EXPERIMENT 3A

Exp No: 1.e

Aim:
Demonstrate an experiment to handle missing data and inappropriate data in a dataset using Python Pandas library for data preprocessing.

Algorithm:

1. Import the necessary libraries (NumPy and Pandas).
2. Read the dataset using Pandas.
3. Identify missing data using the info() function.
4. Replace missing values in categorical columns with mode and numerical columns with mean or median.
5. Encode categorical variables using dummy variables.
6. Replace categorical class labels with numerical values for analysis.
7. Display the final preprocessed dataset.

Code:

```
import numpy as np

import pandas as pd

df = pd.read_csv("pre_process_datasample - pre_process_datasample.csv")

df
```

```
   Country   Age   Salary Purchased
0   France  44.0  72000.0        No
1    Spain  27.0  48000.0       Yes
2  Germany  30.0  54000.0        No
3    Spain  38.0  61000.0        No
4  Germany  40.0      NaN       Yes
5   France  35.0  58000.0       Yes
6    Spain   NaN  52000.0        No
7   France  48.0  79000.0       Yes
8  Germany  50.0  83000.0        No
9   France  37.0  67000.0       Yes
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Country    10 non-null     object
 1   Age        9 non-null      float64
 2   Salary     9 non-null      float64
 3   Purchased  10 non-null     object
dtypes: float64(2), object(2)
memory usage: 452.0+ bytes
```

```
df.Country.mode()
0    France
Name: Country, dtype: object
```

```
df.Country.mode()[0]
'France'
```

```
type(df.Country.mode())
pandas.core.series.Series
```

```
df['Country'] = df['Country'].fillna(df['Country'].mode()[0])
df['Age'] = df['Age'].fillna(df['Age'].median())
df['Salary'] = df['Salary'].fillna(round(df['Salary'].mean()))
df
   Country   Age   Salary Purchased
0   France  44.0  72000.0        No
1    Spain  27.0  48000.0       Yes
2  Germany  30.0  54000.0        No
3    Spain  38.0  61000.0        No
```

```
4 Germany  40.0 63778.0     Yes
5  France  35.0 58000.0     Yes
6   Spain  38.0 52000.0      No
7  France  48.0 79000.0     Yes
8 Germany  50.0 83000.0      No
9  France  37.0 67000.0     Yes
```

```
pd.get_dummies(df.Country)
```

```
  France  Germany  Spain
0   True   False  False
1  False   False   True
2  False    True  False
3  False   False   True
4  False    True  False
5   True   False  False
6  False   False   True
7   True   False  False
8  False    True  False
9   True   False  False
```

```
updated_dataset = pd.concat([pd.get_dummies(df.Country), df.iloc[:, [1, 2, 3]]], axis=1)
updated_dataset
```

```
  France  Germany  Spain   Age   Salary Purchased
0   True   False  False  44.0  72000.0       No
1  False   False   True  27.0  48000.0      Yes
2  False    True  False  30.0  54000.0       No
3  False   False   True  38.0  61000.0       No
4  False    True  False  40.0  63778.0      Yes
5   True   False  False  35.0  58000.0      Yes
6  False   False   True  38.0  52000.0       No
7   True   False  False  48.0  79000.0      Yes
```

```
8  False   True  False  50.0  83000.0      No
9   True   False  False  37.0  67000.0     Yes
```

```
df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Country    10 non-null     object 
 1   Age        10 non-null     float64
 2   Salary     10 non-null     float64
 3   Purchased  10 non-null     object 
dtypes: float64(2), object(2)
memory usage: 452.0+ bytes
```

```
updated_dataset['Purchased'] = updated_dataset['Purchased'].replace(['No', 'Yes'], [0, 1])
updated_dataset
   France  Germany  Spain   Age   Salary  Purchased
0   True   False  False  44.0  72000.0       0
1  False   False   True  27.0  48000.0       1
2  False    True  False  30.0  54000.0       0
3  False   False   True  38.0  61000.0       0
4  False    True  False  40.0  63778.0       1
5   True   False  False  35.0  58000.0       1
6  False   False   True  38.0  52000.0       0
7   True   False  False  48.0  79000.0       1
8  False    True  False  50.0  83000.0       0
9   True   False  False  37.0  67000.0       1
```

Result:
Thus, the Python program to handle missing data and inappropriate data using the Pandas library for data preprocessing was executed successfully, and the output was verified.

EXPERIMENT 3B

Aim:
To demonstrate an experiment for detecting, cleaning, and handling duplicate, missing, and inconsistent data in a dataset using Python Pandas library.

Algorithm:

1. Import the necessary libraries (NumPy and Pandas).
2. Read the dataset using Pandas.
3. Identify and remove duplicate records.
4. Reset the index of the dataset.
5. Drop irrelevant or redundant columns.
6. Replace invalid or negative values with NaN.
7. Handle outliers by replacing inappropriate values with NaN.
8. Identify and correct inconsistent categorical values.
9. Fill missing numerical values using mean or median.
10. Display the cleaned and processed dataset.

Code:

```
import numpy as np

import pandas as pd

df = pd.read_csv("Hotel_Dataset - Hotel_Dataset.csv")

df
```

```
  CustomerID  Age_Group  Rating(1-5)   Hotel  FoodPreference  Bill  NoOfPax  EstimatedSalary
Age_Group.1

0    1.0   20-25       4    Ibis       veg    1300  2    40000    20-25

1    2.0   30-35       5  LemonTree    Non-Veg    2000  3    59000    30-35

2    3.0   25-30       6    RedFox       Veg    1322  2    30000    25-30

3    4.0   20-25-1      4  LemonTree       Veg    1234  2   120000    20-25

4    5.0   35+        3     Ibis    Vegetarian    989  2    45000    35+

5    6.0   35+        3     Ibys     Non-Veg    1909  2    122220    35+
6     7.0   35+          4    RedFox    Vegetarian    1000   -1    21122    35+
7     8.0   20-25         7  LemonTree        Veg    2999  -10    345673    20-25
8     9.0   25-30         2      Ibis     Non-Veg    3456  3    99999    25-30
9     9.0   25-30         2      Ibis     Non-Veg    3456  3    99999    25-30
10    10.0   30-35         5    RedFox     non-Veg     -675  4    87777    30-35
```

---

df.duplicated()

```
0    False
1    False
2    False
3    False
4    False
5    False
6    False
7    False
8    False
9     True
10   False
dtype: bool
```

---

df.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11 entries, 0 to 10
Data columns (total 9 columns):
 #  Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0  CustomerID      11 non-null    int64
 1  Age_Group      11 non-null    object
 2  Rating(1-5)     11 non-null    int64
 3  Hotel          11 non-null    object
 4  FoodPreference  11 non-null    object
 5  Bill          11 non-null    int64
 6  NoOfPax       11 non-null    int64
 7  EstimatedSalary  11 non-null    int64
 8  Age_Group.1     11 non-null    object
dtypes: int64(5), object(4)
memory usage: 924.0+ bytes
```

```
df.drop_duplicates(inplace=True)
df
```

```
  CustomerID  Age_Group  Rating(1-5)   Hotel  FoodPreference  Bill  NoOfPax  EstimatedSalary  Age_Group.1
0       1.0   20-25         4     Ibis        veg    1300  2    40000    20-25
1       2.0   30-35         5  LemonTree    Non-Veg    2000  3    59000    30-35
2       3.0   25-30         6    RedFox       Veg    1322  2    30000    25-30
3       4.0   20-25-1       4  LemonTree       Veg    1234  2   120000    20-25
4       5.0   35+          3     Ibis    Vegetarian    989  2    45000    35+
5       6.0   35+          3     Ibys     Non-Veg    1909  2    122220    35+
6       7.0   35+          4    RedFox   Vegetarian    1000  -1    21122    35+
7       8.0   20-25         7  LemonTree       Veg    2999  -10    345673    20-25
8       9.0   25-30         2     Ibis     Non-Veg    3456  3    99999    25-30
9      10.0   30-35         5    RedFox     non-Veg    -675  4    87777    30-35
```

```
len(df)
```

```
10
```

```
index = np.array(list(range(0, len(df))))
df.set_index(index, inplace=True)
index
```

```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

```
df.drop(['Age_Group.1'], axis=1, inplace=True)
```

```
df
  CustomerID  Age_Group  Rating(1-5)   Hotel  FoodPreference  Bill  NoOfPax  EstimatedSalary
0      1.0   20-25        4     Ibis       veg    1300  2    40000
1      2.0   30-35        5  LemonTree    Non-Veg    2000  3    59000
2      3.0   25-30        6    RedFox       Veg    1322  2    30000
3      4.0   20-25-1       4  LemonTree       Veg    1234  2   120000
4      5.0   35+          3     Ibis    Vegetarian    989  2    45000
5      6.0   35+          3     Ibys     Non-Veg   1909  2    122220
6      7.0   35+          4    RedFox   Vegetarian    1000  -1    21122
7      8.0   20-25        7  LemonTree       Veg    2999  -10    345673
8      9.0   25-30        2     Ibis     Non-Veg   3456  3    99999
9     10.0   30-35        5    RedFox     non-Veg    -675  4    87777
```

```
df.loc[df['CustomerID'] < 0, 'CustomerID'] = np.nan
df.loc[df['Bill'] < 0, 'Bill'] = np.nan
df.loc[df['EstimatedSalary'] < 0, 'EstimatedSalary'] = np.nan
df
```

```
df.loc[(df['NoOfPax'] < 1) | (df['NoOfPax'] > 20), 'NoOfPax'] = np.nan
df
```

```
df.Age_Group.unique()
array(['20-25', '30-35', '25-30', '35+'], dtype=object)
```

```
df.Hotel.unique()
array(['Ibis', 'LemonTree', 'RedFox', 'Ibys'], dtype=object)
```

```
df['Hotel'] = df['Hotel'].replace(['Ibys'], 'Ibis')
```

```
df['FoodPreference'].unique()
array(['veg', 'Non-Veg', 'Veg', 'Vegetarian', 'non-Veg'], dtype=object)
```

```
df['FoodPreference'] = df['FoodPreference'].replace(['Vegetarian', 'veg'], 'Veg')

df['FoodPreference'] = df['FoodPreference'].replace(['non-Veg'], 'Non-Veg')
```

```
df['EstimatedSalary'] = df['EstimatedSalary'].fillna(round(df['EstimatedSalary'].mean()))

df['NoOfPax'] = df['NoOfPax'].fillna(round(df['NoOfPax'].median()))

df['Rating(1-5)'] = df['Rating(1-5)'].fillna(round(df['Rating(1-5)'].median()))

df['Bill'] = df['Bill'].fillna(round(df['Bill'].mean()))

df
```

```
  CustomerID Age_Group  Rating(1-5)    Hotel  FoodPreference    Bill  NoOfPax  EstimatedSalary

0      1.0    20-25        4     Ibis        Veg  1300.0    2.0       40000.0

1      2.0    30-35        5  LemonTree     Non-Veg  2000.0    3.0       59000.0

2      3.0    25-30        6    RedFox        Veg  1322.0    2.0       30000.0

3      4.0    20-25        4  LemonTree        Veg  1234.0    2.0      120000.0

4      5.0     35+       3    Ibis        Veg   989.0    2.0       45000.0

5      6.0     35+       3    Ibis     Non-Veg  1909.0    2.0      122220.0

6      7.0     35+       4    RedFox        Veg  1000.0    2.0       21122.0

7      8.0    20-25        7  LemonTree        Veg  2999.0    2.0      345673.0

8      9.0    25-30        2     Ibis     Non-Veg  3456.0    3.0       96755.0

9     10.0    30-35        4     RedFox     Non-Veg  1801.0    4.0       87777.0
```

Result:
Thus, the Python program to detect, clean, and handle duplicate, missing, and inconsistent data using the Pandas library for data preprocessing was executed successfully, and the output was verified.

EXPERIMENT 3C

Aim:

To demonstrate an experiment to handle missing, duplicate, and inappropriate data in a cricketer dataset using Python Pandas Library for Data Preprocessing.

Algorithm:

1. Import necessary libraries (pandas, numpy).
2. Read the dataset using read_csv().

3. Check for duplicate records and remove them.
4. Identify inappropriate values (like negative runs or matches) and replace them with NaN.
5. Standardize inconsistent text entries.
6. Fill missing values using mean or median.
7. Display the final cleaned dataset.

Code:

```
import pandas as pd

import numpy as np

df = pd.read_csv("Cricketer_Imperfect.csv")

df
```

Output:

PlayerID	**Name**	**Age**	**Country**	**Matches_Played**	**Runs**	**Wickets**	**Batting_Avg**	**Bowling_Avg**	**Role**
1	Virat Kohli	35	India	254	12340	4	59.3	0.0	Batsman
2	Rashid Khan	25	Afghanistan	90	1300	230	18.2	21.4	Bowler
3	Ben Stokes	31	England	120	4500	170	42.5	32.1	All-Rounder
4	Steve Smith	33	Australia	150	-500	0	61.7	0.0	Batsman
5	Jasprit Bumrah	29	India	67	50	108	12.5	24.3	Bowler
6	Kane Williamson	32	New Zealand	150	6700	0	55.2	0.0	Batsman
7	Shakib Al Hasan	34	Bangladesh	-10	6500	300	38.1	31.0	All-Rounder
8	Trent Boult	32	New Zealand	100	900	220	15.3	25.4	Bowler

PlayerID	Name	Age	Country	Matches_Played	Runs	Wickets	Batting_Avg	Bowling_Avg	Role
9	AB de Villiers	38	South Africa	228	9577	2	53.5	0.0	Batsman
9	AB de Villiers	38	South Africa	228	9577	2	53.5	0.0	Batsman
10	Mitchell Starc	32	Australia	97	550	200	12.5	23.5	Bowler

```
df.duplicated()
```

Output:

```
0    False
1    False
2    False
3    False
4    False
5    False
6    False
7    False
8    False
9     True
10   False
dtype: bool
```

```
df.info()
```

Output:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11 entries, 0 to 10
Data columns (total 10 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   PlayerID       11 non-null     int64
```

```
1  Name          11 non-null    object
2  Age           11 non-null    int64
3  Country       11 non-null    object
4  Matches_Played  11 non-null    int64
5  Runs          11 non-null    int64
6  Wickets       11 non-null    int64
7  Batting_Avg    11 non-null    float64
8  Bowling_Avg    11 non-null    float64
9  Role          11 non-null    object
dtypes: float64(2), int64(5), object(3)
memory usage: 1012.0+ bytes
```

```
df.drop_duplicates(inplace=True)
df
```

Output:

PlayerID	Name	Age	Country	Matches_Played	Runs	Wickets	Batting_Avg	Bowling_Avg	Role
1	Virat Kohli	35	India	254	12340	4	59.3	0.0	Batsman
2	Rashid Khan	25	Afghanistan	90	1300	230	18.2	21.4	Bowler
3	Ben Stokes	31	England	120	4500	170	42.5	32.1	All-Rounder
4	Steve Smith	33	Australia	150	-500	0	61.7	0.0	Batsman
5	Jasprit Bumrah	29	India	67	50	108	12.5	24.3	Bowler
6	Kane Williamson	32	New Zealand	150	6700	0	55.2	0.0	Batsman

PlayerID	**Name**	**Age**	**Country**	**Matches_Played**	**Runs**	**Wickets**	**Batting_Avg**	**Bowling_Avg**	**Role**
7	Shakib Al Hasan	34	Bangladesh	-10	6500	300	38.1	31.0	All-Rounder
8	Trent Boult	32	New Zealand	100	900	220	15.3	25.4	Bowler
9	AB de Villiers	38	South Africa	228	9577	2	53.5	0.0	Batsman
10	Mitchell Starc	32	Australia	97	550	200	12.5	23.5	Bowler

```
len(df)
```

Output:

```
10
```

```
df.loc[df['Runs'] < 0, 'Runs'] = np.nan

df.loc[df['Matches_Played'] < 0, 'Matches_Played'] = np.nan

df
```

Output:

PlayerID	**Name**	**Age**	**Country**	**Matches_Played**	**Runs**	**Wickets**	**Batting_Avg**	**Bowling_Avg**	**Role**
1	Virat Kohli	35	India	254.0	12340.0	4.0	59.3	0.0	Batsman
2	Rashid Khan	25	Afghanistan	90.0	1300.0	230.0	18.2	21.4	Bowler
3	Ben Stokes	31	England	120.0	4500.0	170.0	42.5	32.1	All-Rounder
4	Steve Smith	33	Australia	150.0	NaN	0.0	61.7	0.0	Batsman
5	Jasprit Bumrah	29	India	67.0	50.0	108.0	12.5	24.3	Bowler

PlayerID	Name	Age	Country	Matches_Played	Runs	Wickets	Batting_Avg	Bowling_Avg	Role
6	Kane Williamson	32	New Zealand	150.0	6700.0	0.0	55.2	0.0	Batsman
7	Shakib Al Hasan	34	Bangladesh	NaN	6500.0	300.0	38.1	31.0	All-Rounder
8	Trent Boult	32	New Zealand	100.0	900.0	220.0	15.3	25.4	Bowler
9	AB de Villiers	38	South Africa	228.0	9577.0	2.0	53.5	0.0	Batsman
10	Mitchell Starc	32	Australia	97.0	550.0	200.0	12.5	23.5	Bowler

---

```
df['Runs'] = df['Runs'].fillna(round(df['Runs'].mean()))

df['Matches_Played'] = df['Matches_Played'].fillna(round(df['Matches_Played'].median()))

df
```

Output:

PlayerID	Name	Age	Country	Matches_Played	Runs	Wickets	Batting_Avg	Bowling_Avg	Role
1	Virat Kohli	35	India	254.0	12340.0	4.0	59.3	0.0	Batsman
2	Rashid Khan	25	Afghanistan	90.0	1300.0	230.0	18.2	21.4	Bowler
3	Ben Stokes	31	England	120.0	4500.0	170.0	42.5	32.1	All-Rounder
4	Steve Smith	33	Australia	150.0	4713.0	0.0	61.7	0.0	Batsman
5	Jasprit Bumrah	29	India	67.0	50.0	108.0	12.5	24.3	Bowler
6	Kane Williamson	32	New Zealand	150.0	6700.0	0.0	55.2	0.0	Batsman

PlayerID	Name	Age	Country	Matches_Played	Runs	Wickets	Batting_Avg	Bowling_Avg	Role
7	Shakib Al Hasan	34	Bangladesh	120.0	6500.0	300.0	38.1	31.0	All-Rounder
8	Trent Boult	32	New Zealand	100.0	900.0	220.0	15.3	25.4	Bowler
9	AB de Villiers	38	South Africa	228.0	9577.0	2.0	53.5	0.0	Batsman
10	Mitchell Starc	32	Australia	97.0	550.0	200.0	12.5	23.5	Bowler

Result:

The Python program to handle missing, duplicate, and inappropriate data using the Pandas library was successfully executed, and the cleaned dataset was obtained as output.