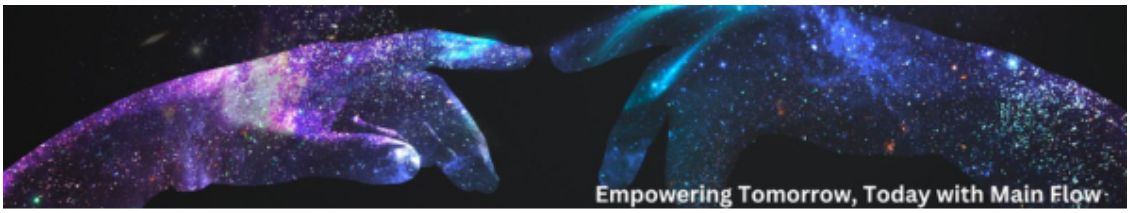
Simple Data Maniputation With Pandas

December 15, 2024

[1]:



PYTHON DEVELOPER

TASK-2

Simple Data Manipulation with Pandas

Description:

The intern will explore basic data analysis using the Pandas library in Python.

Responsibility:

1. Learn how to read CSV files and manipulate data frames using Pandas.
2. Perform simple data cleaning tasks such as handling missing values and removing duplicates.
3. Practice basic data manipulation operations like filtering, sorting, and grouping data.

Step 1: Load the dataset

[2]:
```
import pandas as pd

# Load the dataset
df = pd.read_csv('01.Data Cleaning and Preprocessing.csv')
df
```

[2]:

	Observation	Y-Kappa	ChipRate	BF-CMratio	BlowFlow	ChipLevel4	\
0	31-00:00	23.10	16.520	121.717	1177.607	169.805	
1	31-01:00	27.60	16.810	79.022	1328.360	341.327	
2	31-02:00	23.19	16.709	79.562	1329.407	239.161	
3	31-03:00	23.60	16.478	81.011	1334.877	213.527	
4	31-04:00	22.90	15.618	93.244	1334.168	243.131	
..	…	…	…	…	…	…	
319	10-16:00	23.75	12.667	93.450	1178.252	276.955	
320	9-19:00	19.80	12.558	94.352	1184.119	297.071	
321	9-20:00	23.01	12.550	90.842	1188.517	289.826	
322	9-21:00	24.32	13.083	88.910	1192.879	318.006	
323	9-22:00	25.75	13.417	85.451	1186.342	248.312	

	T-upperExt-2	T-lowerExt-2	UCZAA	WhiteFlow-4	…	SteamFlow-4	\
0	358.282	329.545	1.443	599.253	…	67.122	
1	351.050	329.067	1.549	537.201	…	60.012	
2	350.022	329.260	1.600	549.611	…	61.304	
3	350.938	331.142	1.604	623.362	…	68.496	
4	351.640	332.709	NaN	638.672	…	70.022	
..	…	…	…	…	…	…	
319	347.286	310.970	1.523	513.956	…	61.141	
320	399.135	319.576	1.451	570.058	…	67.667	
321	373.633	314.591	1.457	549.306	…	66.446	
322	364.081	308.559	1.523	504.852	…	61.054	
323	356.289	310.482	1.474	497.375	…	58.247	

	Lower-HeatT-3	Upper-HeatT-3	ChipMass-4	WeakLiquorF	BlackFlow-2	\
0	329.432	303.099	175.964	1127.197	1319.039	
1	330.823	304.879	163.202	665.975	1297.317	
2	329.140	303.383	164.013	677.534	1327.072	
3	328.875	302.254	181.487	767.853	1324.461	
4	328.352	300.954	183.929	888.448	1343.424	
..	…	…	…	…	…	
319	330.117	304.006	148.174	1027.201	1357.271	
320	330.848	304.616	165.178	906.962	1311.177	
321	330.226	304.686	160.841	887.125	1319.226	
322	327.346	304.363	147.589	804.423	1320.225	
323	328.092	304.093	144.218	828.328	1320.848	

```
     WeakWashF   SteamHeatF-3   T-Top-Chips-4   SulphidityL-4
0       257.325         54.612         252.077             NaN
1       241.182         46.603         251.406           29.11
2       237.272         51.795         251.335             NaN
3       239.478         54.846         250.312           29.02
4       215.372         54.186         249.916           29.01
..          …              …              …               …
319     381.643         45.264         252.947           30.86
320      25.494         50.528         252.092           30.70
321       0.638         45.549         252.438             NaN
322       0.000         43.725         253.176           31.13
323       1.276         43.840         253.216             NaN

[324 rows x 23 columns]
```

[ ]:

[3]:
```
# Display the first few rows of the dataset
print("First few rows of the dataset:")
df.head()
```

```
First few rows of the dataset:
```

[3]:

	Observation	Y-Kappa	ChipRate	BF-CMratio	BlowFlow	ChipLevel4	\
0	31-00:00	23.10	16.520	121.717	1177.607	169.805	
1	31-01:00	27.60	16.810	79.022	1328.360	341.327	
2	31-02:00	23.19	16.709	79.562	1329.407	239.161	
3	31-03:00	23.60	16.478	81.011	1334.877	213.527	
4	31-04:00	22.90	15.618	93.244	1334.168	243.131	

	T-upperExt-2	T-lowerExt-2	UCZAA	WhiteFlow-4	…	SteamFlow-4	\
0	358.282	329.545	1.443	599.253	…	67.122	
1	351.050	329.067	1.549	537.201	…	60.012	
2	350.022	329.260	1.600	549.611	…	61.304	
3	350.938	331.142	1.604	623.362	…	68.496	
4	351.640	332.709	NaN	638.672	…	70.022	

	Lower-HeatT-3	Upper-HeatT-3	ChipMass-4	WeakLiquorF	BlackFlow-2	\
0	329.432	303.099	175.964	1127.197	1319.039	
1	330.823	304.879	163.202	665.975	1297.317	
2	329.140	303.383	164.013	677.534	1327.072	
3	328.875	302.254	181.487	767.853	1324.461	
4	328.352	300.954	183.929	888.448	1343.424	

	WeakWashF	SteamHeatF-3	T-Top-Chips-4	SulphidityL-4
0	257.325	54.612	252.077	NaN
1	241.182	46.603	251.406	29.11
2	237.272	51.795	251.335	NaN

```
3     239.478         54.846         250.312          29.02
4     215.372         54.186         249.916          29.01

[5 rows x 23 columns]
```

[ ]:

[4]:
```
# Display column names
print("\nColumn names:")
print(df.columns)
```

```
Column names:
Index(['Observation', 'Y-Kappa', 'ChipRate', 'BF-CMratio', 'BlowFlow',
       'ChipLevel4 ', 'T-upperExt-2 ', 'T-lowerExt-2  ', 'UCZAA',
       'WhiteFlow-4 ', 'AAWhiteSt-4 ', 'AA-Wood-4  ', 'ChipMoisture-4 ',
       'SteamFlow-4 ', 'Lower-HeatT-3', 'Upper-HeatT-3 ', 'ChipMass-4 ',
       'WeakLiquorF ', 'BlackFlow-2 ', 'WeakWashF ', 'SteamHeatF-3 ',
       'T-Top-Chips-4 ', 'SulphidityL-4 '],
      dtype='object')
```

[ ]:

Step 2: Correct column name formatting (remove trailing spaces)

[ ]:

[5]:
```
# Strip trailing and leading spaces from column names
df.columns = df.columns.str.strip()

# Display cleaned column names
print("\nCleaned column names:")
print(df.columns)
```

```
Cleaned column names:
Index(['Observation', 'Y-Kappa', 'ChipRate', 'BF-CMratio', 'BlowFlow',
       'ChipLevel4', 'T-upperExt-2', 'T-lowerExt-2', 'UCZAA', 'WhiteFlow-4',
       'AAWhiteSt-4', 'AA-Wood-4', 'ChipMoisture-4', 'SteamFlow-4',
       'Lower-HeatT-3', 'Upper-HeatT-3', 'ChipMass-4', 'WeakLiquorF',
       'BlackFlow-2', 'WeakWashF', 'SteamHeatF-3', 'T-Top-Chips-4',
       'SulphidityL-4'],
      dtype='object')
```

[ ]:

Step 3: Check for missing values

[ ]:

[6]:
```
# Check for missing values in the dataset
missing_values = df.isnull().sum()

print("\nMissing values in each column:")
print(missing_values)
```

```
Missing values in each column:
Observation          0
Y-Kappa              0
ChipRate             5
BF-CMratio          17
BlowFlow            16
ChipLevel4           1
T-upperExt-2         2
T-lowerExt-2         2
UCZAA               25
WhiteFlow-4          1
AAWhiteSt-4        151
AA-Wood-4            1
ChipMoisture-4       1
SteamFlow-4          1
Lower-HeatT-3        2
Upper-HeatT-3        2
ChipMass-4           1
WeakLiquorF          1
BlackFlow-2          2
WeakWashF            1
SteamHeatF-3         2
T-Top-Chips-4        1
SulphidityL-4      151
dtype: int64
```

[ ]:

[ ]:

Step 4: Fill missing values in specific columns (e.g., `ChipLevel4`)

[ ]:

[7]:
```
# Fill missing values in 'ChipLevel4' with the mean of the column
df['ChipLevel4'] = df['ChipLevel4'].fillna(df['ChipLevel4'].mean())
```

[8]:
```
df['ChipLevel4'].mean()
```

[8]: 258.1644829721362

[ ]:

[9]:
```
df['ChipLevel4']
```

[9]:
```
0      169.805
1      341.327
2      239.161
3      213.527
4      243.131
        …
319    276.955
320    297.071
321    289.826
322    318.006
323    248.312
Name: ChipLevel4, Length: 324, dtype: float64
```

[10]:
```
print("\nMissing values in 'ChipLevel4' after filling:")
print(df['ChipLevel4'].isnull().sum())
```

```
Missing values in 'ChipLevel4' after filling:
0
```

[ ]:

[ ]:

Step 5: Handle missing values for all columns (fill with mean or other strategies)

[ ]:

[11]:
```
# Fill missing values in all numeric columns with the mean
numeric_cols = df.select_dtypes(include=['float64', 'int64']).columns
df[numeric_cols] = df[numeric_cols].fillna(df[numeric_cols].mean())
```

[ ]:

[12]:
```
df[numeric_cols].mean()
```

[12]:
```
Y-Kappa            20.635370
ChipRate           14.347937
BF-CMratio         87.464456
BlowFlow         1237.837614
ChipLevel4        258.164483
T-upperExt-2      356.904295
T-lowerExt-2      324.020180
UCZAA               1.492010
WhiteFlow-4       591.732260
```

```
AAWhiteSt-4          6.140410
AA-Wood-4           17.835885
ChipMoisture-4      46.781969
SteamFlow-4         66.668285
Lower-HeatT-3      325.567820
Upper-HeatT-3      300.525699
ChipMass-4         162.222322
WeakLiquorF        873.828941
BlackFlow-2       1175.917016
WeakWashF          263.543068
SteamHeatF-3        49.696907
T-Top-Chips-4      251.240087
SulphidityL-4       30.411671
dtype: float64
```

[ ]:

[13]:
```
# Verify missing values are handled
print("\nMissing values after handling all numeric columns:")
print(df.isnull().sum())
```

```
Missing values after handling all numeric columns:
Observation       0
Y-Kappa           0
ChipRate          0
BF-CMratio        0
BlowFlow          0
ChipLevel4        0
T-upperExt-2      0
T-lowerExt-2      0
UCZAA             0
WhiteFlow-4       0
AAWhiteSt-4       0
AA-Wood-4         0
ChipMoisture-4    0
SteamFlow-4       0
Lower-HeatT-3     0
Upper-HeatT-3     0
ChipMass-4        0
WeakLiquorF       0
BlackFlow-2       0
WeakWashF         0
SteamHeatF-3      0
T-Top-Chips-4     0
SulphidityL-4     0
dtype: int64
```

[ ]:

Step 6: Remove duplicate rows

[14]:
```
print(f"Shape of the dataset after removing duplicates: {df.shape}")
```

```
Shape of the dataset after removing duplicates: (324, 23)
```

[15]:
```
# Check for duplicate rows
duplicate_count = df.duplicated().sum()
print(f"\nNumber of duplicate rows: {duplicate_count}")
```

```
Number of duplicate rows: 23
```

[ ]:

[16]:
```
# Remove duplicate rows
df = df.drop_duplicates()

print(f"Shape of the dataset after removing duplicates: {df.shape}")
```

```
Shape of the dataset after removing duplicates: (301, 23)
```

[ ]:

Step 7: Perform basic data manipulations

[ ]:

a) Filter data

[ ]:

[17]:
```
# Example: Filter rows where 'ChipRate' > 50
filtered_data = df[df['ChipRate'] > 50]
print("\nFiltered data (ChipRate > 50):")
print(filtered_data.head())
```

```
Filtered data (ChipRate > 50):
Empty DataFrame
Columns: [Observation, Y-Kappa, ChipRate, BF-CMratio, BlowFlow, ChipLevel4,
T-upperExt-2, T-lowerExt-2, UCZAA, WhiteFlow-4, AAWhiteSt-4, AA-Wood-4,
ChipMoisture-4, SteamFlow-4, Lower-HeatT-3, Upper-HeatT-3, ChipMass-4,
WeakLiquorF, BlackFlow-2, WeakWashF, SteamHeatF-3, T-Top-Chips-4, SulphidityL-4]
Index: []

[0 rows x 23 columns]
```

[ ]:

[ ]:

[ ]:

b) Sort data

[ ]:

[18]:
```
# Sort data by 'ChipRate' in descending order
sorted_data = df.sort_values(by='ChipRate', ascending=False)
print("\nData sorted by ChipRate (descending):")
sorted_data.head()
```

```
Data sorted by ChipRate (descending):
```

[18]:

	Observation	Y-Kappa	ChipRate	BF-CMratio	BlowFlow	ChipLevel4	\
64	2-15:00	25.40	16.958	74.405	1230.776	295.522	
99	4-02:00	23.50	16.867	76.937	1284.849	334.170	
73	3-00:00	18.30	16.833	91.341	1236.911	287.397	
72	2-23:00	18.13	16.825	84.836	1246.387	259.527	
71	2-22:00	21.50	16.817	78.969	1237.844	285.892	

	T-upperExt-2	T-lowerExt-2	UCZAA	WhiteFlow-4	…	SteamFlow-4	\
64	350.216	322.740	1.47100	540.151	…	52.962	
99	364.132	329.250	1.55100	620.208	…	72.448	
73	356.787	327.856	1.54000	661.532	…	71.891	
72	355.515	325.985	1.49201	662.577	…	72.407	
71	353.644	324.500	1.43700	621.887	…	68.750	

	Lower-HeatT-3	Upper-HeatT-3	ChipMass-4	WeakLiquorF	BlackFlow-2	\
64	318.913	295.843	146.914	727.366	1052.879	
99	322.259	297.509	171.093	888.854	1019.068	
73	323.285	297.755	187.777	841.552	1144.876	
72	321.904	296.185	186.165	810.272	1132.813	
71	321.235	296.616	176.460	852.132	1141.946	

	WeakWashF	SteamHeatF-3	T-Top-Chips-4	SulphidityL-4
64	476.150	43.701	252.097	30.411671
99	603.462	50.394	248.435	30.411671
73	0.000	53.897	250.212	30.411671
72	0.000	54.044	250.266	29.630000
71	344.089	52.043	250.678	29.700000

```
[5 rows x 23 columns]
```

[ ]:

[19]:
```
sorted_data.head()
```

[19]:
```
    Observation  Y-Kappa  ChipRate  BF-CMratio  BlowFlow  ChipLevel4  \
64     2-15:00    25.40    16.958      74.405  1230.776     295.522
99     4-02:00    23.50    16.867      76.937  1284.849     334.170
73     3-00:00    18.30    16.833      91.341  1236.911     287.397
72     2-23:00    18.13    16.825      84.836  1246.387     259.527
71     2-22:00    21.50    16.817      78.969  1237.844     285.892

    T-upperExt-2  T-lowerExt-2    UCZAA  WhiteFlow-4  …  SteamFlow-4  \
64       350.216       322.740  1.47100      540.151  …       52.962
99       364.132       329.250  1.55100      620.208  …       72.448
73       356.787       327.856  1.54000      661.532  …       71.891
72       355.515       325.985  1.49201      662.577  …       72.407
71       353.644       324.500  1.43700      621.887  …       68.750

    Lower-HeatT-3  Upper-HeatT-3  ChipMass-4  WeakLiquorF  BlackFlow-2  \
64        318.913        295.843     146.914      727.366     1052.879
99        322.259        297.509     171.093      888.854     1019.068
73        323.285        297.755     187.777      841.552     1144.876
72        321.904        296.185     186.165      810.272     1132.813
71        321.235        296.616     176.460      852.132     1141.946

    WeakWashF  SteamHeatF-3  T-Top-Chips-4  SulphidityL-4
64    476.150        43.701        252.097      30.411671
99    603.462        50.394        248.435      30.411671
73      0.000        53.897        250.212      30.411671
72      0.000        54.044        250.266      29.630000
71    344.089        52.043        250.678      29.700000

[5 rows x 23 columns]
```

[ ]:
```

```

c) Group data

[ ]:
```

```

[20]:
```
# Group data by 'Observation' and calculate mean for numeric columns
grouped_data = df.groupby('Observation').mean()
print("\nGrouped data by 'Observation':")
grouped_data.head()
```

```
Grouped data by 'Observation':
```

[20]:
```
             Y-Kappa  ChipRate  BF-CMratio     BlowFlow  ChipLevel4  \
Observation
```

```
1-00:00        20.70    12.573   96.141000  1208.756000     272.985
1-01:00        18.90    12.542   94.741000  1188.212000     302.073
1-02:00        21.35    12.467   94.280000  1194.216000     296.608
1-03:00        21.10    12.667   86.083000  1182.201000     295.825
1-04:00        20.14    13.733   87.464456  1237.837614     274.481

             T-upperExt-2  T-lowerExt-2    UCZAA  WhiteFlow-4  AAWhiteSt-4  \
Observation
1-00:00           347.947       329.301  1.47400      559.122      6.14041
1-01:00           346.808       323.350  1.48800      541.015      6.03100
1-02:00           346.747       317.603  1.49201      518.788      6.04000
1-03:00           346.228       310.421  1.43200      507.502      6.14041
1-04:00           346.553       309.963  1.54600      504.445      6.03700

             …  SteamFlow-4  Lower-HeatT-3  Upper-HeatT-3  ChipMass-4  \
Observation  …
1-00:00      …       65.342        329.249        305.094     162.495
1-01:00      …       60.042        328.464        304.220     155.932
1-02:00      …       63.272        328.447        303.695     153.744
1-03:00      …       62.391        328.654        303.750     152.577
1-04:00      …       62.560        328.914        304.030     151.887

             WeakLiquorF  BlackFlow-2  WeakWashF  SteamHeatF-3  T-Top-Chips-4  \
Observation
1-00:00          738.125     1293.046    330.577        45.731        251.335
1-01:00          803.101     1293.109    377.297        46.899        251.792
1-02:00          910.522     1305.465    426.254        45.610        252.128
1-03:00          666.636     1310.026    427.336        46.053        252.026
1-04:00          760.020     1300.479    436.406        46.632        252.133

             SulphidityL-4
Observation
1-00:00          30.411671
1-01:00          29.640000
1-02:00          30.340000
1-03:00          30.411671
1-04:00          30.070000

[5 rows x 22 columns]
```

[ ]:

[ ]:

[ ]:

[21]:
```
### Step 8: Save the cleaned and processed dataset
# Save the cleaned DataFrame to a new CSV file
df.to_csv('Cleaned_Data.csv', index=False)

print("\nCleaned dataset saved to 'Cleaned_Data.csv'")
```

```
Cleaned dataset saved to 'Cleaned_Data.csv'
```

[ ]:

[ ]:

0.0.1 Outputs

1. **Cleaned Column Names:** Stripped of unnecessary spaces.
2. **Missing Values Report:** Lists columns with missing data and shows how they're handled.
3. **Filtered, Sorted, and Grouped Data:** Examples of basic data manipulation.
4. **Saved Cleaned Dataset:** Outputs a CSV file named `Cleaned_Data.csv`.

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]: