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.info()
```

```
     <class 'pandas.core.frame.DataFrame'>
     RangeIndex: 506 entries, 0 to 505
```

```
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   crim     506 non-null    float64
 1   zn       506 non-null    float64
 2   indus    506 non-null    float64
 3   chas     506 non-null    int64  
 4   nox      506 non-null    float64
 5   rm       506 non-null    float64
 6   age      506 non-null    float64
 7   dis      506 non-null    float64
 8   rad      506 non-null    int64  
 9   tax      506 non-null    int64  
 10  ptratio  506 non-null    float64
 11  b        506 non-null    float64
 12  lstat    506 non-null    float64
 13  medv     506 non-null    float64
dtypes: float64(11), int64(3)
memory usage: 55.5 KB
```

```
ds.describe()
```

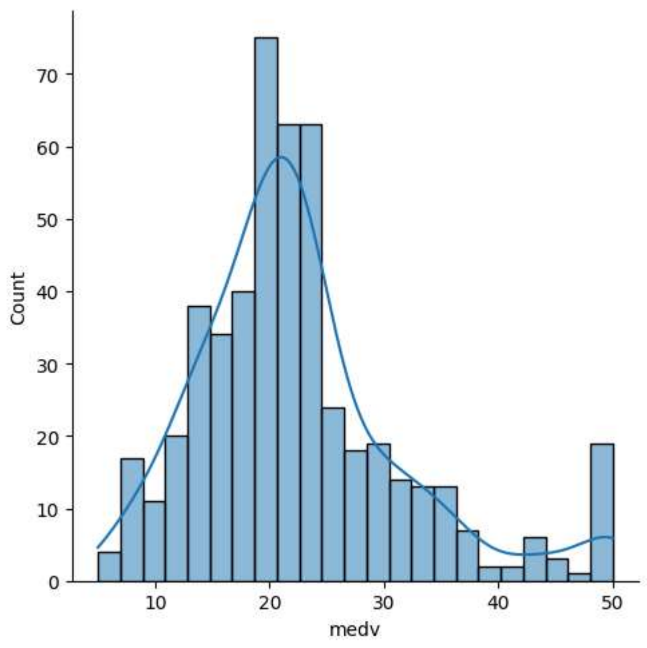
	crim	zn	indus	chas	nox	rm	ag
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.00000
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.57490
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.14886
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	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

```
ds.isna().sum()
```

```
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
```

```
sns.displot(ds.medv, kde = True)
```

```
<seaborn.axisgrid.FacetGrid at 0x7e7b66a09c60>
```



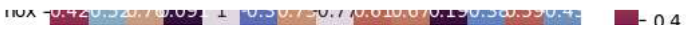
```
ds.corr()
```

	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', annot = True)
```

```
<Axes: >
```

```
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, random_state = 42)
```



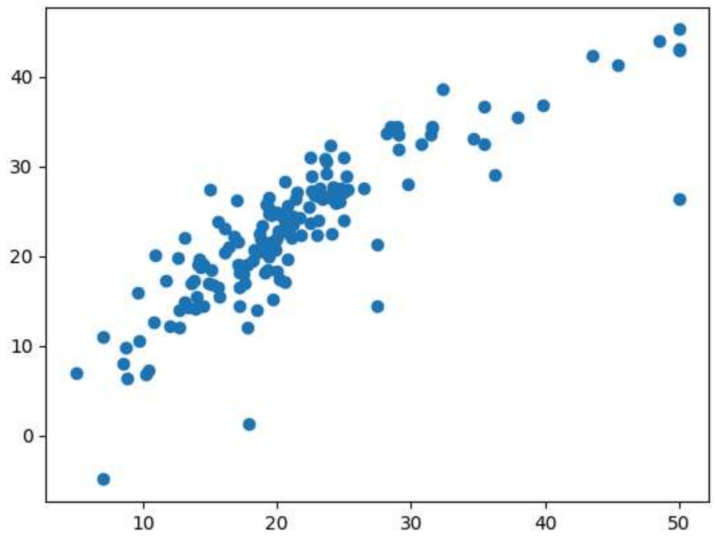
```
from sklearn.preprocessing import StandardScaler

X_train = StandardScaler().fit_transform(X_train)
X_test = StandardScaler().fit_transform(X_test)
```

```
from sklearn.linear_model import LinearRegression

regressor = LinearRegression()
regressor.fit(X_train, y_train)
y_pred = regressor.predict(X_test)
print(y_pred)
```

```
[30.77834822 38.66581541 16.96705276 26.79035818 20.36068008 24.89386356
 18.98948959 15.44554414 24.50205701 22.12455513 27.16709892 20.60218969
 -4.91312377 23.45520373 20.44300739 27.97454642 21.82481134  6.80793603
 43.13876668 19.08804672 28.93737925 31.91943808 12.05148619 25.41493356
 19.60614164 17.23543021 24.61263361 16.907528   24.25623469 20.73183868
 23.98650325 26.66317526 27.45468242 19.57475267 18.15660164 19.94422395
 33.04896681 21.26475982 25.83570427 26.36436267 15.22545112 33.61610719
 45.24385042 18.83620774 28.91584593 18.49470905 14.91054896 27.65199578
 21.75498681 32.36964545 23.36168054 36.5955736  16.82098601 27.59874838
 42.39261672 24.39945725 20.42785358 34.46608842 26.52068304 14.0304141
 24.03170468 32.40960974 33.59067192 17.33583061 22.46180876 18.28308242
 22.01801285 27.57486429 32.54366102 13.95067422 21.96577449 29.21079363
 12.65101384 17.15704754 25.56951945  6.99478235 22.73334313 43.93983367
 20.03571324 10.919632   22.50309934 14.46771561 23.07376266 10.52688537
 24.71136556 34.38262811 21.02382869 27.04400189 30.94450114 21.54068028
 27.48732329  7.31708375 21.56254771 16.57285469 14.43338806 22.26113509
 26.42907376  1.28686278 15.98179144 18.13930581 23.62097091 26.3208465
 12.12867204 20.66733158 25.32719152 14.23008516 19.48076211 26.9289352
 22.02461937 26.11181168  9.83654844 21.30376854 23.13155127 28.99045653
 34.46072055 17.31422502 36.81803955 14.0639201  22.38719669 30.47720506
 16.96724187 26.23079837  6.30719991 25.6891189  27.44337177 24.41723783
 26.72754089 35.52652858 23.8342233  41.30725508 15.51580837 27.19720138
 18.99437335 22.50080685 11.95557604 23.13463621 23.4954008  34.27666272
 33.53948706 16.55984176 17.98511069 31.00757978 26.51337445 18.4765867
  7.99167823 28.32517448 26.1665569  19.032183   14.47502635 42.88762567
 18.69947012 19.82530166]
```

```
import matplotlib.pyplot as plt

plt.scatter(y_test, y_pred)
```

```
<matplotlib.collections.PathCollection at 0x7e7b600ac730>
```



```
from sklearn.metrics import mean_squared_error, r2_score

print("RMSE:  ", np.sqrt(mean_squared_error(y_test, y_pred)))
print("R2 Score:  ", r2_score(y_test, y_pred))
```

```
RMSE:   4.912717301969203
R2 Score:   0.6761000049033603
```