Assignment No 4

```
import pandas as pd
import numpy as np
import io
import seaborn as sns
import matplotlib.pyplot as plt
```

```
%matplotlib inline
```

```
from google.colab import files
uploaded=files.upload()
```

```
<IPython.core.display.HTML object>
```

```
Saving housing.csv to housing.csv
```

```
df=pd.read_csv(io.BytesIO(uploaded['housing.csv']))
```

```
print(df)
```

```
        RM  LSTAT  PTRATIO    MEDV
0    6.575   4.98     15.3  504000
1    6.421   9.14     17.8  453600
2    7.185   4.03     17.8  728700
3    6.998   2.94     18.7  701400
4    7.147   5.33     18.7  760200
..     ...    ...      ...     ...
484  6.593   9.67     21.0  470400
485  6.120   9.08     21.0  432600
486  6.976   5.64     21.0  501900
487  6.794   6.48     21.0  462000
488  6.030   7.88     21.0  249900

[489 rows x 4 columns]
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 489 entries, 0 to 488
Data columns (total 4 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   RM       489 non-null    float64
 1   LSTAT    489 non-null    float64
 2   PTRATIO  489 non-null    float64
 3   MEDV     489 non-null    int64
dtypes: float64(3), int64(1)
memory usage: 15.4 KB
```

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

```
RM         0
LSTAT      0
PTRATIO    0
MEDV       0
dtype: int64
```

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

```
0
```

```
print(np.shape(df))
```

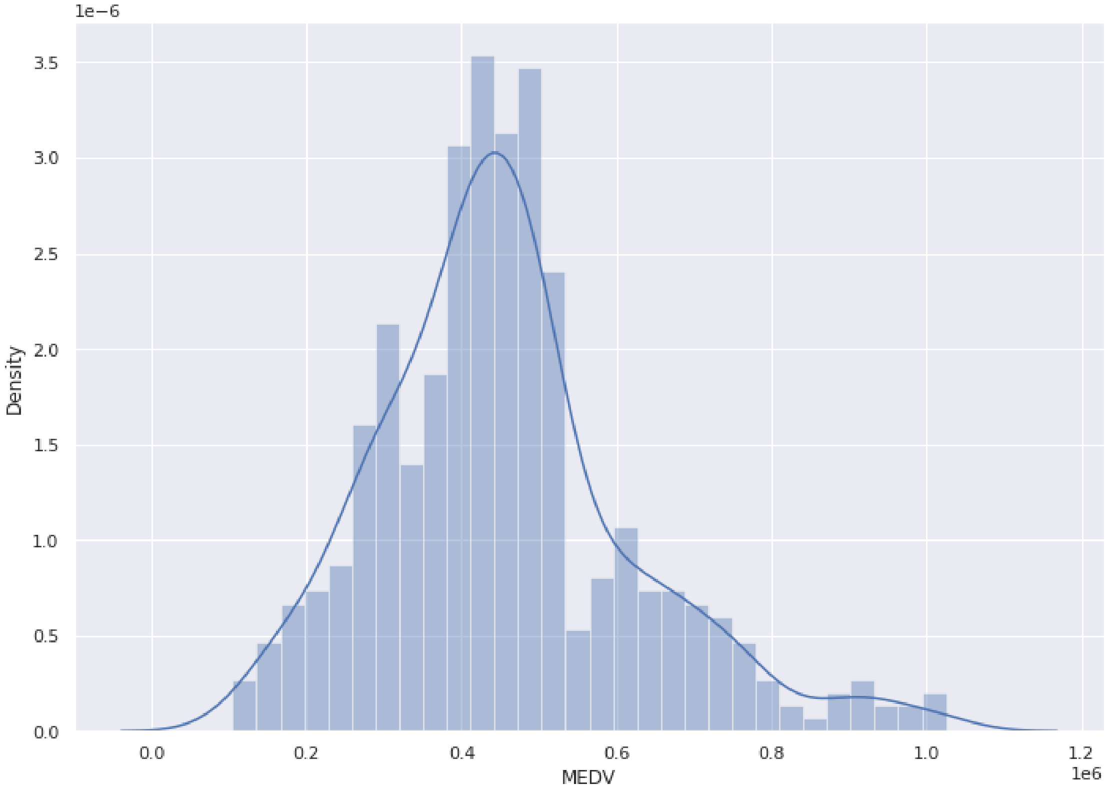
```
(489, 4)
```

```
print(df.describe())
```

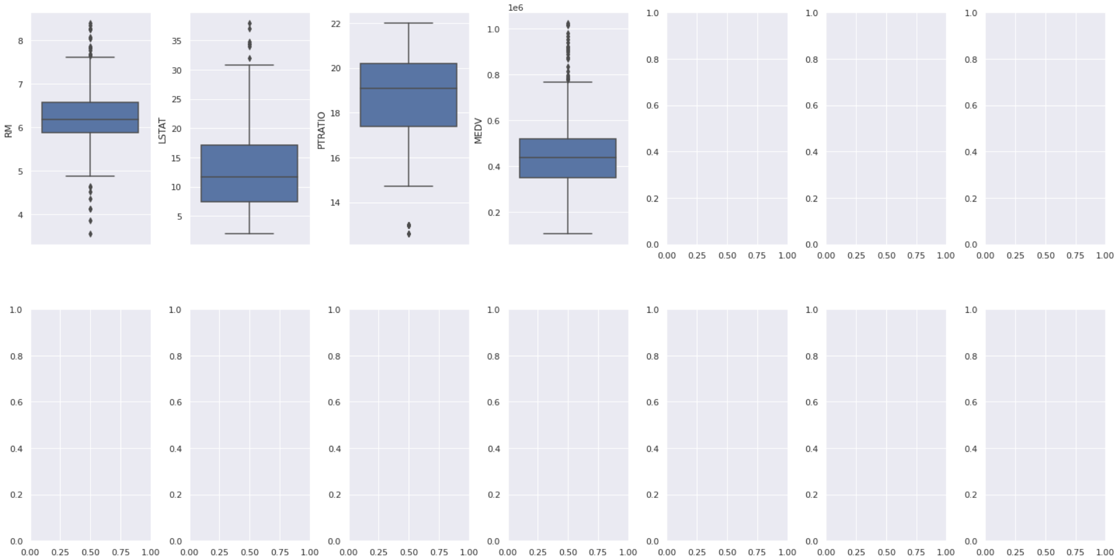
```
               RM       LSTAT     PTRATIO          MEDV
count  489.000000  489.000000  489.000000  4.890000e+02
mean     6.240288   12.939632   18.516564  4.543429e+05
std      0.643650    7.081990    2.111268  1.653403e+05
min      3.561000    1.980000   12.600000  1.050000e+05
25%      5.880000    7.370000   17.400000  3.507000e+05
50%      6.185000   11.690000   19.100000  4.389000e+05
75%      6.575000   17.120000   20.200000  5.187000e+05
max      8.398000   37.970000   22.000000  1.024800e+06
```

```
#set the size of the figure
sns.set(rc={'figure.figsize':(11.7,8.27)})

#Histogram for distribution of the target values
sns.distplot(df['MEDV'], bins =30)
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: 
FutureWarning: `distplot` is a deprecated function and will be removed in a 
future version. Please adapt your code to use either `displot` (a figure-
level function with similar flexibility) or `histplot` (an axes-level 
function for histograms).
  warnings.warn(msg, FutureWarning)
```

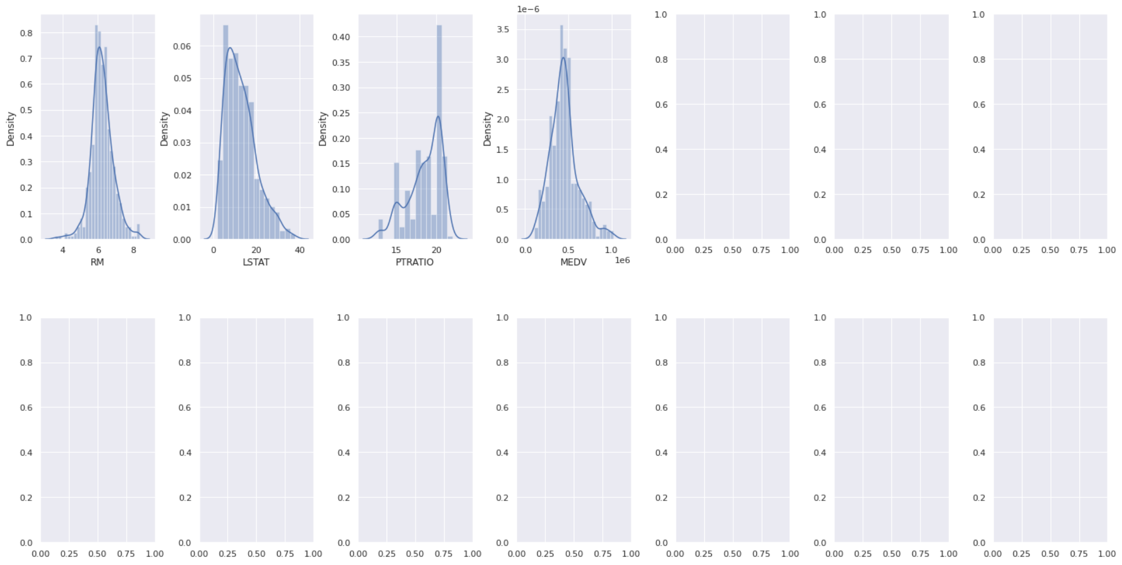


```
import seaborn as sns
import matplotlib.pyplot as plt
from scipy import stats

fig, axs = plt.subplots(ncols=7, nrows=2, figsize=(20, 10))
index = 0
axs = axs.flatten()
for k,v in df.items():
    sns.boxplot(y=k, data=df, ax=axs[index])
    index += 1
plt.tight_layout(pad=0.4, w_pad=0.5, h_pad=5.0)
```



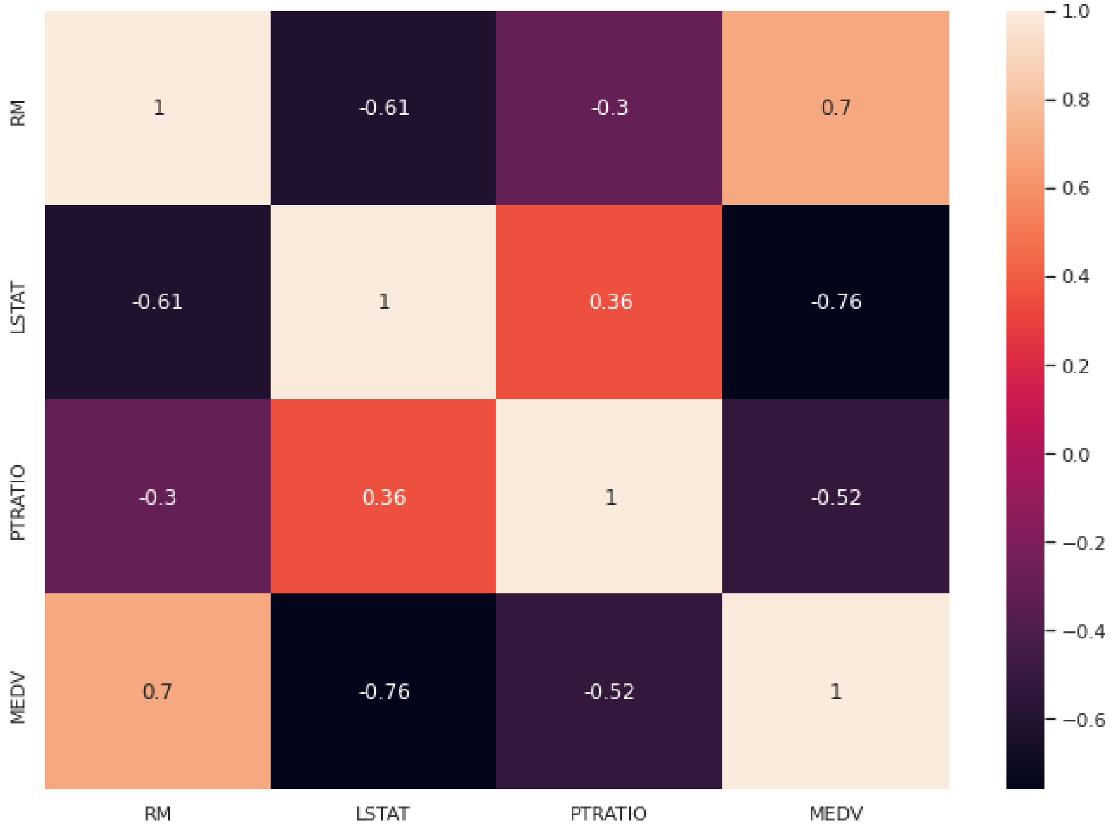
```
fig, axs = plt.subplots(ncols=7, nrows=2, figsize=(20, 10))
index = 0
axs = axs.flatten()
for k,v in df.items():
    sns.distplot(v, ax=axs[index])
    index += 1
plt.tight_layout(pad=0.4, w_pad=0.5, h_pad=5.0)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-
level function with similar flexibility) or `histplot` (an axes-level
function for histograms).
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-
level function with similar flexibility) or `histplot` (an axes-level
function for histograms).
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-
level function with similar flexibility) or `histplot` (an axes-level
function for histograms).
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-
level function with similar flexibility) or `histplot` (an axes-level
function for histograms).
  warnings.warn(msg, FutureWarning)
```



```
correlation_matrix = df.corr().round(2)
sns.heatmap(data=correlation_matrix, annot = True)
```

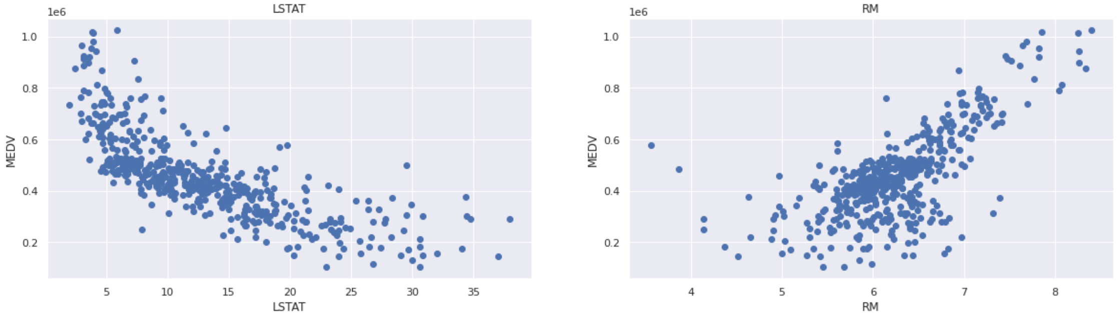
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9d65c7b450>
```



```
plt.figure(figsize=(20,5))
```

```
features = ['LSTAT','RM']
```

```
target = df['MEDV']

for i, col in enumerate(features):
  plt.subplot(1, len(features), i+1)
  x=df[col]
  y = target
  plt.scatter(x,  y,marker='o')
  plt.title(col)
  plt.xlabel(col)
  plt.ylabel('MEDV')
```



```
#Prepare data for training
X = pd.DataFrame(np.c_[df['LSTAT'],df['RM']], columns = ['LSTAT','RM'])
Y = df['MEDV']

#split data into training and testing sets
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,test_size =
0.3,random_state = 42)
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_train.shape)
```

```
(342, 2)
(147, 2)
(342,)
(342,)
```

```
from sklearn import linear_model
# Train the model using sklearn linear regression
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

lin_model = LinearRegression()
lin_model.fit(X_train,y_train)
```

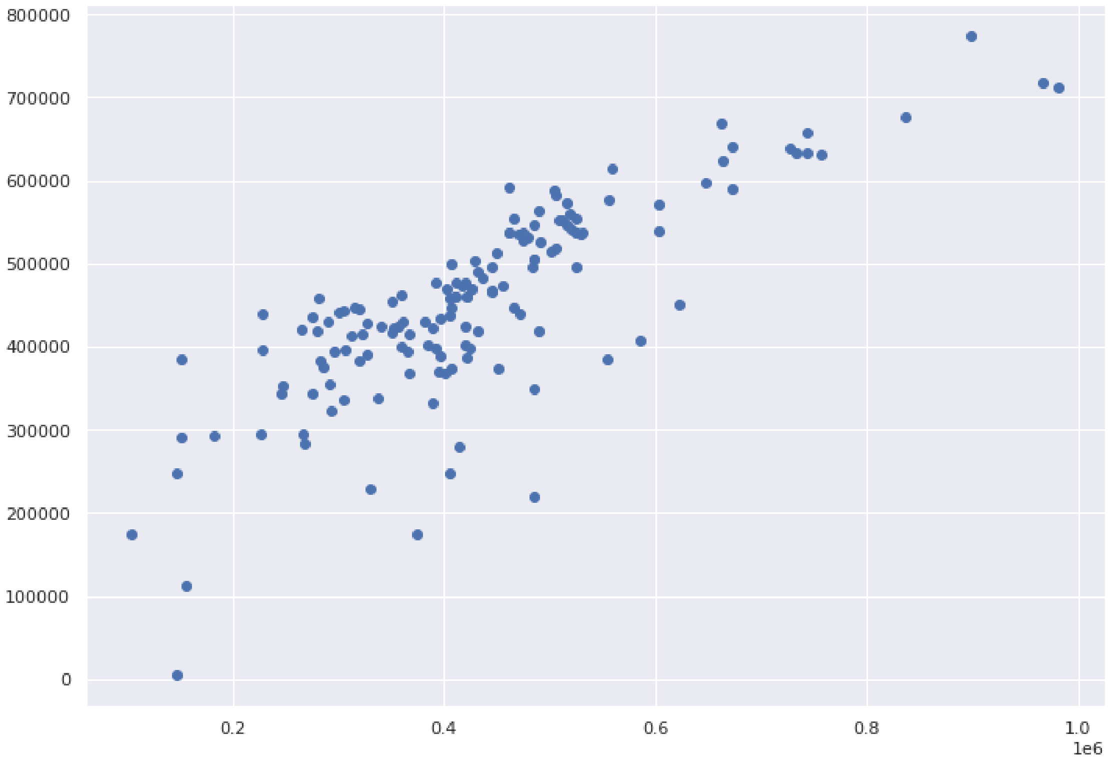
```
LinearRegression()
```

```
# Model evaluation for training set
y_train_predict = lin_model.predict(X_train)
```

```
rmse = (np.sqrt(mean_squared_error(y_train, y_train_predict)))
r2 = r2_score(y_train,y_train_predict)

print("Model performance for training set")
print('RMSE is {}'.format(rmse))
print('R2 score is {}'.format (r2))
print("\n")

# Model evaluation for testing set
y_test_predict = lin_model.predict(X_test)
rmse = (np.sqrt(mean_squared_error(y_test, y_test_predict)))
r2 = r2_score(y_test,y_test_predict)

print("Model performance for testing set")
print('RMSE is {}'.format(rmse))
print('R2 score is {}'.format (r2))
print("\n")
```

```
Model performance for training set
RMSE is 97613.15525868809
R2 score is 0.6701982599508145


Model performance for testing set
RMSE is 92082.48814292479
R2 score is 0.628271585004582
```

```
# Plotting y_test vs y_pred
plt.scatter(y_test, y_test_predict)
plt.show()
```


800000
700000
600000
500000
400000
300000
200000
100000
0
0.2
0.4
0.6
0.8
1.0
1e6