Lab Sheet 1

Exercise 1

1. Read the csv file from the given URL to a dataframe. https://archive.ics.uci.edu/ml/machine-learning-databases/autos/imports-85.data.

```
from pandas import read_csv
path='imports-85.data'
data=read_csv(path, )
data.head()
```

	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.60	...	130	mpfi	3.47	2
0	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	
1	1	?	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68	
2	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	109	mpfi	3.19	
3	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	136	mpfi	3.19	

2. Include the following headers to the above csv file ["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"]

```
import pandas as pd
head = ["symboling","normalized-losses","make","fuel-type","aspiration", "num-of-doors","body-style","drive-wheels","engine-location","wheel
df = pd.read_csv(path, names = head)
df.head()
```

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

5 rows × 26 columns

3. Display the first five rows of the dataset.

```
df.head(5)
```

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

5 rows × 26 columns

4. Replace the “?” in the above file with NaN.

```
import numpy as np
df.replace("?", np.nan, inplace = True)
```

5. Find the missing values in the dataset.

```
df.isnull()
```

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wh
0	False	True	False	False	False	False	False	False	False	I
1	False	True	False	False	False	False	False	False	False	I
2	False	True	False	False	False	False	False	False	False	I
3	False	False	False	False	False	False	False	False	False	I
4	False	False	False	False	False	False	False	False	False	I
...	...	...	...	...	...	...	...	...	...	
200	False	False	False	False	False	False	False	False	False	I
201	False	False	False	False	False	False	False	False	False	I
202	False	False	False	False	False	False	False	False	False	I
203	False	False	False	False	False	False	False	False	False	I
204	False	False	False	False	False	False	False	False	False	I

205 rows × 26 columns

6. Count the missing values in each column.

```
df.isnull().sum()
```

```
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
```

7. Identify the column(s) of a given Data Frame which have at least one missing value.

```
df.columns[df.isnull().any()]
```

```
Index(['normalized-losses', 'num-of-doors', 'bore', 'stroke', 'horsepower',
       'peak-rpm', 'price'],
      dtype='object')
```

8. Find the Indexes of missing values of column “normalized-losses” in the given DataFrame.

```
df[df["normalized-losses"].isnull()].index
```

```
Index([  0,   1,   2,   5,   7,   9,  14,  15,  16,  17,  43,  44,  45,  46,
        48,  49,  63,  66,  71,  73,  74,  75,  82,  83,  84, 109, 110, 113,
       114, 124, 126, 127, 128, 129, 130, 131, 181, 189, 191, 192, 193],
      dtype='int64')
```

9. Replace the missing values in “normalized-losses” ,"stroke", “bore",with the mean.

```
df["normalized-losses"] = pd.to_numeric(df["normalized-losses"], errors='coerce')
df["stroke"] = pd.to_numeric(df["stroke"], errors='coerce')
df["bore"] = pd.to_numeric(df["bore"], errors='coerce')

mean_normalized_losses = df["normalized-losses"].mean()
mean_stroke = df["stroke"].mean()
mean_bore = df["bore"].mean()

df["normalized-losses"].fillna(mean_normalized_losses, inplace=True)
df["stroke"].fillna(mean_stroke, inplace=True)
df["bore"].fillna(mean_bore, inplace=True)

print(df.head())
```

```
   symboling  normalized-losses         make fuel-type aspiration  \
0          3              122.0  alfa-romero       gas        std
1          3              122.0  alfa-romero       gas        std
2          1              122.0  alfa-romero       gas        std
3          2              164.0         audi       gas        std
4          2              164.0         audi       gas        std

  num-of-doors   body-style drive-wheels engine-location  wheel-base  ...  \
0          two  convertible          rwd           front        88.6  ...
1          two  convertible          rwd           front        88.6  ...
2          two    hatchback          rwd           front        94.5  ...
3         four        sedan          fwd           front        99.8  ...
4         four        sedan          4wd           front        99.4  ...

   engine-size  fuel-system  bore  stroke compression-ratio horsepower  \
0          130         mpfi  3.47    2.68               9.0        111
1          130         mpfi  3.47    2.68               9.0        111
2          152         mpfi  2.68    3.47               9.0        154
3          109         mpfi  3.19    3.40              10.0        102
4          136         mpfi  3.19    3.40               8.0        115

   peak-rpm city-mpg  highway-mpg  price
0      5000       21           27  13495
1      5000       21           27  16500
2      5000       19           26  16500
3      5500       24           30  13950
4      5500       18           22  17450
```

```
[5 rows x 26 columns]
```

11.Replace the missing values in “num_doors" with the maximum frequency value

```
max_freq = df['num-of-doors'].mode()[0]
df['num-of-doors'].fillna(max_freq, inplace=True)
df.head()
```

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location
0	3	122.0	alfa-romero	gas	std	two	convertible	rwd	front
1	3	122.0	alfa-romero	gas	std	two	convertible	rwd	front
2	1	122.0	alfa-romero	gas	std	two	hatchback	rwd	front
3	2	164.0	audi	gas	std	four	sedan	fwd	front
4	2	164.0	audi	gas	std	four	sedan	4wd	front

5 rows × 26 columns

12. Replace the missing values in “horse_power”,peak_rpm with the values in the next row.

```
df[['horsepower', 'peak-rpm']] = df[['horsepower', 'peak-rpm']].fillna(method='bfill')
df.head()
```

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location
0	3	122.0	alfa-romero	gas	std	two	convertible	rwd	front
1	3	122.0	alfa-romero	gas	std	two	convertible	rwd	front
2	1	122.0	alfa-romero	gas	std	two	hatchback	rwd	front
3	2	164.0	audi	gas	std	four	sedan	fwd	front
4	2	164.0	audi	gas	std	four	sedan	4wd	front

5 rows × 26 columns

13. Drop the rows of “price”, if value is missing.

```
df.dropna(subset=["price"], axis=0, inplace=True)
```

```
df.head()
```

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location
0	3	122.0	alfa-	gas	std	two	convertible	rwd	front

14. List the datatypes of each column.

```
print(df.dtypes)
```

```
symboling              int64
normalized-losses    float64
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                 float64
stroke               float64
compression-ratio    float64
horsepower            object
peak-rpm              object
city-mpg               int64
highway-mpg            int64
price                 object
dtype: object
```

15. Convert the columns to appropriate datatype.

```
df[["symboling", "normalized-losses"]] = df[["symboling", "normalized-losses"]].astype("int")
df[["bore", "stroke", "price", "peak-rpm"]] = df[["bore", "stroke", "price", "peak-rpm"]].astype("float")
df[["num-of-doors", "num-of-cylinders"]] = df[["num-of-doors", "num-of-cylinders"]].replace({"four": 4, "six": 6, "five": 5, "eight": 8, "tw
df[["num-of-doors", "num-of-cylinders"]] = df[["num-of-doors", "num-of-cylinders"]].astype("int")
print(df.dtypes)
```

```
symboling              int32
normalized-losses      int32
make                  object
fuel-type             object
aspiration            object
num-of-doors           int32
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       int32
engine-size            int64
fuel-system           object
bore                 float64
stroke               float64
compression-ratio    float64
horsepower            object
peak-rpm             float64
city-mpg               int64
highway-mpg            int64
price                float64
dtype: object
```

16. Normalize the columns "length", "width" and "height" so their value ranges from 0 to 1.

```
df['length']=(df['length']-df['length'].min())/(df['length'].max()-df['length'].min())
df['width']=(df['width']-df['width'].min())/(df['width'].max()-df['width'].min())
df['height']=(df['height']-df['height'].min())/(df['height'].max()-df['height'].min())
```

```
df.head()
```

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location
0	3	122	alfa-romero	gas	std	2	convertible	rwd	front
1	3	122	alfa-romero	gas	std	2	convertible	rwd	front
2	1	122	alfa-romero	gas	std	2	hatchback	rwd	front
3	2	164	audi	gas	std	4	sedan	fwd	front
4	2	164	audi	gas	std	4	sedan	4wd	front

5 rows × 26 columns

Exercise 3

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from scipy.stats import norm
from scipy import stats
import warnings
warnings.filterwarnings('ignore')
```

```
data = pd.read_csv('auto-mpg.csv',index_col='car name')
data.head()
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin
car name								
chevrolet chevelle malibu	18.0	8	307.0	130	3504	12.0	70	1
buick skylark 320	15.0	8	350.0	165	3693	11.5	70	1

```
print(data.head())
print(data.index)
print(data.columns)
```

```
                            mpg  cylinders  displacement horsepower  weight  \
car name
chevrolet chevelle malibu  18.0          8         307.0        130    3504
buick skylark 320          15.0          8         350.0        165    3693
plymouth satellite         18.0          8         318.0        150    3436
amc rebel sst              16.0          8         304.0        150    3433
ford torino                17.0          8         302.0        140    3449

                           acceleration  model year  origin
car name
chevrolet chevelle malibu          12.0          70       1
buick skylark 320                  11.5          70       1
plymouth satellite                 11.0          70       1
```

```
amc rebel sst                      12.0          70       1
ford torino                        10.5          70       1
Index(['chevrolet chevelle malibu', 'buick skylark 320', 'plymouth satellite',
       'amc rebel sst', 'ford torino', 'ford galaxie 500', 'chevrolet impala',
       'plymouth fury iii', 'pontiac catalina', 'amc ambassador dpl',
       ...
       'chrysler lebaron medallion', 'ford granada l', 'toyota celica gt',
       'dodge charger 2.2', 'chevrolet camaro', 'ford mustang gl', 'vw pickup',
       'dodge rampage', 'ford ranger', 'chevy s-10'],
      dtype='object', name='car name', length=398)
Index(['mpg', 'cylinders', 'displacement', 'horsepower', 'weight',
       'acceleration', 'model year', 'origin'],
      dtype='object')
```

```
data.shape
```

```
(398, 8)
```

```
data.isnull().any()
```

```
mpg             False
cylinders       False
displacement    False
horsepower      False
weight          False
acceleration    False
model year      False
origin          False
dtype: bool
```

```
data.dtypes
```

```
mpg             float64
cylinders         int64
displacement    float64
horsepower       object
weight            int64
acceleration    float64
model year        int64
origin            int64
dtype: object
```

```
data.horsepower.unique()
```

```
array(['130', '165', '150', '140', '198', '220', '215', '225', '190',
       '170', '160', '95', '97', '85', '88', '46', '87', '90', '113',
       '200', '210', '193', '?', '100', '105', '175', '153', '180', '110',
       '72', '86', '70', '76', '65', '69', '60', '80', '54', '208', '155',
       '112', '92', '145', '137', '158', '167', '94', '107', '230', '49',
       '75', '91', '122', '67', '83', '78', '52', '61', '93', '148',
       '129', '96', '71', '98', '115', '53', '81', '79', '120', '152',
       '102', '108', '68', '58', '149', '89', '63', '48', '66', '139',
       '103', '125', '133', '138', '135', '142', '77', '62', '132', '84',
       '64', '74', '116', '82'], dtype=object)
```

```
data = data[data.horsepower != '?']
```

```
print('?' in data.horsepower)
```

```
False
```

```
data.shape
```

```
(392, 8)
```

```
data.dtypes
```

```
mpg             float64
cylinders         int64
```

```
displacement    float64
horsepower       object
weight            int64
acceleration    float64
model year        int64
origin            int64
dtype: object
```

```
data.horsepower = data.horsepower.astype('float')
data.dtypes
```

```
mpg             float64
cylinders         int64
displacement    float64
horsepower      float64
weight            int64
acceleration    float64
model year        int64
origin            int64
dtype: object
```

```
data.describe()
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	r
count	392.000000	392.000000	392.000000	392.000000	392.000000	392.000000	392.0(
mean	23.445918	5.471939	194.411990	104.469388	2977.584184	15.541327	75.97
std	7.805007	1.705783	104.644004	38.491160	849.402560	2.758864	3.68
min	9.000000	3.000000	68.000000	46.000000	1613.000000	8.000000	70.0(
25%	17.000000	4.000000	105.000000	75.000000	2225.250000	13.775000	73.0(
50%	22.750000	4.000000	151.000000	93.500000	2803.500000	15.500000	76.0(
75%	29.000000	8.000000	275.750000	126.000000	3614.750000	17.025000	79.0(

```
data.mpg.describe()
```

```
count    392.000000
mean      23.445918
std        7.805007
min        9.000000
25%       17.000000
50%       22.750000
75%       29.000000
max       46.600000
Name: mpg, dtype: float64
```

```
sns.distplot(data['mpg'])
```

```
<Axes: xlabel='mpg', ylabel='Density'>
```

```
print("Skewness: %f" % data['mpg'].skew())
print("Kurtosis: %f" % data['mpg'].kurt())
```

```
Skewness: 0.457092
Kurtosis: -0.515993
```

```
def scale(a):
    b = (a-a.min())/(a.max()-a.min())
    return b

data_scale = data.copy()

data_scale ['displacement'] = scale(data_scale['displacement'])
data_scale['horsepower'] = scale(data_scale['horsepower'])
data_scale ['acceleration'] = scale(data_scale['acceleration'])
data_scale ['weight'] = scale(data_scale['weight'])
data_scale['mpg'] = scale(data_scale['mpg'])

data_scale.head()
```

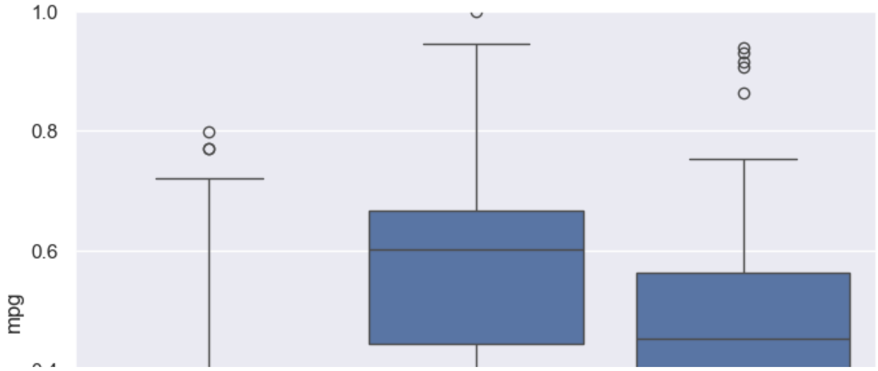
	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	c
car name								
chevrolet chevelle malibu	0.239362	8	0.617571	0.456522	0.536150	0.238095	70	
buick skylark 320	0.159574	8	0.728682	0.646739	0.589736	0.208333	70	

```
data['Country_code'] = data.origin.replace([1,2,3],['USA','Europe','Japan'])
data_scale['Country_code'] = data.origin.replace([1,2,3],['USA','Europe','Japan'])

data_scale.head()
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	c
car name								
chevrolet chevelle malibu	0.239362	8	0.617571	0.456522	0.536150	0.238095	70	
buick skylark 320	0.159574	8	0.728682	0.646739	0.589736	0.208333	70	

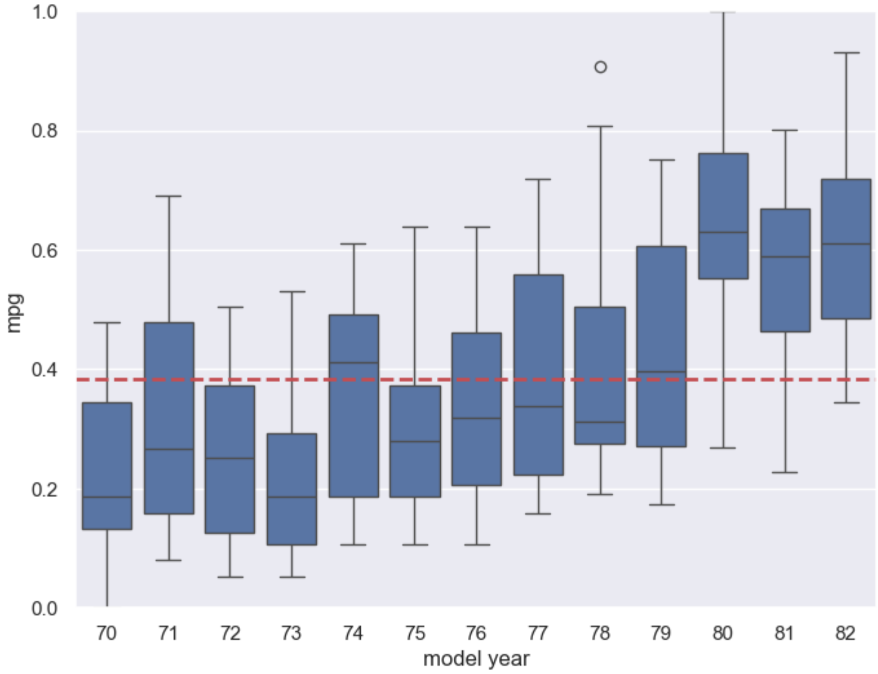
```
var = 'Country_code'
data_plt = pd.concat([data_scale['mpg'], data_scale[var]], axis=1)
f, ax = plt.subplots(figsize=(8, 6))
fig = sns.boxplot(x=var, y="mpg", data=data_plt)
fig.axis(ymin=0, ymax=1)
plt.axhline(data_scale.mpg.mean(),color='r',linestyle='dashed',linewidth=2)
```

```
<matplotlib.lines.Line2D at 0x13e346c5f40>
```



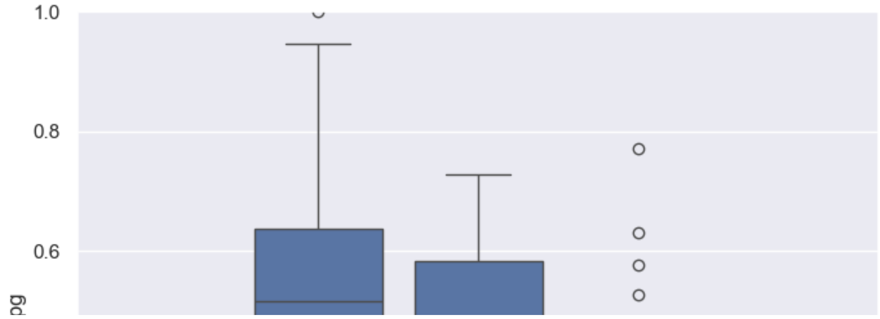
```
var = 'model year'
data_plt = pd.concat([data_scale['mpg'], data_scale[var]], axis=1)
f, ax = plt.subplots(figsize=(8, 6))
fig = sns.boxplot(x=var, y="mpg", data=data_plt)
fig.axis(ymin=0, ymax=1)
plt.axhline(data_scale.mpg.mean(),color='r',linestyle='dashed',linewidth=2)
```

```
<matplotlib.lines.Line2D at 0x13e37ac0980>
```



```
var = 'cylinders'
data_plt = pd.concat([data_scale['mpg'], data_scale[var]], axis=1)
f, ax = plt.subplots(figsize=(8, 6))
fig = sns.boxplot(x=var, y="mpg", data=data_plt)
fig.axis(ymin=0, ymax=1)
plt.axhline(data_scale.mpg.mean(),color='r',linestyle='dashed',linewidth=2)
```

```
<matplotlib.lines.Line2D at 0x13e37fbb290>
```

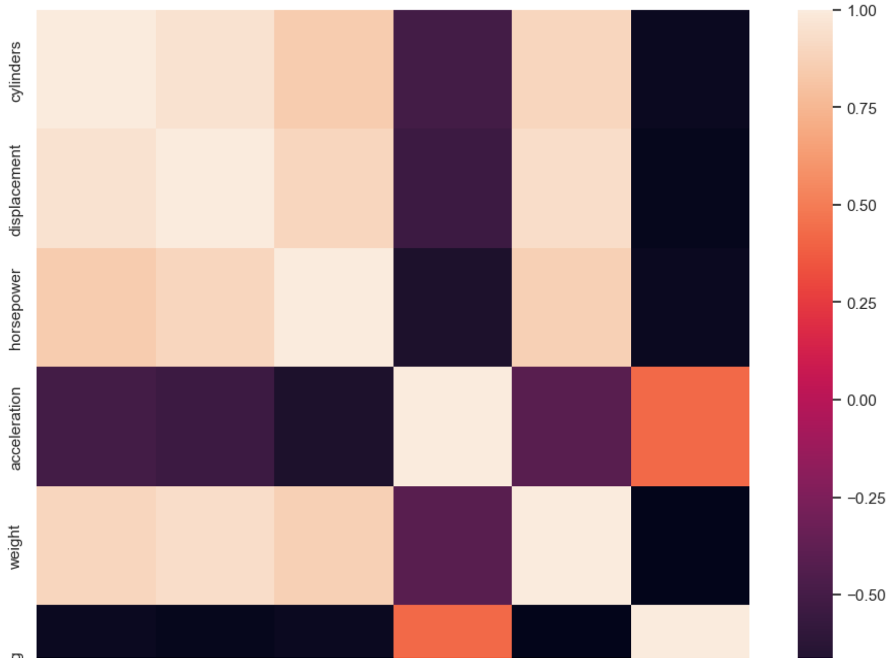


```
corrmat = data.corr()
f, ax = plt.subplots(figsize=(12, 9))
sns.heatmap(corrmat, square=True)
```

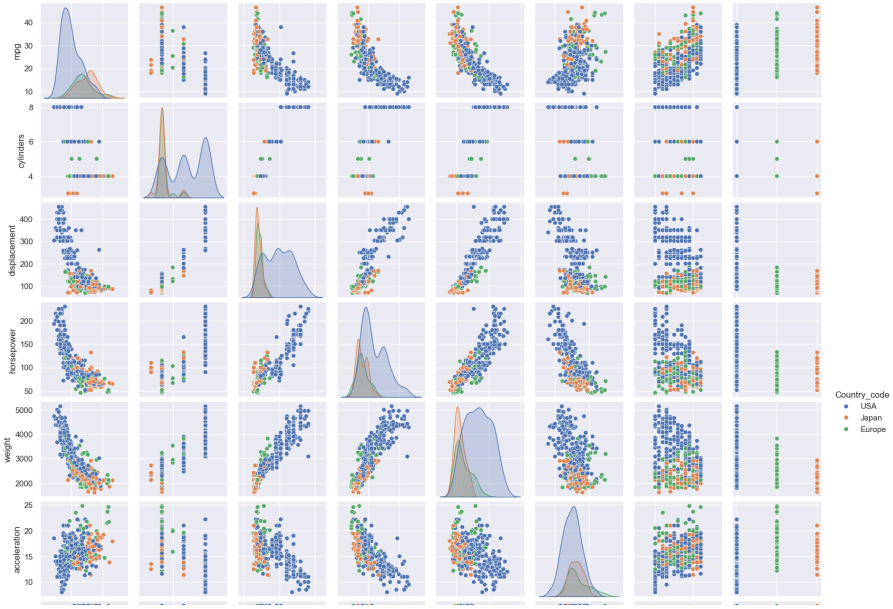
```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
c:\Users\aadit\Desktop\BTech\S5\Foundations Of Data Science\LAB\lab 2\lab2.ipynb Cell
57 line 1
----> <a href='vscode-notebook-
cell:/c%3A/Users/aadit/Desktop/BTech/S5/Foundations%20Of%20Data%20Science/LAB/lab%202/la
line=0'>1</a> corrmat = data.corr()
      <a href='vscode-notebook-
cell:/c%3A/Users/aadit/Desktop/BTech/S5/Foundations%20Of%20Data%20Science/LAB/lab%202/la
line=1'>2</a> f, ax = plt.subplots(figsize=(12, 9))
      <a href='vscode-notebook-
cell:/c%3A/Users/aadit/Desktop/BTech/S5/Foundations%20Of%20Data%20Science/LAB/lab%202/la
line=2'>3</a> sns.heatmap(corrmat, square=True)

File ~\AppData\Roaming\Python\Python312\site-packages\pandas\core\frame.py:10704, in
DataFrame.corr(self, method, min_periods, numeric_only)
  10702 cols = data.columns
  10703 idx = cols.copy()
> 10704 mat = data.to_numpy(dtype=float, na_value=np.nan, copy=False)
  10706 if method == "pearson":
  10707     correl = libalgos.nancorr(mat, minp=min_periods)

File ~\AppData\Roaming\Python\Python312\site-packages\pandas\core\frame.py:1889, in
DataFrame.to_numpy(self, dtype, copy, na_value)
   1887 if dtype is not None:
   1888     dtype = np.dtype(dtype)
-> 1889 result = self._mgr.as_array(dtype=dtype, copy=copy, na_value=na_value)
   1890 if result.dtype is not dtype:
   1891     result = np.array(result, dtype=dtype, copy=False)

File ~\AppData\Roaming\Python\Python312\site-
packages\pandas\core\internals\managers.py:1656, in BlockManager.as_array(self, dtype,
copy, na_value)
   1654         arr.flags.writeable = False
   1655 else:
-> 1656     arr = self._interleave(dtype=dtype, na_value=na_value)
   1657     # The underlying data was copied within _interleave, so no need
   1658     # to further copy if copy=True or setting na_value
   1660 if na_value is lib.no_default:

File ~\AppData\Roaming\Python\Python312\site-
packages\pandas\core\internals\managers.py:1715, in BlockManager._interleave(self,
```

```
factors = ['cylinders','displacement','horsepower','acceleration','weight','mpg']
corrmat = data[factors].corr()
f, ax = plt.subplots(figsize=(12, 9))
sns.heatmap(corrmat, square=True)
```

```
<Axes: >
```



```
sns.set()
sns.pairplot(data, size = 2.0,hue ='Country_code')
plt.show()
```



```
data.index
```

```
      Index(['chevrolet chevelle malibu', 'buick skylark 320', 'plymouth satellite',
             'amc rebel sst', 'ford torino', 'ford galaxie 500', 'chevrolet impala',
             'plymouth fury iii', 'pontiac catalina', 'amc ambassador dpl',
             ...
             'chrysler lebaron medallion', 'ford granada l', 'toyota celica gt',
             'dodge charger 2.2', 'chevrolet camaro', 'ford mustang gl', 'vw pickup',
             'dodge rampage', 'ford ranger', 'chevy s-10'],
            dtype='object', name='car name', length=392)
```

```
data[data.index.str.contains('subaru')].index.str.replace('(.*)', 'subaru dl')
```

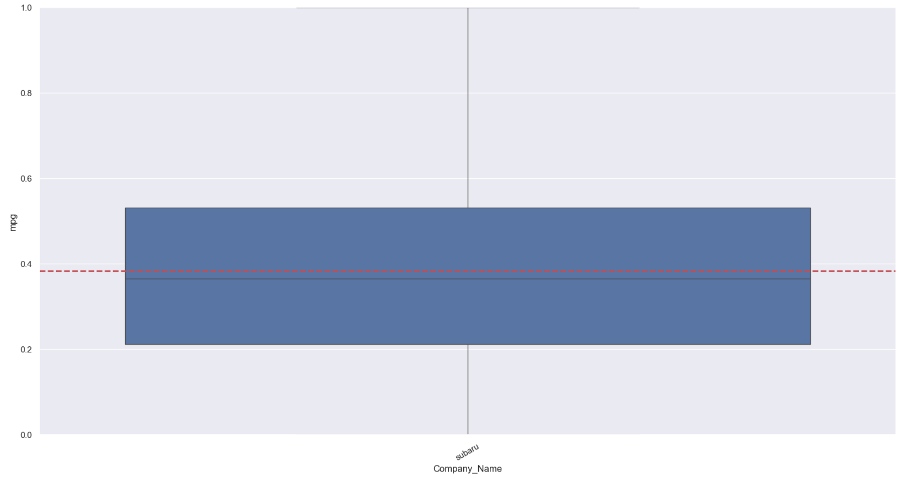
```
      Index(['subaru', 'subaru dl', 'subaru dl', 'subaru'], dtype='object', name='car name')
```

```
data['Company_Name'] = data.index.str.extract('(^.*?)\s')
```

```
data['Company_Name'] = data['Company_Name'].replace(['volkswagen','vokswagen','vw'],'VW')
data['Company_Name'] = data['Company_Name'].replace('maxda','mazda')
data['Company_Name'] = data['Company_Name'].replace('toyouta','toyota')
data['Company_Name'] = data['Company_Name'].replace('mercedes','mercedes-benz')
data['Company_Name'] = data['Company_Name'].replace('nissan','datsun')
data['Company_Name'] = data['Company_Name'].replace('capri','ford')
data['Company_Name'] = data['Company_Name'].replace(['chevroelt','chevy'],'chevrolet')
data['Company_Name'].fillna(value = 'subaru',inplace=True)
```

```
var = 'Company_Name'
data_plt = pd.concat([data_scale['mpg'], data[var]], axis=1)
f, ax = plt.subplots(figsize=(20,10))
fig = sns.boxplot(x=var, y="mpg", data=data_plt)
fig.set_xticklabels(ax.get_xticklabels(),rotation=30)
fig.axis(ymin=0, ymax=1)
plt.axhline(data_scale.mpg.mean(),color='r',linestyle='dashed',linewidth=2)
```

```
<matplotlib.lines.Line2D at 0x13e3a021e20>
```



```
data.Company_Name.isnull().any()
```

```
False
```

```
var='mpg'
data[data[var]== data[var].min()]
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	Co
car name									

```
data[data[var]== data[var].max()]
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	C
car name									

```
var='displacement'
data[data[var]== data[var].min()]
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	Cou
car name									

```
data[data[var]== data[var].max()]
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin
car name								
pontiac catalina	14.0	8	455.0	225.0	4425	10.0	70	1

```
var='horsepower'
data[data[var]== data[var].min()]
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	orig
car name								
volkswagen								

```
data[data[var]== data[var].max()]
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin
car name								

```
var='weight'
data[data[var]== data[var].min()]
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin
car name								

```
data[data[var]== data[var].max()]
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin
car name								

```
var='acceleration'
data[data[var]== data[var].min()]
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin
car name								

```
data[data[var]== data[var].max()]
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin
car name								

```
var = 'horsepower'
plot = sns.lmplot(var,'mpg',data=data,hue='Country_code')
plot.set(ylim = (0,50))
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
c:\Users\aadit\Desktop\BTech\S5\Foundations Of Data Science\LAB\lab 2\lab2.ipynb Cell
76 line 2
      <a href='vscode-notebook-
cell:/c%3A/Users/aadit/Desktop/BTech/S5/Foundations%20Of%20Data%20Science/LAB/lab%202/la
line=0'>1</a> var = 'horsepower'
----> <a href='vscode-notebook-
cell:/c%3A/Users/aadit/Desktop/BTech/S5/Foundations%20Of%20Data%20Science/LAB/lab%202/la
line=1'>2</a> plot = sns.lmplot(var,'mpg',data=data,hue='Country_code')
      <a href='vscode-notebook-
cell:/c%3A/Users/aadit/Desktop/BTech/S5/Foundations%20Of%20Data%20Science/LAB/lab%202/la
```

```
var = 'displacement'
plot = sns.lmplot(var,'mpg',data=data,hue='Country_code')
plot.set(ylim = (0,50))
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
c:\Users\aadit\Desktop\BTech\S5\Foundations Of Data Science\LAB\lab 2\lab2.ipynb Cell
77 line 2
      <a href='vscode-notebook-
cell:/c%3A/Users/aadit/Desktop/BTech/S5/Foundations%20Of%20Data%20Science/LAB/lab%202/la
line=0'>1</a> var = 'displacement'
----> <a href='vscode-notebook-
```

```
var = 'weight'
plot = sns.lmplot(var,'mpg',data=data,hue='Country_code')
plot.set(ylim = (0,50))
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
c:\Users\aadit\Desktop\BTech\S5\Foundations Of Data Science\LAB\lab 2\lab2.ipynb Cell
78 line 2
      <a href='vscode-notebook-
cell:/c%3A/Users/aadit/Desktop/BTech/S5/Foundations%20Of%20Data%20Science/LAB/lab%202/la
line=0'>1</a> var = 'weight'
----> <a href='vscode-notebook-
cell:/c%3A/Users/aadit/Desktop/BTech/S5/Foundations%20Of%20Data%20Science/LAB/lab%202/la
line=1'>2</a> plot = sns.lmplot(var,'mpg',data=data,hue='Country_code')
      <a href='vscode-notebook-
cell:/c%3A/Users/aadit/Desktop/BTech/S5/Foundations%20Of%20Data%20Science/LAB/lab%202/la
```

```
var = 'acceleration'
plot = sns.lmplot(var,'mpg',data=data,hue='Country_code')
plot.set(ylim = (0,50))
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
c:\Users\aadit\Desktop\BTech\S5\Foundations Of Data Science\LAB\lab 2\lab2.ipynb Cell
79 line 2
      <a href='vscode-notebook-
cell:/c%3A/Users/aadit/Desktop/BTech/S5/Foundations%20Of%20Data%20Science/LAB/lab%202/la
line=0'>1</a> var = 'acceleration'
----> <a href='vscode-notebook-
cell:/c%3A/Users/aadit/Desktop/BTech/S5/Foundations%20Of%20Data%20Science/LAB/lab%202/la
line=1'>2</a> plot = sns.lmplot(var,'mpg',data=data,hue='Country_code')
      <a href='vscode-notebook-
cell:/c%3A/Users/aadit/Desktop/BTech/S5/Foundations%20Of%20Data%20Science/LAB/lab%202/la
```

```
data['Power_to_weight'] = ((data.horsepower*0.7457)/data.weight)
```

```
data.sort_values(by='Power_to_weight',ascending=False ).head()
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin
car name								
buick estate wagon (sw)	14.0	8	455.0	225.0	3086	10.0	70	1
pontiac grand	16.0	8	400.0	230.0	4278	9.5	73	1

```
data.head()
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin
car name								
chevrolet chevelle malibu	18.0	8	307.0	130.0	3504	12.0	70	1
buick skylark 320	15.0	8	350.0	165.0	3693	11.5	70	1

```
from sklearn.preprocessing import StandardScaler
from sklearn.tree import DecisionTreeRegressor
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from math import sqrt
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import KFold
```

```
factors = ['cylinders','displacement','horsepower','acceleration','weight','origin','model year']
X = pd.DataFrame(data[factors].copy())
y = data['mpg'].copy()

X = StandardScaler().fit_transform(X)

X_train,X_test,y_train,y_test=train_test_split(X,y,test_size = 0.33,random_state=324)
X_train.shape[0] == y_train.shape[0]
```

```
     True
```

```
regressor = LinearRegression()

regressor.get_params()
```

```
     {'copy_X': True, 'fit_intercept': True, 'n_jobs': None, 'positive': False}
```

```
regressor.fit(X_train,y_train)
```

```
     ▾ LinearRegression
     LinearRegression()
```

```
y_predicted = regressor.predict(X_test)
```

```
rmse = sqrt(mean_squared_error(y_true=y_test,y_pred=y_predicted))
rmse
```

```
     3.486729614901561
```

```
gb_regressor = GradientBoostingRegressor(n_estimators=4000)
gb_regressor.fit(X_train,y_train)
```

```
     ▾         GradientBoostingRegressor
     GradientBoostingRegressor(n_estimators=4000)
```

```
gb_regressor.get_params()
```

```
     {'alpha': 0.9,
      'ccp_alpha': 0.0,
      'criterion': 'friedman_mse',
      'init': None,
      'learning_rate': 0.1,
      'loss': 'squared_error',
      'max_depth': 3,
      'max_features': None,
      'max_leaf_nodes': None,
      'min_impurity_decrease': 0.0,
      'min_samples_leaf': 1,
      'min_samples_split': 2,
      'min_weight_fraction_leaf': 0.0,
      'n_estimators': 4000,
      'n_iter_no_change': None,
      'random_state': None,
      'subsample': 1.0,
      'tol': 0.0001,
      'validation_fraction': 0.1,
      'verbose': 0,
      'warm_start': False}
```

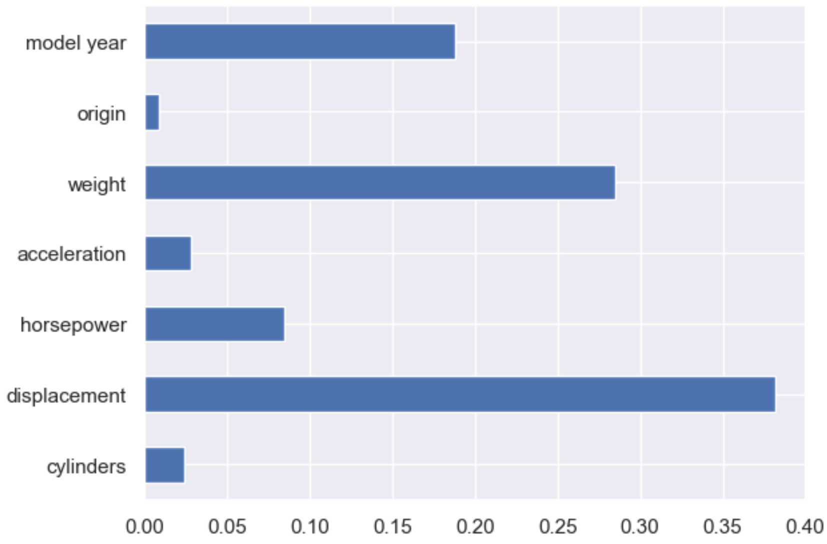
```
y_predicted_gbr = gb_regressor.predict(X_test)
```

```
rmse_bgr = sqrt(mean_squared_error(y_true=y_test,y_pred=y_predicted_gbr))
rmse_bgr
```

```
2.678422792051918
```

```
fi= pd.Series(gb_regressor.feature_importances_,index=factors)
fi.plot.barh()
```

```
<Axes: >
```



```
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
pca.fit(data[factors])
```

```
▾ PCA
PCA(n_components=2)
```

```
pca.explained_variance_ratio_
```

```
array([0.99756151, 0.0020628 ])
```

```
pca1 = pca.components_[0]
pca2 = pca.components_[1]
```

```
transformed_data = pca.transform(data[factors])
pc1 = transformed_data[:,0]
pc2 = transformed_data[:,1]
plt.scatter(pc1,pc2)
```

```
<matplotlib.collections.PathCollection at 0x13e32fec410>
```

```
c = pca.inverse_transform(transformed_data[(transformed_data[:,0]>0 )& (transformed_data[:,1]>250)])
factors
```

```
['cylinders',
 'displacement',
 'horsepower',
 'acceleration',
 'weight',
 'origin',
 'model year']
```

```
c
```

```
array([[9.32016159e+00, 4.65727261e+02, 1.90441442e+02, 5.95699243e+00,
        3.08611199e+03, 6.23550659e-01, 6.93571097e+01]])
```

```
data[(data['model year'] == 70 )&( data.displacement>400)]
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	Country_code	Company_Name	Power_to_weight
car name											
ford galaxie 500	15.0	8	429.0	198.0	4341	10.0	70	1	USA	subaru	0.034013
chevrolet impala	14.0	8	454.0	220.0	4354	9.0	70	1	USA	subaru	0.037679
plymouth fury iii	14.0	8	440.0	215.0	4312	8.5	70	1	USA	subaru	0.037181

```
cv_sets = KFold(n_splits=10, shuffle= True,random_state=100)
params = {'n_estimators' : list(range(40,61)),
         'max_depth' : list(range(1,10)),
         'learning_rate' : [0.1,0.2,0.3] }
grid = GridSearchCV(gb_regressor, params,cv=cv_sets,n_jobs=4)
grid = grid.fit(X_train, y_train)
grid.best_estimator_
```

```
▾                         GradientBoostingRegressor
GradientBoostingRegressor(learning_rate=0.3, max_depth=2, n_estimators=55)
```

```
gb_regressor_t = grid.best_estimator_
gb_regressor_t.fit(X_train,y_train)
```

```
▾                         GradientBoostingRegressor
GradientBoostingRegressor(learning_rate=0.3, max_depth=2, n_estimators=55)
```

```
y_predicted_gbr_t = gb_regressor_t.predict(X_test)
rmse = sqrt(mean_squared_error(y_true=y_test,y_pred=y_predicted_gbr_t))
rmse
```

```
2.6762648633586865
```

```
data.duplicated().any()
```

```
False
```