NAME - AMAN YADAV
CLASS -D15C , ROLL NO - 60

Experiment No-1

Problem Statement:

Introduction to Data Science and Data Preparation using Pandas steps.

Instructions:

1. Load data in Pandas.
2. Provide a description of the dataset.
3. Drop columns that aren't useful.
4. Drop rows with maximum missing values.
5. Take care of missing data.
6. Create dummy variables.
7. Find out outliers (manually).
8. Standardize and normalize columns.

THEORY :

This experiment demonstrates the data preprocessing pipeline applied to a FIFA dataset. The goal is to clean the data, handle missing values, convert categorical variables into numerical format, detect outliers manually, and scale the numeric features. The final output includes both standardized and normalized datasets that can be used for further analysis or modeling.

NAME - AMAN YADAV
CLASS -D15C , ROLL NO - 60

Data Loading and Initial Inspection

```python
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler, MinMaxScaler

# Load the dataset (update the file path accordingly)
data = pd.read_csv("/content/fifa_eda_stats.csv")
print("Initial dataset info:")
print(data.info())
```

```
Initial dataset info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18207 entries, 0 to 18206
Data columns (total 57 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   ID                        18207 non-null  int64
 1   Name                      18207 non-null  object
 2   Age                       18207 non-null  int64
 3   Nationality               18207 non-null  object
 4   Overall                   18207 non-null  int64
 5   Potential                 18207 non-null  int64
 6   Club                      17966 non-null  object
 7   Value                     18207 non-null  object
 8   Wage                      18207 non-null  object
 9   Preferred Foot            18159 non-null  object
```

Overview

- **Dataset Dimensions:** 18,207 entries with 56 columns.
- **Data Types:** Mixed (integer, float, and object).
- **Purpose:** Understand the structure of the dataset and identify columns with missing values.

NAME - AMAN YADAV
CLASS -D15C , ROLL NO - 60

Dropping Unnecessary Columns

```
# Drop unnecessary columns (adjust these based on your analysis)
cols_to_drop = ['ID', 'Name', 'Club', 'Joined', 'Loaned From', 'Contract Valid Until']
data = data.drop(columns=cols_to_drop)
print("\nAfter dropping unnecessary columns:")
print(data.info())
```

```
After dropping unnecessary columns:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18207 entries, 0 to 18206
Data columns (total 51 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Age                       18207 non-null  int64
 1   Nationality               18207 non-null  object
 2   Overall                   18207 non-null  int64
 3   Potential                 18207 non-null  int64
```

Explanation

- **Columns Removed:** Unique identifiers and metadata (e.g., ID, Name, Club), which are not needed for the analysis.
- **Result:** The dataset now contains 51 columns, reducing clutter and focusing on relevant features.

NAME - AMAN YADAV
CLASS -D15C , ROLL NO - 60

Checking for Missing Values

```python
# Check for missing values
print("\nMissing values per column:")
print(data.isnull().sum())
```

```
Missing values per column:
Age                          0
Nationality                  0
Overall                      0
Potential                    0
Value                        0
Wage                         0
Preferred Foot              48
International Reputation    48
Weak Foot                   48
Skill Moves                 48
Work Rate                   48
Body Type                   48
Position                    60
Jersey Number               60
Height                      48
Weight                      48
Crossing                    48
Finishing                   48
HeadingAccuracy             48
ShortPassing                48
Volleys                     48
Dribbling                   48
Curve                       48
FKAccuracy                  48
```

Findings

- Some columns (e.g., Preferred Foot, International Reputation, Work Rate, etc.) have 48–60 missing values.
- The Release Clause column shows 1,564 missing entries.

Note: This step helps determine the strategy for handling missing values in both numeric and categorical features.

NAME - AMAN YADAV
CLASS -D15C , ROLL NO - 60

Removing Rows with Excessive Missing Values

```python
# Remove rows with too many missing entries (keep rows with at least 50% non-null data)
threshold = int(data.shape[1] * 0.5)
data = data.dropna(thresh=threshold)
```

Rows with fewer than 50% non-null entries are dropped. Adjust the threshold if needed.

Explanation

- **Threshold Calculation:** Rows must have non-null entries in at least 50% of the columns.
- **Result:** Rows with insufficient data are removed, ensuring quality in further analysis.

NAME - AMAN YADAV
CLASS -D15C , ROLL NO - 60

Handling Missing Values in Numeric Columns

```
[5] # Fill missing values in numeric columns with the median
    numeric_cols = data.select_dtypes(include=[np.number]).columns
    for col in numeric_cols:
        data[col] = data[col].fillna(data[col].median())
```

Missing numeric values are filled with the median value for each respective column.

Explanation

- **Imputation Method:** Missing numeric values are replaced with the median of the respective column.
- **Rationale:** The median is robust to outliers and appropriate for continuous numerical data.

NAME - AMAN YADAV
CLASS -D15C , ROLL NO - 60

Handling Missing Values in Categorical Columns

```
[6] # Fill missing values in categorical columns with the mode
    categorical_cols = data.select_dtypes(include=['object']).columns
    for col in categorical_cols:
        data[col] = data[col].fillna(data[col].mode()[0])
```

Missing categorical values are filled with the most frequent (mode) value.

Explanation

- **Imputation Method:** Missing categorical values are replaced with the most frequent (mode) value.
- **Rationale:** This ensures that the most common category is preserved for analysis.

NAME - AMAN YADAV
CLASS -D15C , ROLL NO - 60

Creating Dummy Variables

```
[7] # Create dummy variables for categorical columns (drop_first avoids multicollinearity)
    data = pd.get_dummies(data, columns=categorical_cols, drop_first=True)
    print("\nDataset after encoding categorical features:")
    print(data.info())
```

```
Dataset after encoding categorical features:
<class 'pandas.core.frame.DataFrame'>
Index: 18159 entries, 0 to 18206
Columns: 1926 entries, Age to Release Clause_€9M
dtypes: bool(1885), float64(38), int64(3)
memory usage: 38.5 MB
None
```

Categorical features are converted into numerical format using one-hot encoding. The drop_first=True option helps prevent multicollinearity.

Explanation

- **One-Hot Encoding:** Converts categorical features into binary columns.
- **drop_first=True:** Helps reduce multicollinearity by dropping the first category for each feature.
- **Result:** The dataset now contains 1,926 columns (including new dummy variable columns).

NAME - AMAN YADAV
CLASS -D15C , ROLL NO - 60

. Standardizing Numeric Features

```
[8] # Standardize the numeric columns using StandardScaler
    scaler = StandardScaler()
    data_standardized = data.copy()
    data_standardized[numeric_cols] = scaler.fit_transform(data_standardized[numeric_cols])

    # Inspect the first few rows of standardized data
    print("\nFirst few rows of standardized data:")
    print(data_standardized.head())
```

```
First few rows of standardized data:
        Age   Overall  Potential  International Reputation  Weak Foot  \
0  1.258441  4.013364   3.697415                  9.864420   1.593944
1  1.686666  4.013364   3.697415                  9.864420   1.593944
2  0.187878  3.724114   3.534396                  9.864420   3.108090
3  0.401990  3.579489   3.534396                  7.326477   0.079797
4  0.401990  3.579489   3.371377                  7.326477   3.108090

   Skill Moves  Jersey Number  Crossing  Finishing  HeadingAccuracy  ...  \
0     2.167171      -0.598689  1.865922   2.532567         1.018552  ...
1     3.489672      -0.786869  1.865922   2.481351         2.111799  ...
2     3.489672      -0.598689  1.593650   2.122842         0.558238  ...
3    -1.800331      -1.163229 -1.782517  -1.667116        -1.800873  ...
4     2.167171      -0.786869  2.356010   1.866764         0.155463  ...

   Release Clause_€98K  Release Clause_€990K  Release Clause_€991K  \
0                False                 False                 False
1                False                 False                 False
2                False                 False                 False
3                False                 False                 False
4                False                 False                 False

   Release Clause_€992K  Release Clause_€994K  Release Clause_€997K  \
0                 False                 False                 False
1                 False                 False                 False
2                 False                 False                 False
3                 False                 False                 False
4                 False                 False                 False
```

Explanation

- **StandardScaler:** Transforms numeric features so that they have a mean of 0 and a standard deviation of 1.
- **Usage:** Useful when features have different units and scales.

NAME - AMAN YADAV
CLASS -D15C , ROLL NO - 60

Normalizing Numeric Features

```python
# Normalize the numeric columns using MinMaxScaler
minmax_scaler = MinMaxScaler()
data_normalized = data.copy()
data_normalized[numeric_cols] = minmax_scaler.fit_transform(data_normalized[numeric_cols])

# Inspect the first few rows of normalized data
print("\nFirst few rows of normalized data:")
print(data_normalized.head())
```

```
First few rows of normalized data:
        Age   Overall  Potential  International Reputation  Weak Foot  \
0  0.517241  1.000000   0.978723                      1.00       0.75
1  0.586207  1.000000   0.978723                      1.00       0.75
2  0.344828  0.958333   0.957447                      1.00       1.00
3  0.379310  0.937500   0.957447                      0.75       0.50
4  0.379310  0.937500   0.936170                      0.75       1.00

   Skill Moves  Jersey Number  Crossing  Finishing  HeadingAccuracy  ...  \
0         0.75       0.091837  0.897727   1.000000         0.733333  ...
1         1.00       0.061224  0.897727   0.989247         0.944444  ...
2         1.00       0.091837  0.840909   0.913978         0.644444  ...
3         0.00       0.000000  0.136364   0.118280         0.188889  ...
4         0.75       0.061224  1.000000   0.860215         0.566667  ...

   Release Clause_€98K  Release Clause_€990K  Release Clause_€991K  \
0                False                 False                 False
1                False                 False                 False
2                False                 False                 False
3                False                 False                 False
4                False                 False                 False

   Release Clause_€992K  Release Clause_€994K  Release Clause_€997K  \
0                 False                 False                 False
1                 False                 False                 False
2                 False                 False                 False
3                 False                 False                 False
4                 False                 False                 False
```

Explanation

- **MinMaxScaler:** Scales numeric features to a range between 0 and 1.
- **Usage:** Beneficial when comparing features on a similar scale, especially for algorithms sensitive to scale.

NAME - AMAN YADAV
CLASS -D15C , ROLL NO - 60

CONCLUSION :

In conclusion, this experiment demonstrated essential data preprocessing steps using Pandas and Scikit-Learn. By removing irrelevant columns, addressing missing values, creating dummy variables, detecting outliers, and applying both standardization and normalization, we successfully prepared the FIFA dataset for further analysis. This structured approach ensures a robust foundation for effective exploratory data analysis, feature selection, and predictive modeling in subsequent projects.