2019-20

Boston Housing Prices Prediction

Author:

Arnesh Majhi

Table of Contents

Content of the Boston Housing data frame

The Boston data frame contains 489 rows and 4 columns. This data frame contains the following columns:

RM - average number of rooms per dwelling

PTRATIO - pupil-teacher ratio by town

LSTAT - % lower status of the population

MEDV - Median value of owner-occupied homes in $1000's

The **MEDV** variable is the target variable.

Statistics of the Boston Housing data frame

The data was analysed using Python software and various statistics were calculated, which are being tabulated below:

	RM	**LSTAT**	**PTRATIO**	**MEDV**
count	489.000000	489.000000	489.000000	489.000000
mean	6.240288	12.939632	18.516564	4.543429e+05
Standard deviation	0.643650	7.081990	2.111268	1.653403e+05
min	3.561000	1.980000	12.600000	1.050000e+05
25%	5.880000	7.370000	17.400000	3.507000e+05
50%	6.185000	11.690000	19.100000	4.389000e+05
75%	6.575000	17.120000	20.200000	5.187000e+05
max	8.398000	37.970000	22.000000	1.024800e+06

1. Boxplot of the data

Following boxplot gives us the indication of how the values in the data are spread out.

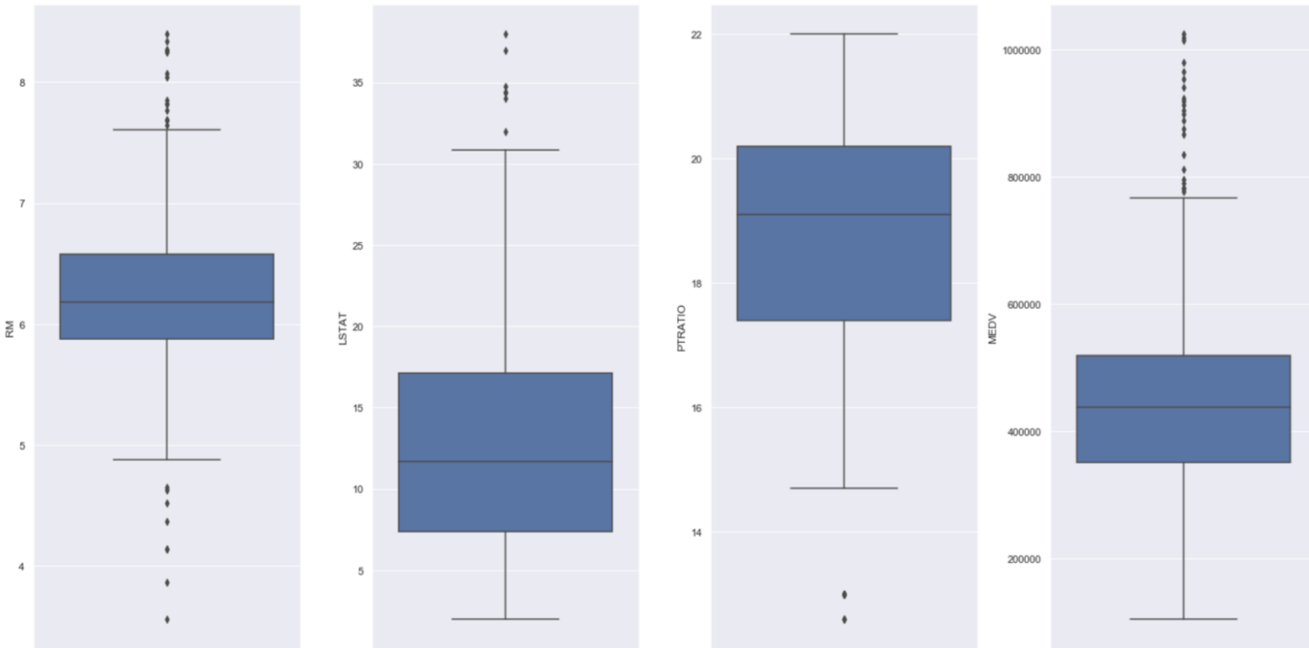
As it can be seen from the boxplot, the % of outliers present in each column are:

RM outliers = 4.50%

LSTAT outliers = 1.43%

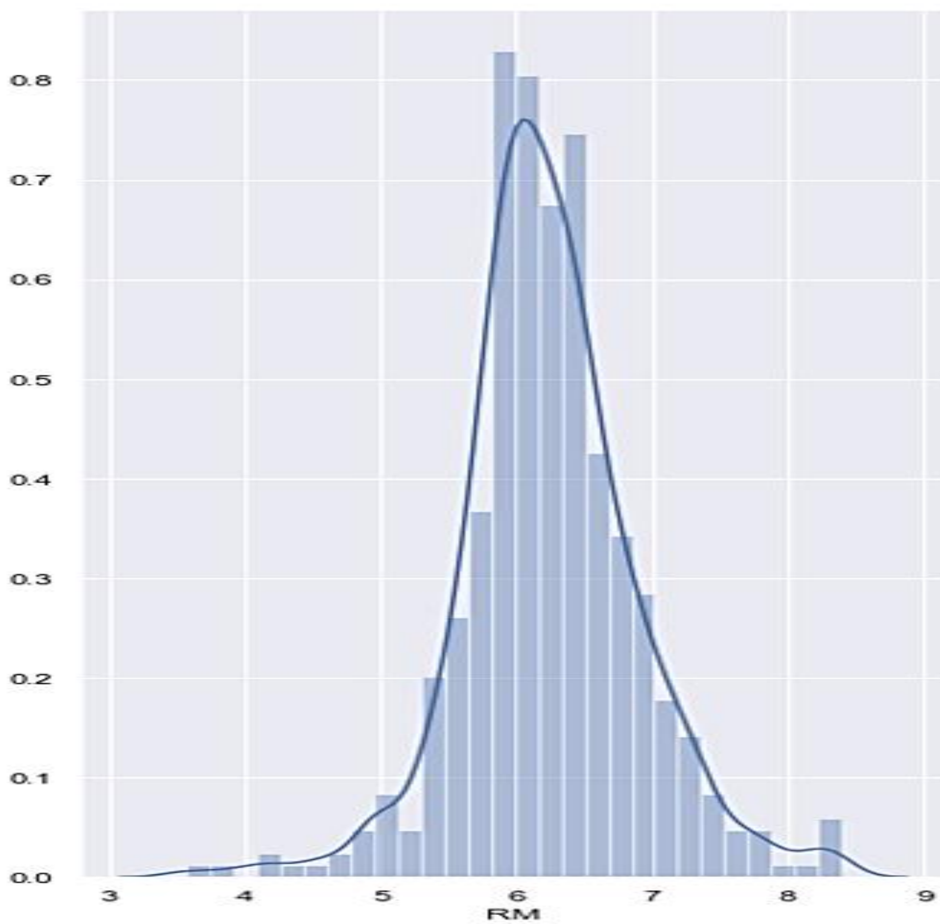
PTRATIO outliers = 2.66%

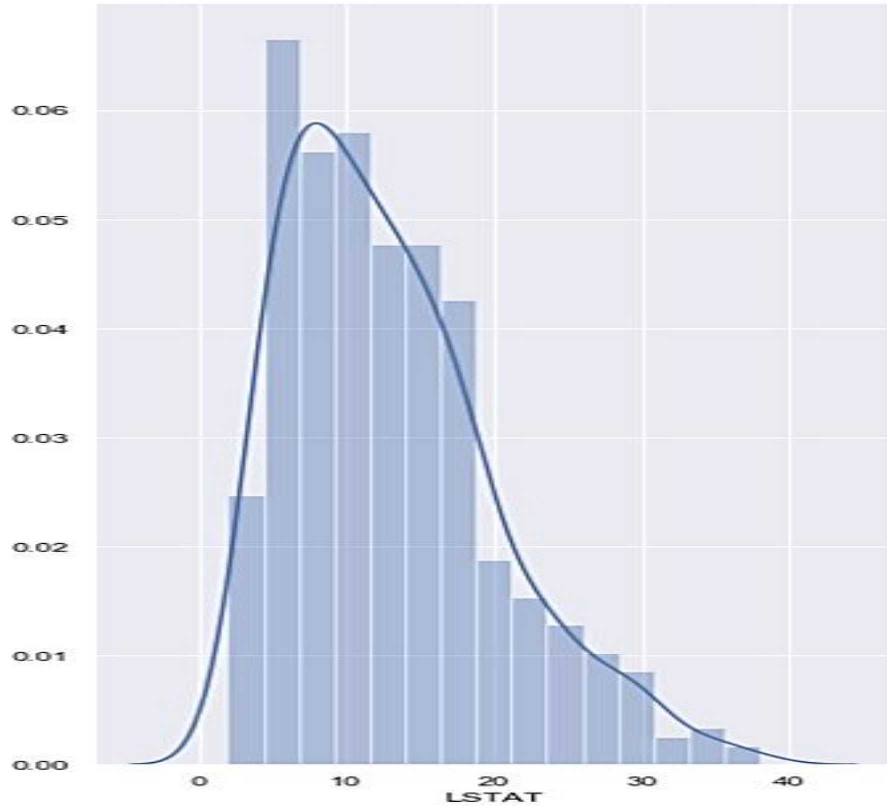
MEDV outliers = 4.50%

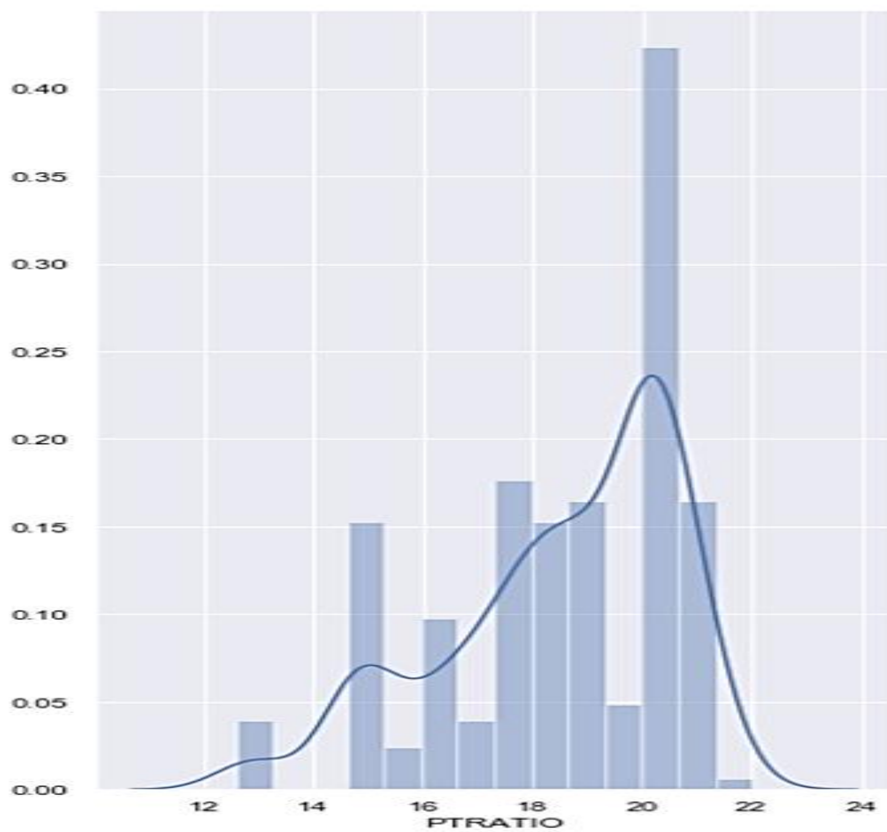


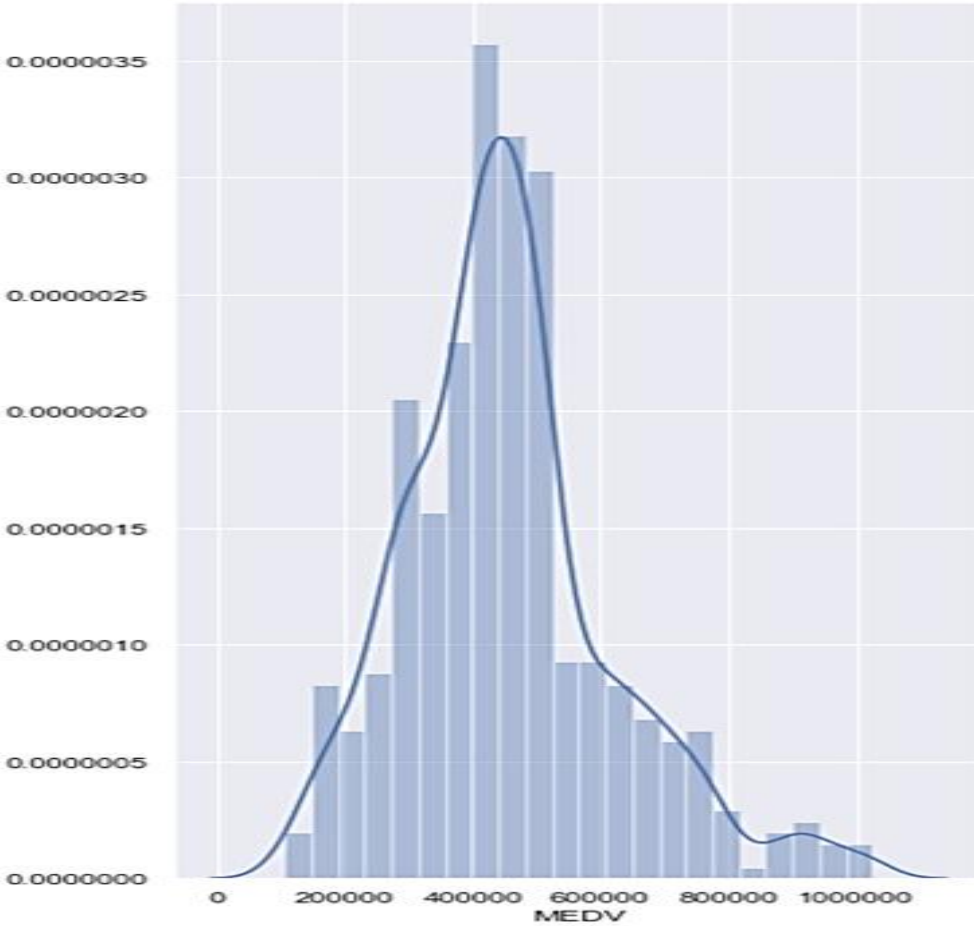
RM	LSTAT	PTRATIO	MEDV

2. Frequency distribution of data



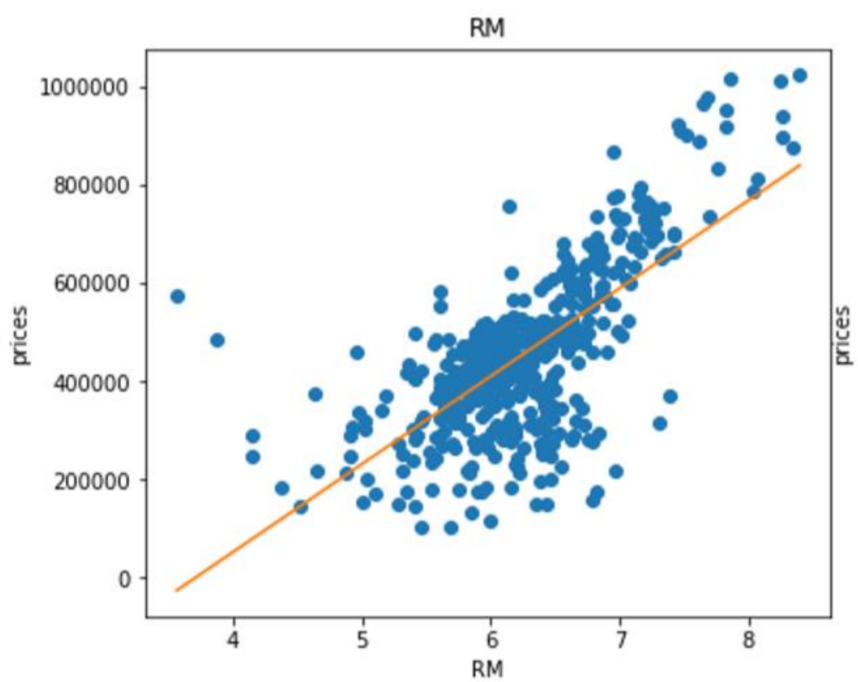

0.06
0.05
0.04
0.03
0.02
0.01
0.00
0
10
20
30
40
LSTAT

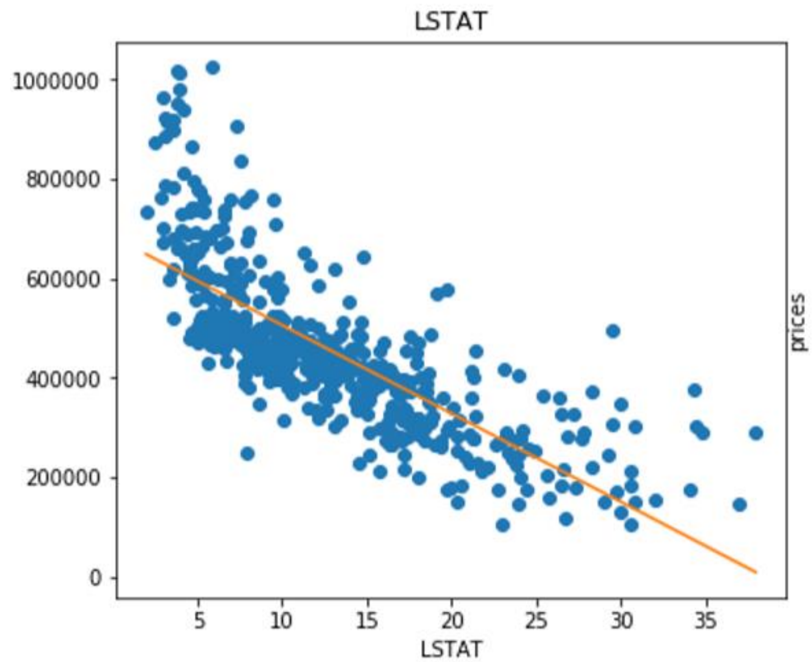

0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
12
14
16
18
20
22
24
PTRATIO

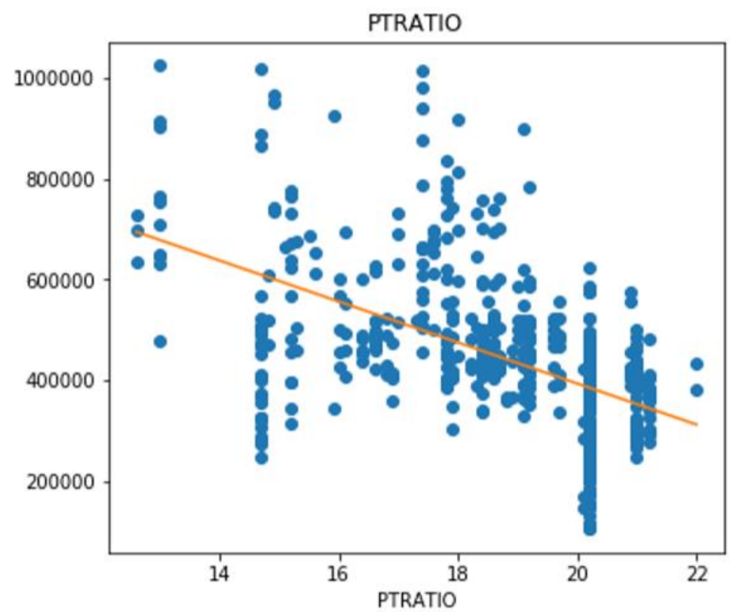


3. Regression analysis

Here prices indicate the MEDV variable. As it can be seen, prices are increasing with the RM variable and decreasing with the LSTAT and PTRATIO variable.







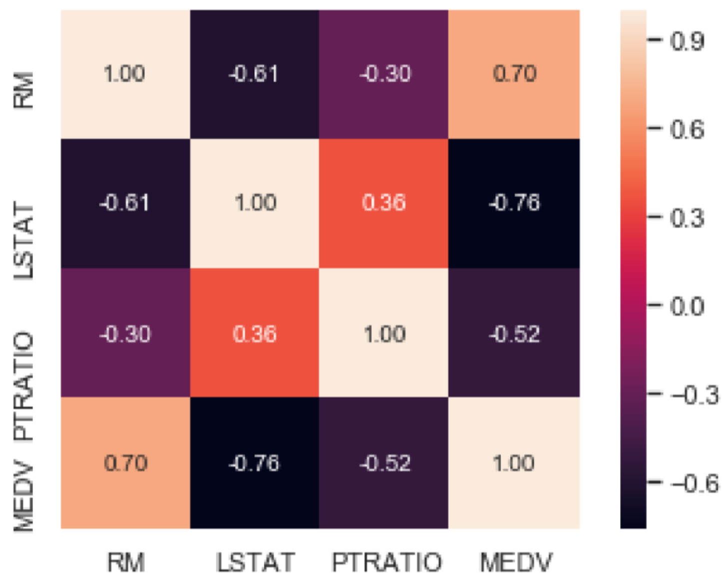
4. Correlation matrix

Linear regression and Random Forest models were applied to the given data and R square values were calculated. Based on R square was found that, ***Random Forest model*** is a better fit for the given data. However, to reach a conclusion with a greater degree of accuracy, we need more data.

A Random Forest is an ensemble technique capable of performing both regression and classification tasks with the use of multiple decision trees and a technique called Bootstrap Aggregation, commonly known as bagging. The basic idea behind this technique is to combine multiple decision trees in determining the final output rather than relying on individual decision trees.

The r square values came out to be 0.6083795028938159 for Random Forest, and for regression modelling r square values came out to be 0.7726270688507346.

Further, a correlation matrix was tabulated, as shown below, indicating the correlation coefficients between various variables. Each cell in the table shows the correlation between two variables. The values of correlation vary in the range of -1 to 1. More the absolute values of the correlation between the two variables, more the two variables are related to each other.



As it can be seen from the above correlation matrix, the variables "RM" and "MEDV" has the highest correlation value of 0.70 in the matrix. Thus, among all the variables, the variables "RM" and "MEDV" are best related to each other. Further, the variables "RM" and "LSTAT" have correlation value of 0.61 in the matrix. Thus, they are also highly related to each other.

5. Z-Test

H0 : μ(actual value) = μ(predicted value)

H1: μ(actual value) != μ(predicted value)

We performed a Z test for linear regression and random forest model at 5% significance.

Z value for linear regression came out to be 0.2684949105149169.

Z value for random forest model came out to be 0.07279780558113479.

Since both the calculated values of Z are less than the critical Z i.e. 1.96, Null Hypothesis is accepted for both the cases.

6. Conclusion

Although, both the models provide fairly accurate results in terms of predicting the housing prices, we should choose the Random Classifier model as it gives better results in terms of accuracy

7. References

https://www.kaggle.com/c/boston-housing/data

https://towardsdatascience.com/linear-regression-detailed-view-ea73175f6e86

https://www.statisticssolutions.com/what-is-linear-regression/

https://towardsdatascience.com/understanding-random-forest-58381e0602d2

Appendix I – Dataset

Following are the variables present in the dataset:

RM - average number of rooms per dwelling

PTRATIO - pupil-teacher ratio by town

LSTAT - % lower status of the population

MEDV - Median value of owner-occupied homes in $1000's

RM	LSTAT	PTRATIO	MEDV
6.575	4.98	15.3	504000
6.421	9.14	17.8	453600
7.185	4.03	17.8	728700
6.998	2.94	18.7	701400
7.147	5.33	18.7	760200
6.43	5.21	18.7	602700
6.012	12.43	15.2	480900
6.172	19.15	15.2	569100
5.631	29.93	15.2	346500
6.004	17.1	15.2	396900
6.377	20.45	15.2	315000
6.009	13.27	15.2	396900
5.889	15.71	15.2	455700
5.949	8.26	21	428400
6.096	10.26	21	382200
5.834	8.47	21	417900
5.935	6.58	21	485100
5.99	14.67	21	367500
5.456	11.69	21	424200
5.727	11.28	21	382200
5.57	21.02	21	285600
5.965	13.83	21	411600
6.142	18.72	21	319200
5.813	19.88	21	304500
5.924	16.3	21	327600

5.599	16.51	21	291900
5.813	14.81	21	348600
6.047	17.28	21	310800
6.495	12.8	21	386400
6.674	11.98	21	441000
5.713	22.6	21	266700
6.072	13.04	21	304500
5.95	27.71	21	277200
5.701	18.35	21	275100
6.096	20.34	21	283500
5.933	9.68	19.2	396900
5.841	11.41	19.2	420000
5.85	8.77	19.2	441000
5.966	10.13	19.2	518700
6.595	4.32	18.3	646800
7.024	1.98	18.3	732900
6.77	4.84	17.9	558600
6.169	5.81	17.9	531300
6.211	7.44	17.9	518700
6.069	9.55	17.9	445200
5.682	10.21	17.9	405300
5.786	14.15	17.9	420000
6.03	18.8	17.9	348600
5.399	30.81	17.9	302400
5.602	16.2	17.9	407400
5.963	13.45	16.8	413700
6.115	9.43	16.8	430500
6.511	5.28	16.8	525000
5.998	8.43	16.8	491400
5.888	14.8	21.1	396900
7.249	4.81	17.9	743400
6.383	5.77	17.3	518700
6.816	3.95	15.1	663600
6.145	6.86	19.7	489300
5.927	9.22	19.7	411600

5.741	13.15	19.7	392700
5.966	14.44	19.7	336000
6.456	6.73	19.7	466200
6.762	9.5	19.7	525000
7.104	8.05	18.6	693000
6.29	4.67	16.1	493500
5.787	10.24	16.1	407400
5.878	8.1	18.9	462000
5.594	13.09	18.9	365400
5.885	8.79	18.9	438900
6.417	6.72	19.2	508200
5.961	9.88	19.2	455700
6.065	5.52	19.2	478800
6.245	7.54	19.2	491400
6.273	6.78	18.7	506100
6.286	8.94	18.7	449400
6.279	11.97	18.7	420000
6.14	10.27	18.7	436800
6.232	12.34	18.7	445200
5.874	9.1	18.7	426300
6.727	5.29	19	588000
6.619	7.22	19	501900
6.302	6.72	19	520800
6.167	7.51	19	480900
6.389	9.62	18.5	501900
6.63	6.53	18.5	558600
6.015	12.86	18.5	472500
6.121	8.44	18.5	466200
7.007	5.5	17.8	495600
7.079	5.7	17.8	602700
6.417	8.81	17.8	474600
6.405	8.2	17.8	462000
6.442	8.16	18.2	480900
6.211	6.21	18.2	525000
6.249	10.59	18.2	432600

6.625	6.65	18	596400
6.163	11.34	18	449400
8.069	4.21	18	812700
7.82	3.57	18	919800
7.416	6.19	18	697200
6.727	9.42	20.9	577500
6.781	7.67	20.9	556500
6.405	10.63	20.9	390600
6.137	13.44	20.9	405300
6.167	12.33	20.9	422100
5.851	16.47	20.9	409500
5.836	18.66	20.9	409500
6.127	14.09	20.9	428400
6.474	12.27	20.9	415800
6.229	15.55	20.9	407400
6.195	13	20.9	455700
6.715	10.16	17.8	478800
5.913	16.21	17.8	394800
6.092	17.09	17.8	392700
6.254	10.45	17.8	388500
5.928	15.76	17.8	384300
6.176	12.04	17.8	445200
6.021	10.3	17.8	403200
5.872	15.37	17.8	428400
5.731	13.61	17.8	405300
5.87	14.37	19.1	462000
6.004	14.27	19.1	426300
5.961	17.93	19.1	430500
5.856	25.41	19.1	363300
5.879	17.58	19.1	394800
5.986	14.81	19.1	449400
5.613	27.26	19.1	329700
5.693	17.19	21.2	340200
6.431	15.39	21.2	378000
5.637	18.34	21.2	300300

6.458	12.6	21.2	403200
6.326	12.26	21.2	411600
6.372	11.12	21.2	483000
5.822	15.03	21.2	386400
5.757	17.31	21.2	327600
6.335	16.96	21.2	380100
5.942	16.9	21.2	365400
6.454	14.59	21.2	359100
5.857	21.32	21.2	279300
6.151	18.46	21.2	373800
6.174	24.16	21.2	294000
5.019	34.41	21.2	302400
5.403	26.82	14.7	281400
5.468	26.42	14.7	327600
4.903	29.29	14.7	247800
6.13	27.8	14.7	289800
5.628	16.65	14.7	327600
4.926	29.53	14.7	306600
5.186	28.32	14.7	373800
5.597	21.45	14.7	323400
6.122	14.1	14.7	451500
5.404	13.28	14.7	411600
5.012	12.12	14.7	321300
5.709	15.79	14.7	407400
6.129	15.12	14.7	357000
6.152	15.02	14.7	327600
5.272	16.14	14.7	275100
6.943	4.59	14.7	867300
6.066	6.43	14.7	510300
6.51	7.39	14.7	489300
6.25	5.5	14.7	567000
5.854	11.64	14.7	476700
6.101	9.81	14.7	525000
5.877	12.14	14.7	499800
6.319	11.1	14.7	499800

6.402	11.32	14.7	468300
5.875	14.43	14.7	365400
5.88	12.03	14.7	401100
5.572	14.69	16.6	485100
6.416	9.04	16.6	495600
5.859	9.64	16.6	474600
6.546	5.33	16.6	617400
6.02	10.11	16.6	487200
6.315	6.29	16.6	516600
6.86	6.92	16.6	627900
6.98	5.04	17.8	781200
7.765	7.56	17.8	835800
6.144	9.45	17.8	760200
7.155	4.82	17.8	795900
6.563	5.68	17.8	682500
5.604	13.98	17.8	554400
6.153	13.15	17.8	621600
6.782	6.68	15.2	672000
6.556	4.56	15.2	625800
7.185	5.39	15.2	732900
6.951	5.1	15.2	777000
6.739	4.69	15.2	640500
7.178	2.87	15.2	764400
6.8	5.03	15.6	653100
6.604	4.38	15.6	611100
7.287	4.08	12.6	699300
7.107	8.61	12.6	636300
7.274	6.62	12.6	726600
6.975	4.56	17	732900
7.135	4.45	17	690900
6.162	7.43	14.7	506100
7.61	3.11	14.7	888300
7.853	3.81	14.7	1018500
5.891	10.87	18.6	474600
6.326	10.97	18.6	512400

5.783	18.06	18.6	472500
6.064	14.66	18.6	512400
5.344	23.09	18.6	420000
5.96	17.27	18.6	455700
5.404	23.98	18.6	405300
5.807	16.03	18.6	470400
6.375	9.38	18.6	590100
5.412	29.55	18.6	497700
6.182	9.47	18.6	525000
5.888	13.51	16.4	489300
6.642	9.69	16.4	602700
5.951	17.92	16.4	451500
6.373	10.5	16.4	483000
6.951	9.71	17.4	560700
6.164	21.46	17.4	455700
6.879	9.93	17.4	577500
6.618	7.6	17.4	632100
8.266	4.14	17.4	940800
8.04	3.13	17.4	789600
7.163	6.36	17.4	663600
7.686	3.92	17.4	980700
6.552	3.76	17.4	661500
5.981	11.65	17.4	510300
7.412	5.25	17.4	665700
8.337	2.47	17.4	875700
8.247	3.95	17.4	1014300
6.726	8.05	17.4	609000
6.086	10.88	17.4	504000
6.631	9.54	17.4	527100
7.358	4.73	17.4	661500
6.481	6.36	16.6	497700
6.606	7.37	16.6	489300
6.897	11.38	16.6	462000
6.095	12.4	16.6	422100
6.358	11.22	16.6	466200

6.393	5.19	16.6	497700
5.593	12.5	19.1	369600
5.605	18.46	19.1	388500
6.108	9.16	19.1	510300
6.226	10.15	19.1	430500
6.433	9.52	19.1	514500
6.718	6.56	19.1	550200
6.487	5.9	19.1	512400
6.438	3.59	19.1	520800
6.957	3.53	19.1	621600
8.259	3.54	19.1	898800
6.108	6.57	16.4	459900
5.876	9.25	16.4	438900
7.454	3.11	15.9	924000
7.333	7.79	13	756000
6.842	6.9	13	632100
7.203	9.59	13	709800
7.52	7.26	13	905100
8.398	5.91	13	1024800
7.327	11.25	13	651000
7.206	8.1	13	766500
5.56	10.45	13	478800
7.014	14.79	13	644700
7.47	3.16	13	913500
5.92	13.65	18.6	434700
5.856	13	18.6	443100
6.24	6.59	18.6	529200
6.538	7.73	18.6	512400
7.691	6.58	18.6	739200
6.758	3.53	17.6	680400
6.854	2.98	17.6	672000
7.267	6.05	17.6	697200
6.826	4.16	17.6	695100
6.482	7.19	17.6	611100
6.812	4.85	14.9	737100

7.82	3.76	14.9	953400
6.968	4.59	14.9	743400
7.645	3.01	14.9	966000
7.088	7.85	15.3	676200
6.453	8.23	15.3	462000
6.23	12.93	18.2	422100
6.209	7.14	16.6	487200
6.315	7.6	16.6	468300
6.565	9.51	16.6	520800
6.861	3.33	19.2	598500
7.148	3.56	19.2	783300
6.63	4.7	19.2	585900
6.127	8.58	16	501900
6.009	10.4	16	455700
6.678	6.27	16	600600
6.549	7.39	16	569100
5.79	15.84	16	426300
6.345	4.97	14.8	472500
7.041	4.74	14.8	609000
6.871	6.07	14.8	520800
6.59	9.5	16.1	462000
6.495	8.67	16.1	554400
6.982	4.86	16.1	695100
7.236	6.93	18.4	758100
6.616	8.93	18.4	596400
7.42	6.47	18.4	701400
6.849	7.53	18.4	592200
6.635	4.54	18.4	478800
5.972	9.97	18.4	426300
4.973	12.64	18.4	338100
6.122	5.98	18.4	464100
6.023	11.72	18.4	407400
6.266	7.9	18.4	453600
6.567	9.28	18.4	499800
5.705	11.5	18.4	340200

5.914	18.33	18.4	373800
5.782	15.94	18.4	415800
6.382	10.36	18.4	485100
6.113	12.73	18.4	441000
6.426	7.2	19.6	499800
6.376	6.87	19.6	485100
6.041	7.7	19.6	428400
5.708	11.74	19.6	388500
6.415	6.12	19.6	525000
6.431	5.08	19.6	516600
6.312	6.15	19.6	483000
6.083	12.79	19.6	466200
5.868	9.97	16.9	405300
6.333	7.34	16.9	474600
6.144	9.09	16.9	415800
5.706	12.43	16.9	359100
6.031	7.83	16.9	407400
6.316	5.68	20.2	466200
6.31	6.75	20.2	434700
6.037	8.01	20.2	443100
5.869	9.8	20.2	409500
5.895	10.56	20.2	388500
6.059	8.51	20.2	432600
5.985	9.74	20.2	399000
5.968	9.29	20.2	392700
7.241	5.49	15.5	686700
6.54	8.65	15.9	346500
6.696	7.18	17.6	501900
6.874	4.61	17.6	655200
6.014	10.53	18.8	367500
5.898	12.67	18.8	361200
6.516	6.36	17.9	485100
6.635	5.99	17	514500
6.939	5.89	19.7	558600
6.49	5.98	19.7	480900

6.579	5.49	18.3	506100
5.884	7.79	18.3	390600
6.728	4.5	17	632100
5.663	8.05	22	382200
5.936	5.57	22	432600
6.212	17.6	20.2	373800
6.395	13.27	20.2	455700
6.127	11.48	20.2	476700
6.112	12.67	20.2	474600
6.398	7.79	20.2	525000
6.251	14.19	20.2	417900
5.362	10.19	20.2	436800
5.803	14.64	20.2	352800
3.561	7.12	20.2	577500
4.963	14	20.2	459900
3.863	13.33	20.2	485100
4.906	34.77	20.2	289800
4.138	37.97	20.2	289800
7.313	13.44	20.2	315000
6.649	23.24	20.2	291900
6.794	21.24	20.2	279300
6.38	23.69	20.2	275100
6.223	21.78	20.2	214200
6.968	17.21	20.2	218400
6.545	21.08	20.2	228900
5.536	23.6	20.2	237300
5.52	24.56	20.2	258300
4.368	30.63	20.2	184800
5.277	30.81	20.2	151200
4.652	28.28	20.2	220500
5	31.99	20.2	155400
4.88	30.62	20.2	214200
5.39	20.85	20.2	241500
5.713	17.11	20.2	317100
6.051	18.76	20.2	487200

5.036	25.68	20.2	203700
6.193	15.17	20.2	289800
5.887	16.35	20.2	266700
6.471	17.12	20.2	275100
6.405	19.37	20.2	262500
5.747	19.92	20.2	178500
5.453	30.59	20.2	105000
5.852	29.97	20.2	132300
5.987	26.77	20.2	117600
6.343	20.32	20.2	151200
6.404	20.31	20.2	254100
5.349	19.77	20.2	174300
5.531	27.38	20.2	178500
5.683	22.98	20.2	105000
4.138	23.34	20.2	249900
5.608	12.13	20.2	585900
5.617	26.4	20.2	361200
6.852	19.78	20.2	577500
5.757	10.11	20.2	315000
6.657	21.22	20.2	361200
4.628	34.37	20.2	375900
5.155	20.08	20.2	342300
4.519	36.98	20.2	147000
6.434	29.05	20.2	151200
6.782	25.79	20.2	157500
5.304	26.64	20.2	218400
5.957	20.62	20.2	184800
6.824	22.74	20.2	176400
6.411	15.02	20.2	350700
6.006	15.7	20.2	298200
5.648	14.1	20.2	436800
6.103	23.29	20.2	281400
5.565	17.16	20.2	245700
5.896	24.39	20.2	174300
5.837	15.69	20.2	214200

6.202	14.52	20.2	228900
6.193	21.52	20.2	231000
6.38	24.08	20.2	199500
6.348	17.64	20.2	304500
6.833	19.69	20.2	296100
6.425	12.03	20.2	338100
6.436	16.22	20.2	300300
6.208	15.17	20.2	245700
6.629	23.27	20.2	281400
6.461	18.05	20.2	201600
6.152	26.45	20.2	182700
5.935	34.02	20.2	176400
5.627	22.88	20.2	268800
5.818	22.11	20.2	220500
6.406	19.52	20.2	359100
6.219	16.59	20.2	386400
6.485	18.85	20.2	323400
5.854	23.79	20.2	226800
6.459	23.98	20.2	247800
6.341	17.79	20.2	312900
6.251	16.44	20.2	264600
6.185	18.13	20.2	296100
6.417	19.31	20.2	273000
6.749	17.44	20.2	281400
6.655	17.73	20.2	319200
6.297	17.27	20.2	338100
7.393	16.74	20.2	373800
6.728	18.71	20.2	312900
6.525	18.13	20.2	296100
5.976	19.01	20.2	266700
5.936	16.94	20.2	283500
6.301	16.23	20.2	312900
6.081	14.7	20.2	420000
6.701	16.42	20.2	344400
6.376	14.65	20.2	371700

6.317	13.99	20.2	409500
6.513	10.29	20.2	424200
6.209	13.22	20.2	449400
5.759	14.13	20.2	417900
5.952	17.15	20.2	399000
6.003	21.32	20.2	401100
5.926	18.13	20.2	401100
5.713	14.76	20.2	422100
6.167	16.29	20.2	417900
6.229	12.87	20.2	411600
6.437	14.36	20.2	487200
6.98	11.66	20.2	625800
5.427	18.14	20.2	289800
6.162	24.1	20.2	279300
6.484	18.68	20.2	350700
5.304	24.91	20.2	252000
6.185	18.03	20.2	306600
6.229	13.11	20.2	449400
6.242	10.74	20.2	483000
6.75	7.74	20.2	497700
7.061	7.01	20.2	525000
5.762	10.42	20.2	457800
5.871	13.34	20.2	432600
6.312	10.58	20.2	445200
6.114	14.98	20.2	401100
5.905	11.45	20.2	432600
5.454	18.06	20.1	319200
5.414	23.97	20.1	147000
5.093	29.68	20.1	170100
5.983	18.07	20.1	285600
5.983	13.35	20.1	422100
5.707	12.01	19.2	457800
5.926	13.59	19.2	514500
5.67	17.6	19.2	485100
5.39	21.14	19.2	413700

5.794	14.1	19.2	384300
6.019	12.92	19.2	445200
5.569	15.1	19.2	367500
6.027	14.33	19.2	352800
6.593	9.67	21	470400
6.12	9.08	21	432600
6.976	5.64	21	501900
6.794	6.48	21	462000
6.03	7.88	21	249900

Appendix II – Python Code

stats_boston_housing

November 13, 2019

In [64]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
df=pd.read_csv('housing.csv')
```

In [65]:
```
#RM - average number of rooms per dwelling
#PTRATIO - pupil-teacher ratio by town
#LSTAT - % lower status of the population #MEDV
- Median value of owner-occupied homes
```

In [66]:
```
df.head()
```

Out[66]:

	T PTRATIO		MEDV 0	6.575	4.98
	15.3	504000.0	1	6.421	9.14
R	17.8	453600.0	2	7.185	4.03
M	17.8	728700.0	3	6.998	2.94
	18.7	701400.0	4	7.147	5.33
L S T A	18.7	760200.0			

In [67]: df.shape

Out[67]: (489, 4)

In [68]: df.describe() RM LSTAT PTRATIO MEDV

Out[68]:

count	489.000000	489.000000	489.000000	4.890000e+02 mean
6.240288	12.939632	18.516564	4.543429e+05	std
0.643650	7.081990	2.111268	1.653403e+05	min
3.561000	1.980000	12.600000	1.050000e+05	25%
5.880000	7.370000	17.400000	3.507000e+05	50%
6.185000	11.690000	19.100000	4.389000e+05	75%
6.575000	17.120000	20.200000	5.187000e+05	max
8.398000	37.970000	22.000000	1.024800e+06	

In [69]:
```
#Checks for blanks
zero_counts={ 'RM': df['RM'].isnull().sum() , 'LSTAT': df['LSTAT'].isnull().sum() ,'PT
 'MEDV': df['MEDV'].isnull().sum()}
zero_counts
```

1

```
Out[69]: {'RM': 0, 'LSTAT': 0, 'PTRATIO': 0, 'MEDV': 0}
```

```
In [70]: prices = df['MEDV']
         features = df.drop('MEDV', axis = 1)
```

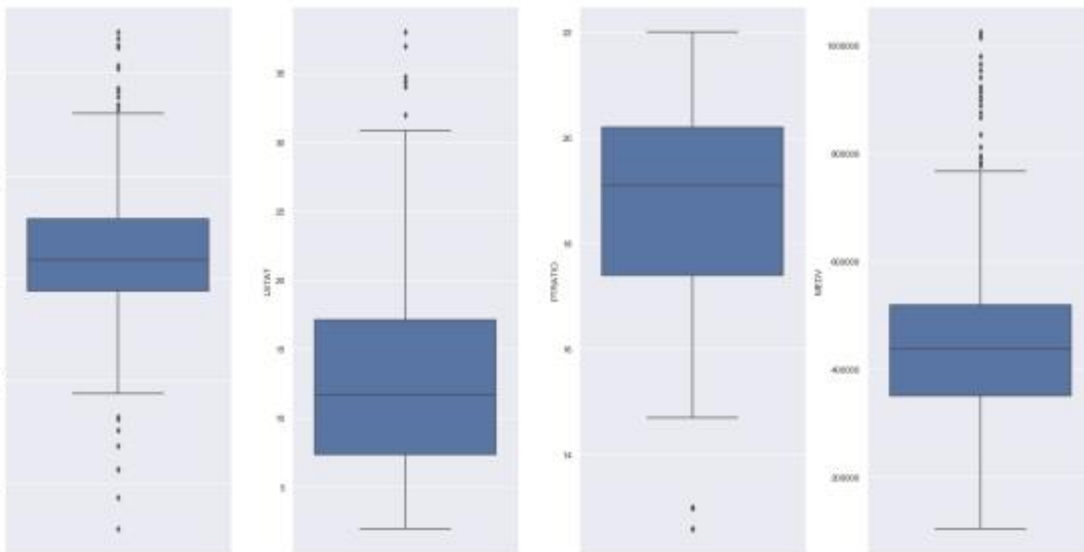
```
In [71]: minimum_price = np.mean(prices)


         maximum_price = np.max(prices)


         mean_price = np.mean(prices)


         median_price = np.median(prices)


         std_price = np.std(prices)


         print("Statistics for Boston housing dataset:\n")
         print("Minimum price: ${:,.2f}".format(minimum_price))
         print("Maximum price: ${:,.2f}".format(maximum_price))
         print("Mean price: ${:,.2f}".format(mean_price))
         print("Median price ${:,.2f}".format(median_price))
         print("Standard deviation of prices: ${:,.2f}".format(std_price))
```

```
Statistics for Boston housing dataset:


Minimum price: $454,342.94
Maximum price: $1,024,800.00
Mean price: $454,342.94
Median price $438,900.00
Standard deviation of prices: $165,171.13
```

```
In [72]: import seaborn as sns
         import matplotlib.pyplot as plt
         from scipy import stats
```

```
fig, axs = plt.subplots(ncols=4, nrows=1, figsize=(20, 10)) index =
0
axs = axs.flatten()
for k,v in df.items():
    sns.boxplot(y=k, data=df, ax=axs[index])
    index += 1
plt.tight_layout(pad=0.4, w_pad=0.5, h_pad=5.0)
```

2



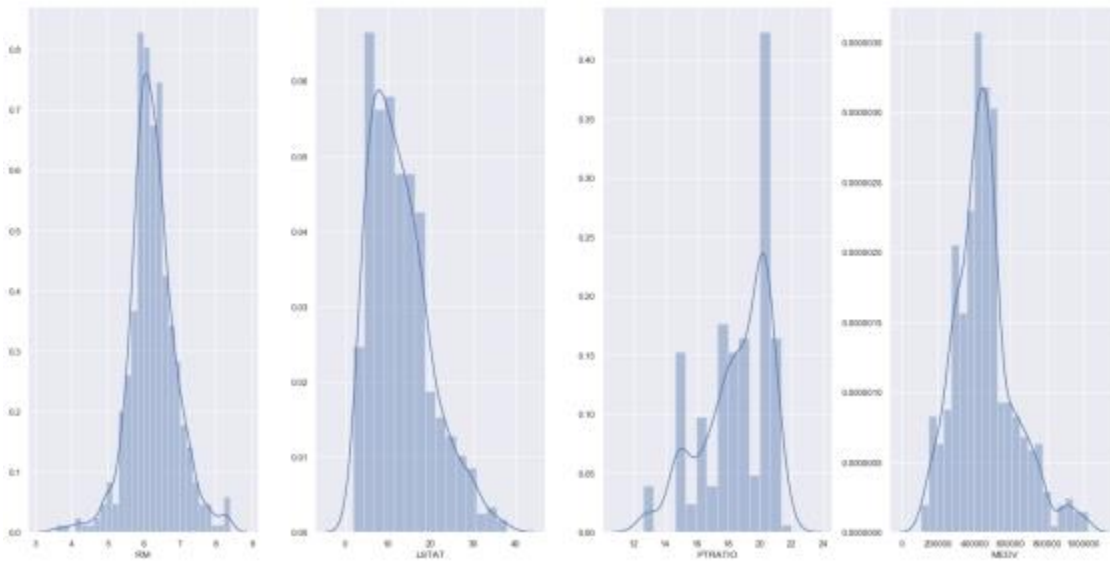
In [73]:
```
for k, v in df.items():
        q1 = v.quantile(0.25)
        q3 = v.quantile(0.75)
        irq = q3 - q1
        v_col = v[(v <= q1 - 1.5 * irq) | (v >= q3 + 1.5 * irq)] perc
        = np.shape(v_col)[0] * 100.0 / np.shape(df)[0]  print("Column
        %s outliers = %.2f%%" % (k, perc))
```

```
Column RM outliers = 4.50%
Column LSTAT outliers = 1.43%
Column PTRATIO outliers = 2.66%
Column MEDV outliers = 4.50%
```

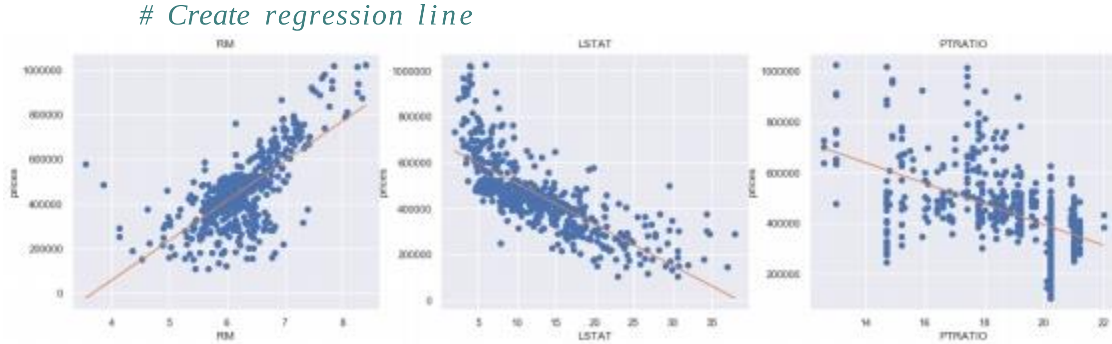
In [74]:
```
fig, axs = plt.subplots(ncols=4, nrows=1, figsize=(20, 10))
index = 0
axs = axs.flatten()
for k,v in df.items():
    sns.distplot(v, ax=axs[index])
    index += 1
plt.tight_layout(pad=0.4, w_pad=0.5, h_pad=5.0)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\scipy\stats\stats.py:1713: FutureWarning: Using a no
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```

3

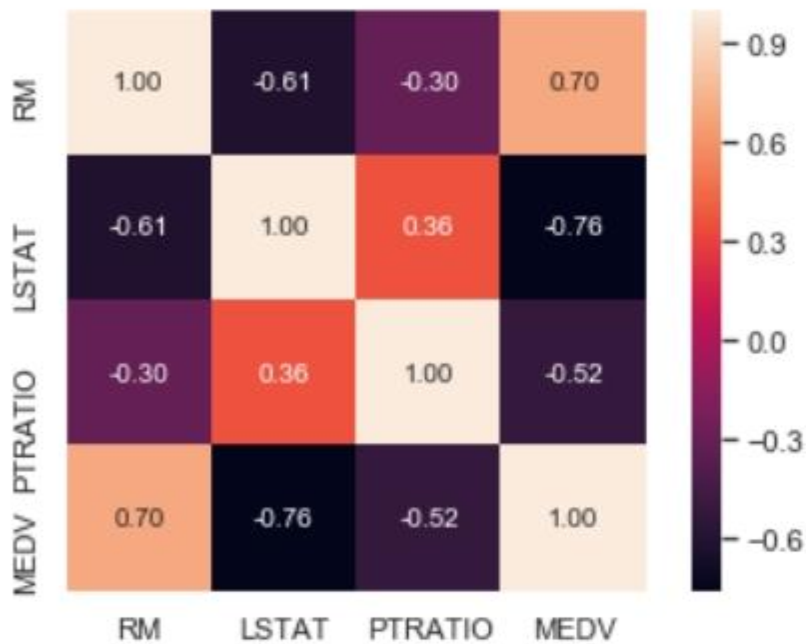



```
In [75]: import matplotlib.pyplot as plt
         plt.figure(figsize=(20, 5))


         for i, col in enumerate(features.columns):
             plt.subplot(1, 3, i+1)
             x = df[col]
             y = prices
             plt.plot(x, y, 'o')
             # Create regression line
```



In [76]: # Correlation Matrix

```
cor_matrix = np.corrcoef(df[df.columns].values.T) # We transpose to get the data by co
sns.set(font_scale=1)
cor_heat_map = sns.heatmap(cor_matrix,
 cbar=True,
 annot=True,
 square=True,
 fmt='.2f',
 annot_kws={'size':10},
 yticklabels=df.columns,
 xticklabels=df.columns)


plt.show()
```



In [77]:
```
# Creating deep copy of original dataframe
dff=df.copy(deep=True)
```

In [78]:
```
dff.head()
```

Out[78]:

RM LSTAT PTRATIO MEDV 0 6.575
4.98 15.3 504000.0 1 6.421 9.14
17.8 453600.0 2 7.185 4.03 17.8
728700.0 3 6.998 2.94 18.7 701400.0 4
7.147 5.33 18.7 760200.0

In [79]:
```
# Standardizing values
#from sklearn.preprocessing import StandardScaler
#num_values1=dff.select_dtypes(['float64','int64']).columns
#scaler = StandardScaler()
#scaler.fit(dff[num_values1])
#dff[num_values1]=scaler.transform(dff[num_values1])
```

In [80]:
```
dff.head()
```

Out[80]:

	RM	LSTAT	PTRATIO	MEDV
0	6.575	4.98	15.3	504000.0
1	6.421	9.14	17.8	453600.0
2	7.185	4.03	17.8	728700.0
3	6.998	2.94	18.7	701400.0
4	7.147	5.33	18.7	760200.0

In [81]:
```
x = dff.drop(['MEDV'],axis = 1)
y = dff['MEDV']
```

In [82]:
```
x.head()
```

Out[82]:

	RM	LSTAT	PTRATIO
0	6.575	4.98	15.3
1	6.421	9.14	17.8
2	7.185	4.03	17.8
3	6.998	2.94	18.7
4	7.147	5.33	18.7

In [83]:
```
y.head()
```

Out[83]:
```
0    504000.0
1    453600.0
2    728700.0
```

```
3       701400.0
4       760200.0
Name: MEDV, dtype: float64
```

In [84]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(x,y,test_size = 0.3,random_state = 33
```

In [85]:
```
# Applying linear regression
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(X_train,y_train)
y_pred_lr = lr.predict(X_test)
```

In [86]:
```
#R^2 score
lr.score(X_test, y_test)
```

Out[86]: 0.6083795028938153

In [87]:
```
# Applying random forest
from sklearn import metrics
from sklearn.ensemble import RandomForestRegressor
rfr = RandomForestRegressor()
rfr.fit(X_train,y_train)
y_pred_rfr = rfr.predict(X_test)
```

In [88]:
```
#R^2 score
rfr.score(X_test,y_test)
```

Out[88]: 0.7878366795445423

In [89]:
```
dff['PRED_MEDV_LR']=lr.predict(df.drop(['MEDV'],axis = 1))
```

In [90]:
```
dff['PRED_MEDV_RF']=rfr.predict(df.drop(['MEDV'],axis = 1))
```

In [91]:
```
dff.head(10)
```

Out[91]:

	RM	LSTAT	PTRATIO	MEDV	PRED_MEDV_LR	PRED_MEDV_RF
0	6.575	4.98	15.3	504000.0	630636.304650	602280.0
1	6.421	9.14	17.8	453600.0	523271.934059	491400.0
2	7.185	4.03	17.8	728700.0	653678.257698	726180.0
3	6.998	2.94	18.7	701400.0	628424.758645	728910.0
4	7.147	5.33	18.7	760200.0	618840.932373	726390.0
5	6.430	5.21	18.7	602700.0	547283.553374	523320.0
6	6.012	12.43	15.2	480900.0	498391.389281	469560.0
7	6.172	19.15	15.2	569100.0	445153.237224	546420.0
8	5.631	29.93	15.2	346500.0	278762.002770	292110.0
9	6.004	17.10	15.2	396900.0	449292.443893	407820.0

In [92]:
```
# standard deviation of the dataframe
dff.std()
```

Out[92]:
```
RM                     0.643650
LSTAT                  7.081990
PTRATIO                2.111268
MEDV              165340.277653
PRED_MEDV_LR      144887.135403
PRED_MEDV_RF      159297.471406
dtype: float64
```

In [93]:
```
dff.mean()
```

Out[93]:
```
RM                     6.240288
LSTAT                 12.939632
PTRATIO               18.516564
MEDV              454342.944785
PRED_MEDV_LR      451673.698602
PRED_MEDV_RF      453587.116564
dtype: float64
```

In [95]:
```
dff.shape
```

Out[95]:
```
(489, 6)
```

In [97]:
```
# Z score for Linear Regression
import math
s1=165340.277653
s2=144887.135403
n1=489
n2=489
x1=454342.944785
x2=451673.698602
den=math.sqrt((s1*s1)/n1+(s2*s2)/n2)
Z=(x1-x2)/den
print(Z)
```

```
    if Z<1.96:
        print("Since calculated value of Z is less than the critical Z i.e. 1.96, Null Hyp
    else:
        print("Null hypothesis is rejected for Linear Regression")
```

```
0.2684949105149169
Since calculated value of Z is less than the critical Z i.e. 1.96, Null Hypothesis is accepted
```

```
In [98]: # Z score for Random Forest
         import math
         s1=165340.277653
         s2=159297.471406
         n1=489
         n2=489
         x1=454342.944785
         x2=453587.116564
         den=math.sqrt((s1*s1)/n1+(s2*s2)/n2)
         Z=(x1-x2)/den
         print(Z)
         if Z<1.96:
             print("Since calculated value of Z is less than the critical Z i.e. 1.96, Null Hyp
         else:
             print("Null hypothesis is rejected for Random Forest")
```

```
0.07279780558113479
Since calculated value of Z is less than the critical Z i.e. 1.96, Null Hypothesis is accepted
```

```
In [ ]
```