In [1]:

```
#Practical No 1
#ajit waman B54
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
#so that we can view the graphs inside the notebook
```

In [2]:

```
#from google.colab import drive
#drive.mount('/content/gdrive')
```

In [3]:

```
s1 = pd.Series(range(1,10,1))
```

In [4]:

```
s1
```

Out[4]:

```
0    1
1    2
2    3
3    4
4    5
5    6
6    7
7    8
8    9
dtype: int64
```

In [5]:

```
s3 = pd.Series({1:21, 2:13,3:45})
```

In [6]:

```
s3
```

Out[6]:

```
1    21
2    13
3    45
dtype: int64
```

In [7]:

```
s2 = pd.Series([1, 2, 3, 4], index=['p', 'q', 'r','s'], name='one')
```

In [8]:

```
s2
```

Out[8]:

```
p    1
q    2
r    3
s    4
Name: one, dtype: int64
```

In [9]:
```
df1 = pd.DataFrame(s2)
df1
```

Out[9]:

	one
p	1
q	2
r	3
s	4

In [11]:
```
df2 = pd.read_csv("/home/kj-comp/california_housing_test.csv")
#dataframe_name = pd.read_<format>(filename)
```

In [12]:
```
df2.head(10)
```

Out[12]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households
0	-122.05	37.37	27.0	3885.0	661.0	1537.0	606.0
1	-118.30	34.26	43.0	1510.0	310.0	809.0	277.0
2	-117.81	33.78	27.0	3589.0	507.0	1484.0	495.0
3	-118.36	33.82	28.0	67.0	15.0	49.0	11.0
4	-119.67	36.33	19.0	1241.0	244.0	850.0	237.0
5	-119.56	36.51	37.0	1018.0	213.0	663.0	204.0
6	-121.43	38.63	43.0	1009.0	225.0	604.0	218.0
7	-120.65	35.48	19.0	2310.0	471.0	1341.0	441.0
8	-122.84	38.40	15.0	3080.0	617.0	1446.0	599.0
9	-118.02	34.08	31.0	2402.0	632.0	2830.0	603.0

In [13]:
```
df2.tail(3)
```

Out[13]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	househo
2997	-119.70	36.30	10.0	956.0	201.0	693.0	22
2998	-117.12	34.10	40.0	96.0	14.0	46.0	1
2999	-119.63	34.42	42.0	1765.0	263.0	753.0	26

In [14]:
```
df2['median_house_value_new']=df2['median_house_value']+111
```

In [15]:
```
df2.tail(3)
```

Out[15]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	househo
2997	-119.70	36.30	10.0	956.0	201.0	693.0	22
2998	-117.12	34.10	40.0	96.0	14.0	46.0	1
2999	-119.63	34.42	42.0	1765.0	263.0	753.0	26

In [16]:
```
# write
# <dataframe's name>.to_<file_format>(<file_name>)
```

In [17]:
```
df2.to_json('data1.json')
```

In [18]:
```
len(df2['total_rooms'])
```

Out[18]:
```
3000
```

In [19]:
```
df2['total_rooms'].count()
```

Out[19]:
```
3000
```

In [20]:
```
df2['total_rooms'].mean()
```

Out[20]:
```
2599.578666666667
```

In [21]:
```
df2['total_rooms'].sum()
```

Out[21]:
```
7798736.0
```

In [22]:
```
df2['total_rooms'].median()
```

Out[22]:
```
2106.0
```

In [23]:
```
df2['total_rooms'].std()
```

Out[23]:
```
2155.59333162558
```

In [24]:
```
df2['total_rooms'].min()
```

Out[24]:
```
6.0
```

In [25]:
```
df2['total_rooms'].max()
```

Out[25]:
```
30450.0
```

In [26]:

```
df2['total_rooms'].describe()
```

Out[26]:

```
count     3000.000000
mean      2599.578667
std       2155.593332
min          6.000000
25%       1401.000000
50%       2106.000000
75%       3129.000000
max      30450.000000
Name: total_rooms, dtype: float64
```

In [27]:

```
df2['total_rooms'].cumsum()
```

Out[27]:

```
0          3885.0
1          5395.0
2          8984.0
3          9051.0
4         10292.0
          ...
2995    7790662.0
2996    7795919.0
2997    7796875.0
2998    7796971.0
2999    7798736.0
Name: total_rooms, Length: 3000, dtype: float64
```

In [28]:

```
# When you give the whole dataframe, then all numerical columns will be analys
is
df2.mean()
```

Out[28]:

```
longitude                  -119.589200
latitude                     35.635390
housing_median_age           28.845333
total_rooms                2599.578667
total_bedrooms              529.950667
population                 1402.798667
households                  489.912000
median_income                 3.807272
median_house_value       205846.275000
median_house_value_new   205957.275000
dtype: float64
```

In [29]:
```
df2.describe()
```

Out[29]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population
count	3000.000000	3000.00000	3000.000000	3000.000000	3000.000000	3000.00000
mean	-119.589200	35.63539	28.845333	2599.578667	529.950667	1402.79866
std	1.994936	2.12967	12.555396	2155.593332	415.654368	1030.54301
min	-124.180000	32.56000	1.000000	6.000000	2.000000	5.00000
25%	-121.810000	33.93000	18.000000	1401.000000	291.000000	780.00000
50%	-118.485000	34.27000	29.000000	2106.000000	437.000000	1155.00000
75%	-118.020000	37.69000	37.000000	3129.000000	636.000000	1742.75000
max	-114.490000	41.92000	52.000000	30450.000000	5419.000000	11935.00000

In [31]:
```
df = pd.read_csv("/home/kj-comp/california_housing_test.csv")
```

In [32]:
```
df.describe()
```

Out[32]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population
count	3000.000000	3000.00000	3000.000000	3000.000000	3000.000000	3000.00000
mean	-119.589200	35.63539	28.845333	2599.578667	529.950667	1402.79866
std	1.994936	2.12967	12.555396	2155.593332	415.654368	1030.54301
min	-124.180000	32.56000	1.000000	6.000000	2.000000	5.00000
25%	-121.810000	33.93000	18.000000	1401.000000	291.000000	780.00000
50%	-118.485000	34.27000	29.000000	2106.000000	437.000000	1155.00000
75%	-118.020000	37.69000	37.000000	3129.000000	636.000000	1742.75000
max	-114.490000	41.92000	52.000000	30450.000000	5419.000000	11935.00000

In [33]:
```
df.columns
```

Out[33]:
```
Index(['longitude', 'latitude', 'housing_median_age', 'total_rooms',
       'total_bedrooms', 'population', 'households', 'median_income',
       'median_house_value'],
      dtype='object')
```

In [34]:
```
df['longitude']
```

Out[34]:
```
0      -122.05
1      -118.30
2      -117.81
3      -118.36
4      -119.67
         ...  
2995   -119.86
2996   -118.14
2997   -119.70
2998   -117.12
2999   -119.63
Name: longitude, Length: 3000, dtype: float64
```

In [35]:
```
df.longitude
```

Out[35]:
```
0      -122.05
1      -118.30
2      -117.81
3      -118.36
4      -119.67
         ...  
2995   -119.86
2996   -118.14
2997   -119.70
2998   -117.12
2999   -119.63
Name: longitude, Length: 3000, dtype: float64
```

In [36]:
```
df.iloc[:,1:3]
```

Out[36]:

	latitude	housing_median_age
0	37.37	27.0
1	34.26	43.0
2	33.78	27.0
3	33.82	28.0
4	36.33	19.0
...	...	...
2995	34.42	23.0
2996	34.06	27.0
2997	36.30	10.0
2998	34.10	40.0
2999	34.42	42.0

3000 rows × 2 columns

In [38]:

```
# importing pandas as pd
import pandas as pd
# making data frame from csv file
data = pd.read_csv("employees.csv")
data.head(10)
```

Out[38]:

	First Name	Gender	Start Date	Last Login Time	Salary	Bonus %	Senior Management	Team
0	Douglas	Male	8/6/1993	12:42 PM	97308.0	6.945	True	Marketing
1	Thomas	Male	3/31/1996	6:53 AM	61933.0	4.170	True	NaN
2	Maria	Female	4/23/1993	11:17 AM	130590.0	11.858	False	Finance
3	Jerry	Male	3/4/2005	1:00 PM	138705.0	9.340	True	Finance
4	Larry	Male	1/24/1998	4:47 PM	101004.0	1.389	True	Client Services
5	Dennis	Male	4/18/1987	1:35 AM	115163.0	10.125	False	Legal
6	Ruby	Female	8/17/1987	4:20 PM	65476.0	10.012	True	Product
7	NaN	Female	7/20/2015	10:43 AM	45906.0	11.598	NaN	Finance
8	Angela	Female	11/22/2005	6:29 AM	95570.0	18.523	True	Engineering
9	Frances	Female	8/8/2002	6:51 AM	139852.0	7.524	True	Business Development

In [39]:

```
data.describe()
```

Out[39]:

	Salary	Bonus %
count	969.000000	1000.000000
mean	90579.972136	10.207555
std	32916.214577	5.528481
min	35013.000000	1.015000
25%	62666.000000	5.401750
50%	90370.000000	9.838500
75%	118733.000000	14.838000
max	149908.000000	19.944000

In [40]:
```
data.isnull()
```

Out[40]:

	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 [41]:
```
data.notnull()
```

Out[41]:

	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 [42]:

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

Out[42]:

```
First Name            67
Gender               145
Start Date             0
Last Login Time        0
Salary                31
Bonus %                0
Senior Management     67
Team                  43
dtype: int64
```

In [43]:

```
# filling a null values using fillna()
data["Gender"].fillna("No Gender", inplace = True)
```

In [44]:

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

Out[44]:

```
First Name           67
Gender                0
Start Date            0
Last Login Time       0
Salary               31
Bonus %               0
Senior Management    67
Team                 43
dtype: int64
```

In [45]:

```
# will replace Nan value in dataframe with value -99
import numpy as np
data.replace(to_replace = np.nan, value = -99)
```

Out[45]:

	First Name	Gender	Start Date	Last Login Time	Salary	Bonus %	Senior Management	Team
0	Douglas	Male	8/6/1993	12:42 PM	97308.0	6.945	True	Marketing
1	Thomas	Male	3/31/1996	6:53 AM	61933.0	4.170	True	-99
2	Maria	Female	4/23/1993	11:17 AM	130590.0	11.858	False	Finance
3	Jerry	Male	3/4/2005	1:00 PM	138705.0	9.340	True	Finance
4	Larry	Male	1/24/1998	4:47 PM	101004.0	1.389	True	Client Services
...	...	...	...	...	...	...	...	...
995	Henry	No Gender	11/23/2014	6:09 AM	132483.0	16.655	False	Distribution
996	Phillip	Male	1/31/1984	6:30 AM	42392.0	19.675	False	Finance
997	Russell	Male	5/20/2013	12:39 PM	96914.0	1.421	False	Product
998	Larry	Male	4/20/2013	4:45 PM	60500.0	11.985	False	Business Development
999	Albert	Male	5/15/2012	6:24 PM	129949.0	10.169	True	Sales

1000 rows × 8 columns

In [46]:
```
# filling a missing value with previous ones
data.fillna(method ='pad')
```

Out[46]:

	First Name	Gender	Start Date	Last Login Time	Salary	Bonus %	Senior Management	Team
0	Douglas	Male	8/6/1993	12:42 PM	97308.0	6.945	True	Marketing
1	Thomas	Male	3/31/1996	6:53 AM	61933.0	4.170	True	Marketing
2	Maria	Female	4/23/1993	11:17 AM	130590.0	11.858	False	Finance
3	Jerry	Male	3/4/2005	1:00 PM	138705.0	9.340	True	Finance
4	Larry	Male	1/24/1998	4:47 PM	101004.0	1.389	True	Client Services
...	...	...	...	...	...	...	...	...
995	Henry	No Gender	11/23/2014	6:09 AM	132483.0	16.655	False	Distribution
996	Phillip	Male	1/31/1984	6:30 AM	42392.0	19.675	False	Finance
997	Russell	Male	5/20/2013	12:39 PM	96914.0	1.421	False	Product
998	Larry	Male	4/20/2013	4:45 PM	60500.0	11.985	False	Business Development
999	Albert	Male	5/15/2012	6:24 PM	129949.0	10.169	True	Sales

1000 rows × 8 columns

In [47]:

```
# filling a missing value with previous ones
data.fillna(method ='pad')
```

Out[47]:

	First Name	Gender	Start Date	Last Login Time	Salary	Bonus %	Senior Management	Team
0	Douglas	Male	8/6/1993	12:42 PM	97308.0	6.945	True	Marketing
1	Thomas	Male	3/31/1996	6:53 AM	61933.0	4.170	True	Marketing
2	Maria	Female	4/23/1993	11:17 AM	130590.0	11.858	False	Finance
3	Jerry	Male	3/4/2005	1:00 PM	138705.0	9.340	True	Finance
4	Larry	Male	1/24/1998	4:47 PM	101004.0	1.389	True	Client Services
...	...	...	...	...	...	...	...	...
995	Henry	No Gender	11/23/2014	6:09 AM	132483.0	16.655	False	Distribution
996	Phillip	Male	1/31/1984	6:30 AM	42392.0	19.675	False	Finance
997	Russell	Male	5/20/2013	12:39 PM	96914.0	1.421	False	Product
998	Larry	Male	4/20/2013	4:45 PM	60500.0	11.985	False	Business Development
999	Albert	Male	5/15/2012	6:24 PM	129949.0	10.169	True	Sales

1000 rows × 8 columns

In [48]:

```
data.dropna(axis=1)
```

Out[48]:

	Gender	Start Date	Last Login Time	Bonus %
0	Male	8/6/1993	12:42 PM	6.945
1	Male	3/31/1996	6:53 AM	4.170
2	Female	4/23/1993	11:17 AM	11.858
3	Male	3/4/2005	1:00 PM	9.340
4	Male	1/24/1998	4:47 PM	1.389
...	...	...	...	...
995	No Gender	11/23/2014	6:09 AM	16.655
996	Male	1/31/1984	6:30 AM	19.675
997	Male	5/20/2013	12:39 PM	1.421
998	Male	4/20/2013	4:45 PM	11.985
999	Male	5/15/2012	6:24 PM	10.169

1000 rows × 4 columns

In [49]:

```
# importing pandas as pd
import pandas as pd
# Creating the dataframe
df = pd.DataFrame({"A":[12, 4, 5, None, 1],
"B":[None, 2, 54, 3, None],
"C":[20, 16, None, 3, 8],
"D":[14, 3, None, None, 6]})
# Print the dataframe
df
```

Out[49]:

	A	B	C	D
0	12.0	NaN	20.0	14.0
1	4.0	2.0	16.0	3.0
2	5.0	54.0	NaN	NaN
3	NaN	3.0	3.0	NaN
4	1.0	NaN	8.0	6.0

In [50]:

```
df.interpolate(method ='linear', limit_direction ='forward')
```

Out[50]:

	A	B	C	D
0	12.0	NaN	20.0	14.0
1	4.0	2.0	16.0	3.0
2	5.0	54.0	9.5	4.0
3	3.0	3.0	3.0	5.0
4	1.0	3.0	8.0	6.0

In [51]:

```
#remove white space everywhere
text="today is Monday"
#df['Col Name'] = df['Col Name'].str.replace(‘ ‘, ‘’)
text.replace(' ','')
```

Out[51]:

```
'todayisMonday'
```

In [52]:

```
text=' Today'
text.lstrip()
```

Out[52]:

```
'Today'
```

In [53]:

```
text='Today '
text.rstrip()
```

Out[53]:

```
'Today'
```

In [54]:

```
text=' Today '
text.strip()
```

Out[54]:

```
'Today'
```

In [55]:

```
import pandas
import scipy
import numpy
from sklearn.preprocessing import MinMaxScaler
# data values
X = [ [110, 200], [120, 800], [310, 400], [140, 900], [510, 200], [653, 400] ,
[310, 880
] ]
# transofrm data
scaler = MinMaxScaler(feature_range=(0,5))
rescaledX = scaler.fit_transform(X)
```

In [56]:

```
X
```

Out[56]:

```
[[110, 200],
 [120, 800],
 [310, 400],
 [140, 900],
 [510, 200],
 [653, 400],
 [310, 880]]
```

In [57]:

```
rescaledX
```

Out[57]:

```
array([[0.        , 0.        ],
       [0.09208103, 4.28571429],
       [1.84162063, 1.42857143],
       [0.27624309, 5.        ],
       [3.68324125, 0.        ],
       [5.        , 1.42857143],
       [1.84162063, 4.85714286]])
```

In [58]:

```
from sklearn.preprocessing import StandardScaler
import pandas
import numpy
# data values
X = [ [110, 200], [120, 800], [310, 400], [140, 900], [510, 200], [653, 400] ,
[310, 880
] ]
# scaler
scaler = StandardScaler().fit(X)
rescaledX = scaler.transform(X)
```

In [59]:

```
rescaledX
```

Out[59]:

```
array([[-1.02004521, -1.17792918],
       [-0.96841602,  0.90076937],
       [ 0.01253852, -0.48502966],
       [-0.86515765,  1.24721913],
       [ 1.04512224, -1.17792918],
       [ 1.78341961, -0.48502966],
       [ 0.01253852,  1.17792918]])
```

In [60]:

```
from sklearn.preprocessing import Normalizer
import pandas
import numpy
# data values
X = [ [110, 200], [120, 800], [310, 400], [140, 900], [510, 200], [653, 400] ,
[310, 880
] ]
# normalize values
scaler = Normalizer().fit(X)
normalizedX = scaler.transform(X)
```

In [61]:

```
normalizedX
```

Out[61]:

```
array([[0.48191875, 0.87621591],
       [0.14834045, 0.98893635],
       [0.61257167, 0.79041505],
       [0.15370701, 0.98811647],
       [0.9309732 , 0.36508753],
       [0.8527326 , 0.52234769],
       [0.33225942, 0.94318804]])
```

In [62]:

```
from sklearn.preprocessing import Binarizer
import pandas
import numpy
# data values
X = [ [501, 200], [120, 800], [310, 400], [140, 900], [510, 200], [653, 400] ,
[310, 880
] ]
# binarize data
binarizer = Binarizer(threshold=500).fit(X)
binaryX = binarizer.transform(X)
```

In [63]:

```
binaryX
```

Out[63]:

```
array([[1, 0],
       [0, 1],
       [0, 0],
       [0, 1],
       [1, 0],
       [1, 0],
       [0, 1]])
```

In [72]:

```
import pandas as pd
import numpy as np
# Read in the CSV file and convert "?" to NaN
headers = ["symboling", "normalized_losses", "make", "fuel_type", "aspiration",
"num_doors", "body_style", "drive_wheels", "engine_location",
"wheel_base", "length", "width", "height", "curb_weight",
"engine_type", "num_cylinders", "engine_size", "fuel_system",
"bore", "stroke", "compression_ratio", "horsepower", "peak_rpm",
"city_mpg", "highway_mpg", "price"]
df = pd.read_csv("https://archive.ics.uci.edu/ml/machine-learning-databases/autos/imports-85.data",header=None, names=headers, na_values="?" )
df.head()
```

Out[72]:

	symboling	normalized_losses	make	fuel_type	aspiration	num_doors	body_style	drive_wh
0	3	NaN	alfa-romero	gas	std	two	convertible	
1	3	NaN	alfa-romero	gas	std	two	convertible	
2	1	NaN	alfa-romero	gas	std	two	hatchback	
3	2	164.0	audi	gas	std	four	sedan	
4	2	164.0	audi	gas	std	four	sedan	

5 rows × 26 columns

In [73]:
```
df.dtypes
```

Out[73]:
```
symboling              int64
normalized_losses    float64
make                  object
fuel_type             object
aspiration            object
num_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_cylinders         object
engine_size            int64
fuel_system           object
bore                 float64
stroke               float64
compression_ratio    float64
horsepower           float64
peak_rpm             float64
city_mpg               int64
highway_mpg            int64
price                float64
dtype: object
```

In [74]:
```
obj_df = df.select_dtypes(include=['object']).copy()
obj_df.head()
```

Out[74]:

	make	fuel_type	aspiration	num_doors	body_style	drive_wheels	engine_location	engine_ty
0	alfa-romero	gas	std	two	convertible	rwd	front	dc
1	alfa-romero	gas	std	two	convertible	rwd	front	dc
2	alfa-romero	gas	std	two	hatchback	rwd	front	ol
3	audi	gas	std	four	sedan	fwd	front	c
4	audi	gas	std	four	sedan	4wd	front	c

In [75]:

```
obj_df[obj_df.isnull().any(axis=1)]
```

Out[75]:

	make	fuel_type	aspiration	num_doors	body_style	drive_wheels	engine_location	engine_t
27	dodge	gas	turbo	NaN	sedan	fwd	front	
63	mazda	diesel	std	NaN	sedan	fwd	front	

In [76]:

```
obj_df["num_doors"].value_counts()
```

Out[76]:

```
four     114
two       89
Name: num_doors, dtype: int64
```

In [77]:

```
obj_df = obj_df.fillna({"num_doors": "four"})
```

In [78]:

```
obj_df[obj_df.isnull().any(axis=1)]
```

Out[78]:

make	fuel_type	aspiration	num_doors	body_style	drive_wheels	engine_location	engine_type

In [79]:

```
obj_df["num_cylinders"].value_counts()
```

Out[79]:

```
four      159
six        24
five       11
eight       5
two         4
twelve      1
three       1
Name: num_cylinders, dtype: int64
```

In [80]:

```
cleanup_nums = {"num_doors": {"four": 4, "two": 2},
"num_cylinders": {"four": 4, "six": 6, "five": 5, "eight": 8,
"two": 2, "twelve": 12, "three":3 }}
```

In [81]:
```
obj_df = obj_df.replace(cleanup_nums)
obj_df.head()
```

Out[81]:

	make	fuel_type	aspiration	num_doors	body_style	drive_wheels	engine_location	engine_ty
0	alfa-romero	gas	std	2	convertible	rwd	front	dc
1	alfa-romero	gas	std	2	convertible	rwd	front	dc
2	alfa-romero	gas	std	2	hatchback	rwd	front	ol
3	audi	gas	std	4	sedan	fwd	front	c
4	audi	gas	std	4	sedan	4wd	front	c

In [82]:
```
obj_df.dtypes
```

Out[82]:
```
make                object
fuel_type           object
aspiration          object
num_doors            int64
body_style          object
drive_wheels        object
engine_location     object
engine_type         object
num_cylinders        int64
fuel_system         object
dtype: object
```

In [83]:
```
obj_df["body_style"].value_counts()
```

Out[83]:
```
sedan          96
hatchback      70
wagon          25
hardtop         8
convertible     6
Name: body_style, dtype: int64
```

In [84]:
```
obj_df["body_style"] = obj_df["body_style"].astype('category')
obj_df.dtypes
```

Out[84]:
```
make                  object
fuel_type             object
aspiration            object
num_doors              int64
body_style          category
drive_wheels          object
engine_location       object
engine_type           object
num_cylinders          int64
fuel_system           object
dtype: object
```

In [85]:
```
obj_df["body_style_cat"] = obj_df["body_style"].cat.codes
obj_df.head()
```

Out[85]:

	make	fuel_type	aspiration	num_doors	body_style	drive_wheels	engine_location	engine_ty
0	alfa-romero	gas	std	2	convertible	rwd	front	dc
1	alfa-romero	gas	std	2	convertible	rwd	front	dc
2	alfa-romero	gas	std	2	hatchback	rwd	front	ol
3	audi	gas	std	4	sedan	fwd	front	c
4	audi	gas	std	4	sedan	4wd	front	c

In [86]:
```
pd.get_dummies(obj_df, columns=["drive_wheels"]).head()
```

Out[86]:

	make	fuel_type	aspiration	num_doors	body_style	engine_location	engine_type	num_cylin
0	alfa-romero	gas	std	2	convertible	front	dohc	
1	alfa-romero	gas	std	2	convertible	front	dohc	
2	alfa-romero	gas	std	2	hatchback	front	ohcv	
3	audi	gas	std	4	sedan	front	ohc	
4	audi	gas	std	4	sedan	front	ohc	

In [87]:

```
from sklearn.preprocessing import OrdinalEncoder
ord_enc = OrdinalEncoder()
obj_df["make_code"] = ord_enc.fit_transform(obj_df[["make"]])
obj_df[["make", "make_code"]].head(11)
```

Out[87]:

	make	make_code
0	alfa-romero	0.0
1	alfa-romero	0.0
2	alfa-romero	0.0
3	audi	1.0
4	audi	1.0
5	audi	1.0
6	audi	1.0
7	audi	1.0
8	audi	1.0
9	audi	1.0
10	bmw	2.0

In [88]:

```
from sklearn.preprocessing import OneHotEncoder
oe_style = OneHotEncoder()
oe_results = oe_style.fit_transform(obj_df[["body_style"]])
pd.DataFrame(oe_results.toarray(), columns=oe_style.categories_).head()
```

Out[88]:

	convertible	hardtop	hatchback	sedan	wagon
0	1.0	0.0	0.0	0.0	0.0
1	1.0	0.0	0.0	0.0	0.0
2	0.0	0.0	1.0	0.0	0.0
3	0.0	0.0	0.0	1.0	0.0
4	0.0	0.0	0.0	1.0	0.0

In [89]:

```
obj_df = obj_df.join(pd.DataFrame(oe_results.toarray(), columns=oe_style.categ
ories_))
```

In [90]:

```
obj_df.head()
```

Out[90]:

	make	fuel_type	aspiration	num_doors	body_style	drive_wheels	engine_location	engine_ty
0	alfa-romero	gas	std	2	convertible	rwd	front	dc
1	alfa-romero	gas	std	2	convertible	rwd	front	dc
2	alfa-romero	gas	std	2	hatchback	rwd	front	ol
3	audi	gas	std	4	sedan	fwd	front	o
4	audi	gas	std	4	sedan	4wd	front	o

In [ ]: