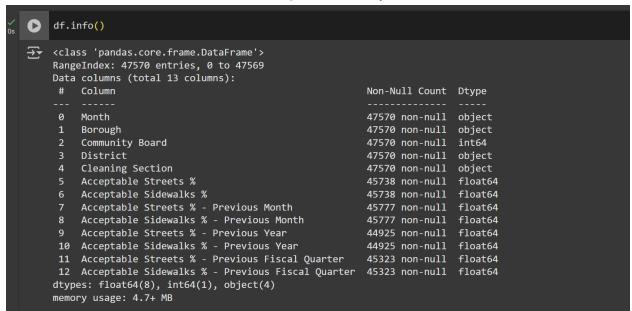
- Improves Data Quality: Ensures data is clean, consistent, and free of errors.
- Boosts Model Performance: Preprocessed data helps machine learning models perform better and converge faster.
- Reduces Bias: Identifies and removes inconsistencies, outliers, and missing values to prevent skewed results.
- Facilitates Interpretability: Organized and clean data is easier to understand and visualize.
- Data preparation is often the most time-consuming yet critical step in a data science project. Properly prepared data leads to more accurate insights and successful outcomes.

Performed Experiment:

Step 1: Firstly import Pandas Library as pd an then Load data in Pandas using pd.read csv.



Step2: Get Description of the Dataset by using following 2 commands df.info() -> Get basic information about the dataset df.describe() -> Summary statistics of the dataset



0	df.describe()										
2		Community Board	Acceptable Streets %	Acceptable Sidewalks %	Acceptable Streets % - Previous Month	Acceptable Sidewalks % - Previous Month	Acceptable Streets % - Previous Year	Acceptable Sidewalks % - Previous Year	Acceptable Streets % - Previous Fiscal Quarter	Acceptable Sidewalks % - Previous Fiscal Quarter	=
	count	47570.000000	45738.000000	45738.000000	45777.000000	45777.000000	44925.000000	44925.000000	45323.000000	45323.000000	
	mean	7.971074	93.135661	96.282435	93.135446	96.268588	92.729644	95.873846	93.105769	96.244739	
	std	4.656491	8.343306	5.276389	8.317449	5.304736	9.360139	6.891127	8.283238	5.295115	
	min	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
	25%	4.000000	90.000000	94.740000	90.000000	94.740000	90.000000	94.290000	90.000000	94.740000	
	50%	8.000000	95.060000	98.000000	95.000000	98.000000	95.000000	97.780000	95.000000	97.960000	
	75%	12.000000	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000	
	max	18.000000	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000	

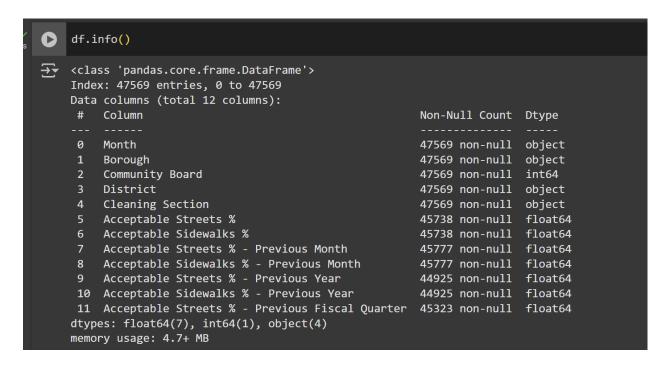
Step 3: Drop Columns that aren't useful. From Our Dataset we are dropping the "Acceptable Sidewalks" column

```
↑ ↓ ◆ 臼 国 章 归 面:

cols = ['Acceptable Sidewalks % - Previous Fiscal Quarter']

df = df.drop(cols,axis=1)
```

We can see that it returned total 9 columns as it dropped the column Acceptable Sidewalks



Step 4: Drop row with maximum missing values. df.isnull().sum(axis=1) -> Computes the number of missing values (NaN) for each row. .idxmax() -> Returns the index of row with max. no. of missing value

```
df = df.drop(df.isnull().sum(axis=1).idxmax())
```

Step 5: Taking care of missing data. We can fill the empty numeric values with mode or median or mean. Below we had filled it with median. Firstly we had fetched the numeric values and then using .fillna().median we had filled it.

```
[9] numeric_columns = df.select_dtypes(include=['float64','int64']).columns

df[numeric_columns] = df[numeric_columns].fillna(df[numeric_columns].median())
```

We can see that all the columns which had empty are filled. As they returned the sum 0

```
print(df.isnull().sum())
→ Month
                                                     0
    Borough
    Community Board
    District
                                                     0
    Cleaning Section
                                                     0
    Acceptable Streets %
    Acceptable Sidewalks %
    Acceptable Streets % - Previous Month
    Acceptable Sidewalks % - Previous Month
    Acceptable Streets % - Previous Year
                                                     0
    Acceptable Sidewalks % - Previous Year
    Acceptable Streets % - Previous Fiscal Quarter
    dtype: int64
```

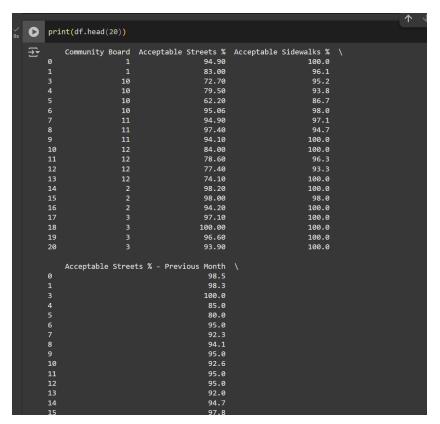
df.head() returns starting 5 values

```
print(df.head())
₹
                    Borough Community Board District Cleaning Section \
    0 2022 / 05 Manhattan
1 2022 / 05 Manhattan
                                                 MN01
                                                  MN01
                                                                  MN013
    3 2022 / 05 Manhattan
4 2022 / 05 Manhattan
                                                  MN10
                                          10
                                                                  MN101
                                                  MN10
                                                                  MN103
    5 2022 / 05 Manhattan
                                                  MN10
                                                                  MN104
       Acceptable Streets % Acceptable Sidewalks % \
                       94.9
                       83.0
                                               96.1
                       72.7
                                               95.2
                                               93.8
    4
                       79.5
                       62.2
       Acceptable Streets % - Previous Month \
                                         98.5
                                        100.0
                                        85.0
       Acceptable Sidewalks % - Previous Month \
    0
                                          100.0
                                           94.1
                                           97.3
       Acceptable Streets % - Previous Year \
                                       100.0
                                       100.0
                                        84.0
                                        96.0
                                        88.0
       Acceptable Sidewalks % - Previous Year
```

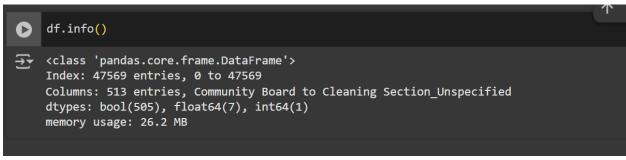
Step 6: Create dummy variables. By using the below commands separate columns are created for each unique value in a column

```
(l5] df= pd.get_dummies(df)
```

```
print(df.head(10))
                                           Borough Community Board District Cleaning Section \
              Month Borough
2022 / 05 Manhattan
                                                                                                      MN01
MN01
                                                                                                                                        MN011
MN013
                                                                                                      MN10
MN10
MN10
                                                                                                                                         MN101
                                                                                                                                        MN103
MN104
                                                                                                       MN10
MN11
MN11
                                                                                                                                        MN111
MN112
                Acceptable Streets % Acceptable Sidewalks % \
                                                                                                  96.1
95.2
                                                83.00
                                               72.70
79.50
                                               97.40
94.10
                                                                                                   94.7
                Acceptable Streets % - Previous Month
```



We can understand the working here, As we can see that we now it have returned 42 columns. But previously our data had 9 columns . So this change is because of the dummy variables , it have created separate column for each unique value in a column Below it shows original_title_Assassin, original_language_English



Step 7: Create Outliers They identify and handle unusual values in a dataset. We are using Z-score to handle the data

```
numerical_columns = df.select_dtypes(include=['float64','int64']).columns
    scaler = StandardScaler()
    df[numerical_columns] = scaler.fit_transform(df[numerical_columns])
    print(df.head())
₹
       Community Board Acceptable Streets % Acceptable Sidewalks % \
                                 0.206396
            -1.497133
                                                         -0.047942
            -1.497133
                                  -1.246695
             0.435699
                                  -2.504412
                                                          -0.221542
                                 -1.674074
              0.435699
    4
                                                         -0.491587
              0.435699
                                 -3.786551
                                                         -1.861099
       Acceptable Streets % - Previous Month \
                                   0.623775
                                   0.831932
                                  -1.004748
                                  -1.616975
       Acceptable Sidewalks % - Previous Month \
    0
                                    0.703112
                                     0.703112
                                    0.703112
                                    -0.428407
                                     0.185298
       Acceptable Streets % - Previous Year \
```

Step 8: Standardization and Normalization Import StandardScaler and MinMaxScaler

```
scaler = MinMaxScaler()
    df[numerical_columns] = scaler.fit_transform(df[numerical_columns])
       Acceptable Sidewalks % - Previous Month \
                                        1.000
                                        1.000
                                        0.941
                                        0.973
      Acceptable Streets % - Previous Year \
                                      1.00
                                      1.00
                                      0.84
                                      0.96
                                      0.88
      Acceptable Sidewalks % - Previous Year
                                        0.92
                                        1.00
      Acceptable Streets % - Previous Fiscal Quarter Month_2005 / 01 \
                                               0.950
                                                               False
                                               0.950
                                                                False
                                               0.593
                                                                False
                                               0.643
```

Standardization (z-score scaling) transforms the data by subtracting the mean and dividing by the standard deviation for each feature.

```
from scipy import stats
    numerical_df = df.select_dtypes(include=['float64','int64'])
    numerical_df = numerical_df.loc[:,numerical_df.nunique()>1]
    numerical_df = numerical_df.dropna(axis=1)
    z_scores = stats.zscore(numerical_df)
    z_scores = pd.DataFrame(z_scores, columns=numerical_df.columns).fillna(0)
    outliers = (abs(z_scores)>3).any(axis=1)
    outlier_rows = df[outliers]
    print(outlier_rows)
          Community Board Acceptable Streets % Acceptable Sidewalks % \
₹
                      10
                                          72.7
                       10
                                          79.5
                                                                 93.8
                       10
                                          62.2
                                                                 86.7
    10
                                          84.0
                                                               100.0
                                         78.6
                                                                96.3
    47504
                                         63.0
                                                                85.2
    47506
                                         80.0
                                                                100.0
    47515
                                        100.0
                                                               100.0
                                         97.1
    47525
                                                                 97.1
    47527
                                          97.1
                                                                100.0
          Acceptable Streets % - Previous Month \
                                          100.0
                                          85.0
                                          80.0
    10
                                          92.6
    11
                                          95.0
    47504
                                          100.0
    47506
                                          93.3
    47515
                                          100.0
    47525
                                          68.6
    47527
                                           51.4
```

Normalization scales numerical data to a fixed range, usually [0, 1]. Use MinMaxScaler for this process

```
scaler = MinMaxScaler()
    df[numerical_columns] = scaler.fit_transform(df[numerical_columns])
    print(df.head())
₹
       Community Board Acceptable Streets % Acceptable Sidewalks % \
             0.000000
                                      0.949
    1
             0.000000
                                      0.830
                                                              0.961
             0.529412
                                                              0.952
                                      0.727
    4
              0.529412
                                      0.795
                                                              0.938
             0.529412
                                      0.622
                                                              0.867
       Acceptable Streets % - Previous Month \
    0
                                      0.985
                                      0.983
                                       1.000
                                       0.850
    5
                                       0.800
       Acceptable Sidewalks % - Previous Month \
    0
                                        1.000
    1
                                        1.000
                                        1.000
    4
                                        0.941
    5
                                        0.973
       Acceptable Streets % - Previous Year \
    0
                                       1.00
    1
                                       1.00
```

Conclusion:

In this experiment, we applied various data preprocessing techniques, including handling missing values, removing irrelevant columns, and detecting outliers using the Z-score method. We then scaled the numerical data using standardization (Z-score method) and normalization (Min-Max scaling) to bring all features onto a uniform scale. Some Challenges we faced: 1. Handling Missing Data: Identifying the appropriate method to handle missing values and replacing them with mean, median, or mode. 2. Scaling and Normalization: Deciding between standardization and normalization for different features can be tricky. Using incorrect scaling methods may distort the data and affect model accuracy. 3. Selection of Columns: Determining which columns are relevant for the model and dropping them is challenging