Data Preprocessing: Turning Raw Data into Insights

Data Preprocessing Steps

1

Data Cleaning

Identify and handle missing values, outliers, and noisy data to ensure data quality.

2

Data Transformation

Normalize, scale, or transform features to improve model performance and interpretability.

3

Feature Selection

Select relevant features to reduce dimensionality and eliminate noise or redundant information.

Handling Missing Data

1 Identify Missing Data

Explore the dataset to locate missing values and understand their patterns.

2 Missing Data Imputation

Apply appropriate techniques such as mean imputation or regression imputation to fill in missing values.

3 Handling Categorical Missing Data

Devise strategies to handle missing values in categorical variables and preserve data integrity.

Data Encoding

One-Hot Encoding

Encode categorical variables using one-hot encoding to represent them numerically for machine learning algorithms.

Label Encoding

Encode ordinal variables using label encoding to transform categorical values into ordered numerical labels.

Binary Encoding

Encode nominal categorical variables using binary encoding to overcome the limitations of one-hot encoding.

Ordinal Encoding

Encode ordinal variables by assigning increasing integers based on their order or importance.

Feature Scaling

1

Standardization

Standardize numerical features to have zero mean and unit variance, making them comparable across different scales.

2

Normalization

Normalize numerical features to a specific range (e.g., 0 to 1) to maintain the relative proportions of their values.

Data Splitting

1

Training Set

Divide the data into a training set, typically 70-80% of the dataset, to build the machine learning model.

2

Validation Set

Set aside a validation set, around 15-20% of the dataset, for hyperparameter tuning and model performance evaluation.

3

Test Set

Keep a separate test set, approximately 10-15% of the dataset, for final model evaluation and generalization assessment.

Data Preprocessing Code

```
import pandas as pd
ADS_Phase3=pd.read_csv("/content/Electricity.csv")
ADS_Phase3
```

	DateTime	Holiday	HolidayFlag	DayOfWeek	WeekOfYear	Day	Month	Year	PeriodOfDay	ForecastWindProduction	SystemLoadEA	SMPEA	ORKTemperature	ORKWindspeed	CO2Intensity	ActualWinc
0	01/11/2011 00:00	None	0	1	44	1	11	2011	0	315.31	3388.77	49.26	6.00	9.30	600.71	
1	01/11/2011 00:30	None	0	1	44	1	11	2011	1	321.80	3196.66	49.26	6.00	11.10	605.42	
2	01/11/2011 01:00	None	0	1	44	1	11	2011	2	328.57	3060.71	49.10	5.00	11.10	589.97	
3	01/11/2011 01:30	None	0	1	44	1	11	2011	3	335.60	2945.56	48.04	6.00	9.30	585.94	
4	01/11/2011 02:00	None	0	1	44	1	11	2011	4	342.90	2849.34	33.75	6.00	11.10	571.52	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
29571	09/07/2013 02:30	None	0	1	28	9	7	2013	5	27.37	2738.80	34.63	18.00	9.30	628.63	
29572	09/07/2013 03:00	None	0	1	28	9	7	2013	6	24.76	2665.11	34.08	17.00	5.60	642.34	
29573	09/07/2013 03:30	None	0	1	28	9	7	2013	7	22.87	2643.98	34.08	17.00	5.60	624.90	
29574	09/07/2013 04:00	None	0	1	28	9	7	2013	8	21.32	2653.09	34.08	17.00	9.30	620.29	
29575	09/07/2013 04:30	None	0	1	28	9	7	2013	9	19.04	2630.13	34.08	17.00	7.40	618.83	

29576 rows × 18 columns

```
# Summary statistics
print(ADS_Phase3.describe())
```

```
       HolidayFlag     DayOfWeek    WeekOfYear           Day         Month  \
count  29576.000000  29576.000000  29576.000000  29576.000000  29576.000000
mean       0.042196      2.999121     24.839600     15.562077      6.131999
std        0.201040      1.999949     15.629816      8.826549      3.597034
min        0.000000      0.000000      1.000000      1.000000      1.000000
25%        0.000000      1.000000     12.000000      8.000000      3.000000
50%        0.000000      3.000000     23.000000     15.000000      6.000000
75%        0.000000      5.000000     39.000000     23.000000      9.000000
max        1.000000      6.000000     52.000000     31.000000     12.000000

              Year   PeriodOfDay
count  29576.000000  29576.000000
mean    2012.208074     23.494996
std        0.602320     13.855188
min     2011.000000      0.000000
25%     2012.000000     11.000000
50%     2012.000000     23.000000
75%     2013.000000     35.000000
max     2013.000000     47.000000
```

```python
# Check for missing values
print("Missing Values:")
print(ADS_Phase3.isnull().sum())
```

```
Missing Values:
DateTime                    0
Holiday                     0
HolidayFlag                 0
DayOfWeek                   0
WeekOfYear                  0
Day                         0
Month                       0
Year                        0
PeriodOfDay                 0
ForecastWindProduction      0
SystemLoadEA                0
SMPEA                       0
ORKTemperature              0
ORKWindspeed                0
CO2Intensity                0
ActualWindProduction        0
SystemLoadEP2               0
SMPEP2                      0
dtype: int64
```

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

df = pd.read_csv("/content/Electricity.csv")

# Display the first few rows of the dataset
print(df.head())

# Check for missing values
print("Missing Values:")
print(df.isnull().sum())

# Handle missing values (if any)
# For example, you can fill missing values with the mean of the column
df = df.fillna(df.mean())
```

```
<ipython-input-11-4f5d79a11f70>:5: DtypeWarning: Columns (9,10,11,14,15,16,17) have mixed types. Specify dtype option on import or set low_memory=False.
  df = pd.read_csv("/content/Electricity.csv")
          DateTime Holiday  HolidayFlag  DayOfWeek  WeekOfYear  Day  Month  \
0  01/11/2011 00:00    None            0          1          44    1     11
1  01/11/2011 00:30    None            0          1          44    1     11
2  01/11/2011 01:00    None            0          1          44    1     11
3  01/11/2011 01:30    None            0          1          44    1     11
4  01/11/2011 02:00    None            0          1          44    1     11

   Year  PeriodOfDay ForecastWindProduction SystemLoadEA  SMPEA  \
0  2011            0                315.31      3388.77  49.26
1  2011            1                321.80      3196.66  49.26
2  2011            2                328.57      3060.71  49.10
3  2011            3                335.60      2945.56  48.04
4  2011            4                342.90      2849.34  33.75

  ORKTemperature ORKWindspeed CO2Intensity ActualWindProduction SystemLoadEP2  \
0           6.00         9.30       600.71                356.00       3159.60
1           6.00        11.10       605.42                317.00       2973.01
2           5.00        11.10       589.97                311.00       2834.00
3           6.00         9.30       585.94                313.00       2725.99
4           6.00        11.10       571.52                346.00       2655.64

  SMPEP2
0  54.32
1  54.23
2  54.23
3  53.47
4  39.87
Missing Values:
DateTime                0
Holiday                 0
HolidayFlag             0
DayOfWeek               0
WeekOfYear              0
Day                     0
Month                   0
Year                    0
PeriodOfDay             0
ForecastWindProduction  0
SystemLoadEA            0
SMPEA                   0
ORKTemperature          0
ORKWindspeed            0
CO2Intensity            0
ActualWindProduction    0
SystemLoadEP2           0
SMPEP2                  0
dtype: int64
<ipython-input-11-4f5d79a11f70>:16: FutureWarning: The default value of numeric_only in DataFrame.mean is deprecated. In a future version, it will default
  df = df.fillna(df.mean())
```

2.Converting all non numerical values into numerical values

In [15]:
```
data_frame["ForecastWindProduction"] = pd.to_numeric(data_frame["ForecastWindProduction"], errors= 'coerce')
data_frame["SystemLoadEA"] = pd.to_numeric(data_frame["SystemLoadEA"], errors= 'coerce')
data_frame["SMPEA"] = pd.to_numeric(data_frame["SMPEA"], errors= 'coerce')
data_frame["ORKTemperature"] = pd.to_numeric(data_frame["ORKTemperature"], errors= 'coerce')
data_frame["ORKWindspeed"] = pd.to_numeric(data_frame["ORKWindspeed"], errors= 'coerce')
data_frame["CO2Intensity"] = pd.to_numeric(data_frame["CO2Intensity"], errors= 'coerce')
data_frame["ActualWindProduction"] = pd.to_numeric(data_frame["ActualWindProduction"], errors= 'coerce')
data_frame["SystemLoadEP2"] = pd.to_numeric(data_frame["SystemLoadEP2"], errors= 'coerce')
data_frame["SMPEP2"] = pd.to_numeric(data_frame["SMPEP2"], errors= 'coerce')
```

3.Splitting our data into features and labels

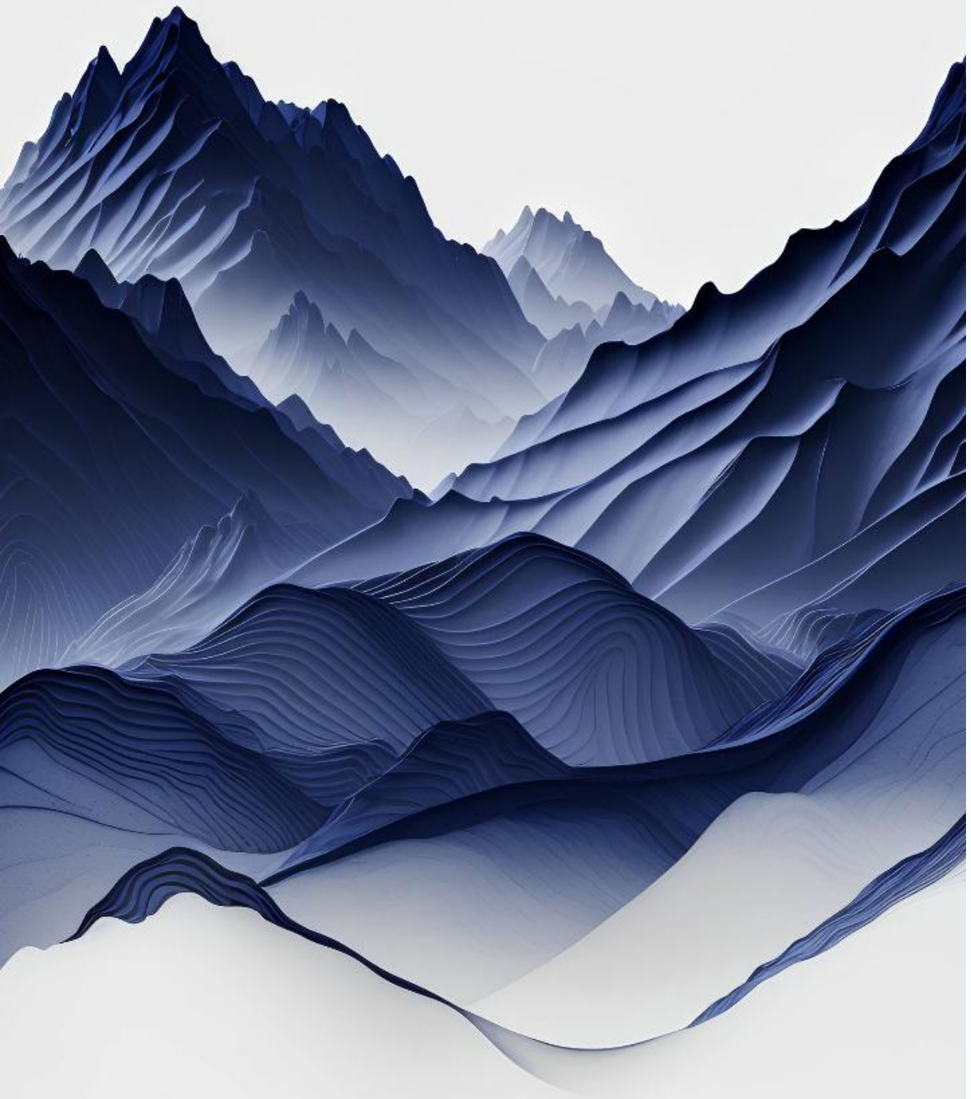
In [16]:
```
x = data_frame[["Day", "Month", "ForecastWindProduction", "SystemLoadEA",
          "SMPEA", "ORKTemperature", "ORKWindspeed", "CO2Intensity",
          "ActualWindProduction", "SystemLoadEP2"]]

y = data_frame["SMPEP2"]
```

4.Splitting our data into training and test set

In [17]:
```
from sklearn.model_selection import train_test_split

xtrain, xtest, ytrain, ytest = train_test_split(x, y,
                                                test_size=0.2,
                                                random_state=42)
```



Conclusion

Data preprocessing involves cleaning and transforming the data to make it suitable for analysis. The goal of data preprocessing is to make the data accurate, consistent, and suitable for analysis. It helps to improve the quality and efficiency of the data mining process.