Data cleaning:

- missing data [replace,dropna,fillna]
- fill the missing values using scikit-learn
- duplicate data

In [1]:
```
import pandas as pd
import numpy as np
```

In [23]:
```
a=np.array([[1,2,np.nan,3,4],
            [10,12,13,14,15],
            [67,34,29,70,55],
            [np.nan,10,23,np.nan,34],
            [67,np.nan,31,54,np.nan],
            [90,np.nan,np.nan,45,np.nan]]
           )
a
```

Out[23]:
```
array([[ 1.,  2., nan,  3.,  4.],
       [10., 12., 13., 14., 15.],
       [67., 34., 29., 70., 55.],
       [nan, 10., 23., nan, 34.],
       [67., nan, 31., 54., nan],
       [90., nan, nan, 45., nan]])
```

In [24]:
```
df=pd.DataFrame(a,columns=["one","two","three","four","five"])
df
```

Out[24]:

	one	two	three	four	five
0	1.0	2.0	NaN	3.0	4.0
1	10.0	12.0	13.0	14.0	15.0
2	67.0	34.0	29.0	70.0	55.0
3	NaN	10.0	23.0	NaN	34.0
4	67.0	NaN	31.0	54.0	NaN
5	90.0	NaN	NaN	45.0	NaN

In [25]:
```
df.isnull().sum()
```

Out[25]:
```
one      1
two      2
three    2
four     1
five     2
dtype: int64
```

In [26]:
```
df[df.isnull().any(axis=1)]
```

Out[26]:

	one	two	three	four	five
0	1.0	2.0	NaN	3.0	4.0
3	NaN	10.0	23.0	NaN	34.0
4	67.0	NaN	31.0	54.0	NaN
5	90.0	NaN	NaN	45.0	NaN

In [27]:
```
# we can handle nan value using two ways
# dropna
# fillna

df.dropna()
```

Out[27]:

	one	two	three	four	five
1	10.0	12.0	13.0	14.0	15.0
2	67.0	34.0	29.0	70.0	55.0

In [28]:
```
df['one']=df['one'].replace(np.nan,0)
```

In [29]:
```
df
```

Out[29]:

	one	two	three	four	five
0	1.0	2.0	NaN	3.0	4.0
1	10.0	12.0	13.0	14.0	15.0
2	67.0	34.0	29.0	70.0	55.0
3	0.0	10.0	23.0	NaN	34.0
4	67.0	NaN	31.0	54.0	NaN
5	90.0	NaN	NaN	45.0	NaN

In [30]:
```
df['two'].mean()
```

Out[30]:
```
14.5
```

In [31]:
```
df['two']=df['two'].fillna(df['two'].mean())
```

In [32]:
```
df
```

Out[32]:

	one	two	three	four	five
0	1.0	2.0	NaN	3.0	4.0
1	10.0	12.0	13.0	14.0	15.0
2	67.0	34.0	29.0	70.0	55.0
3	0.0	10.0	23.0	NaN	34.0
4	67.0	14.5	31.0	54.0	NaN
5	90.0	14.5	NaN	45.0	NaN

In [33]:
```
df['three']=df['three'].fillna(df['three'].median())
```

In [34]:
```
df
```

Out[34]:

	one	two	three	four	five
0	1.0	2.0	26.0	3.0	4.0
1	10.0	12.0	13.0	14.0	15.0
2	67.0	34.0	29.0	70.0	55.0
3	0.0	10.0	23.0	NaN	34.0
4	67.0	14.5	31.0	54.0	NaN
5	90.0	14.5	26.0	45.0	NaN

In [19]:
```
df.fillna(method='ffill')
```

Out[19]:

	one	two	three	four	five
0	1.0	2.0	26.0	3.0	4.0
1	10.0	12.0	13.0	14.0	15.0
2	67.0	34.0	29.0	70.0	55.0
3	0.0	10.0	23.0	70.0	34.0
4	67.0	14.5	31.0	54.0	34.0
5	90.0	14.5	26.0	45.0	89.0

In [20]:
```
df.fillna(method='bfill')
```

Out[20]:

	one	two	three	four	five
0	1.0	2.0	26.0	3.0	4.0
1	10.0	12.0	13.0	14.0	15.0
2	67.0	34.0	29.0	70.0	55.0
3	0.0	10.0	23.0	54.0	34.0
4	67.0	14.5	31.0	54.0	89.0
5	90.0	14.5	26.0	45.0	89.0

In [21]:
```
df
```

Out[21]:

	one	two	three	four	five
0	1.0	2.0	26.0	3.0	4.0
1	10.0	12.0	13.0	14.0	15.0
2	67.0	34.0	29.0	70.0	55.0
3	0.0	10.0	23.0	NaN	34.0
4	67.0	14.5	31.0	54.0	NaN
5	90.0	14.5	26.0	45.0	89.0

In [22]:
```
df.fillna(method='ffill',limit=1)
```

Out[22]:

	one	two	three	four	five
0	1.0	2.0	26.0	3.0	4.0
1	10.0	12.0	13.0	14.0	15.0
2	67.0	34.0	29.0	70.0	55.0
3	0.0	10.0	23.0	70.0	34.0
4	67.0	14.5	31.0	54.0	34.0
5	90.0	14.5	26.0	45.0	89.0

In [35]:
```
df
```

Out[35]:

	one	two	three	four	five
0	1.0	2.0	26.0	3.0	4.0
1	10.0	12.0	13.0	14.0	15.0
2	67.0	34.0	29.0	70.0	55.0
3	0.0	10.0	23.0	NaN	34.0
4	67.0	14.5	31.0	54.0	NaN
5	90.0	14.5	26.0	45.0	NaN

In [37]:
```
df.fillna(method='ffill',limit=2)
```

Out[37]:

	one	two	three	four	five
0	1.0	2.0	26.0	3.0	4.0
1	10.0	12.0	13.0	14.0	15.0
2	67.0	34.0	29.0	70.0	55.0
3	0.0	10.0	23.0	70.0	34.0
4	67.0	14.5	31.0	54.0	34.0
5	90.0	14.5	26.0	45.0	34.0

How to fill the missing values using sckit-learn

In [38]:
```
a=np.array([[1,2,np.nan,3,4],
            [10,12,13,14,15],
            [67,34,29,70,55],
            [np.nan,10,23,np.nan,34],
            [67,np.nan,31,54,np.nan],
            [90,np.nan,np.nan,45,np.nan]]
           )
a
```

Out[38]:
```
array([[ 1.,  2., nan,  3.,  4.],
       [10., 12., 13., 14., 15.],
       [67., 34., 29., 70., 55.],
       [nan, 10., 23., nan, 34.],
       [67., nan, 31., 54., nan],
       [90., nan, nan, 45., nan]])
```

In [39]:
```
df=pd.DataFrame(a,columns=["one","two","three","four","five"])
df
```

Out[39]:

	one	two	three	four	five
0	1.0	2.0	NaN	3.0	4.0
1	10.0	12.0	13.0	14.0	15.0
2	67.0	34.0	29.0	70.0	55.0
3	NaN	10.0	23.0	NaN	34.0
4	67.0	NaN	31.0	54.0	NaN
5	90.0	NaN	NaN	45.0	NaN

In [42]:
```
from sklearn.impute import SimpleImputer
s=SimpleImputer(strategy='mean')
filldata=s.fit_transform(df)
filldata
```

Out[42]:
```
array([[ 1. ,  2. , 24. ,  3. ,  4. ],
       [10. , 12. , 13. , 14. , 15. ],
       [67. , 34. , 29. , 70. , 55. ],
       [47. , 10. , 23. , 37.2, 34. ],
       [67. , 14.5, 31. , 54. , 27. ],
       [90. , 14.5, 24. , 45. , 27. ]])
```

In [43]:
```
missing_df_mean=pd.DataFrame(filldata,columns=df.columns)
missing_df_mean
```

Out[43]:

	one	two	three	four	five
0	1.0	2.0	24.0	3.0	4.0
1	10.0	12.0	13.0	14.0	15.0
2	67.0	34.0	29.0	70.0	55.0
3	47.0	10.0	23.0	37.2	34.0
4	67.0	14.5	31.0	54.0	27.0
5	90.0	14.5	24.0	45.0	27.0

In [44]:
```
# median
from sklearn.impute import SimpleImputer
s=SimpleImputer(strategy='median')
filldata=s.fit_transform(df)
filldata
```

Out[44]:
```
array([[ 1. ,  2. , 26. ,  3. ,  4. ],
       [10. , 12. , 13. , 14. , 15. ],
       [67. , 34. , 29. , 70. , 55. ],
       [67. , 10. , 23. , 45. , 34. ],
       [67. , 11. , 31. , 54. , 24.5],
       [90. , 11. , 26. , 45. , 24.5]])
```

In [45]:
```
# most-frequent
from sklearn.impute import SimpleImputer
s=SimpleImputer(strategy='most_frequent')
filldata=s.fit_transform(df)
filldata
```

Out[45]:
```
array([[ 1.,  2., 13.,  3.,  4.],
       [10., 12., 13., 14., 15.],
       [67., 34., 29., 70., 55.],
       [67., 10., 23.,  3., 34.],
       [67.,  2., 31., 54.,  4.],
       [90.,  2., 13., 45.,  4.]])
```

In [46]:
```
df_most_fre=pd.DataFrame(filldata,columns=df.columns)
df_most_fre
```

Out[46]:

	one	two	three	four	five
0	1.0	2.0	13.0	3.0	4.0
1	10.0	12.0	13.0	14.0	15.0
2	67.0	34.0	29.0	70.0	55.0
3	67.0	10.0	23.0	3.0	34.0
4	67.0	2.0	31.0	54.0	4.0
5	90.0	2.0	13.0	45.0	4.0

In [47]:
```
# constant
from sklearn.impute import SimpleImputer
s=SimpleImputer(strategy='constant',fill_value=0)
filldata=s.fit_transform(df)
filldata
```

Out[47]:
```
array([[ 1.,  2.,  0.,  3.,  4.],
       [10., 12., 13., 14., 15.],
       [67., 34., 29., 70., 55.],
       [ 0., 10., 23.,  0., 34.],
       [67.,  0., 31., 54.,  0.],
       [90.,  0.,  0., 45.,  0.]])
```

drop duplicates

In [52]:
```
df=pd.DataFrame({'sno':[110,101,101,102,102,103,104,105,110],
                 'sname':["e",'a','a','b','b','c','d','f','e']})
df
```

Out[52]:

	sno	sname
0	110	e
1	101	a
2	101	a
3	102	b
4	102	b
5	103	c
6	104	d
7	105	f
8	110	e

In [53]:
```
df.duplicated()
```

Out[53]:
```
0    False
1    False
2     True
3    False
4     True
5    False
6    False
7    False
8     True
dtype: bool
```

In [54]:
```
df[df.duplicated()]
```

Out[54]:

	sno	sname
2	101	a
4	102	b
8	110	e

In [55]:
```
#df.duplicated(subset,keep)
df.duplicated('sno')
```

Out[55]:
```
0    False
1    False
2     True
3    False
4     True
5    False
6    False
7    False
8     True
dtype: bool
```

In [59]:
```
df.duplicated(keep='first')
```

Out[59]:
```
0    False
1    False
2     True
3    False
4     True
5    False
6    False
7    False
8     True
dtype: bool
```

In [60]:
```
df.duplicated(keep='last')
```

Out[60]:
```
0     True
1     True
2    False
3     True
4    False
5    False
6    False
7    False
8    False
dtype: bool
```

In [61]:
```
df.duplicated(keep=False)
```

Out[61]:
```
0     True
1     True
2     True
3     True
4     True
5    False
6    False
7    False
8     True
dtype: bool
```

In [62]:
```
df[df.duplicated(keep=False)]
```

Out[62]:

	sno	sname
0	110	e
1	101	a
2	101	a
3	102	b
4	102	b
8	110	e

In [66]:
```
df.drop_duplicates(inplace=True)
```

In [67]:
```
df
```

Out[67]:

	sno	sname
0	110	e
1	101	a
3	102	b
5	103	c
6	104	d
7	105	f

Task:cars.csv

- replace the ? with nan values
- find the sum of nan values
- fill the nan values using mean or median
- In make feature(column) we have data with (-), ex: alfa-romero. so, keep the before character of (-) in make feature i.e alfa

In [68]:
```
import pandas as pd
import numpy as np
```

In [70]:
```
df=pd.read_csv("cars.csv")
df.head()
```

Out[70]:

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	.
0	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	.
1	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	.
2	1	?	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	.
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	.
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	.

5 rows × 26 columns

In [71]:
```
df.isnull().sum()
```

...

In [74]:
```
df.isnull().any(axis=1).sum()
```

Out[74]:
```
0
```

In [75]:
```
df=df.replace('?',np.nan)
```

In [77]:
```
df.columns
```

Out[77]:
```
Index(['symboling', 'normalized-losses', 'make', 'fuel-type', 'aspiration',
       'num-of-doors', 'body-style', 'drive-wheels', 'engine-location',
       'wheel-base', 'length', 'width', 'height', 'curb-weight', 'engine-type',
       'num-of-cylinders', 'engine-size', 'fuel-system', 'bore', 'stroke',
       'compression-ratio', 'horsepower', 'peak-rpm', 'city-mpg',
       'highway-mpg', 'price'],
      dtype='object')
```

In [78]:
```
df.isnull().sum()
```

Out[78]:
```
symboling             0
normalized-losses    41
make                  0
fuel-type             0
aspiration            0
num-of-doors          2
body-style            0
drive-wheels          0
engine-location       0
wheel-base            0
length                0
width                 0
height                0
curb-weight           0
engine-type           0
num-of-cylinders      0
engine-size           0
fuel-system           0
bore                  4
stroke                4
compression-ratio     0
horsepower            2
peak-rpm              2
city-mpg              0
highway-mpg           0
price                 4
dtype: int64
```

In [79]:
```
#df['normalized-losses']=df['normalized-losses'].fillna(df['normalized-losses'].
```

...

In [81]:
```
df.dtypes
```

Out[81]:
```
symboling              int64
normalized-losses     object
make                  object
fuel-type             object
aspiration            object
num-of-doors          object
body-style            object
drive-wheels          object
engine-location       object
wheel-base           float64
length               float64
width                float64
height               float64
curb-weight            int64
engine-type           object
num-of-cylinders      object
engine-size            int64
fuel-system           object
bore                  object
stroke                object
compression-ratio    float64
horsepower            object
peak-rpm              object
city-mpg               int64
highway-mpg            int64
price                 object
dtype: object
```

In [82]:
```
df["normalized-losses"]=df.replace("mean")
```

In [83]:
```
df.head()
```

Out[83]:

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	.
0	3	3	alfa-romero	gas	std	two	convertible	rwd	front	88.6	.
1	3	3	alfa-romero	gas	std	two	convertible	rwd	front	88.6	.
2	1	1	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	.
3	2	2	audi	gas	std	four	sedan	fwd	front	99.8	.
4	2	2	audi	gas	std	four	sedan	4wd	front	99.4	.

5 rows × 26 columns

In [84]:
```
df.isnull().sum()
```

...

In [87]:
```
df['price']=pd.to_numeric(df['price'])
```

In [89]:
```
df['price']=df['price'].fillna(df['price'].mean())
```

In [90]:
```
df.isnull().sum()
```

Out[90]:
```
symboling           0
normalized-losses   0
make                0
fuel-type           0
aspiration          0
num-of-doors        2
body-style          0
drive-wheels        0
engine-location     0
wheel-base          0
length              0
width               0
height              0
curb-weight         0
engine-type         0
num-of-cylinders    0
engine-size         0
fuel-system         0
bore                4
stroke              4
compression-ratio   0
horsepower          2
peak-rpm            2
city-mpg            0
highway-mpg         0
price               0
dtype: int64
```

In [91]:
```
df.head()
```

Out[91]:

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	.
0	3	3	alfa-romero	gas	std	two	convertible	rwd	front	88.6	.
1	3	3	alfa-romero	gas	std	two	convertible	rwd	front	88.6	.
2	1	1	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	.
3	2	2	audi	gas	std	four	sedan	fwd	front	99.8	.
4	2	2	audi	gas	std	four	sedan	4wd	front	99.4	.

5 rows × 26 columns

In [92]:
```
df['make']
```

...

In [94]:
```
df.sample(10)
```

Out[94]:

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	whe ba
93	1	1	nissan	gas	std	four	wagon	fwd	front	9
176	-1	-1	toyota	gas	std	four	sedan	fwd	front	10
193	0	0	volkswagen	gas	std	four	wagon	fwd	front	10
73	0	0	mercedes-benz	gas	std	four	sedan	rwd	front	12
149	0	0	subaru	gas	turbo	four	wagon	4wd	front	9
188	2	2	volkswagen	gas	std	four	sedan	fwd	front	9
140	2	2	subaru	gas	std	two	hatchback	4wd	front	9
162	0	0	toyota	gas	std	four	sedan	fwd	front	9
132	3	3	saab	gas	std	two	hatchback	fwd	front	9
101	0	0	nissan	gas	std	four	sedan	fwd	front	10

10 rows × 26 columns

In [ ]:

In [ ]: