DSBDAL Assignment 08 - Data Analytics 1

Create a Linear Regression Model using Python/R to predict home prices using Boston Housing Dataset (https://www.kaggle.com/c/boston-housing). The Boston Housing dataset contains information about various houses in Boston through different parameters. There are 506 samples and 14 feature variables in this dataset.

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
import numpy as np
import pandas as pd
import seaborn as sns
```

```
ds = pd.read_csv('/content/drive/My Drive/DSBDL/Assignment8/boston_housing.csv')
ds
```

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	b	l
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	
...	...	...	...	...	...	...	...	...	...	...	...	...	
501	0.06263	0.0	11.93	0	0.573	6.593	69.1	2.4786	1	273	21.0	391.99	
502	0.04527	0.0	11.93	0	0.573	6.120	76.7	2.2875	1	273	21.0	396.90	
503	0.06076	0.0	11.93	0	0.573	6.976	91.0	2.1675	1	273	21.0	396.90	
504	0.10959	0.0	11.93	0	0.573	6.794	89.3	2.3889	1	273	21.0	393.45	
505	0.04741	0.0	11.93	0	0.573	6.030	80.8	2.5050	1	273	21.0	396.90	

506 rows × 14 columns

Next steps: Generate code with ds | View recommended plots

About the Dataset

- CRIM - per capita crime rate by town
- ZN - proportion of residential land zoned for lots over 25,000 sq.ft.
- INDUS - proportion of non-retail business acres per town.
- CHAS - Charles River dummy variable (1 if tract bounds river; 0 otherwise)
- 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
- B - 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
- LSTAT - % lower status of the population
- MEDV - Median value of owner-occupied homes in $1000's

```
ds.columns
```

```
Index(['crim', 'zn', 'indus', 'chas', 'nox', 'rm', 'age', 'dis', 'rad', 'tax',
       'ptratio', 'b', 'lstat', 'medv'],
```

```
          dtype='object')
```

```
ds.dtypes
```

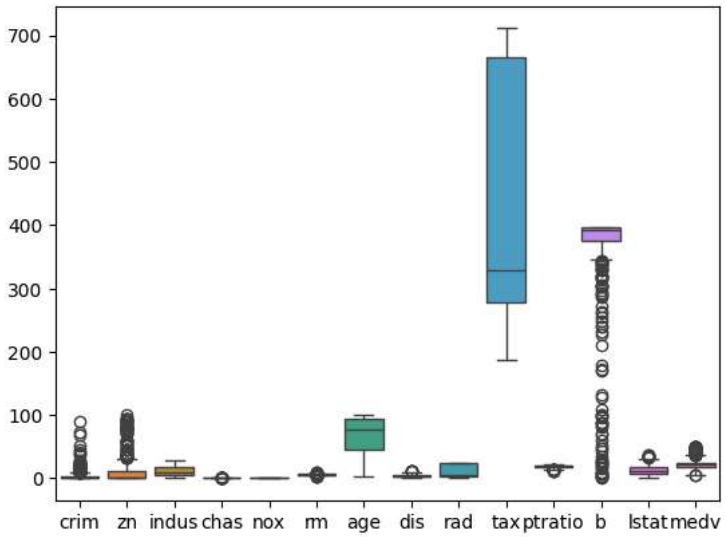
```
crim       float64
zn         float64
indus      float64
chas         int64
nox        float64
rm         float64
age        float64
dis        float64
rad          int64
tax          int64
ptratio    float64
b          float64
lstat      float64
medv       float64
dtype: object
```

```
ds.describe()
```

	crim	zn	indus	chas	nox	rm	ag
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.00000
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.57490
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.14886
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.90000
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.02500
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.50000
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.07500
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.00000

```
sns.boxplot( data=ds )
```

```
<Axes: >
```

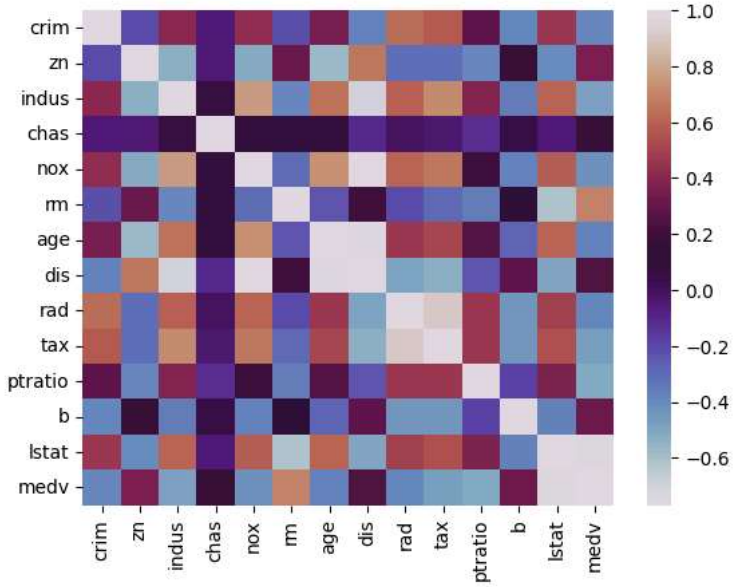



```
ds.corr()
```

	crim	zn	indus	chas	nox	rm	age	d
crim	1.000000	-0.200469	0.406583	-0.055892	0.420972	-0.219247	0.352734	-0.3796
zn	-0.200469	1.000000	-0.533828	-0.042697	-0.516604	0.311991	-0.569537	0.6644
indus	0.406583	-0.533828	1.000000	0.062938	0.763651	-0.391676	0.644779	-0.7080
chas	-0.055892	-0.042697	0.062938	1.000000	0.091203	0.091251	0.086518	-0.0991
nox	0.420972	-0.516604	0.763651	0.091203	1.000000	-0.302188	0.731470	-0.7692
rm	-0.219247	0.311991	-0.391676	0.091251	-0.302188	1.000000	-0.240265	0.2052
age	0.352734	-0.569537	0.644779	0.086518	0.731470	-0.240265	1.000000	-0.7478
dis	-0.379670	0.664408	-0.708027	-0.099176	-0.769230	0.205246	-0.747881	1.0000
rad	0.625505	-0.311948	0.595129	-0.007368	0.611441	-0.209847	0.456022	-0.4945
tax	0.582764	-0.314563	0.720760	-0.035587	0.668023	-0.292048	0.506456	-0.5344
ptratio	0.289946	-0.391679	0.383248	-0.121515	0.188933	-0.355501	0.261515	-0.2324
b	-0.385064	0.175520	-0.356977	0.048788	-0.380051	0.128069	-0.273534	0.2915
lstat	0.455621	-0.412995	0.603800	-0.053929	0.590879	-0.613808	0.602339	-0.4969
medv	-0.388305	0.360445	-0.483725	0.175260	-0.427321	0.695360	-0.376955	0.2499

```
sns.heatmap( ds.corr() , cmap="twilight" )
```

```
<Axes: >
```




```
ds.skew()
```

```
crim       5.223149
zn         2.225666
indus      0.295022
chas       3.405904
nox        0.729308
rm         0.403612
age       -0.598963
dis        1.011781
rad        1.004815
tax        0.669956
ptratio   -0.802325
b         -2.890374
lstat      0.906460
medv       1.108098
dtype: float64
```

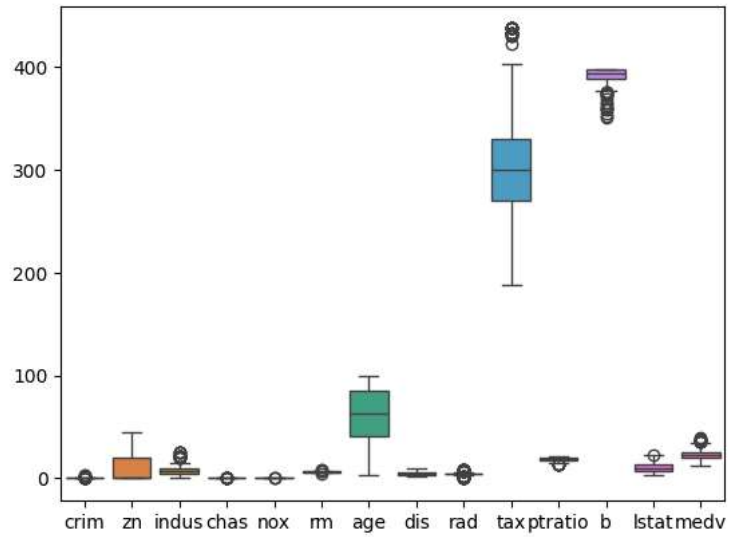
```
def remove_outliers(
    feature_name: str
):
    global ds
    q3 , q1 = np.percentile( ds[ feature_name ] , [ 75 , 25 ] )
    iqr = q3 - q1
    ds = ds[ (ds[ feature_name ] >= q1 - 1.5 * iqr) & (ds[ feature_name ] <= q3 + 1.5 * iqr) ]

remove_outliers( "crim" )
remove_outliers( "zn" )
remove_outliers( "b" )
remove_outliers( "rad" )
remove_outliers( "tax" )
remove_outliers( "lstat" )
remove_outliers( "medv" )
```

```
ds.describe()
```

	crim	zn	indus	chas	nox	rm	ag
count	267.000000	267.000000	267.000000	267.000000	267.000000	267.000000	267.00000
mean	0.269166	7.913858	8.254794	0.063670	0.504317	6.301839	61.38651
std	0.376737	13.134000	5.206476	0.244623	0.071415	0.493938	26.31628
min	0.006320	0.000000	1.250000	0.000000	0.409000	4.973000	2.90000
25%	0.069020	0.000000	4.930000	0.000000	0.447000	5.955500	41.10000
50%	0.131170	0.000000	6.910000	0.000000	0.493000	6.182000	62.80000
75%	0.286760	20.000000	9.900000	0.000000	0.538000	6.582500	84.55000
max	2.733970	45.000000	25.650000	1.000000	0.871000	8.069000	100.00000

```
sns.boxplot( data=ds )
```

```
<Axes: >
```



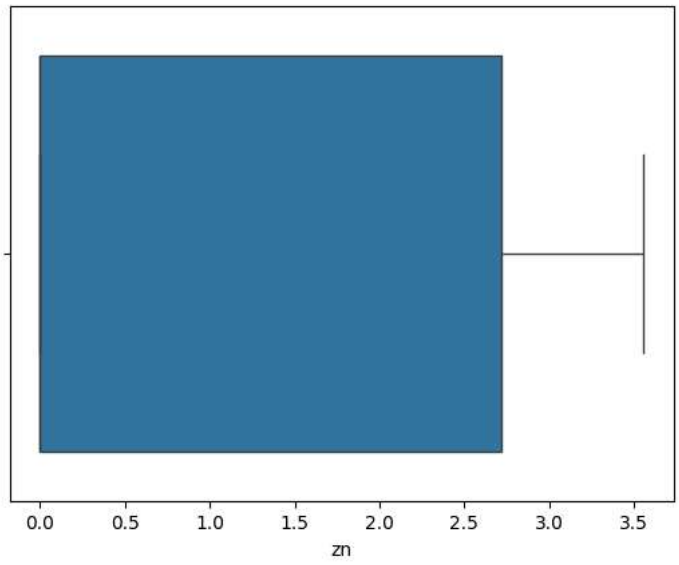

```
ds.skew()
```

```
crim       3.457800
zn         1.372669
indus      1.554100
chas       3.594281
nox        1.776428
rm         0.769068
age       -0.304989
dis        0.575180
rad        0.173594
tax        0.466026
ptratio   -0.538715
b         -2.052233
lstat      0.636089
medv       0.800355
dtype: float64
```

```
ds[ "crim" ] = np.log2( ds["crim"] )
ds[ "zn" ] = np.cbrt( ds["zn"] )
ds[ "nox" ] = np.log2( ds["nox"] )
```

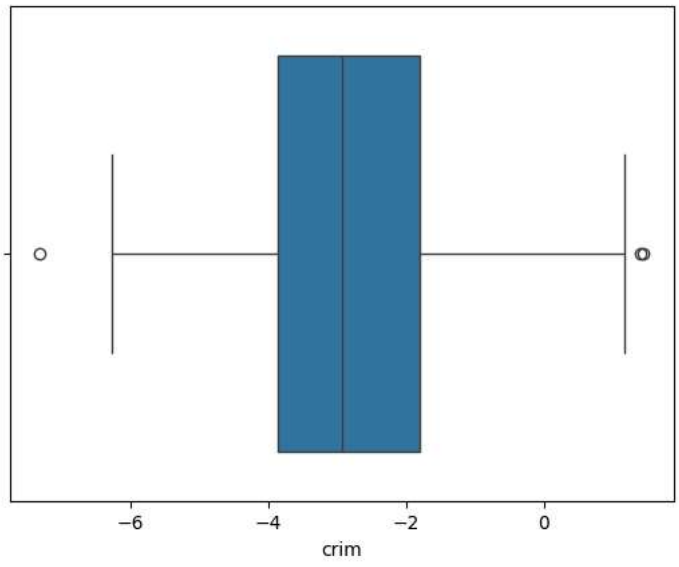
```
sns.boxplot( data=ds , x="zn" )
```

```
<Axes: xlabel='zn'>
```



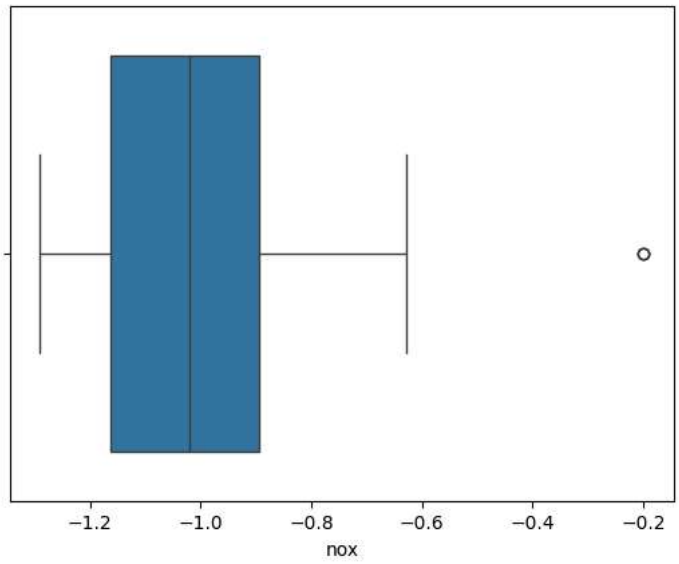

```
sns.boxplot( data=ds , x="crim" )
```

```
<Axes: xlabel='crim'>
```



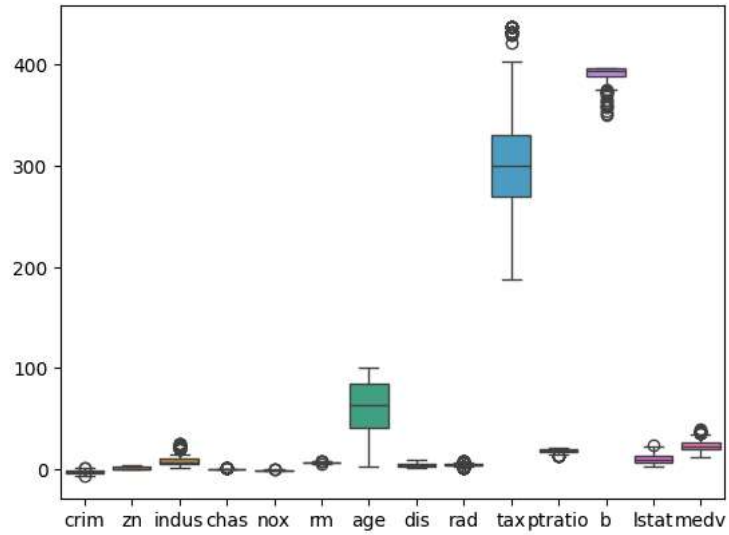

```
sns.boxplot( data=ds , x="nox" )
```

```
<Axes: xlabel='nox'>
```




```
sns.boxplot( data=ds )
```

```
<Axes: >
```




```
ds.skew()
```

```
crim       0.381947
zn         0.943741
indus      1.554100
chas       3.594281
nox        1.042174
rm         0.769068
age       -0.304989
dis        0.575180
rad        0.173594
tax        0.466026
ptratio   -0.538715
b         -2.052233
lstat      0.636089
medv       0.800355
dtype: float64
```

```
def min_max_normalize( name: str ):
    ds[ name ] = (ds[ name ] - ds[ name ].min()) / ( ds[ name ].max() - ds[ name ].min() )

for col in ds.drop( [ "medv" ] , axis=1 ).columns:
    min_max_normalize( col )


from sklearn.model_selection import train_test_split

X = np.asarray( ds.drop( [ "medv" ] , axis=1 ) )
y = np.asarray( ds[ "medv" ] )

X_train, X_test, y_train, y_test = train_test_split( X, y, test_size=0.3 )
```

```
from sklearn.linear_model import LinearRegression

reg = LinearRegression().fit( X_train , y_train )
y_pred = reg.predict( X_test )


sns.boxplot( data=ds , x="medv" )
```

```
<Axes: xlabel='medv'>
```