HANDLING MISSING VALUES:

MISSING VALUES:

→ In python, specifically Pandas, Numpy and scikitlearn, we mark missing values as NaN

→ Values with a NaN value are ignored from operations like sum, count etc.

→ We can mark values as NaN easily with the Pandas DataFrame by using the replace() function on a subset of the columns we are interested in.

HOW TO IDENTIFY MISSING VALUES?

→ df.isnull() - It returns True if the values are missing.

HOW TO DEAL WITH MISSING VALUES?

→ Drop the missing values.

↳ Either drop the entire row.

```
df.dropna('col_name', axis = 0, inplace = True)
```

↳ Or drop the entire column.

```
df.dropna('col_name', axis = 1, inplace = True)
```

→ Replace the missing value.

↳ Use Business understanding.

↳ Statistical methods - Imputation
(Mean, Median, Mode)

```
df['col_name'].fillna(df.col_name.mean(), inplace = True)
```

I/P - import numpy as np
import pandas as pd

I/P - df = pd.read_csv('E:\data.csv')
df

O/P -

	FirstName	LastName	Age	Sex	Pre Test score	Post Test score	Location
0	abc	mno	12.0	m	90.0	65.0	NaN
1	NaN	NaN	NaN	NaN	90.0	?	NaN
2	ghi	pqr	12.0	f	-	65.0	?
3	jkl	stu	12.0	f	90.0	62.0	NaN
4	mno	vwx	12.0	fm	89.0	63.0	NaN

I/P - df.shape

O/P - (5,7)

I/P - df.info()

i/p - missing_val = ['n/a', '-', '?']

```
df = pd.read_csv('E:\data.csv', na_values
                 = missing_val)
df.head()
```

o/p -

	First Name	Last Name	Age	Sex	Pre Test score	Post Test score	Location
0	abc	mno	12.0	m	90.0	65.0	NaN
1	NaN	NaN	NaN	NaN	90.0	NaN	NaN
2	ghi	pqr	12.0	f	NaN	65.0	NaN
3	jkl	stu	12.0	f	90.0	62.0	NaN
4	mno	vwx	12.0	m	89.0	63.0	NaN

HOW TO IDENTIFY MISSING VALUES?

i/p - df.isnull()

o/p -

	First Name	Last Name	Age	Sex	Pre Test score	Post Test score	Location
0	False	False	False	False	False	False	True
1	True	True	True	True	False	True	True
2	False	False	False	False	True	False	True
3	False	False	False	False	False	False	True
4	False	False	False	False	False	False	True

Identifying missing values in columns.

I/p - df.isnull.sum()

O/p - First Name 1

Last Name 1

Age 1

Sex 1

PreTest score 1

PostTest score 1

Location 5

dtype : int64

Columns with atleast one missing value.

I/p - df.isnull.any(axis=0)

O/p - First Name True

Last Name True

Age True

Sex True

PreTest score True

PostTest score True

Location True

dtype : bool

Columns with all missing values.

I/p - df.isnull().all(axis=0)

O/p -

First Name	False
Last Name	False
Age	False
Sex	False
Pre Test Score	False
Post Test Score	False
Location	True

dtype: bool

Number of columns with all missing values

I/p - df.isnull.all(axis=0).sum()

O/p - 1

Rows with atleast one missing value

I/p - df.isnull().any(axis=1)

O/p -

0	True
1	True
2	True
3	True
4	True

dtype: bool

Rows with all missing values

I/p - df.isnull().all(axis = 1)

O/p -
```
0    False
1    False
2    False
3    False
4    False
dtype: bool
```

Number of rows with all missing values

I/p - df.isnull.all(axis = 1).sum()

O/p - 0

MISSING VALUES TREATMENT IN COLUMNS :

I/p - round(100*(df.isnull().sum()/len(df.index)), 2)

O/p -
```
First Name          20.0
Last Name           20.0
Age                 20.0
Sex                 20.0
Pre Test Score      20.0
Post Test score     20.0
Location           100.0
dtype: float64
```

removing the location column

I/p - df.dropna(axis=1, how='all', inplace=True)

round(100*(df.isnull().sum()/len(df.index)), 2)

O/p - First Name 20.0

Last Name 20.0

Age 20.0

Sex 20.0

Pre Test score 20.0

Post Test score 20.0

dtype: float64

MISSING VALUES TREATMENT IN ROWS

I/p - df[df.isnull().sum(axis=1) >= 4]

O/p -

	First Name	Last Name	Age	Sex	Pre Test score	Post Test score
1	NaN	NaN	NaN	NaN	90.0	NaN

retaining the rows having <= 4 NaN's

I/p - df = df[df.isnull().sum(axis=1) <= 4]

look at the summary again

. round(100*(df.isnull.sum()/len(df.index)),2)

O/p -

First Name	0.0
Last Name	0.0
Age	0.0
Sex	0.0
PreTest score	25.0
PostTest score	0.0

dtype: float64

I/p - df.head()

	First Name	Last Name	Age	Sex	Pre Test Score	Post Test score
0	abc	mno	12.0	m	90.0	65.0
2	ghi	pqr	12.0	f	NaN	65.0
3	jkl	stu	12.0	f	90.0	62.0
4	mno	vwx	12.0	m	89.0	63.0

I/p - df['preTestScore'].describe()

O/p -

count	3.0000
mean	89.6667
std	0.5773
min	89.0000
25%	89.5000
50%	90.0000
75%	90.0000
max	90.0000

Name: preTestScore, dtype: float64

Imputing pretestscore by mean values

I/p - df['preTestScore'].fillna(df['PreTestScore'].mean(),
inplace=True)

round(100*(df.isnull().sum()/len(df.index)),2)

O/p -

FirstName	0.0
LastName	0.0
Age	0.0
Sex	0.0
PreTestScore	0.0
PostTestScore	0.0

dtype: float64

i/p - df.head()

o/p -

	First Name	Last Name	Age	sex	Pre Test score	Post Test score
0	abc	mno	12.0	m	90.00	65.0
2	ghi	pqr	12.0	f	89.67	65.0
3	jkl	stu	12.0	f	90.00	62.0
4	mno	vwx	120	m	89.00	63.0

fraction of rows lost

i/p → 1 - len(df.index)/5

o/p - 0.19996