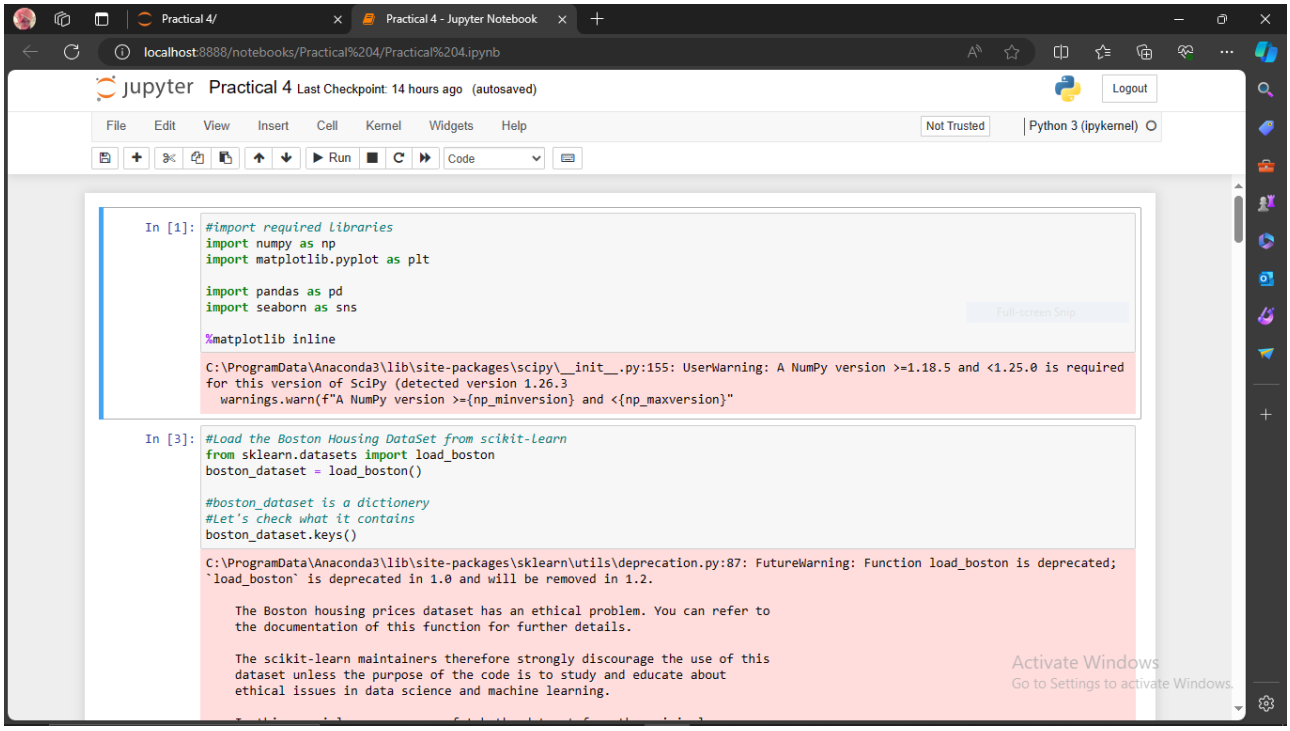
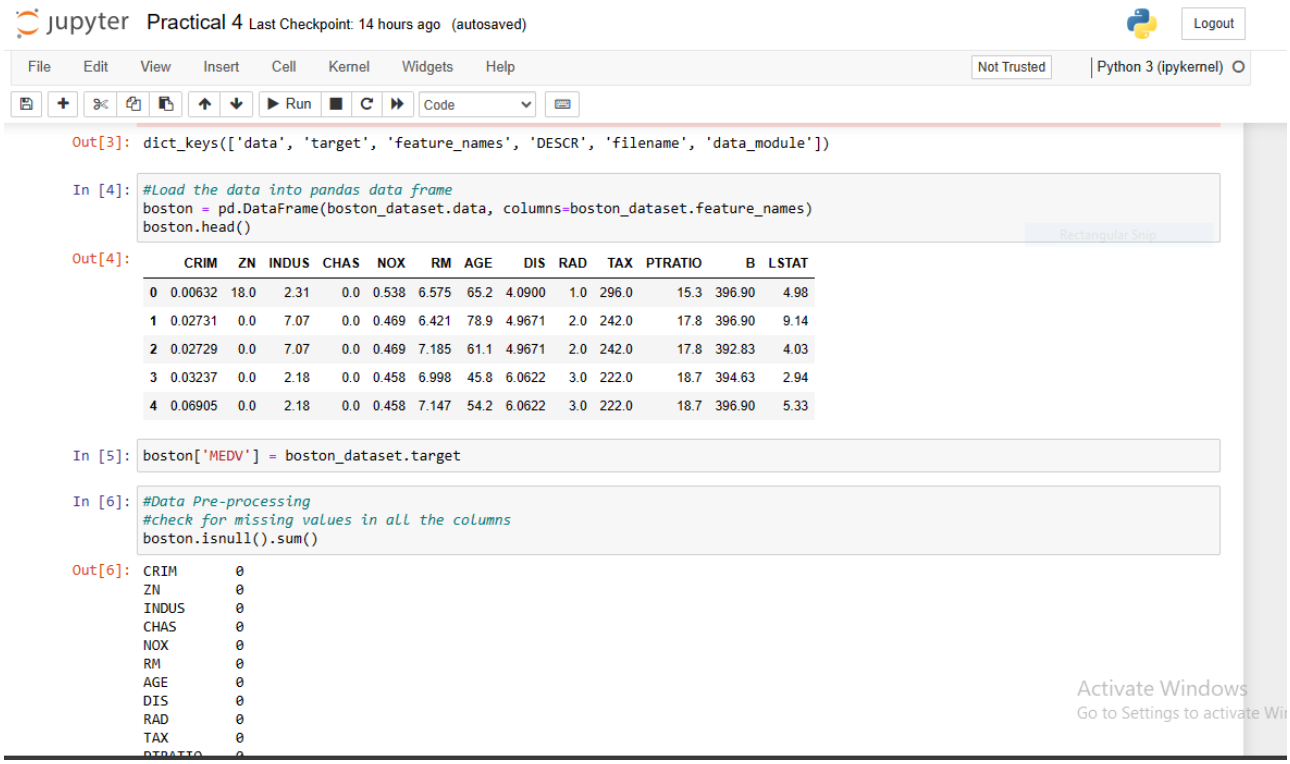

Practical 4/
Practical 4 - Jupyter Notebook
localhost:8888/notebooks/Practical%204/Practical%204.ipynb
jupyter Practical 4 Last Checkpoint: 14 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
In [1]:
#import required libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
%matplotlib inline
C:\ProgramData\Anaconda3\lib\site-packages\scipy__init__.py:155: UserWarning: A NumPy version >=1.18.5 and <1.25.0 is required for this version of SciPy (detected version 1.26.3
warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
In [3]:
#Load the Boston Housing DataSet from scikit-Learn
from sklearn.datasets import load_boston
boston_dataset = load_boston()
#boston_dataset is a dictionery
#Let's check what it contains
boston_dataset.keys()
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\deprecation.py:87: FutureWarning: Function load_boston is deprecated; `load_boston` is deprecated in 1.0 and will be removed in 1.2.
The Boston housing prices dataset has an ethical problem. You can refer to
the documentation of this function for further details.
The scikit-learn maintainers therefore strongly discourage the use of this
dataset unless the purpose of the code is to study and educate about
ethical issues in data science and machine learning.
Activate Windows
Go to Settings to activate Windows.


jupyter Practical 4 Last Checkpoint: 14 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
Out[3]:
dict_keys(['data', 'target', 'feature_names', 'DESCR', 'filename', 'data_module'])
In [4]:
#Load the data into pandas data frame
boston = pd.DataFrame(boston_dataset.data, columns=boston_dataset.feature_names)
boston.head()
Out[4]:
CRIM ZN INDUS CHAS NOX RM AGE DIS RAD TAX PTRATIO B LSTAT
0 0.00632 18.0 2.31 0.0 0.538 6.575 65.2 4.0900 1.0 296.0 15.3 396.90 4.98
1 0.02731 0.0 7.07 0.0 0.469 6.421 78.9 4.9671 2.0 242.0 17.8 396.90 9.14
2 0.02729 0.0 7.07 0.0 0.469 7.185 61.1 4.9671 2.0 242.0 17.8 392.83 4.03
3 0.03237 0.0 2.18 0.0 0.458 6.998 45.8 6.0622 3.0 222.0 18.7 394.63 2.94
4 0.06905 0.0 2.18 0.0 0.458 7.147 54.2 6.0622 3.0 222.0 18.7 396.90 5.33
In [5]:
boston['MEDV'] = boston_dataset.target
In [6]:
#Data Pre-processing
#check for missing values in all the columns
boston.isnull().sum()
Out[6]:
CRIM 0
ZN 0
INDUS 0
CHAS 0
NOX 0
RM 0
AGE 0
DIS 0
RAD 0
TAX 0
Activate Windows
Go to Settings to activate Wi

jupyter **Practical 4** Last Checkpoint: 14 hours ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help | Not Trusted | Python 3 (ipykernel)

Run | Code

In [6]:
```
#Data Pre-processing
#check for missing values in all the columns
boston.isnull().sum()
```

Out[6]:
```
CRIM       0
ZN         0
INDUS      0
CHAS       0
NOX        0
RM         0
AGE        0
DIS        0
RAD        0
TAX        0
PTRATIO    0
B          0
LSTAT      0
MEDV       0
dtype: int64
```

In [7]:
```
#Data visualization
#set he size of the figure

sns.set(rc={'figure.figsize':(11.7,8.27)})

#plot a histogram showing the distributionof the target values

sns.distplot(boston['MEDV'], bins=30)
plt.show()
```

```
C:\ProgramData\Anaconda3\lib\site-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)
```

jupyter **Practical 4** Last Checkpoint: 14 hours ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help | Not Trusted | Python 3 (ipykernel)

Run | Code

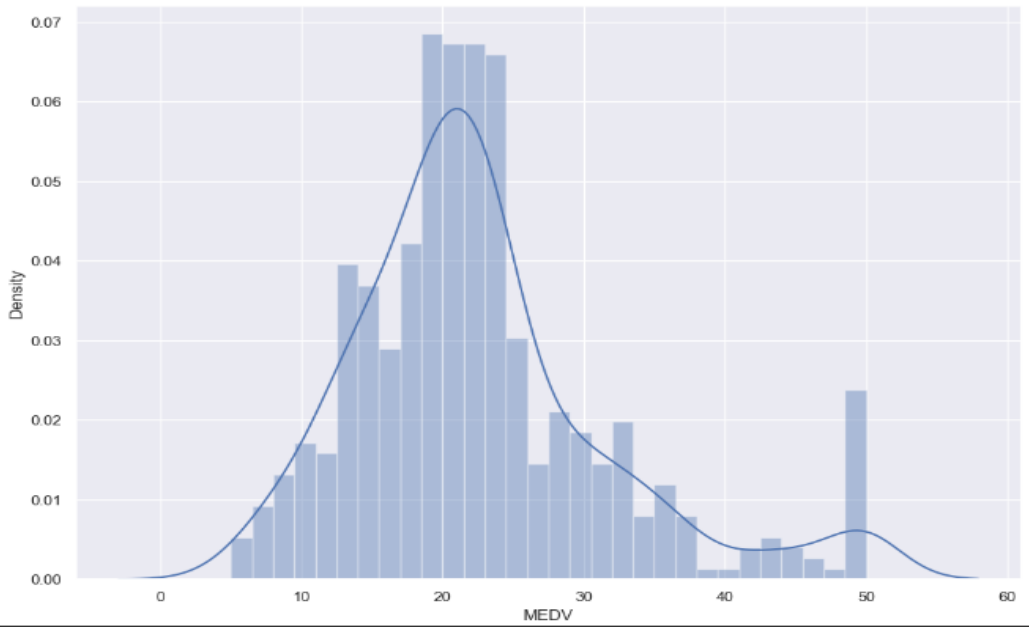
#set he size of the figure

sns.set(rc={'figure.figsize':(11.7,8.27)})

#plot a histogram showing the distributionof the target values

sns.distplot(boston['MEDV'], bins=30)
plt.show()

C:\ProgramData\Anaconda3\lib\site-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)



jupyter **Practical 4** Last Checkpoint: 14 hours ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help | Not Trusted | Python 3 (ipykernel)

Run | Code

In [9]:
```
#Correlation matrix

#compute the pair wise correlation for all columns
correlation_matrix = boston.corr().round(2)
```

In [10]:
```
#use the heatmap function from seaborn to plot the correlation matrix
#annot = True to print the values inside the square
sns.heatmap(data = correlation_matrix, annot=True)
```

Out[10]: <AxesSubplot:>


jupyter **Practical 4** Last Checkpoint: 14 hours ago (autosaved)

Logout

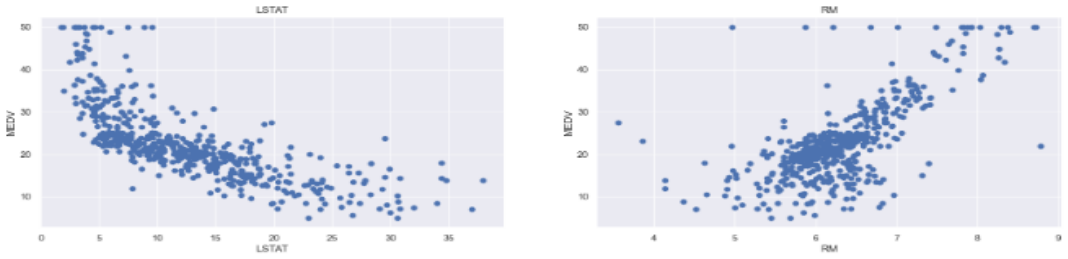
File Edit View Insert Cell Kernel Widgets Help | Not Trusted | Python 3 (ipykernel)

Run | Code

In [15]:
```
plt.figure(figsize=(20,5))

features = ["LSTAT", 'RM']
target = boston['MEDV']

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




In [19]:
```
#Preapare the data for training
X = pd.DataFrame(np.c_[boston['LSTAT'],boston['RM']], columns = ['LSTAT','RM'])
Y = boston['MEDV']
```

In [21]:
```
#Split the data into training and testing sets

from sklearn.model_selection import train_test_split

#splits the training and test data set in 80% : 20%
#assign random_state to any value. This ensures consistency
X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size = 0.2,random_state=20)
print(X_train.shape)
print(X_test.shape)
print(Y_train.shape)
print(Y_test.shape)
```

```
(404, 2)
(102, 2)
(404,)
(102,)
```

jupyter **Practical 4** Last Checkpoint: 14 hours ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help | Not Trusted | Python 3 (ipykernel)

Run | Code

In [23]:
```
#Train the model using sklearnLinearRegression

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)
```

Out[23]:
```
LinearRegression()
```

In [25]:
```
#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("The model performance for testing set")
print("--------------------------------------")
print('RMSE is {}'.format(rmse))
print('R2 score is {}'.format(r2))

#model evaluation for testing set

y_test_predict = lin_model.predict(X_test)

#root mean square error of the model
rmse = (np.sqrt(mean_squared_error(Y_test,y_test_predict)))

#r-squaredscore of the model
r2 = r2_score(Y_test, y_test_predict)

print("The model performance for testing set")
print("--------------------------------------")
print('RMSE is {}'.format(rmse))
print('R2 score is {}'.format(r2))
```

```
The model performance for testing set
--------------------------------------
RMSE is 5.612505753798557
R2 score is 0.6468915821243122
The model performance for testing set
--------------------------------------
RMSE is 5.175217627561771
R2 score is 0.5841519194311253
```

In [26]:
```
#Plotting the y_test vs y_pred
#ideally should have been a straight line
plt.scatter(Y_test,y_test_predict)
plt.show()
```




jupyter **Practical 4** Last Checkpoint: 14 hours ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help | Not Trusted | Python 3 (ipykernel)

Run | Code

In [26]:
```
#Plotting the y_test vs y_pred
#ideally should have been a straight line
plt.scatter(Y_test,y_test_predict)
plt.show()
```


In [ ]: