Multiple Linear Regression Case Study: Boston Housing Prices

Background: The data has 506 cases where each case is a location in Boston. The “median housing price” is a target variable. The data has many other variables related to environment,education,,crime etc.which can influence the housing prices in the specific location

The objective is to identify significant factors affecting housing prices.

Import data and display first 6 rows

```
import os
import pandas as pd


os.chdir("C:/Sankhya")
boston = pd.read_csv('Housing Prices.csv')
boston.head(6)
```

```
      CRIM    ZN  INDUS  CHAS    NOX  ...  RAD  TAX  PTRATIO  LSTAT  MEDV
0  0.00632  18.0   2.31     0  0.538  ...    1  296     15.3   4.98  24.0
1  0.02731   0.0   7.07     0  0.469  ...    2  242     17.8   9.14  21.6
2  0.02729   0.0   7.07     0  0.469  ...    2  242     17.8   4.03  34.7
3  0.03237   0.0   2.18     0  0.458  ...    3  222     18.7   2.94  33.4
4  0.06905   0.0   2.18     0  0.458  ...    3  222     18.7   5.33  36.2
5  0.02985   0.0   2.18     0  0.458  ...    3  222     18.7   5.21  28.7

[6 rows x 13 columns]
```

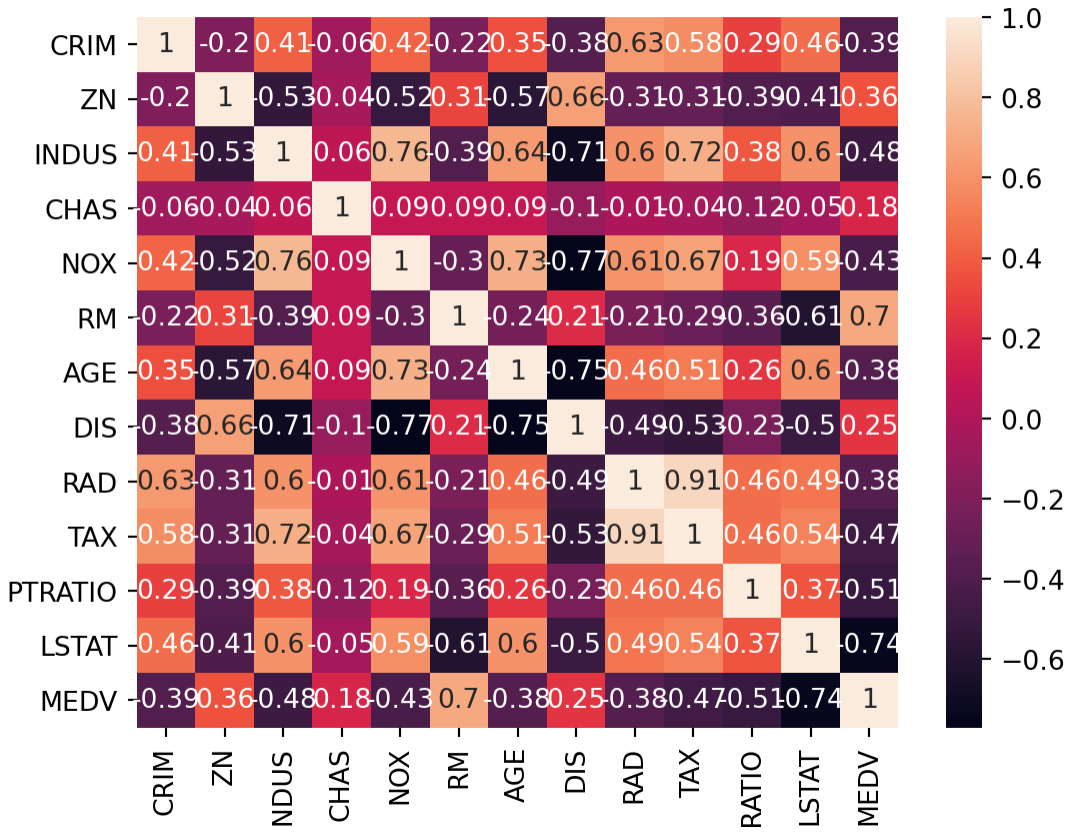
Data Description

Column name	Column description
CRIM	per capita crime rate by town
ZN	proportion of residential land zoned for lots over 25,000 in sq ft
INDUS	proportion of non-retail business acres per town %
CHAS	Charles River dummy variable (=1 if tract bounds river;0 otherwise)

Column name	Column description
NOX	nitric oxides concentration (parts per 10 million)
RM	average number of rooms per dwelling
AGE	proportion of owner-occupied units built prior to 1940 %
DIS	weighted distances to five Boston employment centres
RAD	index of accessibility to radial highways
TAX	full-value property-tax rate per $10,000
PTRATIO	pupil-teacher ratio by town
LSAT	% lower status of the population
MEDV	Median value of owner-occupied homes in $1000s

Correlation matrix using heatmap

```
import seaborn as sns
correlation_matrix = boston.corr().round(2)
sns.heatmap(data=correlation_matrix, annot=True)
```



Check for multicollinearity using vif

```
from patsy import dmatrices
from statsmodels.stats.outliers_influence import variance_inflation_factor

y, X = dmatrices('MEDV~CRIM+ZN+INDUS+CHAS+NOX+RM+AGE+DIS+RAD+TAX+PTRATIO+LSTAT', data=boston,
return_type="dataframe")

vif = pd.Series([variance_inflation_factor(X.values, i)for i in range(X.shape[1])],index=X.co
lumns)
vif
```

```
Intercept    535.526619
CRIM           1.767486
ZN             2.298459
INDUS          3.987181
CHAS           1.071168
NOX            4.369093
RM             1.912532
AGE            3.088232
DIS            3.954037
RAD            7.445301
TAX            9.002158
PTRATIO        1.797060
LSTAT          2.870777
dtype: float64
```

Comment: It is observed that variable TAX has high vif

The variable “TAX” is excluded

```
boston = boston.drop(['TAX'],1)
```

Check for multicollinearity again

```
y2, X2 = dmatrices('MEDV~CRIM+ZN+INDUS+CHAS+NOX+RM+AGE+DIS+RAD+PTRATIO+LSTAT', data=boston, r
eturn_type="dataframe")

vif = pd.Series([variance_inflation_factor(X2.values, i)for i in range(X2.shape[1])],index=X
2.columns)
vif
```

```
Intercept    529.480235
CRIM           1.767349
ZN             2.184172
INDUS          3.217951
CHAS           1.055023
NOX            4.343300
RM             1.902642
AGE            3.085756
DIS            3.952445
RAD            2.772208
PTRATIO        1.787049
LSTAT          2.870408
dtype: float64
```

The multicollinearity problem is resolved as all VIF's are less than 5

Develop Housing Prices Model using Linear Regression

Split original data into training and testing data sets

```
from sklearn.model_selection import train_test_split
boston_train, boston_test = train_test_split(boston, test_size=0.2, random_state=42)
```

Running Linear Regression model using OLS function from statsmodels library

```
import statsmodels.formula.api as smf

hp_model = smf.ols('MEDV~CRIM+ZN+INDUS+CHAS+NOX+RM+AGE+DIS+RAD+PTRATIO+LSTAT',data=boston_tra
in).fit()
```

Displaying model summary

```
print(hp_model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   MEDV   R-squared:                       0.735
Model:                            OLS   Adj. R-squared:                  0.728
Method:                 Least Squares   F-statistic:                     98.99
Date:                Tue, 19 Dec 2023   Prob (F-statistic):          7.75e-106
Time:                        10:55:50   Log-Likelihood:                -1206.6
No. Observations:                 404   AIC:                             2437.
Df Residuals:                     392   BIC:                             2485.
Df Model:                          11
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     35.2203      5.555      6.340      0.000      24.299      46.141
CRIM          -0.1266      0.036     -3.556      0.000      -0.197      -0.057
ZN             0.0189      0.016      1.196      0.232      -0.012       0.050
INDUS         -0.0452      0.064     -0.712      0.477      -0.170       0.080
CHAS           3.3202      0.975      3.406      0.001       1.404       5.237
NOX          -19.3122      4.327     -4.463      0.000     -27.819     -10.805
RM             4.3369      0.473      9.168      0.000       3.407       5.267
AGE           -0.0034      0.015     -0.225      0.822      -0.033       0.026
DIS           -1.4400      0.232     -6.205      0.000      -1.896      -0.984
RAD            0.0794      0.047      1.687      0.092      -0.013       0.172
PTRATIO       -0.9248      0.148     -6.269      0.000      -1.215      -0.635
LSTAT         -0.5345      0.057     -9.412      0.000      -0.646      -0.423
==============================================================================
Omnibus:                      122.472   Durbin-Watson:                   2.171
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              473.320
Skew:                           1.302   Prob(JB):                    1.66e-103
Kurtosis:                       7.620   Cond. No.                     2.06e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.06e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Comment: From the model summary, it can be seen that the variables ZN, INDUS,RAD and AGE are insignificant(since p value > 0.05), so we remove them from the model and run the model again.

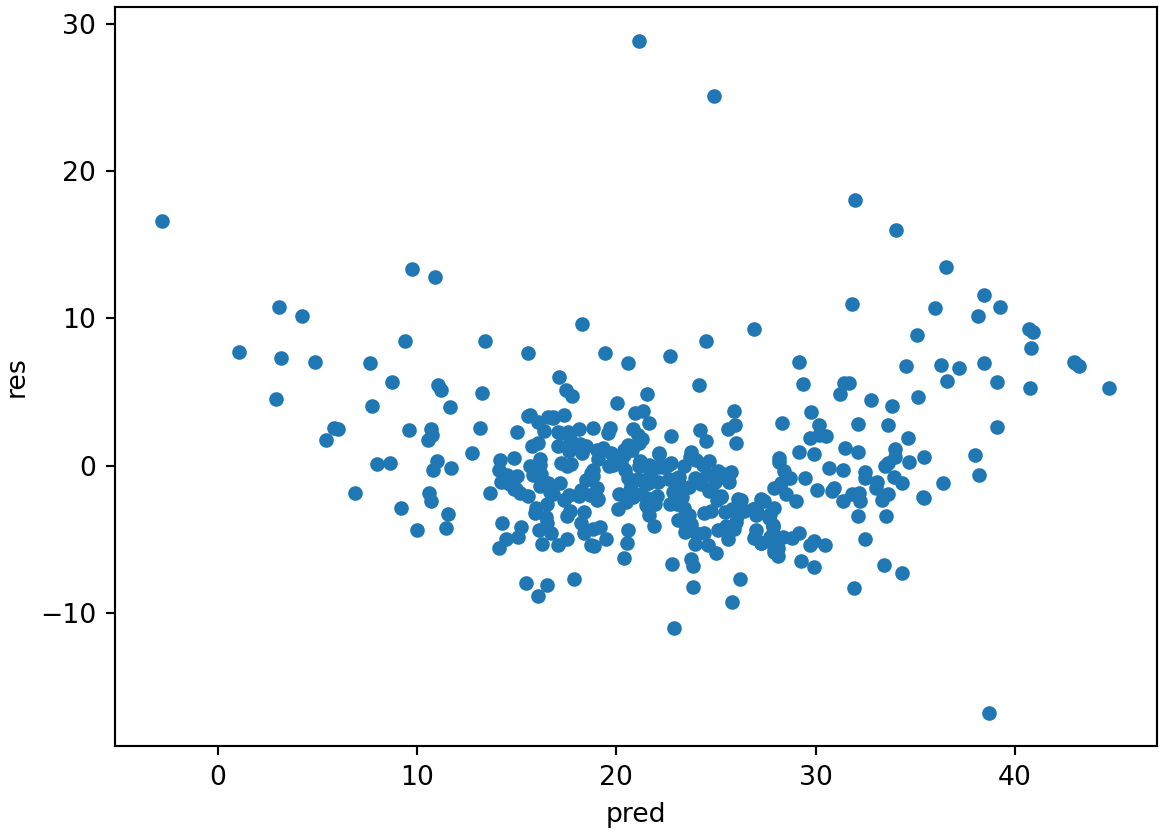
```
hp_model = smf.ols('MEDV~CRIM+CHAS+NOX+RM+DIS+PTRATIO+LSTAT',data=boston_train).fit()
print(hp_model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   MEDV   R-squared:                       0.732
Model:                            OLS   Adj. R-squared:                  0.727
Method:                 Least Squares   F-statistic:                     154.2
Date:                Tue, 19 Dec 2023   Prob (F-statistic):          6.26e-109
Time:                        10:55:51   Log-Likelihood:                -1209.4
No. Observations:                 404   AIC:                             2435.
Df Residuals:                     396   BIC:                             2467.
Df Model:                           7
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     31.8452      5.073      6.278      0.000      21.872      41.818
CRIM          -0.0925      0.031     -2.938      0.003      -0.154      -0.031
CHAS           3.3351      0.974      3.425      0.001       1.421       5.249
NOX          -18.2416      3.613     -5.049      0.000     -25.344     -11.139
RM             4.5754      0.447     10.225      0.000       3.696       5.455
DIS           -1.2243      0.184     -6.660      0.000      -1.586      -0.863
PTRATIO       -0.8962      0.122     -7.317      0.000      -1.137      -0.655
LSTAT         -0.5316      0.054     -9.842      0.000      -0.638      -0.425
==============================================================================
Omnibus:                      137.650   Durbin-Watson:                   2.160
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              610.512
Skew:                           1.426   Prob(JB):                    2.69e-133
Kurtosis:                       8.304   Cond. No.                         558.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

Plot of Residuals vs Predicted values

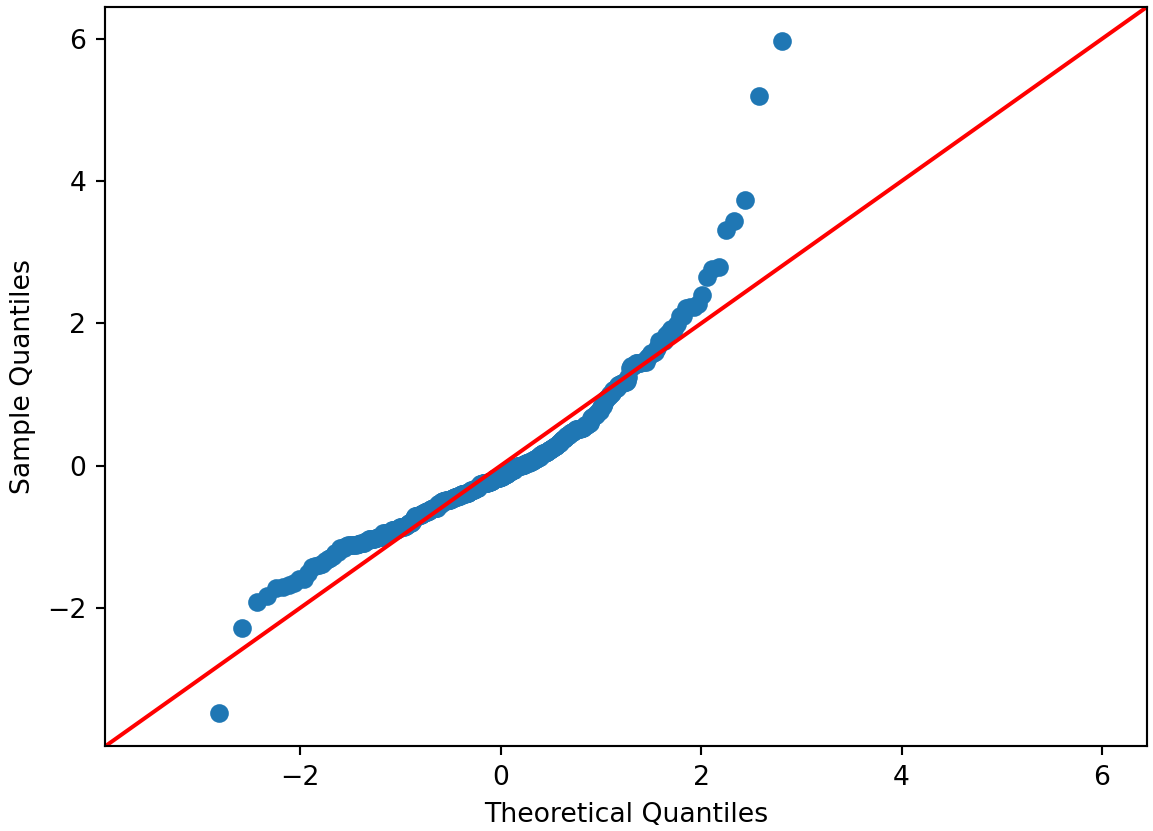
```
boston_train['pred'] = hp_model.predict(boston_train)
boston_train['res']=hp_model.resid
boston_train.plot.scatter(x='pred', y='res')
```



Comment: It is observed that residuals are randomly distributed and uncorelated with predicted values

Check if distribution of errors is “NORMAL”

```
import statsmodels.api as sm
sm.graphics.qqplot(boston_train.res, line='45', fit=True)
```




```
import scipy as sp
sp.stats.shapiro(boston_train.res)
```

```
ShapiroResult(statistic=0.9116807579994202, pvalue=1.2912867356060145e-14)
```

Comment: Although normality of errors is not established we will proceed to evaluate the model performance

Model Validation: Holdout Method using RMSE

```
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
import numpy as np

rmse = (np.sqrt(mean_squared_error(boston_train['MEDV'], boston_train['pred'])))
r2 = r2_score(boston_train['MEDV'], boston_train['pred'])

print('RMSE is {}'.format(rmse))
```

```
RMSE is 4.828905476132975
```

```
print('R2 score is {}'.format(r2))
```

```
R2 score is 0.7315826585744917
```

```
y_test_predict = hp_model.predict(boston_test)
rmse = (np.sqrt(mean_squared_error(boston_test['MEDV'], y_test_predict)))
r2 = r2_score(boston_test['MEDV'], y_test_predict)

print('RMSE is {}'.format(rmse))
```

```
RMSE is 5.10498883392664
```

```
print('R2 score is {}'.format(r2))
```

```
R2 score is 0.6446261208488169
```

K-fold cross validation

```
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.linear_model import LinearRegression


X = boston.drop(labels=['MEDV','ZN', 'INDUS', 'RAD', 'AGE'], axis=1)
Y = boston['MEDV']
hp_model1 = LinearRegression()


folds = KFold(n_splits = 4, shuffle = True, random_state = 100)
scores = cross_val_score(hp_model1, X, Y, scoring='r2', cv=folds)


print("Mean 4-Fold R Squared: {}".format(np.mean(scores)))
```

```
Mean 4-Fold R Squared: 0.7073913275172319
```

```
cv_rmse_scores= cross_val_score(hp_model1, X, Y, cv=folds, scoring='neg_mean_squared_error')
np.sqrt(-(np.mean(cv_rmse_scores)))
```

```
4.971825927756832
```

Comment: RMSE and R squared values using K-fold validation are similar to overall RMSE and R squared values

The model can be implemented for decision making