Cleaning data in Python

```
* Dropna()
* Fillna()
* Replace()
```

- **Which values should be replaced with missing values 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, resulting in a loss of accuracy.**
- **Need to use the mathematical formula and retrieve the outlier data.**

interquartile range(IQR) = Q3(quantile(0.75)) – Q1(quantile(0.25))

In [1]:

```
import pandas as pd
import numpy as np
```

In [2]:

```
di = {
    "A" : [85,56,78,np.nan,76],
    "B" : [np.nan,67,8,4,3],
    "C" : [89,78,np.nan,np.nan,5],
    "D" : [0,9,7,6,5]
}
df = pd.DataFrame(di)
df
```

Out[2]:

	A	B	C	D
0	85.0	NaN	89.0	0
1	56.0	67.0	78.0	9
2	78.0	8.0	NaN	7
3	NaN	4.0	NaN	6
4	76.0	3.0	5.0	5

In [3]:

```
df.shape
```

Out[3]:

```
(5, 4)
```

3. Droping missing data

In [7]:

```
df.dropna(axis = 0)   # it checks row by row
```

Out[7]:

	A	B	C	D
1	56.0	67.0	78.0	9
4	76.0	3.0	5.0	5

In [5]:

```
df.dropna(axis  = 1)   # column wise checking
```

Out[5]:

	D
0	0
1	9
2	7
3	6
4	5

In [8]:

```
df.dropna(how = "all")
```

Out[8]:

	A	B	C	D
0	85.0	NaN	89.0	0
1	56.0	67.0	78.0	9
2	78.0	8.0	NaN	7
3	NaN	4.0	NaN	6
4	76.0	3.0	5.0	5

In [9]:

```
df["C"].dropna()
```

Out[9]:

```
0     89.0
1     78.0
4      5.0
Name: C, dtype: float64
```

4. Droping duplicates

In [17]:

```
# Duplicate -- Repeated data
di2 = {
    "A" : [85,56,78,np.nan,76],
    "B" : [np.nan,67,8,4,6],
    "C" : [89,78,np.nan,np.nan,5],
    "D" : [0,9,7,6,5],
    "F" : [85,56,78,np.nan,76]
}
dff = pd.DataFrame(di2)
dff
```

Out[17]:

	A	B	C	D	F

	A	B	C	D	F
0	85.0	NaN	89.0	0	85.0
1	56.0	67.0	78.0	9	56.0
2	78.0	8.0	NaN	7	78.0
3	NaN	4.0	NaN	6	NaN
4	76.0	6.0	5.0	5	76.0

In [18]:

```
dff.duplicated()
```

Out[18]:

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

In [26]:

```
di2 = {
    "A" : [85,56,78,np.nan,76,56],
    "B" : [np.nan,67,8,4,6,67],
    "C" : [89,56,np.nan,np.nan,5,59],
    "D" : [0,56,7,6,5,56],
    "F" : [85,67,78,np.nan,76,67]
}
dff = pd.DataFrame(di2)
dff
```

Out[26]:

	A	B	C	D	F
0	85.0	NaN	89.0	0	85.0
1	56.0	67.0	56.0	56	67.0
2	78.0	8.0	NaN	7	78.0
3	NaN	4.0	NaN	6	NaN
4	76.0	6.0	5.0	5	76.0
5	56.0	67.0	59.0	56	67.0

In [24]:

```
dff.duplicated()
```

Out[24]:

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

In [25]:

```
dff.drop_duplicates()
```

Out[25]:

	A	B	C	D	F
0	85.0	NaN	89.0	0	85.0

	A	B	C	D	F
1	56.0	67.0	56.0	56	67.0
2	78.0	8.0	NaN	7	78.0
3	NaN	4.0	NaN	6	NaN
4	76.0	6.0	5.0	5	76.0

In [27]:

```
dff["A"].drop_duplicatesicates()
```

Out[27]:

```
0    85.0
1    56.0
2    78.0
3     NaN
4    76.0
Name: A, dtype: float64
```

In [30]:

```
dff["C"].drop_duplicates()
```

Out[30]:

```
0    89.0
1    56.0
2     NaN
4     5.0
5    59.0
Name: C, dtype: float64
```

5. Replace

In [31]:

```
# fillna -- it fills only nan values by given data
# replace -- it fills any value by given value
```

In [34]:

```
dff.replace(to_replace = np.nan, value = 100)
# replaceing null values by constant
```

Out[34]:

	A	B	C	D	F
0	85.0	100.0	89.0	0	85.0
1	56.0	67.0	56.0	56	67.0
2	78.0	8.0	100.0	7	78.0
3	100.0	4.0	100.0	6	100.0
4	76.0	6.0	5.0	5	76.0
5	56.0	67.0	59.0	56	67.0

In [35]:

```
dff.replace(to_replace = 56, value = np.nan)
```

Out[35]:

	A	B	C	D	F
0	85.0	NaN	89.0	0.0	85.0
1	NaN	67.0	NaN	NaN	67.0

	A	B	C	D	E
2	78.0	8.0	NaN	7.0	78.0
3	NaN	4.0	NaN	6.0	NaN
4	76.0	6.0	5.0	5.0	76.0
5	NaN	67.0	59.0	NaN	67.0

7. Identifying and Eliminating Outliers

In [ ]:

```
CSE SESSION a -- 60
58 -- students -- cse
2  -- students -- MBA --- Outlier

Marks 100
2 students 102 -- outliers
```

In [36]:

```
adv = pd.read_csv("https://raw.githubusercontent.com/APSSDC-Data-Analysis/DataAnalysis-Batch-7/main/Datasets/Advertising.csv")
adv.head()
```

Out[36]:

	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

In [37]:

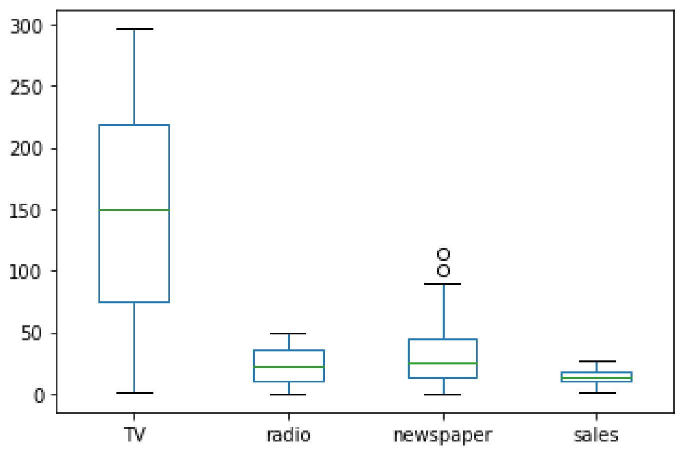
```
import matplotlib.pyplot as plt
```

In [38]:

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

Out[38]:

<matplotlib.axes._subplots.AxesSubplot at 0x19005f0ca00>



In [39]:

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

Out[39]:

```
TV            74.375
radio          9.975
newspaper     12.750
sales         10.375
Name: 0.25, dtype: float64
```

In [40]:

```
Q3 = adv.quantile(0.75)
Q3
```

Out[40]:

```
TV            218.825
radio          36.525
newspaper      45.100
sales          17.400
Name: 0.75, dtype: float64
```

In [41]:

```
IQR = Q3 - Q1
IQR
```

Out[41]:

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

In [43]:

```
min_val = Q1 - 1.5*IQR
max_val = Q3 + 1.5*IQR
```

In [48]:

```
filter_data = adv[((adv < min_val )  | (adv > max_val )).any(axis = 1)]
```

In [49]:

```
filter_data  # outlier data points
```

Out[49]:

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

In [51]:

```
filter_d = adv[~((adv < min_val )  | (adv > max_val )).any(axis = 1)]
```

In [52]:

```
filter_d
```

Out[52]:

	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

	TV	radio	newspaper	sales
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

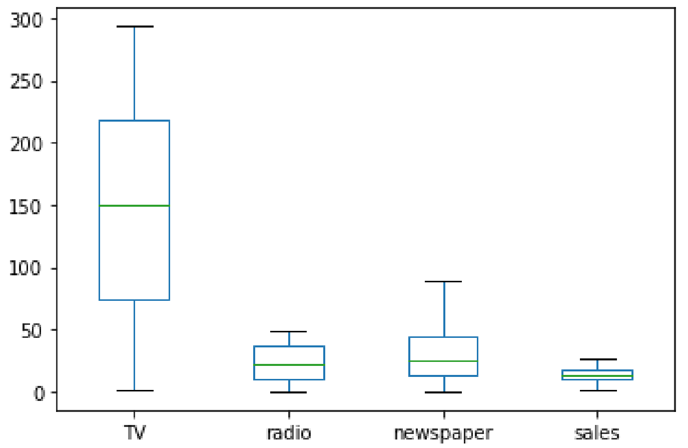
198 rows × 4 columns

In [53]:

```
filter_d.plot(kind = "box")
```

Out[53]:

<matplotlib.axes._subplots.AxesSubplot at 0x190076c2ee0>



In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```

Preprocessing Techniques

- **Data Preprocessing is a technique that is used to convert the raw data into a clean data set**

Data preprocessing steps

- Loading data
- Exploring data
- Cleaning data
- Transforming data
  - will learn data preprocessing techniques with scikit-learn, one of the most popular frameworks used for industry data science
  - The scikit-learn library includes tools for data preprocessing and data mining. It is imported in Python via the statement import sklearn.

Data Imputation

- if the dataset is missing too many values, we just don't use it
- if only a few of the values are missing, we can perform data imputation to substitute the missing data with some other value(s).
- There are many different methods for data imputation
  - Using the mean value
  - Using the median value
  - Using the most frequent value
  - Filling in missing values with a constant

Feature Scaling

1. Standardizing Data

- Data scientists will convert the data into a standard format to make it easier to understand.
- The standard format refers to data that has 0 mean and unit variance (i.e. standard deviation = 1), and the process of converting data into this format is called data standardization.
- improve the performance of models
- it rescales the data to have mean = 0 and varience(statistical measure that provides indicator of data's dispresion) = 1
- Standardization rescales data so that it has a mean of 0 and a standard deviation of 1.
- The formula for this is: (–)/
  - We subtract the mean () from each value (x) and then divide by the standard deviation ()

2. Data Range

- Scale data by compressing it into a fixed range
- One of the biggest use cases for this is compressing data into the range [0, 1]
- Classifier is MinMaxScaler

3. Normalizing Data

- Want to scale the individual data observations (i.e. rows)
- Used in classification Problems and data mining
- ex : columns : salary,ex_yr,position_levels
- when clustering data we need to apply L2 normalization to each row
- L2 normalization applied to a particular row of a data array
- L2 norm of a row is just the square root of the sum of squared values for the row

4. Robust Scaling

- Deal with is outliers (data point that is significantly further away from the other data points)

- **Deal with is outliers (data point that is significantly further away from the other data points)**
- **Robustly scale the data, i.e. avoid being affected by outliers**
- **Scaling by using data's median and Interquartile Range (IQR)**
- **Here mean affected but median remains same**
- **Subtract the median from each data value then scale to the IQR**

In [60]:

```
import sklearn.datasets as sns
```

In [61]:

```
dir(sns)
```

Out[61]:

```
['__all__',
 '__builtins__',
 '__cached__',
 '__doc__',
 '__file__',
 '__loader__',
 '__name__',
 '__package__',
 '__path__',
 '__spec__',
 '_base',
 '_california_housing',
 '_covtype',
 '_kddcup99',
 '_lfw',
 '_olivetti_faces',
 '_openml',
 '_rcv1',
 '_samples_generator',
 '_species_distributions',
 '_svmlight_format_fast',
 '_svmlight_format_io',
 '_twenty_newsgroups',
 'clear_data_home',
 'dump_svmlight_file',
 'fetch_20newsgroups',
 'fetch_20newsgroups_vectorized',
 'fetch_california_housing',
 'fetch_covtype',
 'fetch_kddcup99',
 'fetch_lfw_pairs',
 'fetch_lfw_people',
 'fetch_olivetti_faces',
 'fetch_openml',
 'fetch_rcv1',
 'fetch_species_distributions',
 'get_data_home',
 'load_boston',
 'load_breast_cancer',
 'load_diabetes',
 'load_digits',
 'load_files',
 'load_iris',
 'load_linnerud',
 'load_sample_image',
 'load_sample_images',
 'load_svmlight_file',
 'load_svmlight_files',
 'load_wine',
 'make_biclusters',
 'make_blobs',
 'make_checkerboard',
 'make_circles',
 'make_classification',
 'make_friedman1',
```

```
 'make_friedman2',
 'make_friedman3',
 'make_gaussian_quantiles',
 'make_hastie_10_2',
 'make_low_rank_matrix',
 'make_moons',
 'make_multilabel_classification',
 'make_regression',
 'make_s_curve',
 'make_sparse_coded_signal',
 'make_sparse_spd_matrix',
 'make_sparse_uncorrelated',
 'make_spd_matrix',
 'make_swiss_roll']
```

1. Data Imputation

In [62]:

```
# it handles missing data
```

In [63]:

```
di2 = {
    "A" : [85,56,78,np.nan,76,56],
    "B" : [np.nan,67,8,4,6,67],
    "C" : [89,56,np.nan,np.nan,5,59],
    "D" : [0,56,7,6,5,56],
    "F" : [85,67,78,np.nan,76,67]
}
dff = pd.DataFrame(di2)
dff
```

Out[63]:

	A	B	C	D	F
0	85.0	NaN	89.0	0	85.0
1	56.0	67.0	56.0	56	67.0
2	78.0	8.0	NaN	7	78.0
3	NaN	4.0	NaN	6	NaN
4	76.0	6.0	5.0	5	76.0
5	56.0	67.0	59.0	56	67.0

In [64]:

```
from  sklearn.impute import SimpleImputer
```

In [65]:

```
si = SimpleImputer(strategy= "median")
si.fit_transform(dff)
```

In [66]:

```
si.fit_transform(dff)
```

Out[66]:

```
array([[85. ,  8. , 89. ,  0. , 85. ],
       [56. , 67. , 56. , 56. , 67. ],
       [78. ,  8. , 57.5,  7. , 78. ],
       [76. ,  4. , 57.5,  6. , 76. ],
       [76. ,  6. ,  5. ,  5. , 76. ],
       [56. , 67. , 59. , 56. , 67. ]])
```

In [67]:

In [67]:

```
dff.median()
```

Out[67]:

```
A    76.0
B     8.0
C    57.5
D     6.5
F    76.0
dtype: float64
```

In [68]:

```
si.fit(dff)
```

Out[68]:

```
SimpleImputer(strategy='median')
```

In [69]:

```
si.transform(dff)
```

Out[69]:

```
array([[85. ,  8. , 89. ,  0. , 85. ],
       [56. , 67. , 56. , 56. , 67. ],
       [78. ,  8. , 57.5,  7. , 78. ],
       [76. ,  4. , 57.5,  6. , 76. ],
       [76. ,  6. ,  5. ,  5. , 76. ],
       [56. , 67. , 59. , 56. , 67. ]])
```

In [70]:

```
si = SimpleImputer(strategy= "mean")
si.fit_transform(dff)
```

Out[70]:

```
array([[85.  , 30.4 , 89.  ,  0.  , 85.  ],
       [56.  , 67.  , 56.  , 56.  , 67.  ],
       [78.  ,  8.  , 52.25,  7.  , 78.  ],
       [70.2 ,  4.  , 52.25,  6.  , 74.6 ],
       [76.  ,  6.  ,  5.  ,  5.  , 76.  ],
       [56.  , 67.  , 59.  , 56.  , 67.  ]])
```

In [71]:

```
dff.mean()
```

Out[71]:

```
A    70.200000
B    30.400000
C    52.250000
D    21.666667
F    74.600000
dtype: float64
```

In [73]:

```
si = SimpleImputer(strategy= "most_frequent")
si.fit_transform(dff)
```

Out[73]:

```
array([[85., 67., 89.,  0., 85.],
       [56., 67., 56., 56., 67.],
       [78.,  8.,  5.,  7., 78.],
       [56.,  4.,  5.,  6., 67.],
       [76.,  6.,  5.,  5., 76.],
       [56., 67., 59., 56., 67.]])
```

In [75]:

```
si = SimpleImputer(strategy= "constant", fill_value = -1)
si.fit_transform(dff)
```

Out[75]:

```
array([[85., -1., 89.,  0., 85.],
       [56., 67., 56., 56., 67.],
       [78.,  8., -1.,  7., 78.],
       [-1.,  4., -1.,  6., -1.],
       [76.,  6.,  5.,  5., 76.],
       [56., 67., 59., 56., 67.]])
```

scaling

data -- weights

10gms 250gms 1kg 2tones 100mg

10+250+1+2+100 --- > convert entire data into single units

distance -- cm, m , miles, km...

this is called as scaling

2. Standardizing Data

Z = x - mean(x) / std(x)

In [92]:

```
(adv["TV"][0] - adv["TV"].mean()) / (adv["TV"].std())
```

Out[92]:

```
0.9674245973763037
```

In [76]:

```
adv.head()
```

Out[76]:

	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

In [77]:

```
adv.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   TV         200 non-null    float64
 1   radio      200 non-null    float64
 2   newspaper  200 non-null    float64
 3   sales      200 non-null    float64
```

```
dtypes: float64(4)
memory usage: 6.4 KB
```

In [78]:

```
adv.describe()
```

Out[78]:

	TV	radio	newspaper	sales
count	200.000000	200.000000	200.000000	200.000000
mean	147.042500	23.264000	30.554000	14.022500
std	85.854236	14.846809	21.778621	5.217457
min	0.700000	0.000000	0.300000	1.600000
25%	74.375000	9.975000	12.750000	10.375000
50%	149.750000	22.900000	25.750000	12.900000
75%	218.825000	36.525000	45.100000	17.400000
max	296.400000	49.600000	114.000000	27.000000

In [79]:

```
adv.isnull().sum()
```

Out[79]:

```
TV           0
radio        0
newspaper    0
sales        0
dtype: int64
```

In [80]:

```
from sklearn.preprocessing import scale
```

In [81]:

```
sc = scale(adv)
sc
```

Out[81]:

```
array([[ 9.69852266e-01,  9.81522472e-01,  1.77894547e+00,
         1.55205313e+00],
       [-1.19737623e+00,  1.08280781e+00,  6.69578760e-01,
        -6.96046111e-01],
       [-1.51615499e+00,  1.52846331e+00,  1.78354865e+00,
        -9.07405869e-01],
       [ 5.20496822e-02,  1.21785493e+00,  1.28640506e+00,
         8.60330287e-01],
       [ 3.94182198e-01, -8.41613655e-01,  1.28180188e+00,
        -2.15683025e-01],
       [-1.61540845e+00,  1.73103399e+00,  2.04592999e+00,
        -1.31091086e+00],
       [-1.04557682e+00,  6.43904671e-01, -3.24708413e-01,
        -4.27042783e-01],
       [-3.13436589e-01, -2.47406325e-01, -8.72486994e-01,
        -1.58039455e-01],
       [-1.61657614e+00, -1.42906863e+00, -1.36042422e+00,
        -1.77205942e+00],
       [ 6.16042873e-01, -1.39530685e+00, -4.30581584e-01,
        -6.57617064e-01],
       [-9.45155670e-01, -1.17923146e+00, -2.92486143e-01,
        -1.04190753e+00],
       [ 7.90028350e-01,  4.96973404e-02, -1.22232878e+00,
         6.48970529e-01],
       [-1.43908760e+00,  7.99208859e-01,  1.62704048e+00,
```

```
 -9.26620392e-01],
[-5.78501712e-01, -1.05768905e+00, -1.07502697e+00,
 -8.30547775e-01],
[ 6.66253447e-01,  6.50657027e-01,  7.11007392e-01,
  9.56402904e-01],
[ 5.64664612e-01,  1.65000572e+00,  1.02862691e+00,
  1.60969670e+00],
[-9.25304978e-01,  9.00494200e-01,  3.84117072e+00,
 -2.92541119e-01],
[ 1.56887609e+00,  1.10306488e+00,  1.16211917e+00,
  1.99398717e+00],
[-9.08957349e-01, -1.86635121e-01, -5.64073843e-01,
 -5.23115400e-01],
[ 3.00679600e-03,  4.29449843e-02, -5.27248393e-01,
  1.10963873e-01],
[ 8.33232798e-01,  2.99534513e-01,  1.05164281e+00,
  7.64257669e-01],
[ 1.05509347e+00, -1.22649795e+00, -3.24708413e-01,
 -2.92541119e-01],
[-1.56286250e+00, -4.97243498e-01,  8.76721921e-01,
 -1.61834324e+00],
[ 9.48833887e-01, -4.29719938e-01, -2.00422516e-01,
  2.83894584e-01],
[-9.89527805e-01, -7.20071247e-01, -5.64073843e-01,
 -8.30547775e-01],
[ 1.35285385e+00, -1.33453565e+00, -5.08835667e-01,
 -3.88613736e-01],
[-4.83714657e-02,  4.07572210e-01, -8.26455181e-01,
  1.87821967e-01],
[ 1.08662104e+00, -4.43224650e-01, -3.52327501e-01,
  3.60752677e-01],
[ 1.18820988e+00,  2.59020377e-01, -3.52327501e-01,
  9.37188380e-01],
[-8.92609721e-01, -4.90491142e-01,  4.71641962e-01,
 -6.76831588e-01],
[ 1.70316018e+00,  3.40048650e-01,  5.82118314e-01,
  1.41755147e+00],
[-3.98677796e-01, -3.95958157e-01,  3.70371972e-01,
 -4.07828260e-01],
[-5.82004775e-01, -1.46958277e+00, -2.55016247e-02,
 -8.49762299e-01],
[ 1.38438142e+00, -2.20396901e-01, -1.39264649e+00,
  6.48970529e-01],
[-5.99520091e-01, -1.47633512e+00, -1.06582061e+00,
 -8.68976822e-01],
[ 1.67747105e+00, -1.29402151e+00, -1.01518562e+00,
 -2.34897549e-01],
[ 1.39956136e+00,  1.38666383e+00, -1.17629696e+00,
  2.18613240e+00],
[-8.44734522e-01,  1.76479577e+00,  6.97197848e-01,
  1.30178396e-01],
[-1.21372386e+00,  2.32010953e-01,  2.09260624e-01,
 -7.53689682e-01],
[ 9.45330823e-01,  9.74770116e-01,  6.65620024e-02,
  1.43676599e+00],
[ 6.47570443e-01, -6.50927121e-02,  4.81492770e-02,
  4.95254341e-01],
[ 3.49810063e-01,  6.84418807e-01,  3.74975153e-01,
  5.91326959e-01],
[ 1.71133400e+00,  2.99534513e-01, -1.32359877e+00,
  1.28304980e+00],
[ 6.98948705e-01, -1.00367020e+00, -1.91216154e-01,
 -2.15683025e-01],
[-1.42390765e+00,  1.64487393e-01,  5.86721496e-01,
 -1.06112206e+00],
[ 3.27623995e-01, -5.15880000e-02,  4.35460956e-02,
  1.68607443e-01],
[-6.69581357e-01, -9.02384859e-01,  2.36879713e-01,
 -6.57617064e-01],
[ 1.08428567e+00,  1.23135965e+00, -5.54867481e-01,
  1.76341289e+00],
[ 9.35989321e-01, -5.03995854e-01,  8.90531465e-01,
```

```
        1.49392920e-01],
       [-9.35814168e-01, -7.80842451e-01,  2.87514708e-01,
        -8.30547775e-01],
       [ 6.16042873e-01, -1.36154507e+00,  1.86244718e-01,
        -5.03900877e-01],
       [-5.44638766e-01, -9.22641928e-01, -1.24074150e+00,
        -6.38402541e-01],
       [ 8.09879042e-01,  1.24486436e+00,  4.16403786e-01,
         1.64812575e+00],
       [ 4.15200577e-01,  1.54872038e+00,  1.29561142e+00,
         1.37912242e+00],
       [ 1.35051848e+00,  3.73810430e-01, -6.74550196e-01,
         1.18697718e+00],
       [ 6.05533683e-01,  1.76479577e+00,  1.35545278e+00,
         1.85948550e+00],
       [-1.63175608e+00,  3.26543937e-01,  4.99261050e-01,
        -1.63755776e+00],
       [-1.26606546e-01, -2.74415749e-01, -6.42327927e-01,
        -1.58039455e-01],
       [ 7.44488528e-01,  1.77830048e+00,  3.28943340e-01,
         1.87870003e+00],
       [ 7.43320840e-01,  4.21076922e-01, -9.78360166e-01,
         8.41115763e-01],
       [-1.09228433e+00, -1.43582099e+00, -4.21375221e-01,
        -1.13798015e+00],
       [ 1.33417085e+00,  1.31238792e+00,  1.11148417e+00,
         1.95555812e+00],
       [ 1.07727954e+00, -5.24252922e-01, -1.49787521e-01,
         3.22323631e-01],
       [-5.17781948e-01,  4.27829278e-01, -1.01978880e+00,
        -4.32326777e-03],
       [-1.86158622e-01,  1.31914027e+00, -7.61366196e-02,
         7.64257669e-01],
       [-9.11292725e-01, -9.42898996e-01, -1.36502740e+00,
        -9.07405869e-01],
       [-1.34917564e+00,  9.02114765e-02, -1.30518604e+00,
        -8.68976822e-01],
       [-9.04082253e-02, -5.91776482e-01, -9.36931533e-01,
        -1.19610408e-01],
       [ 1.05509347e+00,  2.86029801e-01, -9.00106083e-01,
         9.37188380e-01],
       [ 8.14549794e-01,  1.39341619e+00, -1.54390703e-01,
         1.59048218e+00],
       [ 6.07869059e-01,  4.95352838e-01,  3.74975153e-01,
         8.21901240e-01],
       [-4.34876116e-01, -6.05281194e-01,  5.27524584e-02,
        -3.11755643e-01],
       [-1.40405696e+00,  6.57409383e-01, -5.18042030e-01,
        -1.00347849e+00],
       [-2.06009314e-01, -1.18598381e+00,  3.43397329e-02,
        -5.80758971e-01],
       [ 7.74848409e-01,  9.02114765e-02, -8.03439274e-01,
         5.72112435e-01],
       [-1.51965805e+00,  1.37991148e+00,  2.70878810e+00,
        -1.02269301e+00],
       [-1.39588315e+00, -1.46283041e+00, -4.53597491e-01,
        -1.36855443e+00],
       [-3.09933525e-01,  3.53553362e-01, -7.52804279e-01,
         3.41057791e-02],
       [-1.65394214e+00,  4.48086346e-01, -9.73756984e-01,
        -1.67598681e+00],
       [-3.62479475e-01, -1.05093669e+00, -3.43121138e-01,
        -5.80758971e-01],
       [-8.24883830e-01,  2.32010953e-01, -3.79946589e-01,
        -4.27042783e-01],
       [ 1.08311798e+00, -1.29402151e+00,  2.92117889e-01,
        -3.30970166e-01],
       [-8.37728396e-01, -2.00139833e-01,  8.95779092e-02,
        -5.23115400e-01],
       [-9.18298852e-01,  1.43393033e+00,  2.32276531e-01,
        -8.11813615e-02],
       [ 7.76016097e-01,  1.33264499e+00,  1.49419267e-01,
```

```
         1.47519504e+00],
       [ 5.38975481e-01, -3.28434597e-01,  1.61783412e+00,
         2.26251013e-01],
       [-8.26051518e-01,  2.86029801e-01, -6.69947015e-01,
        -3.88613736e-01],
       [-4.24366926e-01,  1.17058844e+00,  1.50275459e+00,
         3.79967201e-01],
       [-6.85928986e-01,  1.50982681e-01,  1.97227908e+00,
        -2.15683025e-01],
       [-4.34876116e-01,  1.65675807e+00,  9.59579186e-01,
         5.14468865e-01],
       [-1.48792614e-01, -1.24000266e+00, -9.78360166e-01,
        -5.42329924e-01],
       [-1.38303858e+00, -1.46958277e+00,  1.12593816e-01,
        -1.29169634e+00],
       [ 8.25058983e-01,  6.91171163e-01,  1.30942097e+00,
         1.03326100e+00],
       [ 1.21273132e+00,  8.93741844e-01,  1.92164409e+00,
         1.57126765e+00],
       [-4.62900623e-01, -6.25538262e-01, -9.04709264e-01,
        -4.84686354e-01],
       [ 1.89836839e-01,  5.62876398e-01,  1.02862691e+00,
         5.52897912e-01],
       [ 5.90353742e-01, -1.33453565e+00, -1.13486833e+00,
        -4.46257307e-01],
       [ 4.42057396e-01, -1.52873340e-01, -3.93756133e-01,
         2.83894584e-01],
       [ 1.66579418e+00,  1.28537849e+00,  9.50372823e-01,
         2.18613240e+00],
       [-1.38283424e-01,  1.24486436e+00,  7.06404211e-01,
         6.10541482e-01],
       [ 8.79940308e-01, -1.28051680e+00,  8.85928284e-01,
        -4.46257307e-01],
       [ 1.74402926e+00,  8.80237132e-01,  3.23815396e+00,
         1.87870003e+00],
       [ 1.55486384e+00, -8.88880147e-01, -4.21375221e-01,
         1.49392920e-01],
       [ 4.77088029e-01, -4.09462869e-01, -5.82486569e-01,
         1.30178396e-01],
       [ 1.06443498e+00,  7.45190011e-01, -1.16248742e+00,
         1.28304980e+00],
       [-1.06755854e-01,  1.56222509e+00,  1.30942097e+00,
         9.94831951e-01],
       [-1.42507534e+00, -8.28108943e-01, -3.93111688e-02,
        -1.31091086e+00],
       [-6.61407543e-01, -1.55061104e+00, -3.38517957e-01,
        -1.02269301e+00],
       [-1.56403019e+00, -1.54385868e+00, -2.28041604e-01,
        -1.67598681e+00],
       [ 1.26527727e+00,  2.45515665e-01, -1.15328106e+00,
         1.11011909e+00],
       [ 9.19641692e-01, -1.01717491e+00,  1.19434143e+00,
        -1.19610408e-01],
       [ 1.10530405e+00,  9.95027184e-01, -3.38517957e-01,
         1.49440956e+00],
       [ 3.34630122e-01, -5.31005278e-01, -1.29597968e+00,
         1.48912557e-02],
       [ 7.30476274e-01, -1.79882765e-01, -9.13915627e-01,
         3.60752677e-01],
       [-8.03865450e-01,  1.58923451e+00,  1.81641536e-01,
         1.10963873e-01],
       [-8.40063771e-01,  7.92456503e-01,  1.01942054e+00,
        -2.73326596e-01],
       [-9.15759131e-02, -6.05281194e-01, -2.28041604e-01,
        -3.50184689e-01],
       [-8.24883830e-01, -1.51684926e+00, -7.25185191e-01,
        -8.88191346e-01],
       [-2.49213762e-01,  9.20751268e-01,  2.23926360e+00,
         3.60752677e-01],
       [-1.49046586e+00, -4.90491142e-01, -3.79946589e-01,
        -1.42619800e+00],
       [-6.70544700e-02,  2.38763309e-01,  7.20213755e-01,
```

```
         2.83894584e-01],
       [-1.49747198e+00, -1.05606848e-01,  9.13547372e-01,
        -1.34933991e+00],
       [ 8.98623313e-01, -1.40881156e+00, -6.88359740e-01,
        -4.65471830e-01],
       [-2.79573643e-01,  7.65447079e-01, -8.35661544e-01,
         2.26251013e-01],
       [ 9.62846140e-01,  6.10142891e-01,  2.00910454e+00,
         1.09090457e+00],
       [-6.98773552e-01, -7.74090095e-01, -2.14232060e-01,
        -6.57617064e-01],
       [-1.62591764e+00,  1.05579839e+00,  9.22753735e-01,
        -1.42619800e+00],
       [-7.80511695e-01, -1.57086811e+00, -9.82963347e-01,
        -1.00347849e+00],
       [ 8.55418865e-01,  1.73778635e+00, -1.25915423e+00,
         2.05163074e+00],
       [-1.02105537e+00, -7.60585383e-01,  5.77515133e-01,
        -8.30547775e-01],
       [-1.70882347e+00,  1.10306488e+00, -1.00597925e+00,
        -2.38692417e+00],
       [ 1.37971067e+00, -1.37504978e+00,  5.72911952e-01,
        -2.54112072e-01],
       [-1.61891151e+00,  2.65772733e-01, -1.30978922e+00,
        -1.59912871e+00],
       [ 8.49580427e-01,  6.91171163e-01,  6.69578760e-01,
         1.07169004e+00],
       [-1.28612050e+00,  1.03554132e+00,  1.61323094e+00,
        -6.19188018e-01],
       [-1.15300409e+00,  1.60273923e+00, -1.01518562e+00,
        -4.65471830e-01],
       [-1.41806922e+00,  1.06255074e+00, -9.78360166e-01,
        -8.68976822e-01],
       [ 1.47896413e+00,  3.80562786e-01,  1.34164324e+00,
         1.30226433e+00],
       [-1.21489154e+00,  1.77992105e-01, -4.62803854e-01,
        -8.49762299e-01],
       [ 4.42057396e-01,  1.39341619e+00, -1.32820195e+00,
         1.28304980e+00],
       [-8.59914463e-01, -4.22967582e-01, -8.12645637e-01,
        -5.99973494e-01],
       [ 5.44813920e-01,  8.19465927e-01,  2.07354907e+00,
         9.94831951e-01],
       [ 8.57754241e-01,  6.70914095e-01,  3.38149702e-01,
         1.16776266e+00],
       [-4.95595880e-01, -1.18598381e+00,  1.77038355e-01,
        -6.96046111e-01],
       [-5.93681653e-01, -5.71519414e-01,  3.84181516e-01,
        -5.03900877e-01],
       [-7.87313476e-02, -1.44257334e+00, -9.92169710e-01,
        -7.15260635e-01],
       [ 1.08662104e+00, -1.07794612e+00, -1.00597925e+00,
        -1.58039455e-01],
       [ 1.12281936e+00,  1.73778635e+00,  6.32753309e-01,
         2.18613240e+00],
       [-1.27327593e+00,  1.15033137e+00, -8.58677450e-01,
        -5.99973494e-01],
       [-1.19504085e+00,  1.71239749e-01, -4.58200672e-01,
        -7.53689682e-01],
       [ 1.56070228e+00, -6.32290618e-01,  2.96721070e-01,
         3.99181724e-01],
       [-3.04095087e-01, -1.00367020e+00,  8.35293289e-01,
        -4.65471830e-01],
       [ 5.90353742e-01,  2.43084817e-03, -7.52804279e-01,
         4.95254341e-01],
       [ 2.83251860e-01,  1.10981724e+00,  3.28943340e-01,
         9.56402904e-01],
       [ 4.75920341e-01, -1.46120984e-01, -9.69153803e-01,
         3.03109107e-01],
       [-1.66912209e+00, -7.87594807e-01, -1.14407469e+00,
        -2.07949180e+00],
       [-6.20538471e-01,  1.36640677e+00,  9.18150553e-01,
```

```
  2.45465537e-01],
[ 3.21989902e-02, -1.48308748e+00, -2.87882962e-01,
 -7.53689682e-01],
[-1.58037782e+00,  9.20751268e-01,  6.74181942e-01,
 -1.29169634e+00],
[-1.79152496e-01, -3.28434597e-01,  1.86244718e-01,
 -2.15683025e-01],
[ 2.97264113e-01, -3.48691665e-01,  6.72064478e-03,
  7.25348259e-02],
[-7.16288868e-01,  8.46475352e-01,  8.62912377e-01,
 -1.38824932e-01],
[ 4.82926468e-01, -3.48691665e-01, -2.28041604e-01,
  1.68607443e-01],
[ 1.92172214e-01,  9.13998912e-01, -1.06582061e+00,
  7.64257669e-01],
[-3.48467222e-01, -5.78271770e-01, -1.15788424e+00,
 -4.07828260e-01],
[ 1.02123053e+00, -1.34128800e+00,  2.49704176e+00,
 -4.07828260e-01],
[-1.50798117e+00,  9.68017760e-01, -4.12168859e-01,
 -1.15719467e+00],
[ 6.97781017e-01, -1.21974559e+00, -5.13438849e-01,
 -3.50184689e-01],
[ 7.98202165e-01,  2.26879163e-02,  1.24497643e+00,
  5.91326959e-01],
[ 1.60273904e+00, -8.55118367e-01, -1.11185242e+00,
  1.87821967e-01],
[-1.13315340e+00, -7.87594807e-01, -5.59470662e-01,
 -1.08033658e+00],
[ 2.03849092e-01, -1.59625696e-01,  7.75451931e-01,
  9.17493494e-02],
[-1.48813048e+00, -2.13644545e-01, -6.23915201e-01,
 -1.23405277e+00],
[ 2.49388915e-01, -1.09145083e+00, -8.17248818e-01,
 -4.46257307e-01],
[ 8.79940308e-01, -1.34128800e+00, -8.03439274e-01,
 -4.84686354e-01],
[ 1.51633014e+00,  1.73103399e+00,  5.17673775e-01,
  2.49356478e+00],
[ 1.18353913e+00,  4.68343414e-01, -4.72010216e-01,
  1.18697718e+00],
[ 2.70407294e-01, -1.04418434e+00,  2.13863806e-01,
 -4.46257307e-01],
[ 1.51399477e+00, -1.41556392e+00, -3.15502050e-01,
 -4.27042783e-01],
[ 2.16693657e-01, -8.95632503e-01, -5.96296113e-01,
 -2.73326596e-01],
[ 1.11601758e-01, -1.39530685e+00, -1.02439198e+00,
 -6.76831588e-01],
[ 8.34400486e-01, -1.20624088e+00, -1.45184340e-01,
 -3.50184689e-01],
[-1.06075676e+00, -1.18598381e+00, -3.93111688e-02,
 -1.02269301e+00],
[ 1.64127273e+00,  1.33264499e+00,  1.89862818e+00,
  2.33984859e+00],
[ 1.24659427e+00, -1.32616272e-01, -2.55016247e-02,
  6.87399576e-01],
[ 6.76762637e-01,  1.47444446e+00, -5.04232486e-01,
  1.64812575e+00],
[-8.80728498e-02, -1.42906863e+00, -1.82009791e-01,
 -7.15260635e-01],
[ 5.14454038e-01,  3.67058074e-01, -5.68677025e-01,
  6.29756005e-01],
[ 1.62258973e+00, -6.32290618e-01, -1.23613832e+00,
  3.60752677e-01],
[-1.49863967e+00, -7.53833027e-01, -3.29311594e-01,
 -1.40698348e+00],
[-1.25576062e+00,  1.20435022e+00, -1.13947151e+00,
 -6.19188018e-01],
[-8.35393020e-01, -8.41613655e-01, -1.13026515e+00,
 -7.92118728e-01],
[-1.51615499e+00, -1.29402151e+00,  4.81492770e-02,
```

```
         -1.56069967e+00],
        [ 2.30705910e-01,  1.26512143e+00, -1.24074150e+00,
          1.07169004e+00],
        [ 3.10313024e-02,  8.32970639e-01, -1.13026515e+00,
          6.29756005e-01],
        [-1.27094056e+00, -1.32103093e+00, -7.71217005e-01,
         -1.23405277e+00],
        [-6.17035408e-01, -1.24000266e+00, -1.03359834e+00,
         -8.30547775e-01],
        [ 3.49810063e-01, -9.42898996e-01, -1.11185242e+00,
         -2.34897549e-01],
        [ 1.59456522e+00,  1.26512143e+00,  1.64085003e+00,
          2.20534693e+00],
        [ 9.93206022e-01, -9.90165488e-01, -1.00597925e+00,
         -1.19610408e-01]])
```

In [82]:

```
sc_data = pd.DataFrame(sc,columns = adv.columns)
sc_data
```

Out[82]:

	TV	radio	newspaper	sales
0	0.969852	0.981522	1.778945	1.552053
1	-1.197376	1.082808	0.669579	-0.696046
2	-1.516155	1.528463	1.783549	-0.907406
3	0.052050	1.217855	1.286405	0.860330
4	0.394182	-0.841614	1.281802	-0.215683
...	...	...	...	...
195	-1.270941	-1.321031	-0.771217	-1.234053
196	-0.617035	-1.240003	-1.033598	-0.830548
197	0.349810	-0.942899	-1.111852	-0.234898
198	1.594565	1.265121	1.640850	2.205347
199	0.993206	-0.990165	-1.005979	-0.119610

200 rows × 4 columns

In [83]:

```
adv.mean()
```

Out[83]:

```
TV             147.0425
radio           23.2640
newspaper       30.5540
sales           14.0225
dtype: float64
```

In [84]:

```
adv.std()
```

Out[84]:

```
TV             85.854236
radio          14.846809
newspaper      21.778621
sales           5.217457
dtype: float64
```

In [86]:

```
sc_data.mean().round(3)
```

Out[86]:

```
TV            0.0
radio        -0.0
newspaper     0.0
sales        -0.0
dtype: float64
```

In [87]:

```
sc_data.std()
```

Out[87]:

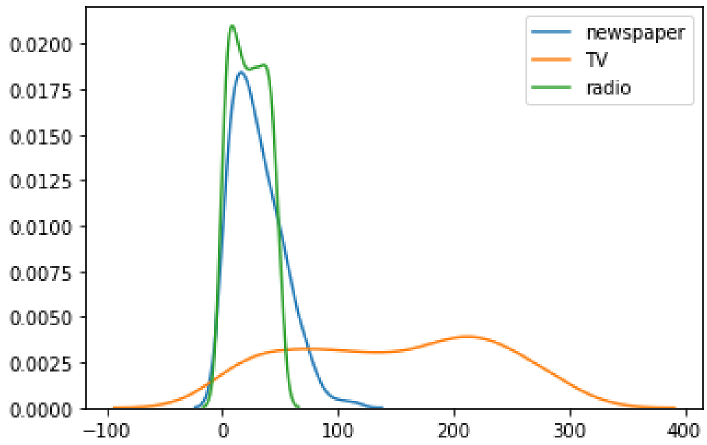
```
TV           1.002509
radio        1.002509
newspaper    1.002509
sales        1.002509
dtype: float64
```

In [88]:

```
# Before scaling
import seaborn as sns
sns.kdeplot(adv["newspaper"])
sns.kdeplot(adv["TV"])
sns.kdeplot(adv["radio"])
```

Out[88]:

<matplotlib.axes._subplots.AxesSubplot at 0x19009b35a30>

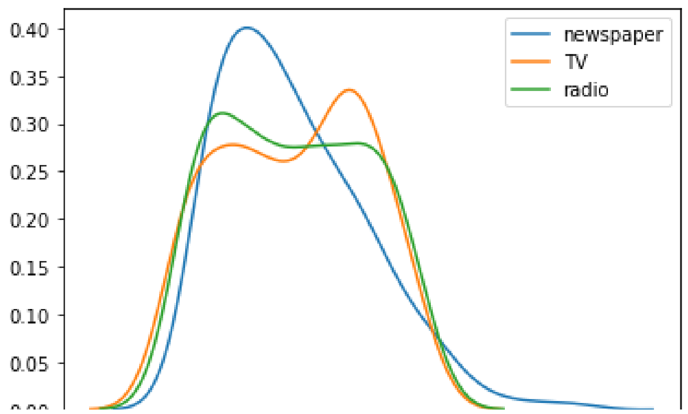


In [89]:

```
# after scaling
import seaborn as sns
sns.kdeplot(sc_data["newspaper"])
sns.kdeplot(sc_data["TV"])
sns.kdeplot(sc_data["radio"])
```

Out[89]:

<matplotlib.axes._subplots.AxesSubplot at 0x19009b288e0>



3. Data range

Z = (X - Xmin) / (Xmax - Xmin)

In [103]:

```
(adv["TV"][0] - adv["TV"].min()) / (adv["TV"].max() - adv["TV"].min())
```

Out[103]:

```
0.7757862698681096
```

In [93]:

```
adv.head()
```

Out[93]:

	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

In [94]:

```
from sklearn.preprocessing import MinMaxScaler
```

In [96]:

```
mnsc = MinMaxScaler()
mnsc
```

Out[96]:

```
MinMaxScaler()
```

In [98]:

```
mnsc = mnsc.fit_transform(adv)
```

In [99]:

```
mnsc_data = pd.DataFrame(mnsc, columns = adv.columns)
mnsc_data.head()
```

Out[99]:

	TV	radio	newspaper	sales
0	0.775786	0.762097	0.605981	0.807087
1	0.148123	0.792339	0.394019	0.346457
2	0.055800	0.925403	0.606860	0.303150
3	0.509976	0.832661	0.511873	0.665354
4	0.609063	0.217742	0.510994	0.444882

In [100]:

```
mnsc_data.min()
```

Out[100]:

Out[100]:

```
TV           0.0
radio        0.0
newspaper    0.0
sales        0.0
dtype: float64
```

In [101]:

```
mnsc_data.max()
```

Out[101]:

```
TV           1.0
radio        1.0
newspaper    1.0
sales        1.0
dtype: float64
```

4. Normalizing Data

appling rescaling techniques for each and evry record

In [105]:

```
home = pd.read_csv("https://raw.githubusercontent.com/APSSDC-Data-Analysis/DataAnalysis-Batch-7/main/Datasets/HomeBuyer.csv")
home.head(10)
```

Out[105]:

	Age	EstimatedSalary	Purchased
0	19	19000	0
1	35	20000	0
2	26	43000	0
3	27	57000	0
4	19	76000	0
5	27	58000	0
6	27	84000	0
7	32	150000	1
8	25	33000	0
9	35	65000	0

In [110]:

```
from sklearn.preprocessing import Normalizer
norm = Normalizer()
norm = norm.fit_transform(home)
```

In [111]:

```
norm_data = pd.DataFrame(norm,columns = home.columns)
norm_data
```

Out[111]:

	Age	EstimatedSalary	Purchased
0	0.001000	1.000000	0.000000
1	0.001750	0.999998	0.000000
2	0.000605	1.000000	0.000000

	Age	EstimatedSalary	Purchased
3	0.000474	1.000000	0.000000
4	0.000250	1.000000	0.000000
...	...	...	...
395	0.001122	0.999999	0.000024
396	0.002217	0.999998	0.000043
397	0.002500	0.999997	0.000050
398	0.001091	0.999999	0.000000
399	0.001361	0.999999	0.000028

400 rows × 3 columns

In [112]:

```
norm_data.max()
```

Out[112]:

```
Age                0.002522
EstimatedSalary    1.000000
Purchased          0.000050
dtype: float64
```

In [113]:

```
norm_data.min()
```

Out[113]:

```
Age                0.000196
EstimatedSalary    0.999997
Purchased          0.000000
dtype: float64
```

5. Robust Scaling

handling outliers

In [114]:

```
adv.head()
```

Out[114]:

	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

In [115]:

```
from sklearn.preprocessing import RobustScaler
Rsc = RobustScaler()
```

In [120]:

```
Rsc = Rsc.fit_transform(adv)
```

In [121]:

```
rsc_data = pd.DataFrame(Rsc, columns = adv.columns)
```

```
rsc_data
```

Out[121]:

	TV	radio	newspaper	sales
0	0.556248	0.561205	1.343122	1.309609
1	-0.728626	0.617702	0.598145	-0.355872
2	-0.917619	0.866290	1.346213	-0.512456
3	0.012115	0.693032	1.012365	0.797153
4	0.214953	-0.455744	1.009274	0.000000
...	...	...	...	...
195	-0.772240	-0.723164	-0.369397	-0.754448
196	-0.384562	-0.677966	-0.545595	-0.455516
197	0.188647	-0.512241	-0.598145	-0.014235
198	0.926618	0.719397	1.250386	1.793594
199	0.570093	-0.538606	-0.527048	0.071174

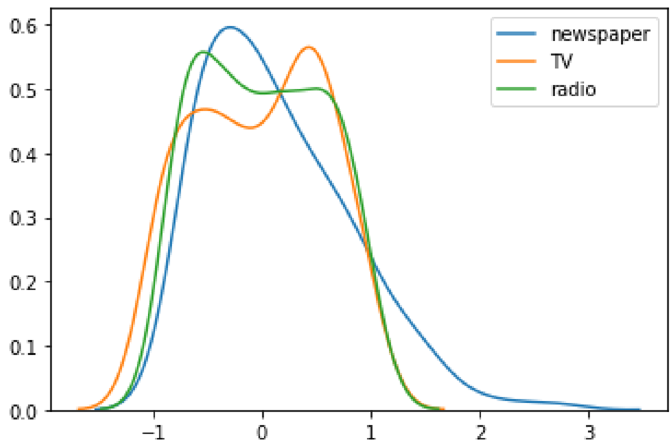
200 rows × 4 columns

In [122]:

```
sns.kdeplot(rsc_data["newspaper"])
sns.kdeplot(rsc_data["TV"])
sns.kdeplot(rsc_data["radio"])
```

Out[122]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1900a3157f0>
```



In [ ]: