MIT World Peace University

Data Science for Cybersecurity and Forensics
Third Year B. Tech, Semester 6

Data Pre Processing in Python

Assignment 2

Prepared By

Krishnaraj Thadesar
Cyber Security and Forensics
Batch A1, PA 10

April 16, 2024

Contents

1 Aim

Using python perform some Preprocessing using Python Libraries on any dataset.

2 Objectives

1. To perform data preprocessing on a dataset using Python.
2. To understand the importance of data preprocessing in data science.
3. To learn how to use Python libraries for data preprocessing.

3 Theory

Data preprocessing is a crucial step in the data science pipeline. It involves cleaning and transforming raw data into a more understandable format. Real-world data is often incomplete, inconsistent, and/or lacking in certain behaviors or trends, and is likely to contain many errors.

Data preprocessing is a data mining technique that involves transforming raw data into an understandable format. It is a proven method for handling such data. This process is essential because data scientists cannot work with raw data directly due to its inherent complexities and imperfections.

The process of data preprocessing encompasses various tasks, including handling missing values, dealing with outliers, normalizing data, transforming features, and integrating multiple datasets. Each of these tasks contributes to ensuring that the data is of high quality and suitable for analysis and modeling.

By performing data preprocessing, data scientists can enhance the quality of their analyses and improve the performance of machine learning models. Preprocessed data is easier to work with, interpret, and analyze, leading to more reliable insights and predictions.

4 Data Preprocessing Techniques

1. Data Cleaning
2. Data Transformation
3. Data Reduction
4. Data Normalization
5. Data Integration

5 Data Preprocessing Techniques

5.1 Data Cleaning

- Identification and handling of missing values, which can involve imputation techniques such as mean, median, or mode imputation, or removal of incomplete records.
- Detection and treatment of outliers using statistical methods like Z-score, interquartile range (IQR), or visualizations.

- Consistency checks to identify and rectify errors or inconsistencies in the data.

5.2 Data Transformation

- Encoding categorical variables through techniques like one-hot encoding, label encoding, or ordinal encoding.
- Feature scaling or normalization to ensure that all features have a similar scale, which can include methods such as Min-Max scaling or standardization.
- Creation of new features through techniques like polynomial features, interaction terms, or feature extraction from existing ones.

5.3 Data Reduction

- Dimensionality reduction methods to reduce the number of features in the dataset, such as Principal Component Analysis (PCA) or Singular Value Decomposition (SVD).
- Feature selection techniques to identify and retain the most relevant features, including filter methods, wrapper methods, and embedded methods.

5.4 Data Normalization

- Ensuring data consistency and conformity by bringing it to a common scale or format.
- Standardization of data to have a mean of 0 and a standard deviation of 1, making it easier to compare and interpret different features.
- Min-Max scaling to rescale data to a fixed range, typically between 0 and 1, preserving the relationships between data points.

5.5 Data Integration

- Combining data from multiple sources or datasets into a single unified dataset, ensuring consistency and coherence.
- Handling conflicts or inconsistencies in data schemas, formats, or values during the integration process.
- Resolving duplicate records or redundant information to create a clean and comprehensive dataset.

6 Platform

Operating System: Windows 11
IDEs or Text Editors Used: Visual Studio Code
Compilers or Interpreters: Python 3.10.1

7 Requirements

```
python==3.10.1
matplotlib==3.8.3
numpy==1.26.4
pandas==2.2.2
seaborn==0.13.2
```

8 Code

[ ]:
```python
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
```

[2]:
```python
data = pd.read_csv('uber.csv')
```

[3]:
```python
data.head()
```

[3]:
```
   Unnamed: 0                            key  fare_amount  \
0    24238194    2015-05-07 19:52:06.0000003          7.5
1    27835199    2009-07-17 20:04:56.0000002          7.7
2    44984355   2009-08-24 21:45:00.00000061         12.9
3    25894730    2009-06-26 08:22:21.0000001          5.3
4    17610152  2014-08-28 17:47:00.000000188         16.0

           pickup_datetime  pickup_longitude  pickup_latitude  \
0  2015-05-07 19:52:06 UTC        -73.999817        40.738354
1  2009-07-17 20:04:56 UTC        -73.994355        40.728225
2  2009-08-24 21:45:00 UTC        -74.005043        40.740770
3  2009-06-26 08:22:21 UTC        -73.976124        40.790844
4  2014-08-28 17:47:00 UTC        -73.925023        40.744085

   dropoff_longitude  dropoff_latitude  passenger_count
0         -73.999512         40.723217                1
1         -73.994710         40.750325                1
2         -73.962565         40.772647                1
3         -73.965316         40.803349                3
4         -73.973082         40.761247                5
```

[4]:
```python
# drop the first column
data.drop(data.columns[0], axis=1, inplace=True)
```

[5]:
```python
data
```

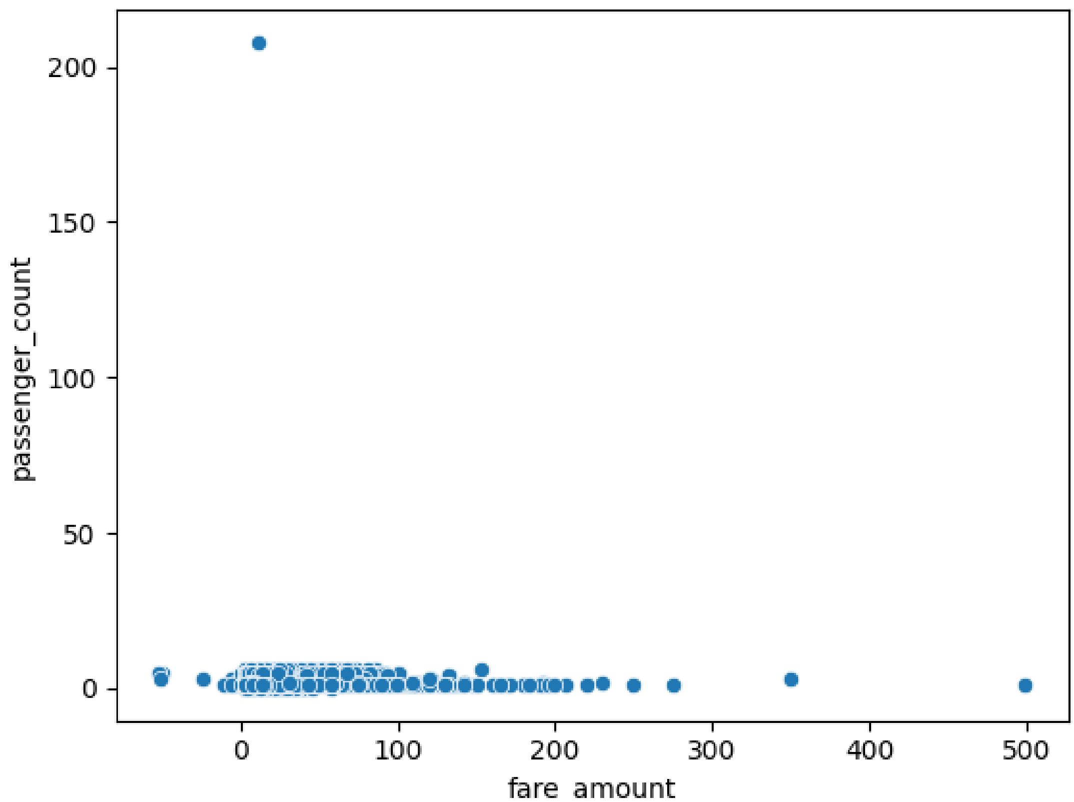
[5]:
```
                                  key  fare_amount          pickup_datetime
↪\
0          2015-05-07 19:52:06.0000003          7.5  2015-05-07 19:52:06 UTC
1          2009-07-17 20:04:56.0000002          7.7  2009-07-17 20:04:56 UTC
2         2009-08-24 21:45:00.00000061         12.9  2009-08-24 21:45:00 UTC
```

```
3        2009-06-26 08:22:21.0000001          5.3  2009-06-26 08:22:21 UTC
4       2014-08-28 17:47:00.000000188        16.0  2014-08-28 17:47:00 UTC
...                                   ...     ...                      ...
199995   2012-10-28 10:49:00.00000053         3.0  2012-10-28 10:49:00 UTC
199996    2014-03-14 01:09:00.0000008         7.5  2014-03-14 01:09:00 UTC
199997   2009-06-29 00:42:00.00000078        30.9  2009-06-29 00:42:00 UTC
199998    2015-05-20 14:56:25.0000004        14.5  2015-05-20 14:56:25 UTC
199999   2010-05-15 04:08:00.00000076        14.1  2010-05-15 04:08:00 UTC

        pickup_longitude  pickup_latitude  dropoff_longitude  \
0             -73.999817        40.738354         -73.999512
1             -73.994355        40.728225         -73.994710
2             -74.005043        40.740770         -73.962565
3             -73.976124        40.790844         -73.965316
4             -73.925023        40.744085         -73.973082
...                  ...              ...                ...
199995        -73.987042        40.739367         -73.986525
199996        -73.984722        40.736837         -74.006672
199997        -73.986017        40.756487         -73.858957
199998        -73.997124        40.725452         -73.983215
199999        -73.984395        40.720077         -73.985508

        dropoff_latitude  passenger_count
0              40.723217                1
1              40.750325                1
2              40.772647                1
3              40.803349                3
4              40.761247                5
...                  ...              ...
199995         40.740297                1
199996         40.739620                1
199997         40.692588                2
199998         40.695415                1
199999         40.768793                1

[200000 rows x 8 columns]
```

9 EDA

[8]:
```
# is there a relationship between fare amoutn and passenger count?
sns.scatterplot(x='fare_amount', y='passenger_count', data=data)
```

[8]:
```
<Axes: xlabel='fare_amount', ylabel='passenger_count'>
```



[11]:

```
# find value of correlation r using pandas
r = data['fare_amount'].corr(data['passenger_count'])
r
```

[11]:

```
0.010149925554531453
```

[15]:

```
# lets convert pickup_datetime to
data['pickup_datetime'] = pd.to_datetime(data['pickup_datetime'])
data.dtypes
```

[15]:

```
key                               object
fare_amount                      float64
pickup_datetime      datetime64[ns, UTC]
pickup_longitude                 float64
pickup_latitude                  float64
dropoff_longitude                float64
dropoff_latitude                 float64
passenger_count                    int64
dtype: object
```

[20]:
```
# let us split datetime to hours, minutes, seconds, day, month, year into␣
↪new columns
data['pickup_hour'] = data['pickup_datetime'].dt.hour
data['pickup_minute'] = data['pickup_datetime'].dt.minute
data['pickup_second'] = data['pickup_datetime'].dt.second
data['pickup_day'] = data['pickup_datetime'].dt.day
data['pickup_month'] = data['pickup_datetime'].dt.month
data['pickup_year'] = data['pickup_datetime'].dt.year
```

[22]:
```
data.drop('pickup_datetime', axis=1, inplace=True)
```

[24]:
```
data.head()
```

[24]:
```
                             key  fare_amount  pickup_longitude  \
0    2015-05-07 19:52:06.0000003          7.5        -73.999817
1    2009-07-17 20:04:56.0000002          7.7        -73.994355
2   2009-08-24 21:45:00.00000061         12.9        -74.005043
3    2009-06-26 08:22:21.0000001          5.3        -73.976124
4  2014-08-28 17:47:00.000000188         16.0        -73.925023

   pickup_latitude  dropoff_longitude  dropoff_latitude  passenger_count  \
0        40.738354         -73.999512         40.723217                1
1        40.728225         -73.994710         40.750325                1
2        40.740770         -73.962565         40.772647                1
3        40.790844         -73.965316         40.803349                3
4        40.744085         -73.973082         40.761247                5

   pickup_hour  pickup_minute  pickup_second  pickup_day  pickup_month  \
0           19             52              6           7             5
1           20              4             56          17             7
2           21             45              0          24             8
3            8             22             21          26             6
4           17             47              0          28             8

   pickup_year
0         2015
1         2009
2         2009
3         2009
4         2014
```

[27]:
```
# drop key
new_data = data.drop('key', axis=1)
```

[28]:
```
new_data.corr()
```

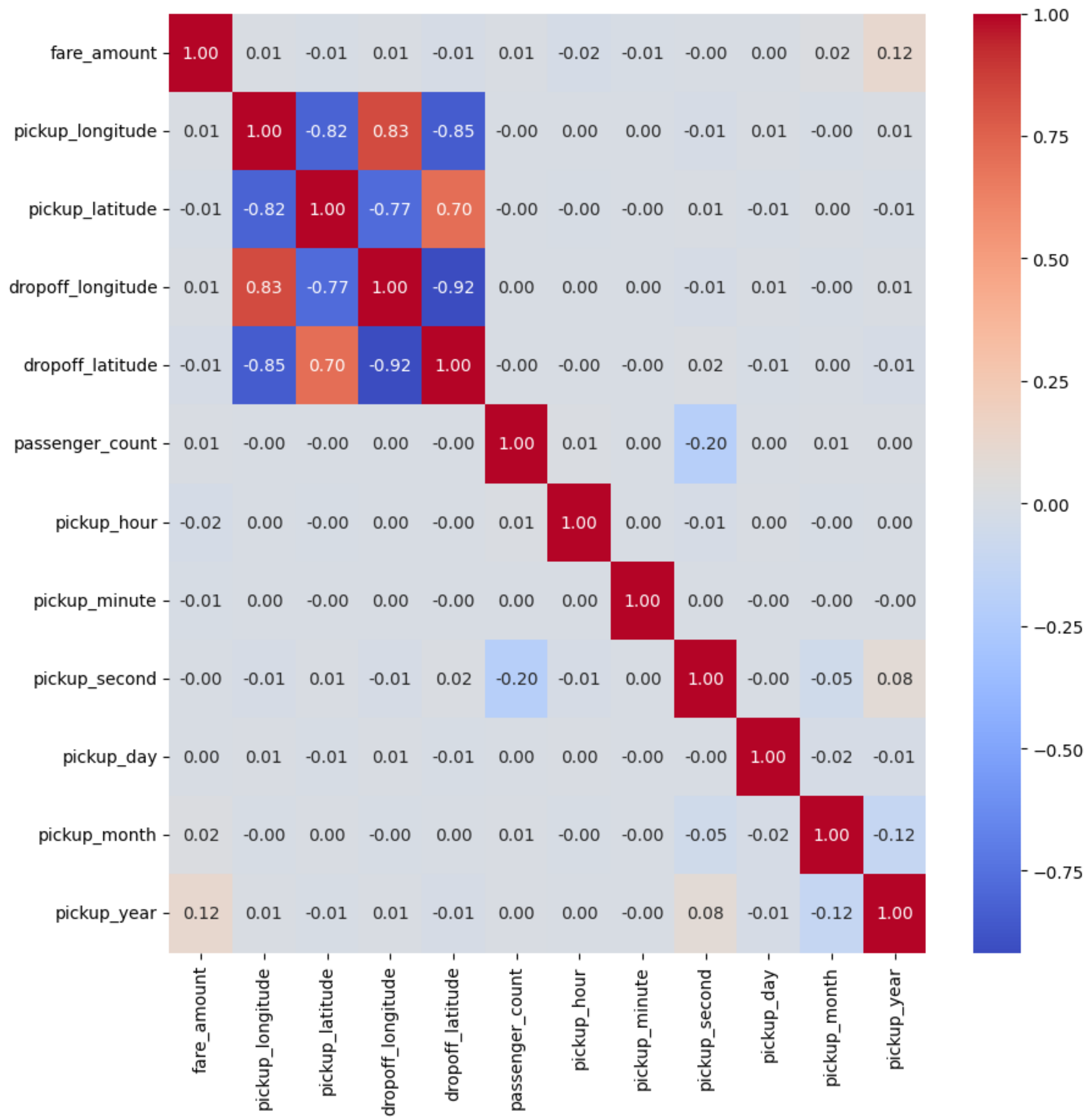
[28]:

```
                   fare_amount  pickup_longitude  pickup_latitude  \
fare_amount           1.000000          0.010457        -0.008481
pickup_longitude      0.010457          1.000000        -0.816461
pickup_latitude      -0.008481         -0.816461         1.000000
dropoff_longitude     0.008986          0.833026        -0.774787
dropoff_latitude     -0.011014         -0.846324         0.702367
passenger_count       0.010150         -0.000414        -0.001560
pickup_hour          -0.021473          0.002433        -0.003822
pickup_minute        -0.008035          0.002781        -0.002919
pickup_second        -0.001259         -0.011270         0.011046
pickup_day            0.001374          0.005184        -0.008264
pickup_month          0.023814         -0.004665         0.004625
pickup_year           0.118335          0.009966        -0.010233

                   dropoff_longitude  dropoff_latitude  passenger_count  \
fare_amount                 0.008986         -0.011014         0.010150
pickup_longitude            0.833026         -0.846324        -0.000414
pickup_latitude            -0.774787          0.702367        -0.001560
dropoff_longitude           1.000000         -0.917010         0.000033
dropoff_latitude           -0.917010          1.000000        -0.000659
passenger_count             0.000033         -0.000659         1.000000
pickup_hour                 0.003478         -0.002544         0.013196
pickup_minute               0.002557         -0.001620         0.000688
pickup_second              -0.011077          0.015280        -0.203017
pickup_day                  0.005055         -0.007835         0.003252
pickup_month               -0.003605          0.003818         0.009773
pickup_year                 0.008467         -0.011239         0.004798

                   pickup_hour  pickup_minute  pickup_second  pickup_day  \
fare_amount          -0.021473      -0.008035      -0.001259    0.001374
pickup_longitude      0.002433       0.002781      -0.011270    0.005184
pickup_latitude      -0.003822      -0.002919       0.011046   -0.008264
dropoff_longitude     0.003478       0.002557      -0.011077    0.005055
dropoff_latitude     -0.002544      -0.001620       0.015280   -0.007835
passenger_count       0.013196       0.000688      -0.203017    0.003252
pickup_hour           1.000000       0.001138      -0.013240    0.004677
pickup_minute         0.001138       1.000000       0.001987   -0.001217
pickup_second        -0.013240       0.001987       1.000000   -0.002107
pickup_day            0.004677      -0.001217      -0.002107    1.000000
pickup_month         -0.003926      -0.001485      -0.049937   -0.017360
pickup_year           0.002156      -0.002805       0.083345   -0.012170

                   pickup_month  pickup_year
fare_amount            0.023814     0.118335
pickup_longitude      -0.004665     0.009966
pickup_latitude        0.004625    -0.010233
dropoff_longitude     -0.003605     0.008467
```

```
dropoff_latitude        0.003818    -0.011239
passenger_count         0.009773     0.004798
pickup_hour            -0.003926     0.002156
pickup_minute          -0.001485    -0.002805
pickup_second          -0.049937     0.083345
pickup_day             -0.017360    -0.012170
pickup_month            1.000000    -0.115859
pickup_year            -0.115859     1.000000
```

[31]:
```
# visualize the correlation matrix
fig, ax = plt.subplots(figsize=(10, 10))
# round the values to 2 decimal places
sns.heatmap(new_data.corr(), annot=True, ax=ax, cmap='coolwarm', fmt='.2f')
```

[31]:
```
<Axes: >
```


fare_amount
pickup_longitude
pickup_latitude
dropoff_longitude
dropoff_latitude
passenger_count
pickup_hour
pickup_minute
pickup_second
pickup_day
pickup_month
pickup_year
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75

[ ]:

```

```

10 FAQs

1. **What is Preprocessing technique?**
   Data preprocessing involves a series of steps aimed at cleaning, transforming, and organizing raw data into a format that is more suitable for analysis and modeling. These steps include handling missing values, dealing with outliers, normalizing data, transforming features, and integrating multiple datasets.

2. **What is the use of Preprocessing technique in data science?**
   Preprocessing techniques are essential in data science for several reasons:
   - Enhancing data quality by addressing issues like missing values, outliers, and inconsistencies.
   - Improving the performance of machine learning models by ensuring that the data meets the assumptions and requirements of the algorithms.
   - Facilitating feature engineering by transforming and creating new features from existing ones.
   - Enabling effective data visualization and exploration by preparing the data in a standardized and interpretable format.

3. **What is the difference between the data with preprocessing and without preprocessing?**
   The differences between preprocessed and unprocessed data are significant and can impact the outcomes of data analysis and modeling:
   - Data Quality: Preprocessed data tends to have higher quality, with missing values handled, outliers addressed, and inconsistencies resolved, leading to more reliable results.
   - Model Performance: Preprocessing improves the performance of machine learning models by ensuring that the data meets the assumptions of the algorithms, resulting in more accurate predictions and better generalization.
   - Interpretability: Preprocessed data is often easier to interpret and analyze, as it is in a standardized format with normalized scales and transformed features, facilitating effective data exploration and visualization.

11 Conclusion

In this assignment, we have explored the importance of data preprocessing in data science and learned about various preprocessing techniques. We have also implemented data preprocessing using Python libraries like Pandas, Numpy, Matplotlib, and Seaborn. Data preprocessing is a crucial step in the data science pipeline, as it helps clean, transform, and organize raw data into a format that is more suitable for analysis and modeling.

By applying preprocessing techniques, we can enhance data quality, improve model performance, and facilitate effective data exploration and visualization. Data preprocessing is an essential skill for data scientists and analysts, as it enables them to work with real-world data effectively and derive meaningful insights from it.