LAB TASK

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
import statsmodels.api as sm
import matplotlib.pyplot as plt
import numpy as np

# 1. Read the BostonHousing Dataset
df = pd.read_csv('BostonHousing.csv')

# show null vals of dataset
print(df.isnull().sum())

# Clean dataset
df = df.dropna()
print("------------- AFTER -------------")
print(df.isnull().sum())
```

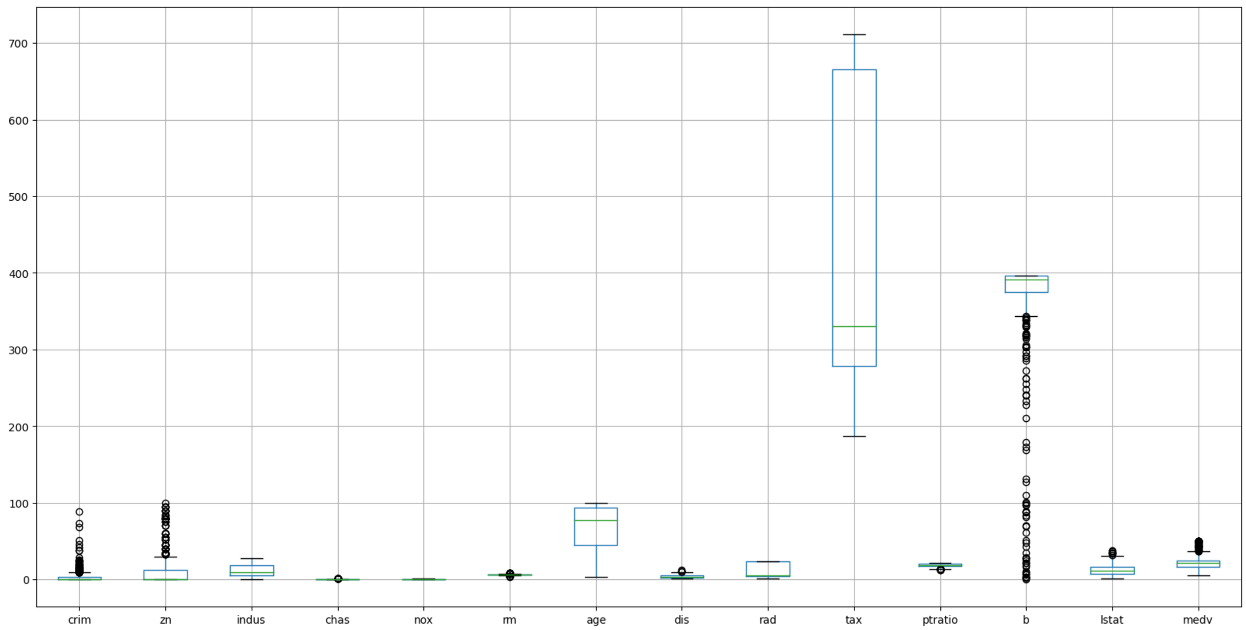
```
crim       0
zn         0
indus      0
chas       0
nox        0
rm         5
age        0
dis        0
rad        0
tax        0
ptratio    0
b          0
lstat      0
medv       0
dtype: int64
------------- AFTER -------------
crim       0
zn         0
indus      0
chas       0
nox        0
rm         0
age        0
dis        0
rad        0
tax        0
ptratio    0
```

```
b            0
lstat        0
medv         0
dtype: int64

# Print outliers
for column in df.columns:
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    outliers = df[(df[column] < (Q1 - 1.5 * IQR)) | (df[column] > (Q3
+ 1.5 * IQR))][column]
    print(f"Outliers for {column} are {outliers}")

# Show boxplot
df.boxplot(figsize=(20,10))
plt.show()

Outliers for crim are 367    13.5222
371     9.2323
373    11.1081
374    18.4982
375    19.6091
        ...
468    15.5757
469    13.0751
477    15.0234
478    10.2330
479    14.3337
Name: crim, Length: 66, dtype: float64
Outliers for zn are 39     75.0
40     75.0
54     75.0
55     90.0
56     85.0
       ...
351    60.0
352    60.0
353    90.0
354    80.0
355    80.0
Name: zn, Length: 68, dtype: float64
Outliers for indus are Series([], Name: indus, dtype: float64)
Outliers for chas are 142    1
152    1
154    1
155    1
160    1
162    1
163    1
```

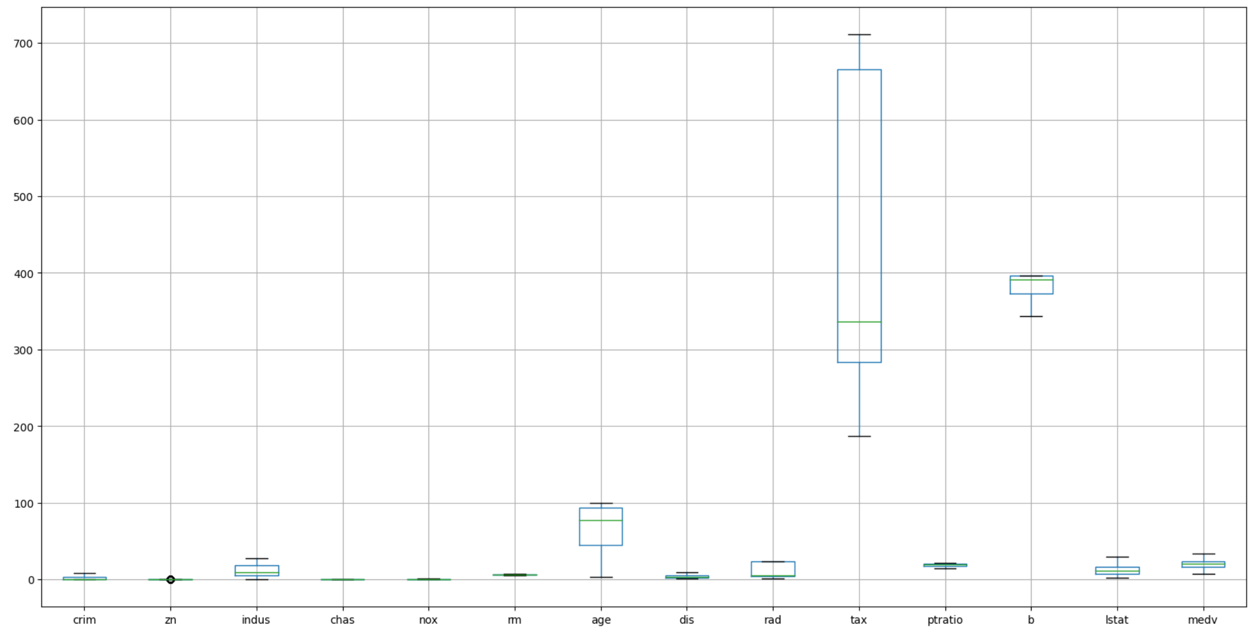
```
208    1
209    1
210    1
211    1
212    1
216    1
218    1
219    1
220    1
221    1
222    1
234    1
236    1
269    1
273    1
274    1
276    1
277    1
282    1
283    1
356    1
357    1
358    1
363    1
364    1
369    1
370    1
372    1
Name: chas, dtype: int64
Outliers for nox are Series([], Name: nox, dtype: float64)
Outliers for rm are 97     8.069
98     7.820
162    7.802
163    8.375
166    7.929
180    7.765
186    7.831
195    7.875
203    7.853
204    8.034
224    8.266
225    8.725
226    8.040
232    8.337
233    8.247
253    8.259
257    8.704
262    8.398
267    8.297
```

```
280    7.820
283    7.923
364    8.780
365    3.561
367    3.863
374    4.138
384    4.368
386    4.652
406    4.138
412    4.628
414    4.519
Name: rm, dtype: float64
Outliers for age are Series([], Name: age, dtype: float64)
Outliers for dis are 351    10.7103
352    10.7103
353    12.1265
354    10.5857
355    10.5857
Name: dis, dtype: float64
Outliers for rad are Series([], Name: rad, dtype: int64)
Outliers for tax are Series([], Name: tax, dtype: int64)
Outliers for ptratio are 196    12.6
197    12.6
198    12.6
257    13.0
258    13.0
259    13.0
260    13.0
261    13.0
262    13.0
263    13.0
264    13.0
265    13.0
266    13.0
267    13.0
268    13.0
Name: ptratio, dtype: float64
Outliers for b are 18     288.99
25     303.42
27     306.38
32     232.60
34     248.31
        ...
465    334.40
466     22.01
467    331.29
475    302.76
490    318.43
Name: b, Length: 76, dtype: float64
```

```
Outliers for lstat are 141    34.41
373    34.77
374    37.97
387    31.99
412    34.37
414    36.98
438    34.02
Name: lstat, dtype: float64
Outliers for medv are 97     38.7
98     43.8
157    41.3
161    50.0
162    50.0
163    50.0
166    50.0
179    37.2
180    39.8
182    37.9
186    50.0
195    50.0
202    42.3
203    48.5
204    50.0
224    44.8
225    50.0
226    37.6
228    46.7
232    41.7
233    48.3
253    42.8
256    44.0
257    50.0
261    43.1
262    48.8
267    50.0
268    43.5
280    45.4
282    46.0
283    50.0
291    37.3
368    50.0
369    50.0
370    50.0
371    50.0
372    50.0
Name: medv, dtype: float64
```




```
for column in df.columns:
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    mean = df[column].mean()
    df[column] = np.where(((df[column] < (Q1 - 1.5 * IQR)) |
(df[column] > (Q3 + 1.5 * IQR))), mean, df[column])


# Show boxplot after replacing outliers
df.boxplot(figsize=(20,10))
plt.show()
```




```python
# 2. Fit the linear Multi regression.
X = df[['crim', 'zn', 'indus', 'chas', 'nox', 'rm', 'age', 'dis',
'rad', 'tax', 'ptratio', 'b', 'lstat']]
Y = df['medv']

# 3. Split the data into 80%:20% Ratio.

X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
test_size=0.2, random_state=12)

# 4. Use the 80% data to fit the regression function.

model = LinearRegression()
model.fit(X_train, Y_train)
```

```
LinearRegression()
```

```python
# 5. Use remaining 20% data for the testing.
Y_pred = model.predict(X_test)

# 6. Find the Squared R and MSE.

print('R2 Score:', r2_score(Y_test, Y_pred))
print('Mean Squared Error:', mean_squared_error(Y_test, Y_pred))
```

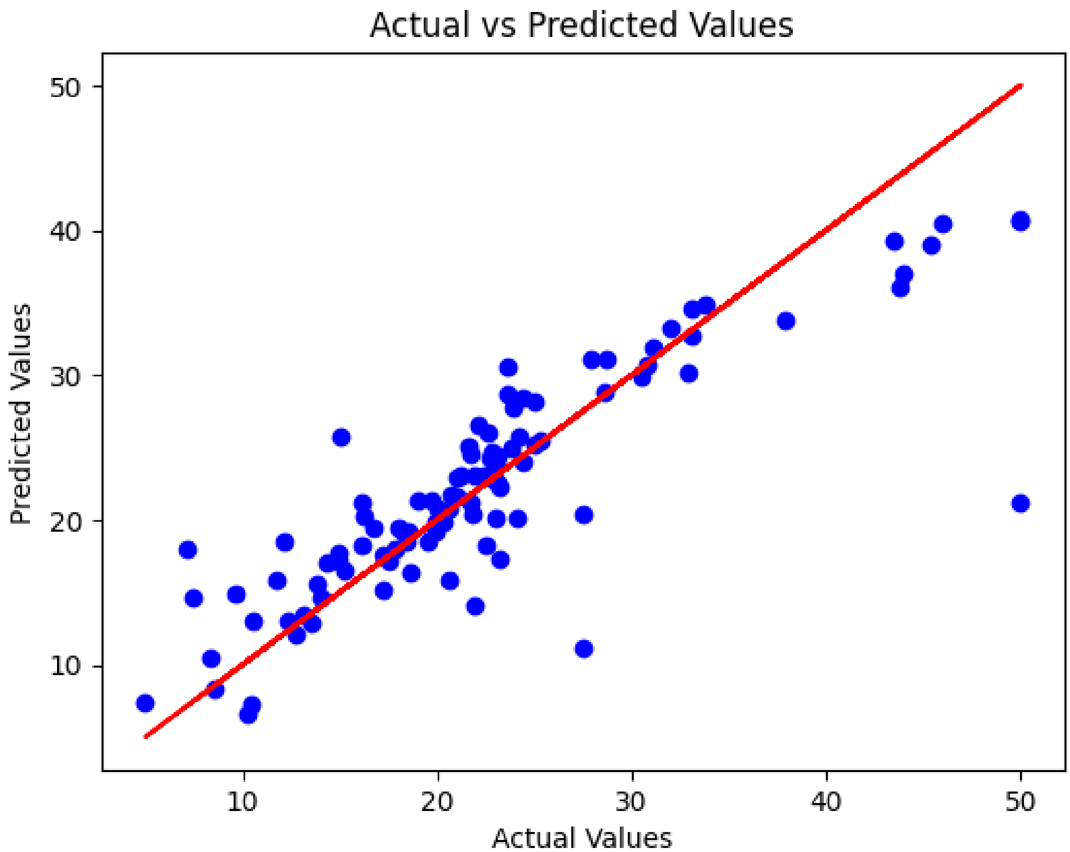
```
R2 Score: 0.7332757970303041
Mean Squared Error: 24.092041824144875
```

```python
# 8. Find the P-Value against each variable.

X2 = sm.add_constant(X)
est = sm.OLS(Y, X2)
```

```
est2 = est.fit()
print(est2.summary())

                            OLS Regression Results

==============================================================================
========
Dep. Variable:                   medv   R-squared:
0.741
Model:                            OLS   Adj. R-squared:
0.735
Method:                 Least Squares   F-statistic:
107.4
Date:                Wed, 27 Mar 2024   Prob (F-statistic):
8.84e-134
Time:                        13:00:36   Log-Likelihood:
-1485.1
No. Observations:                 501   AIC:
2998.
Df Residuals:                     487   BIC:
3057.
Df Model:                          13

Covariance Type:            nonrobust

==============================================================================
========
                 coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
--------
const         37.0421      5.154      7.187      0.000      26.915
47.169
crim          -0.1086      0.033     -3.296      0.001      -0.173
-0.044
zn             0.0455      0.014      3.295      0.001       0.018
0.073
indus          0.0140      0.062      0.226      0.821      -0.108
0.136
chas           2.6673      0.864      3.087      0.002       0.970
4.365
nox          -18.0847      3.837     -4.713      0.000     -25.624
-10.545
rm             3.7811      0.421      8.983      0.000       2.954
4.608
age            0.0020      0.013      0.148      0.882      -0.024
0.028
dis           -1.4814      0.203     -7.305      0.000      -1.880
-1.083
rad            0.3059      0.067      4.579      0.000       0.175
```

```
0.437
tax             -0.0122      0.004     -3.235      0.001      -0.020
-0.005
ptratio         -0.9662      0.132     -7.306      0.000      -1.226
-0.706
b                0.0093      0.003      3.460      0.001       0.004
0.015
lstat           -0.5236      0.051    -10.261      0.000      -0.624
-0.423
======================================================================
========
Omnibus:                      176.092   Durbin-Watson:
1.083
Prob(Omnibus):                  0.000   Jarque-Bera (JB):
769.213
Skew:                           1.520   Prob(JB):
9.28e-168
Kurtosis:                       8.254   Cond. No.
1.51e+04
======================================================================
========

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
[2] The condition number is large, 1.51e+04. This might indicate that
there are
strong multicollinearity or other numerical problems.
```

```python
import matplotlib.pyplot as plt

# Scatter plot of actual test values
plt.scatter(Y_test, Y_pred, color='blue')

# Regression line
plt.plot(Y_test, Y_test, color='red')

plt.title('Actual vs Predicted Values')
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')

plt.show()
```

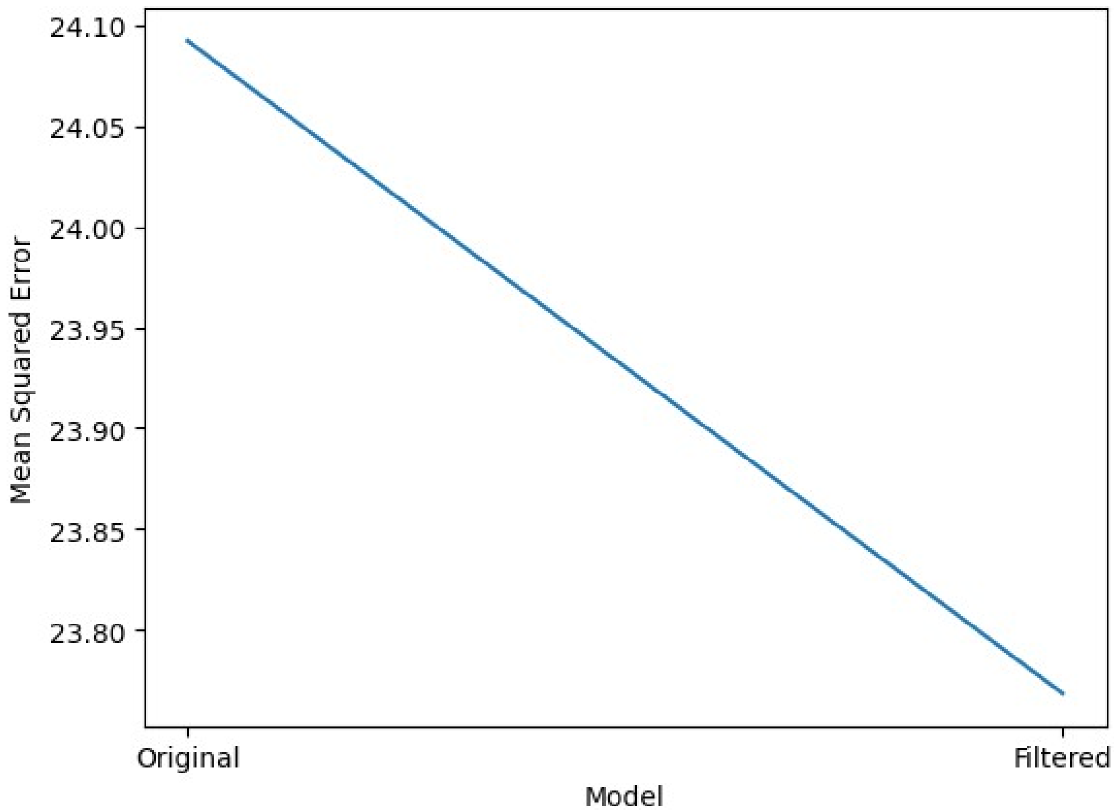



```
# 9. Remove variables having with P>0.005 and do the  multi regression
on remaining attributes.


# Get p-values
p_values = est2.pvalues

# Get variables with P>0.005
vars_with_p_greater_than_005 = p_values[p_values > 0.005].index

# Remove the constant
vars_with_p_greater_than_005 = [var for var in
vars_with_p_greater_than_005 if var != 'const']

# Drop these variables from X
X_filtered = X.drop(vars_with_p_greater_than_005, axis=1)

# Split the data again
X_train_filtered, X_test_filtered, Y_train, Y_test =
train_test_split(X_filtered, Y, test_size=0.2, random_state=12)

# Fit the model again
model_filtered = LinearRegression()
```

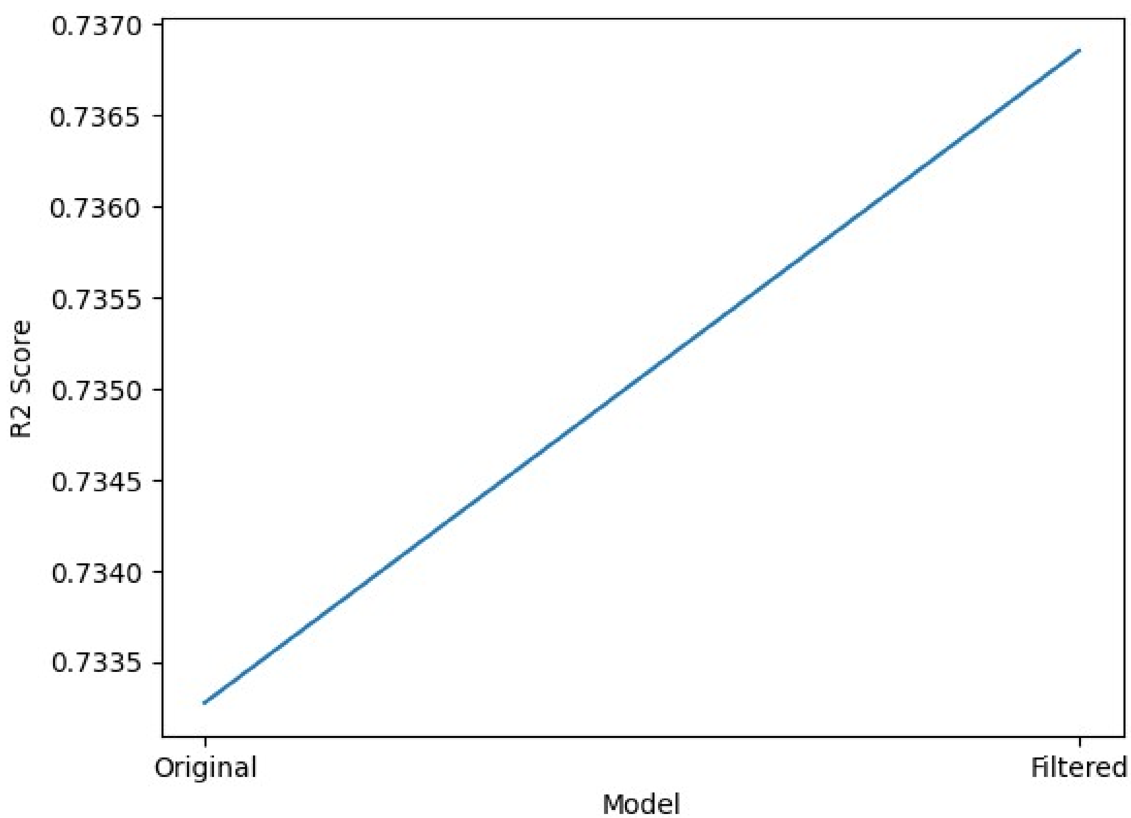
```
model_filtered.fit(X_train_filtered, Y_train)

# Make predictions
Y_pred_filtered = model_filtered.predict(X_test_filtered)

# Print R2 Score and MSE
print('Filtered R2 Score:', r2_score(Y_test, Y_pred_filtered))
print('Filtered Mean Squared Error:', mean_squared_error(Y_test,
Y_pred_filtered))
```

```
Filtered R2 Score: 0.7368557200229189
Filtered Mean Squared Error: 23.768682888191492
```

```
# 11. Plot all MSE, and Squared R line graph and submit the resultant
Jupyter notebook along with a seperate PDF file of the results and
submit.

plt.plot(['Original', 'Filtered'], [mean_squared_error(Y_test,
Y_pred), mean_squared_error(Y_test, Y_pred_filtered)])
plt.xlabel('Model')
plt.ylabel('Mean Squared Error')
plt.show()

plt.plot(['Original', 'Filtered'], [r2_score(Y_test, Y_pred),
r2_score(Y_test, Y_pred_filtered)])
plt.xlabel('Model')
plt.ylabel('R2 Score')
plt.show()
```



24.10
24.05
24.00
23.95
23.90
23.85
23.80
Mean Squared Error
Original
Filtered
Model



0.7370
0.7365
0.7360
0.7355
0.7350
0.7345
0.7340
0.7335
R2 Score
Original
Filtered
Model