Name : Naman Thaker

Roll No : 20BCE529

Subject : Data Mining

Checking missing values in the given data in practical list

```
import numpy as np
import pandas as pd
import seaborn as sns
import·matplotlib.pyplot·as·plt
data_list·=·pd.read_csv('data_practical.csv')
data_list1 = pd.read_csv('data_practical.csv')
```

```
data_list
```

	Name	Value
0	A	45
1	B	37
2	C	59
3	D	?
4	E	47
5	F	39
6	G	?
7	H	43
8	I	52
9	J	?

```
data_list.describe
```

```
<bound method NDFrame.describe of   Name Value
0    A    45
1    B    37
2    C    59
3    D     ?
4    E    47
5    F    39
6    G     ?
7    H    43
8    I    52
9    J     ?>
```

```
data_list.shape
data_list.dtypes
```

```
Name     object
Value    object
dtype: object
```

```
data_list.isnull().sum()
```

```
Name     0
Value    0
dtype: int64
```

```
data_list['Value'] = data_list['Value'].replace(['?'],np.nan)
data_list['Value']
```

```
0     45
1     37
2     59
3    NaN
4     47
5     39
6    NaN
7     43
8     52
9    NaN
Name: Value, dtype: object
```

Filling NaN values with mode of all values

```
#1. Function to replace NAN values with mode value this both rows are categorical,
#not numeric based with datatype of float or int
def impute_nan_most_frequent_category(data_list,ColName):
    # .mode()[0] - gives first category name
     most_frequent_category=data_list[ColName].mode()[0]

    # replace nan values with most occured category
     #data[ColName + "_Imputed"] = data[ColName]
     #data[ColName + "_Imputed"].fillna(most_frequent_category,inplace=True)
     data_list[ColName] = data_list[ColName]
     data_list[ColName].fillna(most_frequent_category,inplace=True)


#2. Call function to impute most occured category
for Columns in ['Value']:
    impute_nan_most_frequent_category(data_list,Columns)

# Display imputed result
data_list[['Value']]
```

	Value
0	45
1	37
2	59
3	37
4	47
5	39
6	37
7	43

Filling NaN values with mean of all values

```
data_list1
```

	Name	Value
0	A	45
1	B	37
2	C	59
3	D	?
4	E	47
5	F	39
6	G	?
7	H	43
8	I	52
9	J	?

```
data_list1['Value'] = data_list1['Value'].replace(['?'],np.nan)
data_list1['Value']
```

```
0     45
1     37
2     59
3    NaN
4     47
5     39
6    NaN
7     43
8     52
9    NaN
Name: Value, dtype: object
```

```
data_list1.shape
```

```
(10, 2)
```

```
data_list1['Value'] = pd.to_numeric(data_list1['Value'],errors='coerce')
data_list1
```

	Name	Value
0	A	45.0
1	B	37.0
2	C	59.0
3	D	NaN
4	E	47.0
5	F	39.0
6	G	NaN
7	H	43.0
8	I	52.0
9	J	NaN

```
values= data_list1.Value[1:]
values
```

```
1    37.0
2    59.0
3     NaN
4    47.0
5    39.0
6     NaN
7    43.0
8    52.0
9     NaN
Name: Value, dtype: float64
```

```
mean_value=data_list1['Value'].mean()

# Replace NaNs in column S2 with the
# mean of values in the same column
data_list1['Value'].fillna(value=mean_value, inplace=True)
print('Updated Dataframe:')
print(data_list1)
```

```
Updated Dataframe:
  Name  Value
0    A   45.0
1    B   37.0
2    C   59.0
3    D   46.0
4    E   47.0
```

```
5    F   39.0
6    G   46.0
7    H   43.0
8    I   52.0
9    J   46.0
```

Checking missing values in Insurance claims fraud Detection Analysis

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

```
data = pd.read_csv('Automobile_insurance_fraud.csv')
data
```

	months_as_customer	age	policy_number	policy_bind_date	policy_state	policy_
0	328	48	521585	17-10-2014	OH	250/
1	228	42	342868	27-06-2006	IN	250/
2	134	29	687698	06-09-2000	OH	100/
3	256	41	227811	25-05-1990	IL	250/
4	228	44	367455	06-06-2014	IL	500/1
...	...	...	...	...	...	
995	3	38	941851	16-07-1991	OH	500/1
996	285	41	186934	05-01-2014	IL	100/
997	130	34	918516	17-02-2003	OH	250/
998	458	62	533940	18-11-2011	IL	500/1
999	456	60	556080	11-11-1996	OH	250/

1000 rows × 40 columns

```
data.describe()
```

	months_as_customer	age	policy_number	policy_deductable	policy_annu
count	1000.000000	1000.000000	1000.000000	1000.000000	
mean	203.954000	38.948000	546238.648000	1136.000000	
std	115.113174	9.140287	257063.005276	611.864673	
min	0.000000	19.000000	100804.000000	500.000000	
25%	115.750000	32.000000	335980.250000	500.000000	
50%	199.500000	38.000000	533135.000000	1000.000000	
75%	276.250000	44.000000	759099.750000	2000.000000	
max	479.000000	64.000000	999435.000000	2000.000000	

```
data.shape
data.dtypes
```

```
months_as_customer               int64
age                              int64
policy_number                    int64
policy_bind_date                object
policy_state                    object
policy_csl                      object
policy_deductable                int64
policy_annual_premium          float64
umbrella_limit                   int64
insured_zip                      int64
insured_sex                     object
insured_education_level         object
insured_occupation              object
insured_hobbies                 object
insured_relationship            object
capital-gains                    int64
capital-loss                     int64
incident_date                   object
incident_type                   object
collision_type                  object
incident_severity               object
authorities_contacted           object
incident_state                  object
incident_city                   object
incident_location               object
incident_hour_of_the_day         int64
number_of_vehicles_involved      int64
property_damage                 object
bodily_injuries                  int64
witnesses                        int64
police_report_available         object
total_claim_amount               int64
injury_claim                     int64
```

```
property_claim                  int64
vehicle_claim                   int64
auto_make                      object
auto_model                     object
auto_year                       int64
fraud_reported                 object
_c39                          float64
dtype: object
```

Checking Null Values in Dataset

```
data.isnull().sum()
```

```
months_as_customer                0
age                               0
policy_number                     0
policy_bind_date                  0
policy_state                      0
policy_csl                        0
policy_deductable                 0
policy_annual_premium             0
umbrella_limit                    0
insured_zip                       0
insured_sex                       0
insured_education_level           0
insured_occupation                0
insured_hobbies                   0
insured_relationship              0
capital-gains                     0
capital-loss                      0
incident_date                     0
incident_type                     0
collision_type                    0
incident_severity                 0
authorities_contacted             0
incident_state                    0
incident_city                     0
incident_location                 0
incident_hour_of_the_day          0
number_of_vehicles_involved       0
property_damage                   0
bodily_injuries                   0
witnesses                         0
police_report_available           0
total_claim_amount                0
injury_claim                      0
property_claim                    0
vehicle_claim                     0
auto_make                         0
auto_model                        0
auto_year                         0
fraud_reported                    0
_c39                           1000
dtype: int64
```

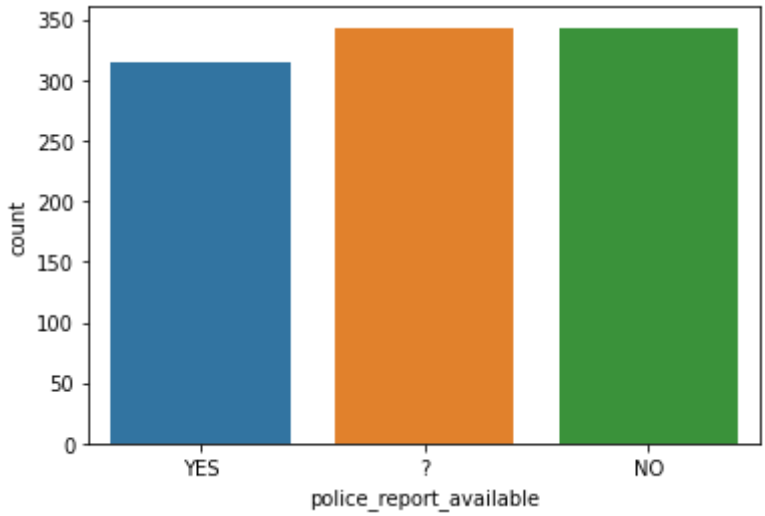
Oservation: 1) we have 1000 number of data and _c39 contains all if the data as null so need to

```
data.drop('_c39',
  axis='columns', inplace=True)
#_c39 is dropped
```

```
import seaborn as sns
alpha = sns.countplot(x="police_report_available",data=data)
print(data["police_report_available"].value_counts())

#Observation: we can change ? with nan and then with mean or median. But if will encode it
```

```
?       343
NO      343
YES     314
Name: police_report_available, dtype: int64
```




```
data['police_report_available'] = data['police_report_available'].replace(['?'],np.nan)
data['police_report_available']
```

```
0      YES
1      NaN
2       NO
3       NO
4       NO
      ...
995    NaN
996    NaN
997    YES
998    YES
999    NaN
Name: police_report_available, Length: 1000, dtype: object
```

```
data.isnull().sum()
```

```
months_as_customer               0
age                              0
policy_number                    0
policy_bind_date                 0
policy_state                     0
```

```
policy_csl                      0
policy_deductable               0
policy_annual_premium           0
umbrella_limit                  0
insured_zip                     0
insured_sex                     0
insured_education_level         0
insured_occupation              0
insured_hobbies                 0
insured_relationship            0
capital-gains                   0
capital-loss                    0
incident_date                   0
incident_type                   0
collision_type                  0
incident_severity               0
authorities_contacted           0
incident_state                  0
incident_city                   0
incident_location               0
incident_hour_of_the_day        0
number_of_vehicles_involved     0
property_damage                 0
bodily_injuries                 0
witnesses                       0
police_report_available       343
total_claim_amount              0
injury_claim                    0
property_claim                  0
vehicle_claim                   0
auto_make                       0
auto_model                      0
auto_year                       0
fraud_reported                  0
dtype: int64
```

Handling null values of police_report_available

```
#1. Function to replace NAN values with mode value this both rows are categorical,
#not numeric based with datatype of float or int
def impute_nan_most_frequent_category(data,ColName):
    # .mode()[0] - gives first category name
     most_frequent_category=data[ColName].mode()[0]

    # replace nan values with most occured category
     #data[ColName + "_Imputed"] = data[ColName]
     #data[ColName + "_Imputed"].fillna(most_frequent_category,inplace=True)
     data[ColName] = data[ColName]
     data[ColName].fillna(most_frequent_category,inplace=True)


#2. Call function to impute most occured category
for Columns in ['police_report_available']:
    impute_nan_most_frequent_category(data,Columns)
```

```
# Display imputed result
data[['police_report_available']].head(10)
```

	police_report_available
0	YES
1	NO
2	NO
3	NO
4	NO
5	NO
6	NO
7	YES
8	YES
9	NO

```
#Rechecking null values in dataset
data.isnull().sum()
```

```
months_as_customer             0
age                            0
policy_number                  0
policy_bind_date               0
policy_state                   0
policy_csl                     0
policy_deductable              0
policy_annual_premium          0
umbrella_limit                 0
insured_zip                    0
insured_sex                    0
insured_education_level        0
insured_occupation             0
insured_hobbies                0
insured_relationship           0
capital-gains                  0
capital-loss                   0
incident_date                  0
incident_type                  0
collision_type                 0
incident_severity              0
authorities_contacted          0
incident_state                 0
incident_city                  0
incident_location              0
incident_hour_of_the_day       0
number_of_vehicles_involved    0
property_damage                0
bodily_injuries                0
witnesses                      0
police_report_available        0
total_claim_amount             0
```

```
injury_claim                 0
property_claim               0
vehicle_claim                0
auto_make                    0
auto_model                   0
auto_year                    0
fraud_reported               0
dtype: int64
```

✓ 0s completed at 12:45 PM