Data cleaning with pandas and preprocessing using scikit learn

Cleaning data

- NaN : not a number
- working with duplicates and missing values
- isnull()
- notnull()'
- dropna()
- fillna()
- replace()
- which values should be replaced with missing values is done based on data identifying and eliminating outliers
- Dropping duplicate data

Identifying and eliminating outliers

- Outliers are observations that are significantly different from other data points
- Outliers can adversely affect the training process of a machine learning algorithm which results in loss of accuracy
- Need to use mathematical formula and retrieve outliers data-InterQuartileRange(IQR)=Q3(Quantile(0.75))-Q1(Quantile(0.25))

In [1]:

```
import pandas as pd
```

In [2]:

```
emp=pd.read_csv("employee.csv")
```

In [3]:

```
emp
```

Out[3]:

	First Name	Gender	Start Date	Last Login Time	Salary	Bonus %	Senior Management	Team
0	Douglas	Male	8/6/1993	12:42 PM	97308	6.945	True	Marketing
1	Thomas	Male	3/31/1996	6:53 AM	61933	4.170	True	NaN
2	Maria	Female	4/23/1993	11:17 AM	130590	11.858	False	Finance
3	Jerry	Male	3/4/2005	1:00 PM	138705	9.340	True	Finance
4	Larry	Male	1/24/1998	4:47 PM	101004	1.389	True	Client Services
...	...	...	...	...	...	...	...	...
995	Henry	NaN	11/23/2014	6:09 AM	132483	16.655	False	Distribution
996	Phillip	Male	1/31/1984	6:30 AM	42392	19.675	False	Finance
997	Russell	Male	5/20/2013	12:39 PM	96914	1.421	False	Product
998	Larry	Male	4/20/2013	4:45 PM	60500	11.985	False	Business Development
999	Albert	Male	5/15/2012	6:24 PM	129949	10.169	True	Sales

1000 rows × 8 columns

In [4]:

```
emp.head()
```

Out[4]:

	First Name	Gender	Start Date	Last Login Time	Salary	Bonus %	Senior Management	Team
0	Douglas	Male	8/6/1993	12:42 PM	97308	6.945	True	Marketing
1	Thomas	Male	3/31/1996	6:53 AM	61933	4.170	True	NaN
2	Maria	Female	4/23/1993	11:17 AM	130590	11.858	False	Finance
3	Jerry	Male	3/4/2005	1:00 PM	138705	9.340	True	Finance
4	Larry	Male	1/24/1998	4:47 PM	101004	1.389	True	Client Services

In [5]:

```
emp.shape
```

Out[5]:

```
(1000, 8)
```

isnull()

- Detect missing values for an array-like object
- isnull() returns True for null values otherwise return False

notnull()

- Detect non-missing values for an array-like object
- notnull() returns True for NOT NULL Values otherwise False

In [6]:

```
emp.isnull()
```

Out[6]:

	First Name	Gender	Start Date	Last Login Time	Salary	Bonus %	Senior Management	Team
0	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	True
2	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...
995	False	True	False	False	False	False	False	False
996	False	False	False	False	False	False	False	False
997	False	False	False	False	False	False	False	False
998	False	False	False	False	False	False	False	False
999	False	False	False	False	False	False	False	False

1000 rows × 8 columns

In [7]:

```
emp.isnull().sum()  # returns the columns in our pandas dataframe along with noo of null
```

Out[7]:

```
First Name            67
Gender               145
Start Date             0
Last Login Time        0
Salary                 0
Bonus %                0
Senior Management     67
Team                  43
dtype: int64
```

In [8]:

```
emp.notnull()
```

Out[8]:

	First Name	Gender	Start Date	Last Login Time	Salary	Bonus %	Senior Management	Team
0	True	True	True	True	True	True	True	True
1	True	True	True	True	True	True	True	False
2	True	True	True	True	True	True	True	True
3	True	True	True	True	True	True	True	True
4	True	True	True	True	True	True	True	True
...	...	...	...	...	...	...	...	...
995	True	False	True	True	True	True	True	True
996	True	True	True	True	True	True	True	True
997	True	True	True	True	True	True	True	True
998	True	True	True	True	True	True	True	True
999	True	True	True	True	True	True	True	True

1000 rows × 8 columns

In [9]:

```
emp.notnull().sum()
```

Out[9]:

```
First Name            933
Gender                855
Start Date           1000
Last Login Time      1000
Salary               1000
Bonus %              1000
Senior Management     933
Team                  957
dtype: int64
```

In [10]:

```
emp.columns
```

Out[10]:

```
Index(['First Name', 'Gender', 'Start Date', 'Last Login Time', 'Salary',
       'Bonus %', 'Senior Management', 'Team'],
      dtype='object')
```

In [11]:

```
emp['Gender']
```

Out[11]:

```
0        Male
1        Male
2      Female
3        Male
4        Male
        ...  
995       NaN
996      Male
997      Male
998      Male
999      Male
Name: Gender, Length: 1000, dtype: object
```

In [12]:

```
emp['Gender'].isnull().sum()
```

Out[12]:

```
145
```

In [14]:

```
pd.isnull(emp['Gender']).sum()
```

Out[14]:

```
145
```

dropna()

- dropna() method removes the rows that contains Null Values
- dropna() returns a new DataFrame object unless the inplace parameter is set to TRue, if inplace is True dropna() does the removing in Original DataFrame
- Syntax:
  - DataFrame.dropna(axis=0,how='any',thresh=None,subset=None,inplace=False)

In [15]:

```
emp.dropna()
```

Out[15]:

	First Name	Gender	Start Date	Last Login Time	Salary	Bonus %	Senior Management	Team
0	Douglas	Male	8/6/1993	12:42 PM	97308	6.945	True	Marketing
2	Maria	Female	4/23/1993	11:17 AM	130590	11.858	False	Finance
3	Jerry	Male	3/4/2005	1:00 PM	138705	9.340	True	Finance
4	Larry	Male	1/24/1998	4:47 PM	101004	1.389	True	Client Services
5	Dennis	Male	4/18/1987	1:35 AM	115163	10.125	False	Legal
...	...	...	...	...	...	...	...	...
994	George	Male	6/21/2013	5:47 PM	98874	4.479	True	Marketing
996	Phillip	Male	1/31/1984	6:30 AM	42392	19.675	False	Finance
997	Russell	Male	5/20/2013	12:39 PM	96914	1.421	False	Product
998	Larry	Male	4/20/2013	4:45 PM	60500	11.985	False	Business Development
999	Albert	Male	5/15/2012	6:24 PM	129949	10.169	True	Sales

764 rows × 8 columns

In [16]:

```
1000-764
```

Out[16]:

```
236
```

In [17]:

```
emp.dropna().shape
```

Out[17]:

```
(764, 8)
```

Filling missing values with meaningful data

- mean
- median
- mode
- constant

fillna()

- fillna() - replaces the missing values with user specified values

In [19]:

```
emp['Gender'].fillna('No gender')   # replace NaN with User specified value i.e No gende
```

Out[19]:

```
0           Male
1           Male
2         Female
3           Male
4           Male
         ...
995    No gender
996         Male
997         Male
998         Male
999         Male
Name: Gender, Length: 1000, dtype: object
```

In [20]:

```
emp['Gender'].fillna(0)
```

Out[20]:

```
0        Male
1        Male
2      Female
3        Male
4        Male
        ...
995         0
996      Male
997      Male
998      Male
999      Male
Name: Gender, Length: 1000, dtype: object
```

In [21]:

```
emp['Gender'].fillna(method="pad")  #filling values with previous once
```

Out[21]:

```
0        Male
1        Male
2      Female
3        Male
4        Male
        ...
995      Male
996      Male
997      Male
998      Male
999      Male
Name: Gender, Length: 1000, dtype: object
```

In [22]:

```
emp['Gender'].fillna(method="bfill")  # backward value
```

Out[22]:

```
0        Male
1        Male
2      Female
3        Male
4        Male
        ...  
995      Male
996      Male
997      Male
998      Male
999      Male
Name: Gender, Length: 1000, dtype: object
```

replace

- the replace() replaces the specified value with another specified value
- replace() searches the entire dataframe and replace everycase of the specified value

- Syntax:
  - dataframe.replace(to_replace,value,inplace,limit,regex,method)

In [23]:

```
emp.replace(to_replace="Male",value=0)  # replace male with 0
```

Out[23]:

	First Name	Gender	Start Date	Last Login Time	Salary	Bonus %	Senior Management	Team
0	Douglas	0	8/6/1993	12:42 PM	97308	6.945	True	Marketing
1	Thomas	0	3/31/1996	6:53 AM	61933	4.170	True	NaN
2	Maria	Female	4/23/1993	11:17 AM	130590	11.858	False	Finance
3	Jerry	0	3/4/2005	1:00 PM	138705	9.340	True	Finance
4	Larry	0	1/24/1998	4:47 PM	101004	1.389	True	Client Services
...	...	...	...	...	...	...	...	...
995	Henry	NaN	11/23/2014	6:09 AM	132483	16.655	False	Distribution
996	Phillip	0	1/31/1984	6:30 AM	42392	19.675	False	Finance
997	Russell	0	5/20/2013	12:39 PM	96914	1.421	False	Product
998	Larry	0	4/20/2013	4:45 PM	60500	11.985	False	Business Development
999	Albert	0	5/15/2012	6:24 PM	129949	10.169	True	Sales

1000 rows × 8 columns

In [24]:

```
help(emp.replace)
```

...

Drop

In [25]:

```
import numpy as np
```

In [26]:

```
di={"First":[100,np.nan,67,87,69,4],
   "second":[90,89,np.nan,78,78,56],
    "third":[23,46,67,789,9,np.nan]
    }

df=pd.DataFrame(di)
df
```

Out[26]:

	First	second	third
0	100.0	90.0	23.0
1	NaN	89.0	46.0
2	67.0	NaN	67.0
3	87.0	78.0	789.0
4	69.0	78.0	9.0
5	4.0	56.0	NaN

In [27]:

```
df.dropna()
```

Out[27]:

	First	second	third
0	100.0	90.0	23.0
3	87.0	78.0	789.0
4	69.0	78.0	9.0

In [28]:

```
df.dropna(axis=0) # removes rows containing missing values
```

Out[28]:

	First	second	third
0	100.0	90.0	23.0
3	87.0	78.0	789.0
4	69.0	78.0	9.0

In [29]:

```
df.dropna(axis=1)   # removes cols having missing values
```

Out[29]:

0
1
2
3
4
5

In [30]:

```
df.isna().sum()
```

Out[30]:

```
First     1
second    1
third     1
dtype: int64
```

Droping duplicate values

dataframe.duplicated()

- the duplicated() returns a series with TRue and False values that decribe which rows in the dataframe are duplicated and not

drop_duplicates

- the drop_dupliactes() removes the duplicate rows
- use the subset parameter if only some columns should be considered when looking for duplicates
- syntax:
  - dataframe.drop_duplicates(subset,keep,inplace,ignore_index)

In [31]:

```
di={"First":[100,89,np.nan,67,87,89],
   "second":[90,80,np.nan,78,78,78],
    "third":[23,46,67,789,9,np.nan]
    }

df=pd.DataFrame(di)
df
```

Out[31]:

	First	second	third
0	100.0	90.0	23.0
1	89.0	80.0	46.0
2	NaN	NaN	67.0
3	67.0	78.0	789.0
4	87.0	78.0	9.0
5	89.0	78.0	NaN

In [32]:

```
df.duplicated()  # returns true if row contains duplicate values
```

Out[32]:

```
0    False
1    False
2    False
3    False
4    False
5    False
dtype: bool
```

In [33]:

```
df.shape
```

Out[33]:

```
(6, 3)
```

In [34]:

```
df.drop_duplicates()
```

Out[34]:

	First	second	third
0	100.0	90.0	23.0
1	89.0	80.0	46.0
2	NaN	NaN	67.0
3	67.0	78.0	789.0
4	87.0	78.0	9.0
5	89.0	78.0	NaN

In [35]:

```
df.drop_duplicates(subset="second")
```

Out[35]:

	First	second	third
0	100.0	90.0	23.0
1	89.0	80.0	46.0
2	NaN	NaN	67.0
3	67.0	78.0	789.0

In [37]:

```
df.drop_duplicates(subset=["second","third"])
```

Out[37]:

	First	second	third
0	100.0	90.0	23.0
1	89.0	80.0	46.0
2	NaN	NaN	67.0
3	67.0	78.0	789.0
4	87.0	78.0	9.0
5	89.0	78.0	NaN

Identifying and Eliminating Outliers

In [38]:

```
adv=pd.read_csv("Advertising.csv")
adv
```

Out[38]:

	TV	radio	newspaper	sales
0	230.1	37.8	69.2	22.1
1	44.5	39.3	45.1	10.4
2	17.2	45.9	69.3	9.3
3	151.5	41.3	58.5	18.5
4	180.8	10.8	58.4	12.9
...	...	...	...	...
195	38.2	3.7	13.8	7.6
196	94.2	4.9	8.1	9.7
197	177.0	9.3	6.4	12.8
198	283.6	42.0	66.2	25.5
199	232.1	8.6	8.7	13.4

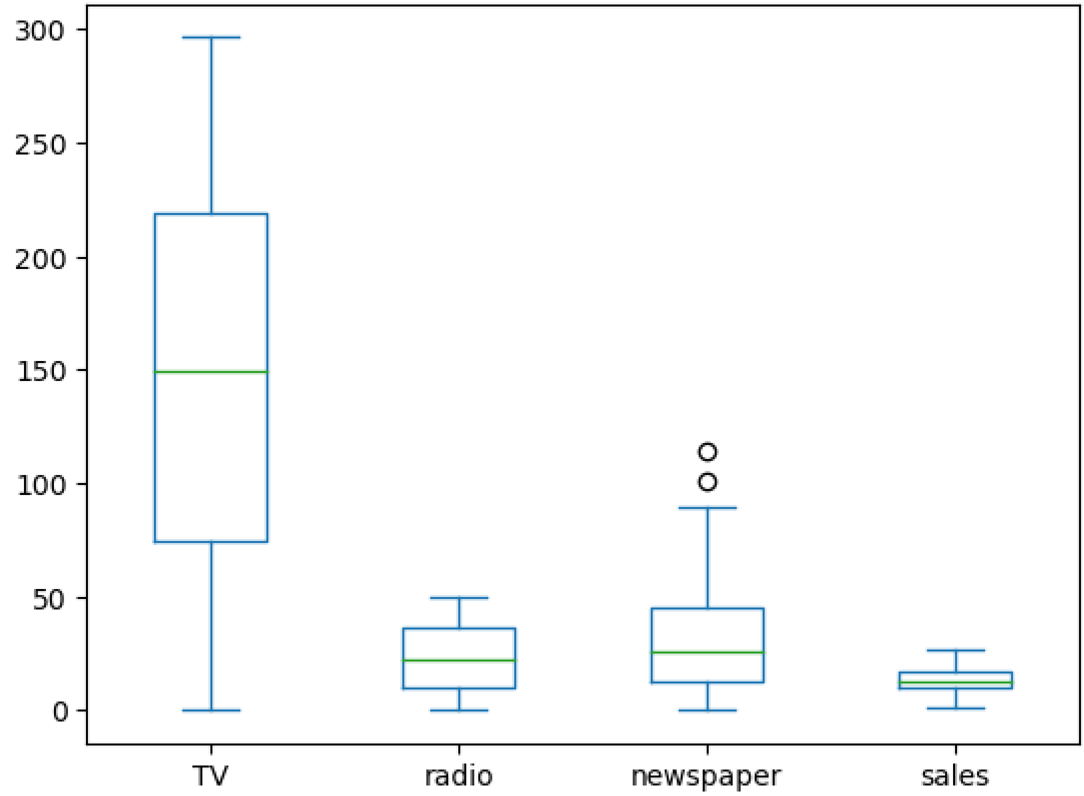
200 rows × 4 columns

In [39]:

```
import matplotlib.pyplot as plt
```

In [40]:

```
adv.plot(kind="box")
plt.show()
```



In [41]:

```
### Interquartile range(IQR)=Q3(Quantile(0.75))-Q1(Quantile(0.25))

Q3=adv.quantile(0.75)
Q1=adv.quantile(0.25)
IQR=Q3-Q1
IQR
```

Out[41]:

```
TV           144.450
radio         26.550
newspaper     32.350
sales          7.025
dtype: float64
```

In [42]:

```
filter_data=adv[(  (adv<(Q1-1.5*IQR)) | (adv>(Q3+1.5*IQR)) ).any(axis=1)]
filter_data
```

Out[42]:

	TV	radio	newspaper	sales
16	67.8	36.6	114.0	12.5
101	296.4	36.3	100.9	23.8

In [43]:

```
filter_data=adv[~((adv<(Q1-1.5*IQR)) | (adv>(Q3+1.5*IQR)) ).any(axis=1)]
filter_data
```

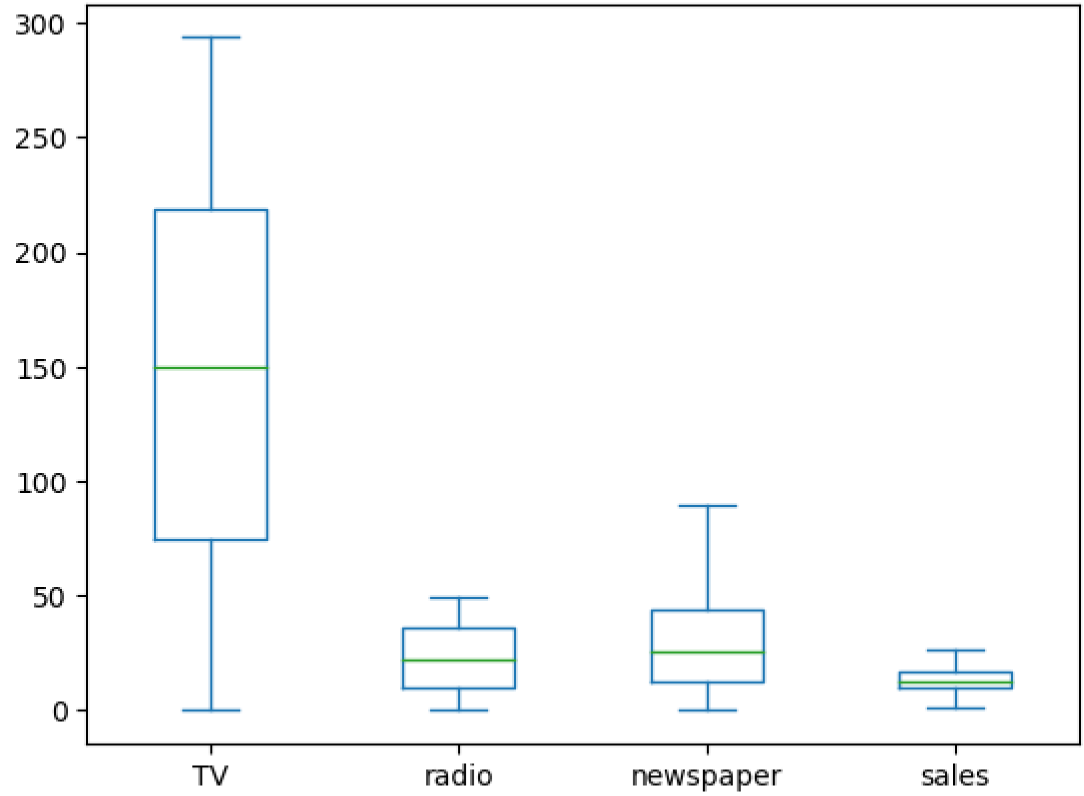
Out[43]:

	TV	radio	newspaper	sales
0	230.1	37.8	69.2	22.1
1	44.5	39.3	45.1	10.4
2	17.2	45.9	69.3	9.3
3	151.5	41.3	58.5	18.5
4	180.8	10.8	58.4	12.9
...	...	...	...	...
195	38.2	3.7	13.8	7.6
196	94.2	4.9	8.1	9.7
197	177.0	9.3	6.4	12.8
198	283.6	42.0	66.2	25.5
199	232.1	8.6	8.7	13.4

198 rows × 4 columns

In [44]:

```
filter_data.plot(kind="box")
plt.show()
```



Data Preprocessing with Scikit learn

Scikit learn

- It is most popular framework used for DataScience
- scikitlearn library includes tools for data preprocessing and data mining
- provides machine learning algorithms classification,regression,clustering,model validation etc
- built on numpy,scipy,matplotlib
- it is imported in python by using import sklearn

Data Preprocessing

- It is a technique that is used to convert raw data into a clean dataset

steps for data preprocessing

- Loading data(reading files)
- exploing data(summarizing data,statistics etc)
- cleaning data(handling missing values)
- Transforming data(Scaling, feature engineering etc)

In [ ]:

```

```