Boston Housing Data

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

%matplotlib inline

sns.set_style('whitegrid')
plt.style.use("fivethirtyeight")
```

```
data = pd.read_csv('train.csv')
```

```
data.head()
```

	ID	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
0	0	0.95577	0.0	8.14	0	0.538	6.047	88.8	4.4534	4	307.0	21.0	306.38	17.28	14.8
1	1	0.02875	28.0	15.04	0	0.464	6.211	28.9	3.6659	4	270.0	18.2	396.33	6.21	25.0
2	2	1.22358	0.0	19.58	0	0.605	6.943	97.4	1.8773	5	403.0	14.7	363.43	4.59	41.3
3	3	5.66637	0.0	18.10	0	0.740	6.219	100.0	2.0048	24	666.0	20.2	395.69	16.59	18.4
4	4	0.04544	0.0	3.24	0	0.460	6.144	32.2	5.8736	4	430.0	16.9	368.57	9.09	19.8

```
data.shape
```

```
(400, 15)
```

```
data.describe().style.background_gradient()
```

	ID	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	
count	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400.0
mean	199.500000	3.757191	10.970000	10.936425	0.072500	0.552817	6.292165	68.086000	3.819462	9.462500	403.7
std	115.614301	9.155496	22.796261	6.848042	0.259639	0.115488	0.709923	28.386888	2.132445	8.687478	169.6
min	0.000000	0.009060	0.000000	0.460000	0.000000	0.385000	4.138000	2.900000	1.129600	1.000000	187.0
25%	99.750000	0.077820	0.000000	5.130000	0.000000	0.449000	5.877500	42.375000	2.109150	4.000000	277.0
50%	199.500000	0.242170	0.000000	8.560000	0.000000	0.532000	6.208500	76.950000	3.272100	5.000000	329.0
75%	299.250000	3.543427	12.500000	18.100000	0.000000	0.624000	6.620500	93.825000	5.214600	24.000000	666.0
max	399.000000	88.976200	95.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.126500	24.000000	711.0

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 15 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   ID       400 non-null    int64  
 1   CRIM     400 non-null    float64
 2   ZN       400 non-null    float64
 3   INDUS    400 non-null    float64
 4   CHAS     400 non-null    int64  
 5   NOX      400 non-null    float64
 6   RM       400 non-null    float64
 7   AGE      400 non-null    float64
 8   DIS      400 non-null    float64
 9   RAD      400 non-null    int64  
 10  TAX      400 non-null    float64
 11  PTRATIO  400 non-null    float64
 12  B        400 non-null    float64
 13  LSTAT    400 non-null    float64
 14  MEDV     400 non-null    float64
dtypes: float64(12), int64(3)
memory usage: 47.0 KB
```

```
data.isnull().sum()
```

```
ID         0
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
```

```
data.duplicated().sum()
```

```
0
```

```
cols = data.columns
cols
```

```
Index(['ID', 'CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD',
       'TAX', 'PTRATIO', 'B', 'LSTAT', 'MEDV'],
      dtype='object')
```

```
for col in cols:
    print(col,data[col].nunique())
```

```
ID 400
CRIM 398
ZN 23
INDUS 72
CHAS 2
NOX 80
RM 362
AGE 296
DIS 339
RAD 9
TAX 63
PTRATIO 44
B 286
LSTAT 365
MEDV 205
```

```
for col in cols:

    print(col,data[col].unique())
```

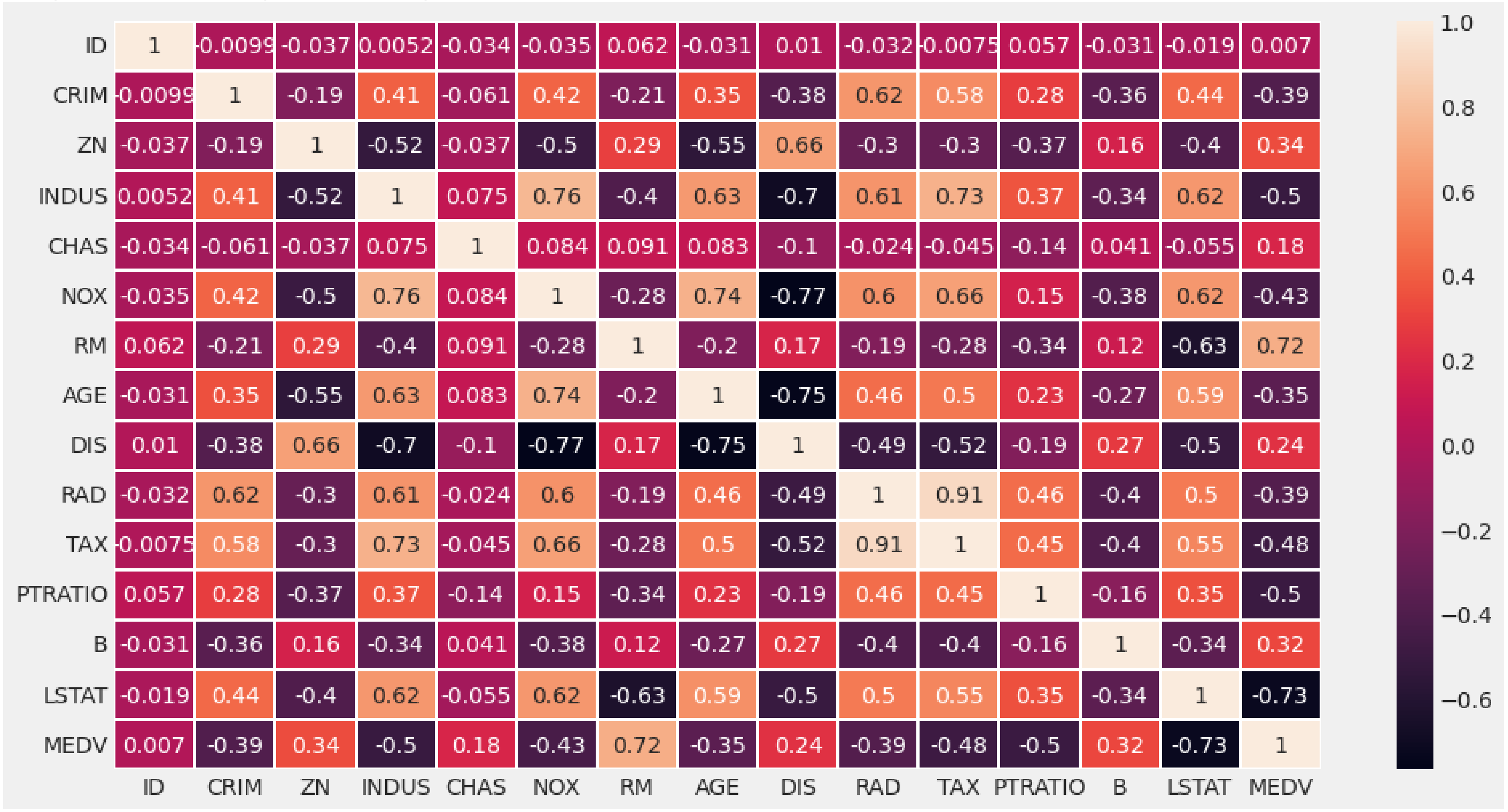
```
ID [  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17
  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35
  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53
  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71
  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89
  90  91  92  93  94  95  96  97  98  99 100 101 102 103 104 105 106 107
 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125
 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143
 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161
 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179
 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197
 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215
 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233
 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251
 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269
 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287
 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305
 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323
```

```
  324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341
  342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359
  360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377
  378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395
  396 397 398 399]
 CRIM [9.55770e-01 2.87500e-02 1.22358e+00 5.66637e+00 4.54400e-02 1.06590e-01
  5.11358e+01 3.32105e+00 1.05393e+00 2.45220e-01 5.78340e-01 3.53700e-02
  3.84970e+00 3.83518e+01 3.58090e-01 1.58603e+01 6.44405e+00 5.82115e+00
  3.50200e-02 1.20482e+01 6.66400e-02 5.37000e-01 5.57780e-01 7.90410e-01
  3.30450e-01 6.58800e-02 7.72990e-01 5.37200e-02 8.26500e-02 1.64390e-01
  3.67367e+00 3.40060e-01 9.74400e-02 6.04700e-02 5.69175e+00 2.43938e+01
  1.40520e-01 5.56100e-02 1.28160e-01 2.37857e+00 6.91100e-02 2.17190e-01
  5.40110e-01 2.29270e-01 5.47900e-02 8.52040e-01 4.93200e-02 8.20058e+00
  8.98296e+00 7.02259e+00 1.36781e+01 2.44668e+00 1.12658e+00 6.21100e-02
  4.83567e+00 2.53870e-01 2.29690e-01 6.26300e-02 1.43900e-02 1.50980e-01
  8.01400e-02 2.25971e+01 2.86558e+01 9.37800e-02 6.71910e-01 3.49400e-01
  2.22120e-01 1.05740e-01 1.32620e-01 6.23560e-01 2.24236e+00 1.50100e-02
  1.17470e-01 1.96091e+01 7.61620e-01 8.26725e+00 2.14090e-01 3.03470e-01
  9.16400e-02 4.52700e-02 1.84982e+01 8.25260e-01 3.93200e-02 5.64400e-02
  1.91330e-01 5.75290e-01 7.85700e-01 1.70900e-02 2.59406e+01 2.00849e+01
  1.87000e-02 2.98500e-02 5.34120e-01 4.46200e-02 1.95390e-01 1.15172e+00
  2.05500e-02 2.63548e+00 4.12380e-01 8.82600e-02 5.73500e-02 2.76300e-02
  1.88360e-01 2.06080e-01 3.65900e-02 6.14700e-01 2.21880e-01 8.89762e+01
  2.48017e+01 8.44700e-02 8.15174e+00 7.01300e-02 7.50300e-02 3.15330e-01
  5.50070e-01 3.58400e-02 1.52880e+01 1.77800e-02 2.77974e+00 9.18702e+00
  1.39134e+01 9.72418e+00 7.87500e-02 3.16360e+00 1.10690e-01 1.74460e-01
  6.63510e-01 1.38100e-02 1.11081e+01 4.11300e-02 1.83377e+00 3.31470e-01
  2.49800e-01 1.28020e-01 3.51140e-01 9.39063e+00 1.43337e+01 3.73800e-02
  8.19900e-02 4.03841e+00 5.44520e-01 1.69020e-01 4.37900e-02 4.64689e+00
  8.24809e+00 3.56868e+00 4.09740e+00 4.89822e+00 1.14600e-01 6.39312e+00
  1.14250e-01 8.37000e-02 1.25790e-01 1.11604e+01 1.10270e-01 2.61690e-01
  3.23700e-02 8.38700e-02 1.96570e-01 2.36862e+00 1.28023e+01 1.20742e+00
  4.29700e-02 8.49213e+00 3.29820e-01 1.51772e+01 3.87100e-02 2.44953e+00
  6.41700e-02 2.51990e-01 1.22040e-01 4.35710e-01 1.78990e-01 1.02330e+01
  4.47910e-01 1.39140e-01 1.13290e-01 9.25200e-02 3.54800e-02 5.58107e+00
  2.24380e-01 7.89600e-02 6.11540e-01 3.82140e-01 3.69695e+00 5.82401e+00
  9.06500e-02 3.15000e-02 3.69311e+00 9.82349e+00 1.96500e-02 2.83920e-01
  2.15505e+00 9.88430e-01 1.71340e-01 7.15100e-02 7.95000e-02 1.25179e+00
  1.31100e-02 3.76800e-02 1.77830e-01 8.71675e+00 5.44114e+00 4.92980e-01
  4.87141e+00 1.26500e-01 3.57800e-02 5.78000e-02 3.30600e-02 5.09017e+00
```

```
plt.figure(figsize=(15,8))

sns.heatmap(data.corr() , annot=True , lw=1 )
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe3fcc79fd0>
```



	ID	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
ID	1	-0.0099	-0.037	0.0052	-0.034	-0.035	0.062	-0.031	0.01	-0.032	-0.0075	0.057	-0.031	-0.019	0.007
CRIM	-0.0099	1	-0.19	0.41	-0.061	0.42	-0.21	0.35	-0.38	0.62	0.58	0.28	-0.36	0.44	-0.39
ZN	-0.037	-0.19	1	-0.52	-0.037	-0.5	0.29	-0.55	0.66	-0.3	-0.3	-0.37	0.16	-0.4	0.34
INDUS	0.0052	0.41	-0.52	1	0.075	0.76	-0.4	0.63	-0.7	0.61	0.73	0.37	-0.34	0.62	-0.5
CHAS	-0.034	-0.061	-0.037	0.075	1	0.084	0.091	0.083	-0.1	-0.024	-0.045	-0.14	0.041	-0.055	0.18
NOX	-0.035	0.42	-0.5	0.76	0.084	1	-0.28	0.74	-0.77	0.6	0.66	0.15	-0.38	0.62	-0.43
RM	0.062	-0.21	0.29	-0.4	0.091	-0.28	1	-0.2	0.17	-0.19	-0.28	-0.34	0.12	-0.63	0.72
AGE	-0.031	0.35	-0.55	0.63	0.083	0.74	-0.2	1	-0.75	0.46	0.5	0.23	-0.27	0.59	-0.35
DIS	0.01	-0.38	0.66	-0.7	-0.1	-0.77	0.17	-0.75	1	-0.49	-0.52	-0.19	0.27	-0.5	0.24
RAD	-0.032	0.62	-0.3	0.61	-0.024	0.6	-0.19	0.46	-0.49	1	0.91	0.46	-0.4	0.5	-0.39
TAX	-0.0075	0.58	-0.3	0.73	-0.045	0.66	-0.28	0.5	-0.52	0.91	1	0.45	-0.4	0.55	-0.48
PTRATIO	0.057	0.28	-0.37	0.37	-0.14	0.15	-0.34	0.23	-0.19	0.46	0.45	1	-0.16	0.35	-0.5
B	-0.031	-0.36	0.16	-0.34	0.041	-0.38	0.12	-0.27	0.27	-0.4	-0.4	-0.16	1	-0.34	0.32
LSTAT	-0.019	0.44	-0.4	0.62	-0.055	0.62	-0.63	0.59	-0.5	0.5	0.55	0.35	-0.34	1	-0.73
MEDV	0.007	-0.39	0.34	-0.5	0.18	-0.43	0.72	-0.35	0.24	-0.39	-0.48	-0.5	0.32	-0.73	1

```
x = data.iloc[:, :-1]

y = data.iloc[:,-1]
```

```
x.head()
```

	ID	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
0	0	0.95577	0.0	8.14	0	0.538	6.047	88.8	4.4534	4	307.0	21.0	306.38	17.28
1	1	0.02875	28.0	15.04	0	0.464	6.211	28.9	3.6659	4	270.0	18.2	396.33	6.21
2	2	1.22358	0.0	19.58	0	0.605	6.943	97.4	1.8773	5	403.0	14.7	363.43	4.59
3	3	5.66637	0.0	18.10	0	0.740	6.219	100.0	2.0048	24	666.0	20.2	395.69	16.59
4	4	0.04544	0.0	3.24	0	0.460	6.144	32.2	5.8736	4	430.0	16.9	368.57	9.09

```
y.head()
```

```
0    14.8
1    25.0
2    41.3
3    18.4
4    19.8
Name: MEDV, dtype: float64
```

Training and testing the model

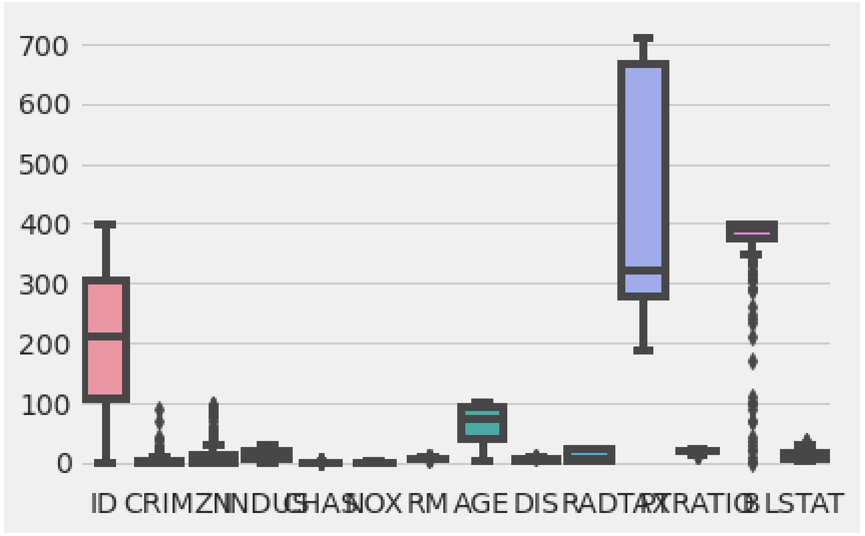
```
from sklearn.model_selection import train_test_split
```

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=.30, random_state=10)
```

```
#to check the outliers
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe3f667bd90>
```



Model Creation

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error,mean_absolute_error
from sklearn.metrics import r2_score,plot_confusion_matrix,confusion_matrix
from sklearn.model_selection import RandomizedSearchCV, GridSearchCV
from sklearn.preprocessing import MinMaxScaler
import numpy as np


# model instantiation
linear_model = LinearRegression()
linear_model.fit(x_train, y_train)
y_pred = linear_model.predict(x_test)
print('R2 Score : ',r2_score(y_test,y_pred))
print('MAE : ', mean_absolute_error(y_test,y_pred))
print("MSE : ", mean_squared_error(y_test,y_pred))
print("RSME : ",np.sqrt(mean_squared_error(y_test,y_pred)))
```

```
R2 Score :  0.7612316373010546
MAE :  3.402458519434511
MSE :  21.238691097077055
RSME :  4.6085454426616055
```

```
# print the intercept
print(linear_model.intercept_)
```

```
34.482470408771334
```

```
coeff_df = pd.DataFrame(linear_model.coef_, x.columns, columns=['Coefficient'])
coeff_df
```

	Coefficient
ID	0.000912

```
predicted_house_price = linear_model.predict(x_test)
print("Predicted House Prices")
print(predicted_house_price)
```

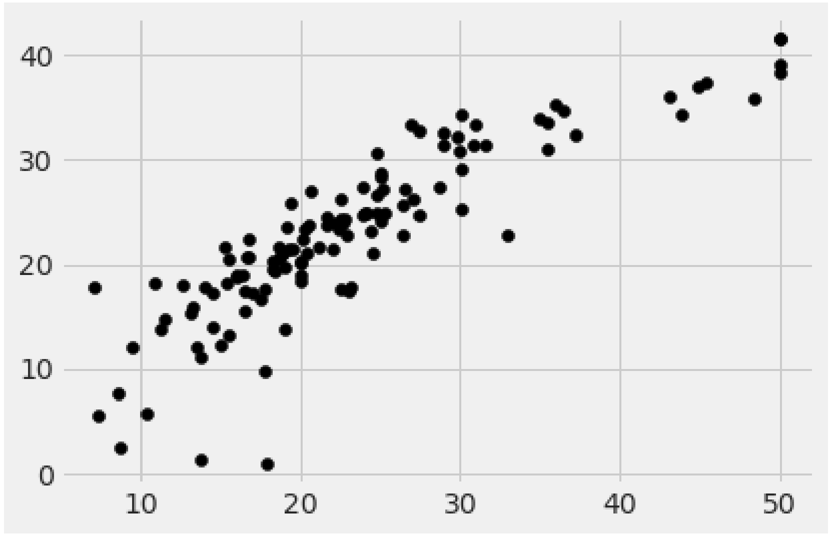
```
Predicted House Prices
[22.68762739 27.42490739  2.54844767 20.59402223 23.2086487  24.68101306
 33.23995838 18.93067954 25.85437299 14.70905505 13.81200059 18.38582099
 34.30693483 20.81423796 24.83326769 38.27780498 24.33983423 23.34947297
 17.45134564 21.7059009  12.02402327 29.11620999 30.55333567 17.77931724
 20.51951703 36.9390843   9.80567908 18.86308327 18.2649806  21.10496159
 28.65908178 12.20232548 41.55606719 19.30433527 17.76834863 17.34453684
 22.45319501 32.24479217 23.94431808 23.73778029 33.94362798 25.71494294
 20.47573189 31.38586956 27.22124752 35.77943688 34.16349858 22.86706338
 22.80395221 24.46289412 18.68584549 18.08506167 21.65842049 31.43721568
 27.43981806 21.40102722 17.55014886  5.82751811 16.68139283 24.14039056
  5.63077443 18.90388894 21.40042545 19.43611268 15.23684074 18.18979915
 35.27878604 11.10408947 21.0745016  26.24891847 17.31146471 30.85114095
 24.81566875 31.45778945 21.6284874  13.23294536 27.06268576  7.64814156
 15.56805606 26.28338835 17.79255839  1.03192922 15.85463014 27.23271242
 32.49783115 20.18310563 17.65532449 14.0208535  20.26156445 32.13211715
 24.31537431 32.7914202  31.08170851 13.78459989 39.03702136 24.97484269
 33.55971361 19.69117721 20.6153669   1.29153924 23.28944328 25.26669876
 41.59105091 24.61377733 21.42880124 33.36333806 17.14285204 35.89804727
 12.14939757 26.60869972 21.04869516 37.39136332 20.26203486 24.01332439
 28.35574017 22.48576316 23.66168423 24.95626851 23.52653107 34.72346863]
```

```
predicted_dataframe = pd.DataFrame(
    {"Predicted_Prices": predicted_house_price, "Actual_Prices": y_test})
print("Predicted and Actual Price Data frame")
predicted_dataframe
```

```
Predicted and Actual Price Data frame
```

	Predicted_Prices	Actual_Prices
332	22.687627	26.4
111	27.424907	28.7
352	2.548448	8.8
205	20.594022	16.7
56	23.208649	24.4
...	...	...
17	22.485763	20.2
225	23.661684	20.5
223	24.956269	25.3
59	23.526531	19.2

```
plt.scatter(y_test, predicted_house_price, color='black')
plt.show()
```



0
10
20
30
40
10
20
30
40
50