LINEAR REGRESSION

1. What is linear regression

Linear regression is a linear model, e.g. a model that assumes a **linear relationship between the input variables** (x) and the single **output variable** (y).

More specifically, that **y can be calculated from a linear combination of the input variables (x).**

When there is a **single input variable (x), the method is referred to as simple linear regression**. When there are **multiple input variables, often refers to the method as multiple linear regression.**

2. What is Linear Relationship

A linear relationship (or linear association) is a statistical term used to describe a **straight-line relationship between two variables.**

Linear relationships can be expressed : **y = mx + b** where: m=slope b=y-intercept

3. Example of Linear Regression

Estimation of Bad Loans in a bank. The bank had disbursed 100 loans in the quarter. Additionally, we had noticed around 2.5% of bad rate. So, build a regression model for estimating bad rate in next quarter.

4. Assumptions in a linear regression model

Linear relationship: A linear relationship (or linear association) is a statistical term used to describe a straight-line relationship between two variables.

Multivariate normality: The error terms must be normally distributed.

No or little multicollinearity: The independent variables should not be correlated. Absence of this phenomenon is known as multicollinearity.

No autocorrelation: There should be no correlation between the residual (error) terms. Absence of this phenomenon is known as Autocorrelation.

Homoscedasticity: The error terms must have constant variance. This phenomenon is known as homoskedasticity. The presence of non-constant variance is referred to heteroskedasticity.

5. How does multicollinearity affect the linear regression

Multicollinearity causes the following two basic types of problems:

1. The coefficient estimates can swing wildly based on which other independent variables are in the model. The coefficients become very sensitive to small changes in the model.
2. **Multicollinearity reduces the precision of the estimate coefficients**, which weakens the statistical power of your regression model. You might **not be able to trust the p-values to identify independent variables that are statistically significant**.

Example of High Multicollinearity is Age and Date of Birth

6. How does heteroscedasticity affect the linear regression

Example:

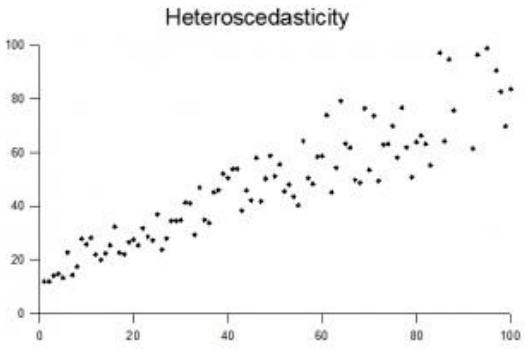
Imagine we have data on **family income and spending on luxury items**. Using bivariate regression, we use family income to predict luxury spending. **As expected, there is a strong, positive association between income and spending.**

Upon examining the residuals we detect a problem – **the residuals are very small for low values of family income** (almost all families with low incomes don't spend much on luxury items) while there is **great variation in the size of the residuals for wealthier families** (some families spend a great deal on luxury items while some are more moderate in their luxury spending).

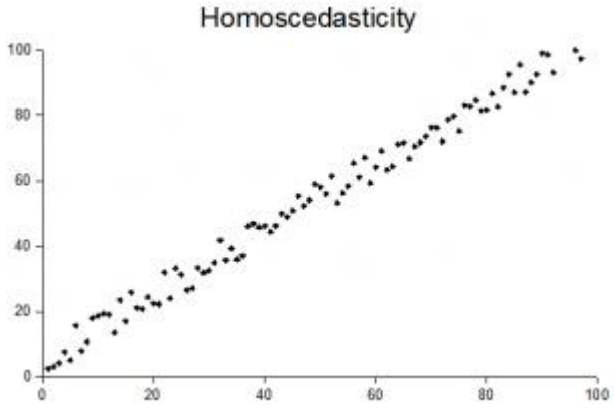
This **situation represents heteroscedasticity** because the size of the error varies across values of the independent variable.

The problem that heteroscedasticity presents for regression models is simple. Recall that ordinary least-squares (OLS) regression seeks to minimize residuals and in turn produce the smallest possible standard errors.

By definition, OLS regression gives equal weight to all observations, but when heteroscedasticity is present, the cases with larger disturbances have more "pull" than other observations. In this case, weighted least squares regression would be more appropriate, as it down-weights those observations with larger disturbances.



The points higher on the x-axis have a larger variance than smaller values.



7. How does Autocorrelation affect the linear regression

Autocorrelation refers to the **degree of correlation between the values of the same variables across different observations in the data.**

Example

In a survey, for instance, one might expect people from nearby geographic locations to provide more similar answers to each other than people who are more geographically distant. Similarly, students from the same class might perform more similarly to each other than students from different classes.

Thus, **autocorrelation can occur if observations are dependent in aspects other than time.** Autocorrelation can **cause problems in conventional analyses (such as ordinary least squares regression) that assume independence of observations.**

In a regression analysis, autocorrelation of the regression residuals can also occur if the model is incorrectly specified. For example, if you are attempting to model a simple linear relationship but the observed relationship is non-linear (i.e., it follows a curved or U-shaped function), then the residuals will be autocorrelated.

8. How does Normality affect the linear regression

When the **sample size is sufficiently large** (>200), the **normality assumption is not needed** at all as the Central Limit Theorem ensures that the distribution of disturbance term will approximate normality.

For small samples, the statistics to assess samples stability are less, so their results should be interpreted with caution. When the distribution of the disturbance term is found to deviate from normality, the best solution is to use a more conservative p value (.01 rather than .05) for conducting significance tests and constructing confidence intervals.

9. What is Central Limit Theorem

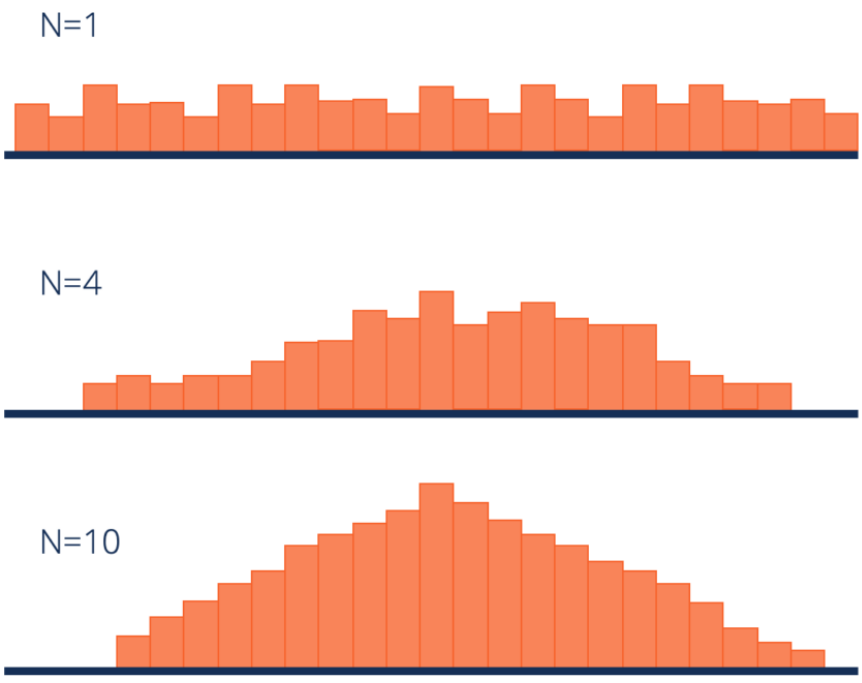
The central limit theorem (CLT) states that the **distribution of sample means approximates a normal distribution as the sample size gets larger.**

Sample sizes equal to or greater than 30 are considered sufficient for the CLT to hold.

A key aspect of CLT is that the average of the sample mean and standard deviations will equal the population mean and standard deviation.

A sufficiently large sample size can predict the characteristics of a population accurately.

As the sample sizes get bigger, the distribution of the means from the repeated samples tend to normalize and resemble a normal distribution. The result remains the same regardless of what the original shape of the distribution was. It can be illustrated in the figure below:



10. How to detect Multicollinearity

One popular detection method is based on the **bivariate correlation between two predictor variables.** If it's above .8 (or .7 or .9 or some other high number), the rule of thumb says you have multicollinearity.

But remember two factors, First, how high that correlation has to be before you're finding inflated variances depends on the sample size. There is no one good cut off number. **Second, it's possible that while no two variables are highly correlated, three or more together are multicollinear.**

Indicators of Multicollinearity:

1. **Very high standard errors for regression coefficients**

When standard errors are orders of magnitude higher than their coefficients, that's an indicator.

2. **The overall model is significant, but none of the coefficients are**

Remember that a p-value for a coefficient tests whether the unique effect of that predictor on Y is zero. If all predictors overlap in what they measure, there is little unique effect, even if the predictors as a group have an effect on Y.

3. **Large changes in coefficients when adding predictors**

If the predictors are completely independent of each other, their coefficients won't change at all when you add or remove one. But the more they overlap, the more drastically their coefficients will change.

4. **Coefficients have signs opposite what you'd expect from theory**

Be careful here as you don't want to disregard an unexpected finding as problematic. Not all effects opposite theory indicate a problem with the model. That said, it could be multicollinearity and warrants taking a second look at other indicators.

5. **Coefficients on different samples are wildly different**

If you have a large enough sample, split the sample in half and run the model separately on each half. Wildly different coefficients in the two models could be a sign of multicollinearity.

6. **High Variance Inflation Factor (VIF) and Low Tolerance**

These two useful statistics are reciprocals of each other. So, either a high VIF or a low tolerance is indicative of multicollinearity. VIF is a direct measure of how much the variance of the coefficient (i.e. its standard error) is being inflated due to multicollinearity.

7. **High Condition Indices**

Condition indices are a bit strange. The basic idea is to run a Principal Components Analysis on all predictors. If they have a lot of shared information, the first Principal Component will be much higher than the last. Their ratio, the Condition Index, will be high if multicollinearity is present.

11. What is Standard Error?

- The standard error is the approximate standard deviation of a statistical sample population.
- The **standard error can include the variation between the calculated mean of the population and one which is considered known or accepted as accurate.**
- The more data points involved in the calculations of the mean, the smaller the standard error tends to be. When the standard error is small, the data is said to be more representative of the true mean. In cases where the standard error is large, the data may have some notable irregularities.

12. Regression model evaluation metrics

The MSE, MAE, RMSE and R-Squared metrics are **mainly used to evaluate the prediction error rates** and model performance in regression analysis.

1. **MAE (Mean absolute error)** represents the difference between the original and predicted values extracted by averaged the absolute difference over the data set.
2. **MSE (Mean Squared Error)** represents the difference between the original and predicted values extracted by squared the average difference over the data set.
3. **RMSE (Root Mean Squared Error)** is the error rate by the square root of MSE.
4. **R-squared (Coefficient of determination)** represents the coefficient of how well the values fit compared to the original values. The value from 0 to 1 interpreted as percentages. The higher the value is, the better the model is.

13. What is the difference between correlation and regression?

- **Regression establishes how x causes y to change**, and the results will change if x and y are swapped. With **correlation, x and y are variables that can be interchanged and get the same result**.
- **Correlation is a single statistic, or data point**, whereas **regression is the entire equation** with all of the data points that are represented with a line.
- **Correlation shows the relationship between the two variables**, while **regression allows us to see how one affects the other.**
- The data shown with **regression establishes a cause and effect**, when one changes, so does the other, and not always in the same direction. With **correlation, the variables move together.**

14. What is VIF? How do you calculate it?

- A variance inflation factor is basically a tool to help **identify the degree of multicollinearity**.
- Variance inflation factor measures how much the **behavior (variance) of an independent variable is influenced**, or inflated, by its interaction/correlation with the other independent variables.
- Variance inflation factors allow a quick measure of how much a **variable is contributing to the standard error in the regression**.
- When **significant multicollinearity issues exist, the variance inflation factor will be very large** for the variables involved.

For example,

If an economist wants to test whether there is a statistically significant relationship between the unemployment rate (as an independent variable) and the inflation rate (as the dependent variable). Including additional independent variables that are related to the unemployment rate, such a new initial jobless claim, would be likely to introduce multicollinearity into the model. The overall model might show strong, statistically sufficient explanatory power but be unable to identify if the effect is mostly due to the unemployment rate or to the new initial jobless claims. This is what the VIF would detect, and it would suggest possibly dropping one of the variables out of the model or finding some way to consolidate them to capture their joint effect, depending on what specific hypothesis the researcher is interested in testing.

VIF Values - A **VIF of 1** indicates two variables are **not correlated**, a VIF **between 1 and 5 indicates moderate correlation**, and a VIF **above 5 indicates high correlation.**

15. How is hypothesis testing used in linear regression?

The test focuses on the slope of the regression line -

$$Y = B0 + B1X$$

where B0 is a constant, B1 is the slope (also called the regression coefficient), X is the value of the independent variable, and Y is the value of the dependent variable.

If we find that the **slope of the regression line is significantly different from zero, we will conclude that there is a significant relationship between the independent and dependent variables.**

State the Hypotheses - If there is a **significant linear relationship** between the independent variable X and the dependent variable Y, the slope will not equal zero.

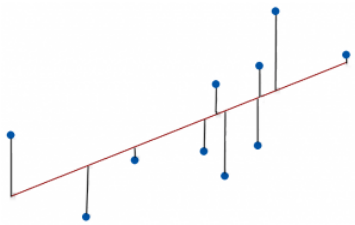
$$Ho: B1 = 0 \qquad Ha: B1 \neq 0$$

The **null hypothesis** states that the slope is equal to zero, and the **alternative hypothesis** states that the slope is not equal to zero.

16. What is OLS (Ordinary Least Square) method.

Ordinary least squares, or linear least squares, estimates the parameters in a regression model **by minimizing the sum of the squared residuals.**

This method draws a line through the data points that minimizes the sum of the squared differences between the observed values and the corresponding fitted values.



Residuals - Residual is the **difference between the observed value and the mean value that the model predicts** for that observation.

Fitted Value - A fitted value is a statistical model's prediction of the mean response value when you input the values of the predictors, factor levels, or components into the model. Suppose you have the following regression equation: y = 3X + 5. If you enter a value of 5 for the predictor, the fitted value is 20**. Fitted values are also called predicted values.**

17. What is the assumption of normality?

I. It means the normal distribution of the error term
II. The mean of the residuals should be zero
III. The standard deviation of the residuals should be constant

18. What is the assumption of homoscedasticity?

In simple terms it means the equal variance. There is no relationship between the error term and the predicted Y.

19. How to prevent heteroscedasticity?

I. It may be due to outliers
II. It may be due to omitted variable bias
III. Log transformation

20. How to detect autocorrelation?

DW (Durbin Watson) test is used to detect autocorrelation

I. If DW test statistics is less than 1 then there is strong autocorrelation
II. If DW test statistics is close to 2 then there is no autocorrelation
III. If DW test statistics is more than 3 then there is strong autocorrelation

21. What are the remedies to remove autocorrelation?

There is no remedy in linear regression. The modelers can try different models like AR, MA, ARMA or ARIMA

22. What is the generalized linear model?

Generalized linear model (GLM) is a flexible **generalization of ordinary linear regression that allows for response variables that have error distribution models other than a normal distribution.** The GLM generalizes linear regression by allowing the linear model to be related to the response variable via a link function and by allowing the magnitude of the variance of each measurement to be a function of its predicted value.

Maximum-likelihood estimation remains popular and is the default method on many statistical computing packages. Other approaches, including Bayesian approaches and least squares fits to variance stabilized responses, have been developed.

23. How to identify if the model is overfitting or underfitting?

Underfit model performs bad (low accuracy) on training and bad (low accuracy) on test.

Overfit model performs good (high accuracy) on training and bad (low accuracy) on test.

A **good model** performs good (high accuracy) on training and good (high accuracy) on test.

24. How do you interpret a linear regression model?

Residuals:

Min	1Q	Median	3Q	Max
-404.62	-135.97	-9.47	95.58	4484.79

Distribution of the Residuals.

- This shows the distribution of the residuals.
- The residuals are the difference between the prices in the training data set and the predicted prices by this model.
- **A negative residual** is an overestimate and **a positive residual** is an underestimate. Ideally, you should see a symmetrical distribution with a median near zero.
- In this case, the median of -9.47 is very close to zero.

Coefficients:

```
Coefficients:
                          Estimate Std. Error t value Pr(>|t|)
(Intercept)            -5191.3914  1405.3794  -3.694 0.000820 ***
Age                        0.9171     0.3502   2.619 0.013373 *
Income                   329.9655    83.0036   3.975 0.000375 ***
Gender                    -1.8965     0.4753  -3.990 0.000360 ***
City                       7.3211    13.6187   0.538 0.594587
City + State             336.9234   130.7021  -2.578 0.014753 *
```

Coefficient Name. Column 1 displays the names of the coefficients.

Estimate.

- These are the **estimated values for the coefficients.**
- The displayed coefficients are not standardized, for example, they are measured in their natural units, and thus cannot be compared with one another to determine which one is more influential in the model.

Standard Error.

- These are the **standard errors of the coefficients.**
- They can be used to **construct the lower and upper bounds for the coefficient**.
- **An example is Coefficient ± Standard Error, which provides an indication where the value may fall if another sample data set is used.**
- The standard error is also used to test whether the parameter is significantly different from 0. If a coefficient is significantly different from 0, then it has impact on the dependent variable (see t-value below).

t-value.

- The t-value is the ratio of the regression coefficient β to its standard error (**t = coefficient ÷ standard error**).
- The t statistic tests the hypothesis that a population regression coefficient is 0. **If a coefficient is different from zero, then it has a genuine effect on the dependent variable.**
- However, a coefficient may be different from zero, but if the difference is due to random variation, then it has no impact on the dependent variable. **In this example**, City is different from zero due to random variation and thus has no real impact on the dependent variable. The t-values are used to determine the P values (see below).

Pr(>|t|).

- The P value indicates whether the independent variable has statistically significant predictive capability.
- It essentially shows the probability of the coefficient being attributed to random variation.
- The lower the probability, the more significant the impact of the coefficient. **For example**, there is less than a 1.3% chance that the Age impact is due to random variation.

- The P value is automatically calculated by R by comparing the t-value against the Student's T distribution table. As a rule, a P value of less than 5% indicates significance. In theory, the P value for the constant could be used to determine whether the constant could be removed from the model.

*. The asterisks in the last column indicate the significance ranking of the P values.

Significance. codes: 0 '*' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1**

The significance codes indicate how certain we can be that the coefficient has an impact on the dependent variable. For example, a significance level of 0.01 indicates that there is less than a 0.1% chance that the coefficient might be equal to 0 and thus be insignificant. Stated differently, we can be 99.9% sure that it is significant. The significance codes (shown by asterisks) are intended for quickly ranking the significance of each variable.

Another Output:

Residual standard error: 237.4 on 37 degrees of freedom.

Multiple R-squared: 0.7958

Adjusted R-squared: 0.7384

F-statistic: 13.86 on 9 and 32 DF, p-value: 0.000000009629.

Residual Standard Error.

- This is the standard deviation of the error term in the regression equation (see Simple Regression, Error). The sample mean and the standard error can be used to construct the confidence interval for the mean.
- For example, it is the range of values within which the mean is expected to be if another representative data set is used.

Degrees of Freedom (Df).

- This column shows the degrees of freedom associated with the sources of variance. The total variance has N -1 degrees of freedom.
- A data set contains a number of observations - 60 in Fixed Deposit data set. The cases are individual pieces of information that can be used either to estimate parameters or variability. Each item (coefficient) being estimated costs one degree of freedom; the rest are used to estimate variability. For the FD data set, you are estimating nine coefficients. Therefore, there are 50 degrees of freedom for the residuals. For example, 50 = 60 - 1- 9.

R-squared.

- R-squared is a measure of the proportion of variability explained by the regression.
- It is a number between zero and one, and a value close to zero suggests a poor model. In a multiple regression, each additional independent variable may increase the R-squared without improving the actual fit.
- An adjusted R-squared is calculated that represents the more accurate fit with multiple independent variables. The adjusted R-squared takes into account both the number of observations and the number of independent variables. It is always lower than R-squared.
    - R-Squared takes a value between 0 and 1.
    - If R-Sq = 0 then the model does not explain any variability
    - If R-Sq = 1 then the model explains entire variability

F-statistics and P-value.

- The F-value, like the t-value (see t-value above), is calculated to measure the overall quality of the regression.
- The F-value is the Mean Square Regression divided by the Mean Square Residual. It is calculated on 9 Df for the coefficients and 50 Df for the residuals.
- The P-value associated with this F-value is very small (5.596e-15). The P-value is a measure of how confident you can be that the independent variables reliably predict the dependent variable.
- P stands for probability and is usually interpreted as the probability that test data does not represent accurately the population from which it is drawn.
- If the P-value is 0.10, there is a 10% probability that the calculation for the test data is not true for the population. Conversely, you can be 90% certain that the results of the test data are true of the population.
- For example, if the P-value were greater than 0.05, the group of independent variables does not show a statistically significant relationship with the dependent variable, or that the group of independent variables does not reliably predict the dependent variable. Note that this is an overall significance test assessing whether the group of independent variables when used together reliably predict the dependent variable and does not address the ability of any of the particular independent variables to predict the dependent variable. The ability of each individual independent variable to predict the dependent variable is addressed in the coefficients table. (See P-values for the regression coefficients.)

25. What do you mean by adjusted R2? How is it different from R2?

Meaning of Adjusted R2

- Both R2 and the adjusted R2 give you an idea of how many data points fall within the line of the regression equation.
- However, there is one main difference between R2 and the adjusted R2: **R2 assumes that every single variable explains the variation in the dependent variable.**
- The **adjusted R2 tells you the percentage of variation explained by only the independent variables that actually affect the dependent variable.**

How Adjusted R2 Penalizes You

- The adjusted R2 will penalize you for adding independent variables (K in the equation) that do not fit the model.
- Why? In regression analysis, it can be tempting to add more variables to the data as you think of them. Some of those variables will be significant, but you can't be sure that significance is just by chance.
- The adjusted R2 will compensate for this by that penalizing you for those extra variables.
- While values are usually positive, they can be negative as well. This could happen if your R2 is zero; After the adjustment, the value can dip below zero.
- This usually indicates that your model is a poor fit for your data. Other problems with your model can also cause sub-zero values, such as not putting a constant term in your model.

Problems with R2 that are corrected with an adjusted R2

- R2 increases with every predictor added to a model.
- As R2 always increases and never decreases, it can appear to be a better fit with the more terms you add to the model. This can be completely misleading.
- Similarly, if your model has too many terms and too many high-order polynomials you can run into the problem of over-fitting the data. When you over-fit data, a misleadingly high R2 value can lead to misleading projections.

26. What is Pseudo R Square?

When running an ordinary least squares (OLS) regression, one common metric to assess model fit is the R-squared (R2). The R2 metric can is calculated as follows.

$$R^2 = 1 - [\Sigma_i(y_i - \hat{y}_i)^2]/[\Sigma_i(y_i - \bar{y})^2]$$

R-squared as explained variability –

- The denominator of the ratio can be thought of as the total variability in the dependent variable, or how much y varies from its mean.
- The numerator of the ratio can be thought of as the variability in the dependent variable that is not predicted by the model.
- Thus, this ratio is the proportion of the total variability unexplained by the model.
- Subtracting this ratio from one results in the proportion of the total variability explained by the model. The more variability explained, the better the model.

R-squared as improvement from null model to fitted model –

- The denominator of the ratio can be thought of as the sum of squared errors from the null model–a model predicting the dependent variable without any independent variables.
- In the null model, each y value is predicted to be the mean of the y values.
- Consider being asked to predict a y value without having any additional information about what you are predicting.
- The mean of the y values would be your best guess if your aim is to minimize the squared difference between your prediction and the actual y value.
- The numerator of the ratio would then be the sum of squared errors of the fitted model.
- The ratio is indicative of the degree to which the model parameters improve upon the prediction of the null model. The smaller this ratio, the greater the improvement and the higher the R-squared.

R-squared as the square of the correlation –

- The term “R-squared” is derived from this definition. R-squared is the square of the correlation between the model’s predicted values and the actual values. T
- his correlation can range from -1 to 1, and so the square of the correlation then ranges from 0 to 1.
- The greater the magnitude of the correlation between the predicted values and the actual values, the greater the R-squared, regardless of whether the correlation is positive or negative.

So then what is a pseudo R-squared?

- When running a logistic regression, many people would like a similar goodness of fit metric.
- An R-squared value does not exist, however, for logit regressions since these regressions rely on “maximum likelihood estimates arrived at through an iterative process.
- They are not calculated to minimize variance, so the OLS approach to goodness-of-fit does not apply.” However, there are a few variations of a pseudo R-squared which are analogs to the OLS R-squared. For instance:
    - Efron’s Pseudo R-Squared
    - McFadden’s Pseudo R-Squared
    - McFadden’s Pseudo R-Squared (adjusted).

27. What is TSS, ESS and RSS?

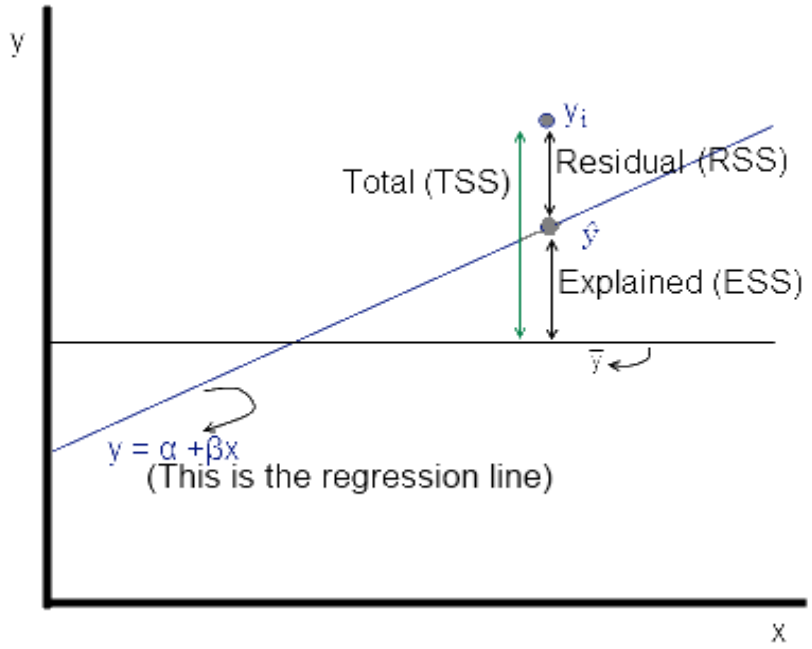
TSS, RSS and ESS (Total Sum of Squares, Residual Sum of Squares and Explained Sum of Squares)

TSS = ESS + RSS

Total variability = Explained variability + Unexplained variability

Consider the diagram below.

- Yi is the actual observed value of the dependent variable, y-hat is the value of the dependent variable according to the regression line, as predicted by our regression model.
- What we want to get is a feel for is the variability of actual y around the regression line, ie, the volatility of ϵ. This is given by the distance yi minus y-hat.



$$TSS = \Sigma (y_i - \bar{y})^2$$
$$RSS = \Sigma (y_i - \hat{y})^2$$
$$ESS = \Sigma (\hat{y} - \bar{y})^2$$

28. How good is the regression?

- Intuitively, the regression line given by α + βx will be a more accurate prediction of y if the correlation between x and y is high.
- Generally, R2, called the coefficient of determination, is used to evaluate how good the 'fit' of the regression model is. R2 is calculated as ESS/TSS, ie the ratio of the explained variation to the total variation.

R2 = ESS/TSS

- R2 is also the same thing as the square of the correlation, which means that our initial intuition that the quality of our regression model depends upon the correlation of the variables was correct. (Note that in the ratio ESS/TSS, both the numerator and denominator are squares of some sort – which means this ratio explains how much of the 'variance' is explained, not standard deviation. Variance is always in terms of the square of the units, which makes it slightly difficult to interpret intuitively, which is why we have standard deviation.)

29. What is the difference between coefficient of determination, and coefficient of correlation?

- Coefficient of correlation is "R" value which is given in the summary table in the Regression output. R square is also called coefficient of determination.
- Multiply R times R to get the R square value. In other words Coefficient of Determination is the square of Coefficient of Correlation.
- R square or coeff. of determination shows percentage variation in y which is explained by all the x variables together. Higher the better. It is always between 0 and 1. It can never be negative – since it is a squared value.

It is easy to explain the R square in terms of regression. It is not so easy to explain the R in terms of regression.

Model Summary[b]

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate
1	.850[a]	.723	.690	4.57996

a. Predictors: (Constant), weight, horsepower

b. Dependent Variable: mpg

Coefficient of Correlation is the R value i.e. .850 (or 85%). Coefficient of Determination is the R square value i.e. .723 (or 72.3%). R square is simply square of R i.e. R times R.

Coefficient of Correlation

- It is the degree of relationship between two variables say x and y. It can go between -1 and 1. 1 indicates that the two variables are moving in unison.
- They rise and fall together and have perfect correlation.
- -1 means that the two variables are in perfect opposites. One goes up and other goes down, in perfect negative way.
- Any two variables in this universe can be argued to have a correlation value. If they are not correlated then the correlation value can still be computed which would be 0.
- The correlation value always lies between -1 and 1 (going thru 0 – which means no correlation at all – perfectly not related).
- Correlation can be rightfully explalined for simple linear regression – because you only have one x and one y variable.
- For multiple linear regression R is computed, but then it is difficult to explain because we have multiple variables invovled here.
- That's why R square is a better term. You can explain R square for both simple linear regressions and also for multiple linear regressions.

CODE (REGRESSION ASSUMPTIONS)

```
# Libraries

from sklearn import datasets
import pandas as pd
import seaborn as sns
import numpy as np
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
from statsmodels.stats.diagnostic import normal_ad
from statsmodels.stats.outliers_influence import variance_inflation_factor
from statsmodels.stats.stattools import durbin_watson
import pandas.util.testing as tm


# Boston Dataset for Regression

boston = datasets.load_boston()
df = pd.DataFrame(boston.data)
df.columns = boston.feature_names
df["HousePrice"] = boston.target
print("Top 5 Rows of the dataset")
df.head()
```

Top 5 Rows of the dataset

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	HousePrice
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33	36.2

```
# Initial Setup

# Before we test the assumptions, we'll need to fit our linear regression models.
# Fitting the model
boston_model = LinearRegression()
boston_model.fit(boston.data, boston.target)

# Returning the R^2 for the model
boston_r2 = boston_model.score(boston.data, boston.target)
print('R^2: {0}'.format(boston_r2))
```

```
R^2: 0.7406426641094095
```

*# **ASSUMPTION 1: Linearity***

 # This assumes that there is a linear relationship between the predictors e.g. Independent variables or features & the response variable (e.g. Dependent variable/label). This also assumes that the predictors are additive.

 # Why it can happen:
 # There may not just be a linear relationship among the data. Modeling is about trying to estimate a function that explains a process. Linear regression would not be a fitting estimator if there is no linear relationship.

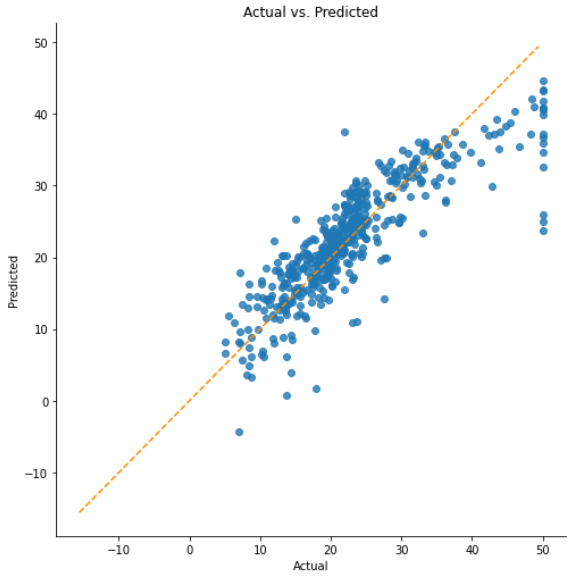
 # What it will affect:
 # The predictions will be extremely inaccurate because our model is underfitting.

 # How to detect it:
 # If there is only one predictor, this is pretty easy to test with a scatter plot.
 # Most cases aren't so simple, so we'll have to modify this by using a scatter plot to see our predicted values versus the actual values (in other words, view the residuals).
Ideally, the points should lie on or around a diagonal line on the scatter plot.

 # How to fix it:
 # Either adding polynomial terms to some of the predictors/applying nonlinear transformations.
If those do not work, try adding additional variables to help capture the relationship between the predictors and the label.

```
print('Assumption 1: Linear Relationship between the Target and the Feature')

# Initializing Variable name like model, independent features and label in dataset
model = boston_model
features = boston.data
label = boston.target

# Prediction and a table of Actual vs Predicted
predictions = model.predict(features)
df_results = pd.DataFrame({'Actual': label, 'Predicted': predictions})

# Calculating Residuals
df_results['Residuals'] = abs(df_results['Actual']) - abs(df_results['Predicted'])
print("Residual Calculation")
print(df_results.head())
print("\n")

# Plotting the actual vs predicted values
sns.lmplot(x='Actual', y='Predicted', data=df_results, fit_reg=False, size=7)
```

```
# Plotting the diagonal line
line_coords = np.arange(df_results.min().min(), df_results.max().max())
plt.plot(line_coords, line_coords, color='darkorange', linestyle='--')  # X and y points
plt.title('Actual vs. Predicted')
plt.show()

# Inference:
# We can see in this case that there is not a perfect linear relationship.
# Our predictions are biased towards lower values in both the lower end (around 5-10) and especially at the higher values (above 40).

Assumption 1: Linear Relationship between the Target and the Feature
Residual Calculation
   Actual  Predicted  Residuals
0    24.0  30.003843  -6.003843
1    21.6  25.025562  -3.425562
2    34.7  30.567597   4.132403
3    33.4  28.607036   4.792964
4    36.2  27.943524   8.256476
```




```
# ASSUMPTION 2: Normality of the Error Terms
# More specifically, this assumes that the error terms of the model are normally distributed. Linear regressions other than Ordinary Least Squares (OLS) may also assume normality of the predictors or the label, but that is not the case here

# Why it can happen:
# This can actually happen if either the predictors or the label are significantly non-normal. Other potential reasons could include the linearity assumption being violated or outliers affecting our model.
```

```
# What it will affect:
# A violation of this assumption could cause issues with either shrinking or inflati
ng our confidence intervals.

# How to detect it:
# We'll look at both a histogram and the p-value from the Anderson-Darling test for
normality.

# How to fix it:
# It depends on the root cause, but there are a few options. Nonlinear
transformations of the variables, excluding specific variables (such as long-tailed
variables)or removing outliers may solve this problem.

print('Assumption 2: The error terms are normally distributed', '\n')
print('Using the Anderson-Darling test for normal distribution')

# Performing the test on the residuals
p_value = normal_ad(df_results['Residuals'])[1]
print('p-value from the test - below 0.05 generally means non-normal:', p_value)

# Reporting the normality of the residuals
p_value_thresh=0.05
if p_value < p_value_thresh:
  print('Residuals are not normally distributed')
else:
  print('Residuals are normally distributed')

# Plotting the residuals distribution
plt.subplots(figsize=(12, 6))
plt.title('Distribution of Residuals')
sns.distplot(df_results['Residuals'])
plt.show()
print()
if p_value > p_value_thresh:
  print('Assumption satisfied')
else:
  print('Assumption not satisfied')
  print('Confidence intervals will likely be affected')
  print('Try performing nonlinear transformations on variables')

# Check for another one variable
print("Skewness of RESIDUAL :",df_results['Residuals'].skew())
if df_results['Residuals'].skew() > 1:
  print('Residual is Positive skewed')
elif df_results['Residuals'].skew() < -1:
  print('Residual is Negative skewed')
else:
  print('Residual is not skewed')
```

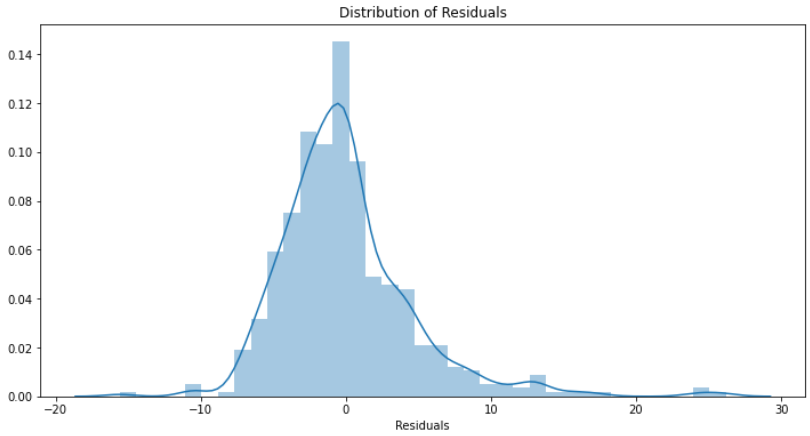
 # Inference: *This isn't ideal, and we can see that our model is biasing towards under-estimating*

Assumption 2: The error terms are normally distributed

Using the Anderson-Darling test for normal distribution

p-value from the test - below 0.05 generally means non-normal: 8.311128328210816e-25

Residuals are not normally distributed



Assumption not satisfied

Confidence intervals will likely be affected

Try performing nonlinear transformations on variables

Skewness of RESIDUAL: 1.533464025711795

Residual is Positive skewed

 # ASSUMPTION 3 - No Multicollinearity among Predictors

 # This assumes that the predictors used in the regression are not correlated with ea ch
other. This won't render our model unusable if violated, but it will cause issues wi th the interpretability of the model.

 # Why it can happen:

 # A lot of data is just naturally correlated. For example, if trying to predict a house price with square footage, the number of bedrooms, and the number of bathrooms , we can expect to see correlation between those three variables because bedrooms and bathrooms make up a portion of square footage.

 # What it will affect:

 # Multicollinearity causes issues with the interpretation of the coefficients. Specifically, you can interpret a coefficient as "an increase of 1 in this predictor results in a change of (coefficient) in the response variable, holding all other predictors constant. This becomes problematic when multicollinearity is present because we can't hold correlated predictors constant. Additionally, it increases the standard error of the coefficients, which results in them potentially showing as sta tistically insignificant when they might actually be significant.

 # How to detect it:

 # There are a few ways, but we will use a heatmap of the correlation as a visual aid and examine the variance inflation factor (VIF).

```
# How to fix it:
# This can be fixed by other removing predictors with a high variance inflation fact
or (VIF) or performing dimensionality reduction.

print('Assumption 3: Little to no multicollinearity among predictors')
# Plotting the heatmap
plt.figure(figsize = (10,8))
feature_names = boston.feature_names
sns.heatmap(pd.DataFrame(features, columns=feature_names).corr(), annot=True)
plt.title('Correlation of Variables')
plt.show()

print('Variance Inflation Factors (VIF)')
print('> 10: An indication that multicollinearity may be present')
print('> 100: Certain multicollinearity among the variables')
print('-------------------------------------')

# Gathering the VIF for each variable
VIF = [variance_inflation_factor(features, i) for i in range(features.shape[1])]
for idx, vif in enumerate(VIF):
  print('{0}: {1}'.format(feature_names[idx], vif))

# Gathering and printing total cases of possible or definite multicollinearity
possible_multicollinearity = sum([1 for vif in VIF if vif > 10])
definite_multicollinearity = sum([1 for vif in VIF if vif > 100])
print()
print('{0} cases of possible multicollinearity'.format(possible_multicollinearity))
print('{0} cases of definite multicollinearity'.format(definite_multicollinearity))
print()

if definite_multicollinearity == 0:
  if possible_multicollinearity == 0:
    print('Assumption satisfied')
  else:
    print('Assumption possibly satisfied')
    print()
    print('Coefficient interpretability may be problematic')
    print('Consider removing variables with a high Variance Inflation Factor (VIF)')
else:
  print('Assumption not satisfied')
  print()
  print('Coefficient interpretability will be problematic')
  print('Consider removing variables with a high Variance Inflation Factor (VIF)')

# Inference:
# This isn't quite as egregious as our normality assumption violation, but there is
possible multicollinearity for most of the variables in this dataset.
```

Assumption 3: Little to no multicollinearity among predictors


```
Variance Inflation Factors (VIF)
> 10: An indication that multicollinearity may be present
> 100: Certain multicollinearity among the variables
--------------------------------------
CRIM: 2.1003728199615224
ZN: 2.8440132669462637
INDUS: 14.485757706539331
CHAS: 1.1529518589418777
NOX: 73.89494652814788
RM: 77.94828304638538
AGE: 21.38685048994314
DIS: 14.699652383749175
RAD: 15.167724857920897
TAX: 61.227274009649456
PTRATIO: 85.02954731061801
B: 20.104942636229136
LSTAT: 11.102024772203539

10 cases of possible multicollinearity
0 cases of definite multicollinearity
Assumption possibly satisfied
Coefficient interpretability may be problematic
Consider removing variables with a high Variance Inflation Factor (VIF)
```

```
# ASSUMPTION 4:  No Autocorrelation of the Error Terms

# This assumes no autocorrelation of the error terms. Autocorrelation being present typically indicates that we are missing some information that should be captured by model.

# Why it can happen:
# In a time series scenario, there could be information about the past that we aren't capturing. In a non-time series scenario, our model could be systematically biased by either under or over predicting in certain conditions. Lastly, this could be a result of a violation of the linearity assumption.

# What it will affect:
# This will impact our model estimates.

# How to detect it:
# We will perform a Durbin-Watson test to determine if either positive or negative correlation is present.
# Alternatively, you could create plots of residual autocorrelations.

# How to fix it:
# A simple fix of adding lag variables can fix this problem.
# Alternatively, interaction terms, additional variables, or additional transformations may fix this.

print('Assumption 4: No Autocorrelation')

print('\nPerforming Durbin-Watson Test')
print('Values of 1.5 < d < 2.5 generally show that there is no autocorrelation in th
e data')
print('0 to 2< is positive autocorrelation')
print('>2 to 4 is negative autocorrelation')
print('-------------------------------------')
durbinWatson = durbin_watson(df_results['Residuals'])
print('Durbin-Watson:', durbinWatson)

if durbinWatson < 1.5:
  print('Signs of positive autocorrelation', '\n')
  print('Assumption not satisfied')
elif durbinWatson > 2.5:
  print('Signs of negative autocorrelation', '\n')
  print('Assumption not satisfied')
else:
  print('Little to no autocorrelation', '\n')
  print('Assumption satisfied')
```

```
# Inference :
# We're having signs of positive autocorrelation here, but we should expect this since we know our model is consistently under-predicting and our linearity assumption is being violated. Since this isn't a time series dataset, lag variables aren't possible.
# Instead, we should look into either interaction terms/additional transformations.
```

```
Assumption 4: No Autocorrelation

Performing Durbin-Watson Test
Values of 1.5 < d < 2.5 generally show that there is no autocorrelation in the data
0 to 2< is positive autocorrelation
>2 to 4 is negative autocorrelation
-------------------------------------
Durbin-Watson: 1.0715916506006853
Signs of positive autocorrelation

Assumption not satisfied
```

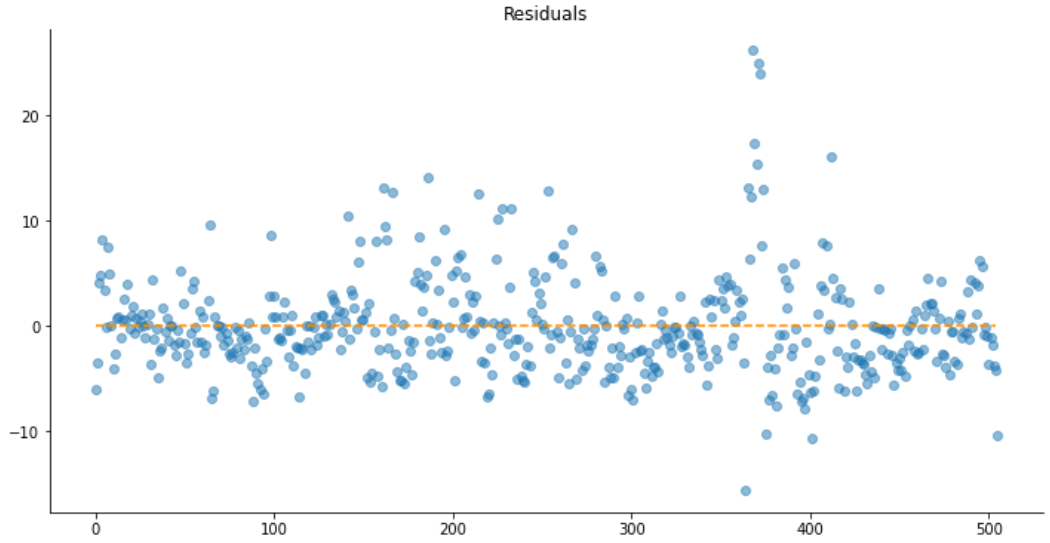
```
# ASSUMPTION 5: Homoscedasticity

# This assumes homoscedasticity, which is the same variance within our error terms.
# Heteroscedasticity, the violation of homoscedasticity, occurs when we don't have an even variance across the error terms.

# Why it can happen:
# Our model may be giving too much weight to a subset of the data, particularly where the error variance was the largest.

# What it will affect:
# Significance tests for coefficients due to the standard errors being biased.
# Additionally, the confidence intervals will be either too wide or too narrow.

# How to detect it:
# Plot the residuals and see if the variance appears to be uniform.

# How to fix it:
# Heteroscedasticity (can you tell I like the scedasticity words?) can be solved either by using weighted least squares regression instead of the standard OLS or transforming either the dependent or highly skewed variables. Performing a log transformation on the dependent variable is not a bad place to start.

print('Assumption 5: Homoscedasticity of Error Terms', '\n')
print('Residuals should have relative constant variance')

# Plotting the residuals
plt.subplots(figsize=(12, 6))
ax = plt.subplot(111)  # To remove spines
plt.scatter(x=df_results.index, y=df_results.Residuals, alpha=0.5)
plt.plot(np.repeat(0, df_results.index.max()), color='darkorange', linestyle='--')
```

```
ax.spines['right'].set_visible(False)  # Removing the right spine
ax.spines['top'].set_visible(False)  # Removing the top spine
plt.title('Residuals')
plt.show()
```

 # Inference:

 # We can't see a fully uniform variance across our residuals, so this is potentially problematic.
 # However, we know from our other tests that our model has several issues and is under predicting in many cases.

Assumption 5: Homoscedasticity of Error Terms
Residuals should have relative constant variance



 # Conclusion:

 # We can clearly see that a linear regression model on the Boston dataset violates number of assumptions which cause significant problems with the interpretation of the model itself.
 # It is dangerous to make decisions on a model that has violated assumptions because those decisions are effectively being formulated on made-up numbers. Not only that, but it also provides a false sense of security due to trying to be empirical in the decision-making process. Empiricism requires due diligence, which is why these assumptions exist and are stated up front.

Simple and Multiple Linear Regression Code

```
# Libraries
from sklearn import datasets
import pandas as pd
import seaborn as sns
import numpy as np
import statsmodels.stats.api as sms
from statsmodels.compat import lzip
from statsmodels.stats.outliers_influence import variance_inflation_factor
import statsmodels.api as sm
import pandas.util.testing as tm

# Load the Datasets

# Boston Dataset for Regression
boston = datasets.load_boston()
boston_pd = pd.DataFrame(boston.data)
boston_pd.columns = boston.feature_names
boston_pd["HOUSEPRICE"] = boston.target
boston_pd.head()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	HOUSEPRICE
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33	36.2

```
# Dataset overall Information
boston_pd.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    float64
 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    float64
 9   TAX         506 non-null    float64
```

```
# Let set the BASE MODEL on which we will improve

# Assigning Independent and Dependent variables
dependent_variable = boston_pd[['HOUSEPRICE']]
print("DEPENDENT VARIABLE : ",dependent_variable.columns)
independent_variables = boston_pd[boston_pd.columns[0:12]]
print("INDEPENDENT VARIABLES : ")
print(independent_variables.columns)
```

```
DEPENDENT VARIABLE :  Index(['HOUSEPRICE'], dtype='object')
INDEPENDENT VARIABLES :
Index(['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX',
       'PTRATIO', 'B'],
      dtype='object')
```

In [5]:

```
# Add constant to the dataframe
# In Statsmodel library, intercept is not considered. So we need to add Manually
independent_variables = sm.add_constant(independent_variables)
print("After Adding constant, Independent Variable names are")
print(independent_variables.columns)
```

```
After Adding constant, Independent Variable names are
Index(['const', 'CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS',
       'RAD', 'TAX', 'PTRATIO', 'B'],
      dtype='object')
```

In [6]:

```
# Base Model

import statsmodels.api as sm
from statsmodels.api import OLS
base_model = OLS(dependent_variable, independent_variables).fit()
base_model.summary()
```

Dep. Variable:	HOUSEPRICE	**R-squared:**	0.684
Model:	OLS	**Adj. R-squared:**	0.677
Method:	Least Squares	**F-statistic:**	89.01
Date:	Tue, 26 Jan 2021	**Prob (F-statistic):**	4.90e-115
Time:	13:35:59	**Log-Likelihood:**	-1548.6
No. Observations:	506	**AIC:**	3123.
Df Residuals:	493	**BIC:**	3178.
Df Model:	12		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	20.6526	5.368	3.848	0.000	10.106	31.199
CRIM	-0.1599	0.036	-4.467	0.000	-0.230	-0.090
ZN	0.0389	0.015	2.573	0.010	0.009	0.069
INDUS	-0.0279	0.068	-0.413	0.680	-0.161	0.105
CHAS	3.2166	0.948	3.393	0.001	1.354	5.079
NOX	-20.4846	4.201	-4.877	0.000	-28.738	-12.231
RM	6.1231	0.389	15.731	0.000	5.358	6.888
AGE	-0.0459	0.014	-3.356	0.001	-0.073	-0.019
DIS	-1.5549	0.220	-7.077	0.000	-1.987	-1.123
RAD	0.2816	0.073	3.852	0.000	0.138	0.425
TAX	-0.0117	0.004	-2.832	0.005	-0.020	-0.004
PTRATIO	-1.0142	0.144	-7.040	0.000	-1.297	-0.731
B	0.0136	0.003	4.657	0.000	0.008	0.019

Omnibus:	267.269	Durbin-Watson:	0.934
Prob(Omnibus):	0.000	Jarque-Bera (JB):	2542.250
Skew:	2.108	Prob(JB):	0.00
Kurtosis:	13.139	Cond. No.	1.48e+04

```
# Model Properties

# Normality of the residuals
# Jarque-Bera test:
name = ['Jarque-Bera', 'Chi^2 two-tail prob.', 'Skew', 'Kurtosis']
test = sms.jarque_bera(base_model.resid)
base_model_properties_1 = lzip(name, test)
print("BASE MODEL PARAMETERS")
print("\n")
print("Normality of the residuals")
print("Jarque-Bera test : ")
print("JB, ideal is close to 0")
print("Probability(JB), less than 0.05")
print("Skewness, ideal is -1 to 1")
print("Kurtosis, ideal value is 3")
print(pd.DataFrame(base_model_properties_1,columns = ['Test','Values']).round(2))

# Omni test:
print("\n")
print("Normality of the residuals (also)")
print("Omni Test : ")
print("Omni, ideal is close to 0")
print("Probability(Omni), less than 0.05")
```

```
name = ['Chi^2', 'Two-tail probability']
test = sms.omni_normtest(base_model.resid)
print(pd.DataFrame(lzip(name, test),columns = ['Test','Values']).round(2))

# Multicollinearity
print("\n")
print("Multicollinearity")
print("Ideal Condition Number will be less than 30")
print("Condition number:",np.linalg.cond(base_model.model.exog).round(2))

# Adjusted R Square
print("\n")
print("Model Performance")
print("Adjusted R Square, ideal is close to 1")
print("Adjusted R Square : ",base_model.rsquared_adj.round(2))

# F Statistics
print("\n")
print("Significance of Independent Variables")
print("Probability(F Stat), less than 0.05 suggests independent variables are
important")
print("F Statistics : ",base_model.fvalue.round(2))
print("Probability of F Statistics : ",base_model.f_pvalue.round(2))

# Durbin Watson Test
print("\n")
print("Homoscedasticity of Error")
print("Durbin Watson, ideal is between 1 to 2")
print("Durbin Watson Value : 0.78")
```

BASE MODEL PARAMETERS

```
Normality of the residuals
Jarque-Bera test :
JB, ideal is close to 0
Probability(JB), less than 0.05
Skewness, ideal is -1 to 1
Kurtosis, ideal value is 3
                   Test   Values
0           Jarque-Bera  2542.25
1  Chi^2 two-tail prob.     0.00
2                  Skew     2.11
3              Kurtosis    13.14
Normality of the residuals (also)
Omni Test :
Omni, ideal is close to 0
Probability(Omni), less than 0.05
                   Test  Values
0                Chi^2  267.27
1  Two-tail probability    0.00
```

Multicollinearity
Ideal Condition Number will be less than 30
Condition number: 14793.16

Model Performance
Adjusted R Square, ideal is close to 1
Adjusted R Square : 0.68

Significance of Independent Variables
Probability(F Stat), less than 0.05 suggests independent variables are important
F Statistics : 89.01
Probability of F Statistics : 0.0

Homoscedasticity of Error
Durbin Watson, ideal is between 1 to 2
Durbin Watson Value : 0.78

 # Output Explaination

 # Omnibus/Prob(Omnibus) -
 # A test of the skewness and kurtosis of the residual. We hope to see a value close to zero which would indicate normalcy. The Prob (Omnibus) indicates the probability that the residuals are normally distributed. We hope to see something close to 1 here. In this case Omnibus = 267 (way higher than 1) and Prob(Omnibus) = 0 which is way high (normally = 1) and the Prob (Omnibus) is 0 which is way too low so the data is not normal, not ideal.

 # Skew -
 # Measure of data symmetry. We want to see something close to zero, indicating the residual distribution is normal. Note that this value also drives the Omnibus
 # In this case, Skewness = 2.1, way higher than 0 so error/residual is skewed

 # Kurtosis -
 # Measure of "peakiness", or curvature of the data. Kurtosis of the normal distribu tion is 3.0. In this case, Kurtosis = 13.14 which is way too higher

 # Durbin-Watson -
 # Tests for homoscedasticity, We hope to have a value between 1 and 2. In this case , Durbin-Watson = 0.78 is close, but within limits.

 # Jarque-Bera (JB)/Prob(JB) -
 # It’s like the Omnibus test in that it tests both skew and kurtosis. It is also performed for the distribution analysis of the regression errors. A large value of JB test indicates that the errors are not normally distributed. In this case, JB = 2542 which is way too higher, so error are not normally distributed

 # Condition Number -

 # This test measures the sensitivity of a function's output as compared to its Input. When we have multicollinearity, we can expect much higher fluctuations to sm all changes in the data hence, we hope to see a relatively small number, something below 30. In this case, Condition Number = well above 30, so multicollinearity Present.

 # R Square and Adjusted R Square -

 # Both measures model performance and Possible values range from 0.0 to 1.0. The Ad justed R Squared value is always a bit lower than the Multiple R-Squared value. Adjusted R Square consequently is a more accurate measure of model performance. Adding an additional explanatory variable to the model will likely increase the Mul tiple R-Squared value but decrease the Adjusted R-Squared value Adjusted R Square w ill only increase when good variables are added in the model Higher the Adjusted R Square, better is the model. In this case, Adjusted R Square = 0.68 which can be im prove

 # F Statistics and Prob(F Statistics) -

 # This test for overall significance has the following two hypothesis:
 # Null hypothesis: Model with no independent variables fits the data as well as you r model.
 # Alternative hypothesis: Model fits the data better than the intercept-only model.
 # In this case, P(F-Statistics) = less than 0.05 suggests Independent variables are important

 # P Value -

 # If the p-value for a variable is less than your significance level (0.05) sample data provide enough evidence to reject the null hypothesis for the entire populatio n. Null hypothesis that the variable has no correlation with the dependent variable . If No correlation, no association between the changes in the independent & depend ent variable. In this case, Remove all variables with P values greater than 0.05

 # Log-Likelihood, AIC and BIC are for LOGISTIC REGRESSION MODEL

 # Let test the Base Model
 # Dividing Data into training and test set

```
from sklearn.model_selection import train_test_split
data_train, data_test, house_price_train, house_price_test = train_test_split(indep endent_variables,dependent_variable,train_size = 0.7,random_state = 21)
print("Training Data Shape :", data_train.shape)
print("Training House Price Shape :", house_price_train.shape)
print("Test Data Shape :",data_test.shape)
print("Test House Price Shape :",house_price_test.shape)
```

```
Training Data Shape : (354, 13)
Training House Price Shape : (354, 1)
Test Data Shape : (152, 13)
Test House Price Shape : (152, 1)
```

```
# Base Model
Model_1 = OLS(house_price_train,data_train).fit()
Model_1.summary()
```

Dep. Variable:	HOUSEPRICE	**R-squared:**	0.691
Model:	OLS	**Adj. R-squared:**	0.680
Method:	Least Squares	**F-statistic:**	63.54
Date:	Tue, 26 Jan 2021	**Prob (F-statistic):**	2.32e-79
Time:	13:35:59	**Log-Likelihood:**	-1069.2
No. Observations:	354	**AIC:**	2164.
Df Residuals:	341	**BIC:**	2215.
Df Model:	12		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	25.7285	6.186	4.159	0.000	13.562	37.895
CRIM	-0.2415	0.055	-4.364	0.000	-0.350	-0.133
ZN	0.0495	0.018	2.676	0.008	0.013	0.086
INDUS	-0.0334	0.075	-0.442	0.658	-0.182	0.115
CHAS	2.6172	1.052	2.488	0.013	0.548	4.686
NOX	-20.1834	4.897	-4.121	0.000	-29.816	-10.550
RM	5.3255	0.443	12.028	0.000	4.455	6.196
AGE	-0.0414	0.016	-2.615	0.009	-0.073	-0.010
DIS	-1.5677	0.252	-6.222	0.000	-2.063	-1.072
RAD	0.3606	0.085	4.229	0.000	0.193	0.528
TAX	-0.0131	0.005	-2.837	0.005	-0.022	-0.004
PTRATIO	-1.1124	0.173	-6.418	0.000	-1.453	-0.772
B	0.0175	0.004	4.936	0.000	0.011	0.025

Omnibus:	186.735	**Durbin-Watson:**	2.016
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	1626.044
Skew:	2.039	**Prob(JB):**	0.00
Kurtosis:	12.675	**Cond. No.**	1.49e+04

```
# Model Properties

# Normality of the residuals
# Jarque-Bera test:
name = ['Jarque-Bera', 'Chi^2 two-tail prob.', 'Skew', 'Kurtosis']
test = sms.jarque_bera(Model_1.resid)
model_properties_1 = lzip(name, test)
print("MODEL 1 PARAMETERS")
print("\n")
print("Normality of the residuals")
```

```
print("Jarque-Bera test : ")
print("JB, ideal is close to 0")
print("Probability(JB), less than 0.05")
print("Skewness, ideal is -1 to 1")
print("Kurtosis, ideal value is 3")
print(pd.DataFrame(model_properties_1,columns = ['Test','Values']).round(2))

# Omni test:
print("Omni Test : ")
print("Omni, ideal is close to 0")
print("Probability(Omni), less than 0.05")
name = ['Chi^2', 'Two-tail probability']
test = sms.omni_normtest(Model_1.resid)
print(pd.DataFrame(lzip(name, test),columns = ['Test','Values']).round(2))

# Multicollinearity
print("\n")
print("Multicollinearity")
print("Ideal Condition Number will be less than 30")
print("Condition number:",np.linalg.cond(Model_1.model.exog).round(2))

# Adjusted R Square
print("\n")
print("Model Performance")
print("Adjusted R Square, ideal is close to 1")
print("Adjusted R Square : ",Model_1.rsquared_adj.round(2))

# F Statistics
print("\n")
print("Significance of Independent Variables")
print("Probability(F Stat), less than 0.05 suggests independent variables are impor
tant")
print("F Statistics : ",base_model.fvalue.round(2))
print("Probability of F Statistics : ",Model_1.f_pvalue.round(2))

# Durbin Watson Test
print("\n")
print("Homoscedasticity of Error")
print("Durbin Watson, ideal is between 1 to 2")
print("Durbin Watson Value : 1.99")
```

```
MODEL 1 PARAMETERS

Normality of the residuals
Jarque-Bera test :
JB, ideal is close to 0
Probability(JB), less than 0.05
Skewness, ideal is -1 to 1
```

```
Kurtosis, ideal value is 3
                   Test   Values
0           Jarque-Bera  1626.04
1  Chi^2 two-tail prob.     0.00
2                  Skew     2.04
3              Kurtosis    12.68
```

Omni Test:

```
Omni, ideal is close to 0
Probability(Omni), less than 0.05
                   Test  Values
0                Chi^2  186.74
1  Two-tail probability    0.00
```

Multicollinearity

```
Ideal Condition Number will be less than 30
Condition number: 14949.83
```

Model Performance

```
Adjusted R Square, ideal is close to 1
Adjusted R Square :  0.68
```

Significance of Independent Variables

```
Probability(F Stat), less than 0.05 suggests independent variables are important
F Statistics :  89.01
Probability of F Statistics :  0.0
```

Homoscedasticity of Error

```
Durbin Watson, ideal is between 1 to 2
Durbin Watson Value : 1.99
```

 # Base Model Test and Metrics

```
from sklearn import metrics
Prediction_1 = Model_1.predict(data_test)
print("Mean Absolute Error :",metrics.mean_absolute_error(house_price_test,Predicti
on_1).round(2))
print("Mean Squared Error :",metrics.mean_squared_error(house_price_test,Prediction
_1).round(2))
print("Root Mean Squared Error :",metrics.mean_squared_error(house_price_test,Predi
ction_1).round(2))
def mape(actual, pred):
    actual, pred = np.array(actual), np.array(pred)
    return np.mean(np.abs((actual - pred) / actual)) * 100

print("Mean Absolute Percentage Error :",mape(house_price_test,Prediction_1).round(
2))
```

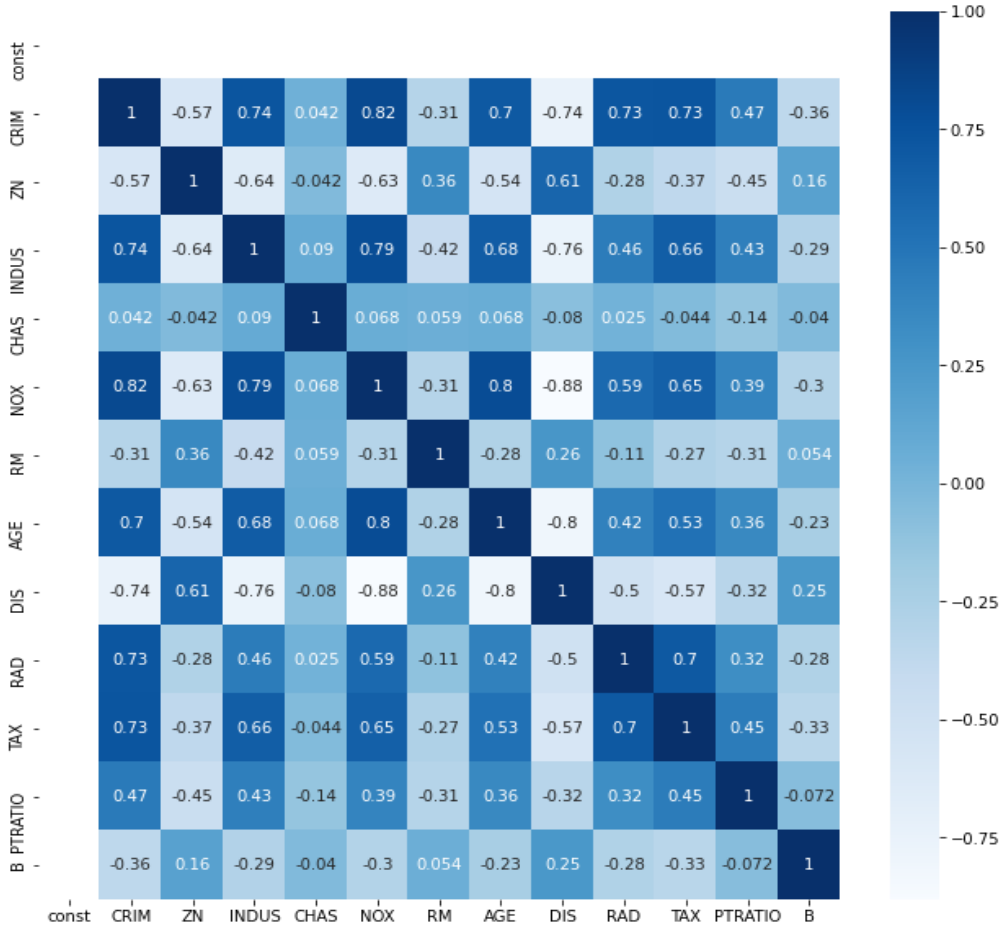
```
# MAE is Absolute difference between the Model Predictions & True(Actual) values
# MSE is Square difference between Model Predictions & True values
# MSE affected by Outliers, because we Square the distance
# RMSE, square root(MSE) therefore cancelling the affect the square
# MAE value can range from 0 to infinity, difficult to interpret
# MAPE is equivalent to MAE but provide the error in a percentage(0-100)
# MAPE = 52,average difference between predicted & actual is 52% which is very high
```

```
Mean Absolute Error : 4.01
Mean Squared Error : 33.66
Root Mean Squared Error : 33.66
Mean Absolute Percentage Error : 52.39
```

```
# Model 2, first Data Cleaning
```

```
# Correlation
# Increasing the size of the plot
import matplotlib.pyplot as plt
fig, ax = plt.subplots(figsize=(11,11)) # Sample figsize in inches, change the numb
er accordingly
sns.heatmap(independent_variables.corr(method = 'spearman'), annot= True, cmap = 'B
lues',ax = ax)
```

```
# Inference: Lots of High Correlation Values
```



```
# Filtering Highly Positive or Negative Correlated Values

# Create correlation matrix
corr_matrix = independent_variables.corr().round(2)
corr_matrix
```

	const	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B
const	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
CRIM	NaN	1.00	-0.20	0.41	-0.06	0.42	-0.22	0.35	-0.38	0.63	0.58	0.29	-0.39
ZN	NaN	-0.20	1.00	-0.53	-0.04	-0.52	0.31	-0.57	0.66	-0.31	-0.31	-0.39	0.18
INDUS	NaN	0.41	-0.53	1.00	0.06	0.76	-0.39	0.64	-0.71	0.60	0.72	0.38	-0.36
CHAS	NaN	-0.06	-0.04	0.06	1.00	0.09	0.09	0.09	-0.10	-0.01	-0.04	-0.12	0.05
NOX	NaN	0.42	-0.52	0.76	0.09	1.00	-0.30	0.73	-0.77	0.61	0.67	0.19	-0.38
RM	NaN	-0.22	0.31	-0.39	0.09	-0.30	1.00	-0.24	0.21	-0.21	-0.29	-0.36	0.13
AGE	NaN	0.35	-0.57	0.64	0.09	0.73	-0.24	1.00	-0.75	0.46	0.51	0.26	-0.27
DIS	NaN	-0.38	0.66	-0.71	-0.10	-0.77	0.21	-0.75	1.00	-0.49	-0.53	-0.23	0.29
RAD	NaN	0.63	-0.31	0.60	-0.01	0.61	-0.21	0.46	-0.49	1.00	0.91	0.46	-0.44
TAX	NaN	0.58	-0.31	0.72	-0.04	0.67	-0.29	0.51	-0.53	0.91	1.00	0.46	-0.44
PTRATIO	NaN	0.29	-0.39	0.38	-0.12	0.19	-0.36	0.26	-0.23	0.46	0.46	1.00	-0.18
B	NaN	-0.39	0.18	-0.36	0.05	-0.38	0.13	-0.27	0.29	-0.44	-0.44	-0.18	1.00

```
# Find index of feature columns with correlation greater than 0.75
Pos_corr_var=np.where(corr_matrix > 0.75)
Pos_corr_var=[(corr_matrix.columns[x],corr_matrix.columns[y]) for x,y in zip(*Pos_c
orr_var) if x!=y and x<y]
print("Highly Positive Related (> 0.75) : ")
print(Pos_corr_var)

# Find index of feature columns with correlation less than - 0.75
Neg_corr_var=np.where(corr_matrix < -0.75)
Neg_corr_var=[(corr_matrix.columns[x],corr_matrix.columns[y]) for x,y in zip(*Neg_c
orr_var) if x!=y and x<y]
print("Highly Negatively Related (< - 0.75) : ")
print(Neg_corr_var)

# Inference:
# Remove 'INDUS' / 'NOX', 'RAD'/'TAX , 'NOX'/'DIS'
# Removing NOX
independent_variables = independent_variables.drop(['NOX'], axis = 1)
# Now out of 'RAD' & 'TAX' which one to remove ? VIF
```

```
Highly Positive Related (> 0.75) : [('INDUS', 'NOX'), ('RAD', 'TAX')]
Highly Negatively Related (< - 0.75) : [('NOX', 'DIS')]
```

```
# Function for VIF
def calc_vif(X):
  # Calculating VIF
  vif = pd.DataFrame()
  vif["variables"] = X.columns
  vif["VIF"] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]
  return(vif)

print(pd.DataFrame(calc_vif(independent_variables).sort_values("VIF")))

# VIF = 1, no correlation between the independent variable and the other variables
# VIF exceeding 20 indicates high multicollinearity between this independent
variable and the others

# Inference: Suppose RM and TAX have more than 20 VIF then
# Remove 'RM' & 'TAX'
# independent_variables = independent_variables.drop(['RM', 'TAX'] ,axis = 1)
```

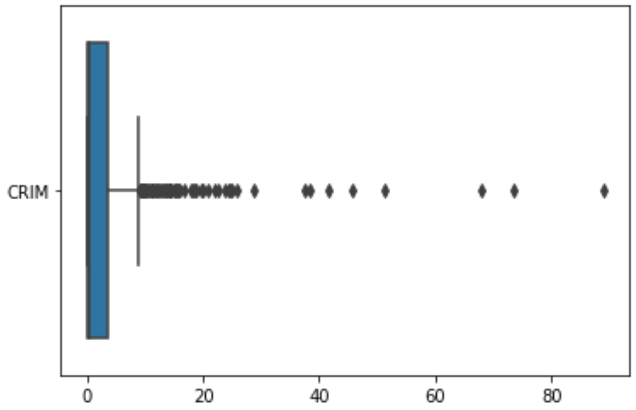
	variables	VIF
4	CHAS	1.069265
11	B	1.306190
5	RM	1.347445
10	PTRATIO	1.602691
1	CRIM	1.746058
2	ZN	2.289884
6	AGE	2.480534
7	DIS	3.640058
3	INDUS	3.677764
8	RAD	7.317727
9	TAX	8.957485
0	const	344.083623

```
# Outlier in the data (Column Wise Checking)

# CRIM Column
print("Quantile Distribution")
print(pd.DataFrame(independent_variables['CRIM'].quantile([0, 0.1, 0.2, 0.3, 0.4, 0
.5, 0.6, 0.7, 0.8, 0.9,0.95 ,1])))
sns.boxplot(data = independent_variables, orient = 'h', order = ['CRIM'])

# Inference :
# Capping to 95% Datapoints
independent_variables['CRIM'] = np.where(independent_variables['CRIM'].round(1) > 1
5.0, 15.0, independent_variables['CRIM'])
```

Quantile Distribution

	CRIM
0.00	0.006320
0.10	0.038195
0.20	0.064170
0.30	0.099245
0.40	0.150380

```
0.50   0.256510
0.60   0.550070
0.70   1.728440
0.80   5.581070
0.90  10.753000
0.95  15.789150
1.00  88.976200
```

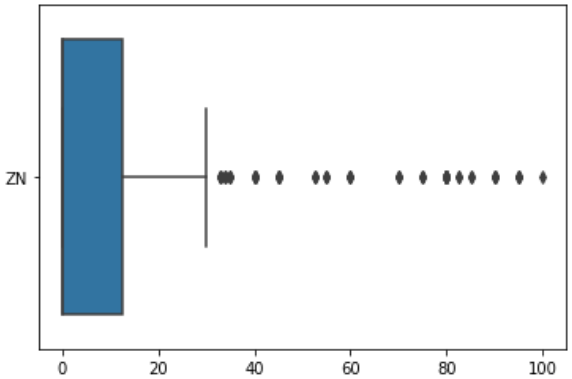


 # ZN Column

```
print("Quantile Distribution")
print(pd.DataFrame(independent_variables['ZN'].quantile([0, 0.1, 0.2, 0.3, 0.4, 0.5
, 0.6, 0.7, 0.8, 0.9,0.95 ,1])))
sns.boxplot(data = independent_variables, orient = 'h', order = ['ZN'])
```

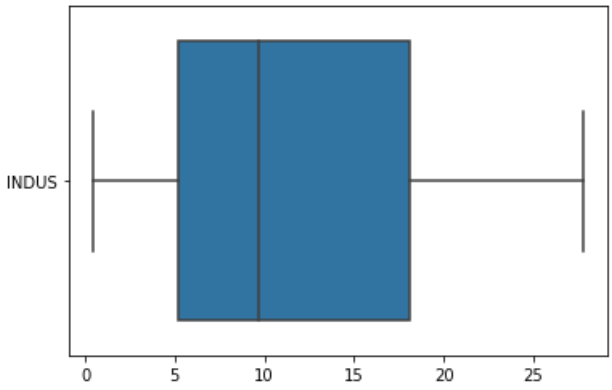
 # Inference: Most Values (70%) data points are zero. So let's not do any capping !!

```
Quantile Distribution
          ZN
0.00    0.0
0.10    0.0
0.20    0.0
0.30    0.0
0.40    0.0
0.50    0.0
0.60    0.0
0.70    0.0
0.80   20.0
0.90   42.5
0.95   80.0
1.00  100.0
```



```
# CRIM Column

print("Quantile Distribution")
print(pd.DataFrame(independent_variables['INDUS'].quantile([0, 0.1, 0.2, 0.3, 0.4,
0.5, 0.6, 0.7, 0.8, 0.9,0.95 ,1])))
sns.boxplot(data = independent_variables, orient = 'h', order = ['INDUS'])

# Inference : NO OUTLIERS
```

```
Quantile Distribution
       INDUS
0.00   0.46
0.10   2.91
0.20   4.39
0.30   5.96
0.40   7.38
0.50   9.69
0.60  12.83
0.70  18.10
0.80  18.10
0.90  19.58
0.95  21.89
1.00  27.74
```




```
# AGE Column
print("Quantile Distribution")
print(pd.DataFrame(independent_variables['AGE'].quantile([0, 0.1, 0.2, 0.3, 0.4, 0.
5, 0.6, 0.7, 0.8, 0.9,0.95 ,1])))
sns.boxplot(data = independent_variables, orient = 'h', order = ['AGE'])

# Inference : NO OUTLIERS
```

```
Quantile Distribution
         AGE
0.00    2.90
0.10   26.95
0.20   37.80
0.30   52.40
0.40   65.40
0.50   77.50
0.60   85.90
0.70   91.80
0.80   95.60
0.90   98.80
0.95  100.00
1.00  100.00
```

```
# Skewness in the Data
# If Skewness = 0, data is perfectly symmetrical
# If Skewness is less than -1 or greater +1, distribution is highly skewed
# If Skewness is between -1 and -0.5 or between 0.5 and 1, distribution is moderate
ly skewed
# If Skewness is between -0.5 and 0.5 then it is approximately symmetric

pd.DataFrame(independent_variables.skew().round(2),columns = ["Skewness"])

# Inference: Positive Skew - 'CRIM','ZN' & Negative Skew - 'Age' 'B'
```

	Skewness
const	0.00
CRIM	1.71
ZN	2.23
INDUS	0.30
RM	0.40
AGE	-0.60
DIS	0.55
RAD	1.00
TAX	0.67
PTRATIO	-0.80
B	-1.76

```
# Remove Skewness one by one (Column Wise)

# CRIM Variable
# Log Transformation
CRIM_Log = np.log(independent_variables['CRIM'])

# Square Root Transformation
CRIM_sqrt = np.sqrt(independent_variables['CRIM'])

# Cube Root Transformation
CRIM_cbrt = np.cbrt(independent_variables['CRIM'])

# Box Cox Transformation
from scipy import stats
CRIM_BoxCox = stats.boxcox(independent_variables['CRIM'])[0]
Variable = pd.Series(CRIM_BoxCox)

print("Comparing all Skewness values")
print(pd.DataFrame([independent_variables['CRIM'].skew(),CRIM_Log.skew(),CRIM_sqrt.
skew(),CRIM_cbrt.skew(),Variable.skew()],
             index=['Normal','Log','Square Root','Cube Root','Box Cox'],
             columns = ["Skewness"]))
```

```
# Inference: Best transformations is Log Transformation (Near to 0)

independent_variables['CRIM_Log'] = CRIM_Log
```

```
Comparing all Skewness values
             Skewness
Normal       1.705324
Log          0.321681
Square Root  1.168043
Cube Root    0.935891
Box Cox      0.083939
```

```
# ZN Variable
# Log Transformation
ZN_Log = np.log(independent_variables['ZN'])
# Square Root Transformation
ZN_sqrt = np.sqrt(independent_variables['ZN'])
# Cube Root Transformation
ZN_cbrt = np.cbrt(independent_variables['ZN'])

print("Comparing all Skewness values")
print(pd.DataFrame([independent_variables['ZN'].skew(),ZN_Log.skew(),ZN_sqrt.skew()
,ZN_cbrt.skew()],index=['Normal','Log','Square Root','Cube Root'], columns = ["Skew
ness"]))

# Inference: Best transformations is Cube Transformation (Near to 0)

independent_variables['ZN_Cbrt'] = ZN_cbrt
```

```
Comparing all Skewness values
             Skewness
Normal       2.225666
Log               NaN
Square Root  1.476293
Cube Root    1.262563
```

```
# B Variable
# Log Transformation
B_Log = np.log(independent_variables['B'])
# Square Root Transformation
B_sqrt = np.sqrt(independent_variables['B'])
# Cube Root Transformation
B_cbrt = np.cbrt(independent_variables['B'])

# Box Cox Transformation
from scipy import stats
B_BoxCox = stats.boxcox(independent_variables['B'])[0]
Variable = pd.Series(B_BoxCox)
```

```
print("Comparing all Skewness values")
print(pd.DataFrame([independent_variables['B'].skew(),B_Log.skew(),B_sqrt.skew(),B_
cbrt.skew(),Variable.skew()],
             index=['Normal','Log','Square Root','Cube Root','Box Cox'],
             columns = ["Skewness"]))

# Inference : Best transformations is Box Cox Transformation (Approx to zero)
# But avoiding Box Cox because it give values e raise to, second best is Square
Root Transformation
independent_variables['B_Sqrt'] = B_sqrt
```

```
Comparing all Skewness values
             Skewness
Normal      -1.760681
Log         -1.837693
Square Root -1.799513
Cube Root   -1.812319
Box Cox     -0.867610
```

```
# Removing variables (raw variables) which are transformed and keep one set also
(no removing)
independent_variables_v2 = independent_variables

# Removing variables
independent_variables = independent_variables.drop(['CRIM', 'ZN', 'B'], axis = 1)

# Round to 2 decimal point
independent_variables = independent_variables.round(2)
independent_variables.head()
```

	const	INDUS	RM	AGE	DIS	RAD	TAX	PTRATIO	CRIM_Log	ZN_Cbrt	B_Sqrt
0	1.0	2.31	6.58	65.2	4.09	1.0	296.0	15.3	-5.06	2.62	19.92
1	1.0	7.07	6.42	78.9	4.97	2.0	242.0	17.8	-3.60	0.00	19.92
2	1.0	7.07	7.18	61.1	4.97	2.0	242.0	17.8	-3.60	0.00	19.82
3	1.0	2.18	7.00	45.8	6.06	3.0	222.0	18.7	-3.43	0.00	19.87
4	1.0	2.18	7.15	54.2	6.06	3.0	222.0	18.7	-2.67	0.00	19.92

```
# Now Train the model

# Splitting Data into 70:30
data_train, data_test, house_price_train, house_price_test = train_test_split(indep
endent_variables,dependent_variable,train_size = 0.7,random_state = 21)

# Fitting the model
Model_2 = OLS(house_price_train,data_train).fit()
Model_2.summary()
```

Dep. Variable:	HOUSEPRICE	**R-squared:**	0.639
Model:	OLS	**Adj. R-squared:**	0.628
Method:	Least Squares	**F-statistic:**	60.65
Date:	Tue, 26 Jan 2021	**Prob (F-statistic):**	9.29e-70
Time:	13:36:02	**Log-Likelihood:**	-1096.8
No. Observations:	354	**AIC:**	2216.
Df Residuals:	343	**BIC:**	2258.
Df Model:	10		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	-7.1318	8.474	-0.842	0.401	-23.799	9.535
INDUS	-0.0841	0.080	-1.058	0.291	-0.240	0.072
RM	5.9865	0.461	12.981	0.000	5.079	6.894
AGE	-0.0709	0.017	-4.126	0.000	-0.105	-0.037
DIS	-1.5377	0.322	-4.768	0.000	-2.172	-0.903
RAD	0.2449	0.105	2.343	0.020	0.039	0.450
TAX	-0.0149	0.005	-3.054	0.002	-0.025	-0.005
PTRATIO	-1.0319	0.183	-5.641	0.000	-1.392	-0.672
CRIM_Log	-0.4998	0.381	-1.313	0.190	-1.248	0.249
ZN_Cbrt	0.2892	0.308	0.940	0.348	-0.316	0.895
B_Sqrt	1.3286	0.372	3.572	0.000	0.597	2.060

Omnibus:	202.916	**Durbin-Watson:**	2.107
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	2161.246
Skew:	2.191	**Prob(JB):**	0.00
Kurtosis:	14.283	**Cond. No.**	1.32e+04

```
# Prediction of Model 2

Mean Absolute Error : 3.9
Mean Squared Error : 34.54
Root Mean Squared Error : 34.54
Mean Absolute Percentage Error : 50.5
```

```
# Model 3
# Now lets drop variable based on P Values,suppose we want to drop RAD and CRIM_Log
independent_variables = independent_variables.drop(['RAD', 'CRIM_Log'], axis = 1)

# Splitting Data into 70:30
data_train, data_test, house_price_train, house_price_test = train_test_split(indep
endent_variables,dependent_variable,train_size = 0.7, random_state = 21)

# Fitting the model
Model_3 = OLS(house_price_train,data_train).fit()
Model_3.summary()
```

Dep. Variable:	HOUSEPRICE	**R-squared:**	0.633
Model:	OLS	**Adj. R-squared:**	0.624
Method:	Least Squares	**F-statistic:**	74.38
Date:	Tue, 26 Jan 2021	**Prob (F-statistic):**	1.83e-70
Time:	13:36:02	**Log-Likelihood:**	-1099.6
No. Observations:	354	**AIC:**	2217.
Df Residuals:	345	**BIC:**	2252.
Df Model:	8		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-11.3930	8.273	-1.377	0.169	-27.664	4.878
INDUS	-0.1324	0.077	-1.715	0.087	-0.284	0.019
RM	6.1019	0.461	13.241	0.000	5.196	7.008
AGE	-0.0769	0.017	-4.602	0.000	-0.110	-0.044
DIS	-1.5708	0.318	-4.943	0.000	-2.196	-0.946
TAX	-0.0068	0.003	-2.518	0.012	-0.012	-0.001
PTRATIO	-0.9154	0.177	-5.184	0.000	-1.263	-0.568
ZN_Cbrt	0.4088	0.302	1.356	0.176	-0.184	1.002
B_Sqrt	1.4215	0.360	3.947	0.000	0.713	2.130

Omnibus:	210.014	**Durbin-Watson:**	2.095
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	2347.177
Skew:	2.277	**Prob(JB):**	0.00
Kurtosis:	14.764	**Cond. No.**	1.28e+04

```
# Prediction of Model 3
Prediction_3 = Model_3.predict(data_test)
print("Mean Absolute Error :",metrics.mean_absolute_error(house_price_test,Predicti
on_3).round(2))
print("Mean Squared Error :",metrics.mean_squared_error(house_price_test,Prediction
_3).round(2))
print("Root Mean Squared Error :",metrics.mean_squared_error(house_price_test,Predi
ction_3).round(2))
print("Mean Absolute Percentage Error :",mape(house_price_test,Prediction_3).round(
2))

# Inference :
# Model 3 is performed well out of 3 models (Model 3 should be selected)
```

```
Mean Absolute Error : 3.83
Mean Squared Error : 34.57
Root Mean Squared Error : 34.57
Mean Absolute Percentage Error : 50.33
```

***# Overall Comparison* - ALL MODEL COMPARISION**

Normality of the residuals

Jarque-Bera test :
JB, ideal is close to 0
Probability(JB), less than 0.05
Base Model
('Jarque-Bera', 2542.250488008214)
('Chi^2 two-tail prob.', 0.0)
Model 1
('Jarque-Bera', 1626.0437912680602)
('Chi^2 two-tail prob.', 0.0)
Model 2
('Jarque-Bera', 2161.2461350595318)
('Chi^2 two-tail prob.', 0.0)
Model 3
('Jarque-Bera', 2347.1769653084443)
('Chi^2 two-tail prob.', 0.0)

Skewness and Kurtosis of Residual

Skewness, ideal is -1 to 1
Kurtosis, ideal value is 3
Base Model
('Skew', 2.1082399036809814)
('Kurtosis', 13.139149315831563)
Model 1
('Skew', 2.0386930060436312)
('Kurtosis', 12.675493803602887)
Model 2
('Skew', 2.191454152691215)
('Kurtosis', 14.283407131073531)
Model 3
('Skew', 2.2769330038791407)
('Kurtosis', 14.764053132284399)

Normality of the Residual also
Omni Test :

Omni, ideal is close to 0
Probability(Omni), less than 0.05
Base Model

	Test	Values
0	Chi^2	267.27
1	Two-tail probability	0.00

Model 1

	Test	Values
0	Chi^2	186.74
1	Two-tail probability	0.00

Model 2

	Test	Values
0	Chi^2	202.92
1	Two-tail probability	0.00

Model 3

	Test	Values
0	Chi^2	210.01
1	Two-tail probability	0.00

Multicollinearity

Ideal Condition Number will be less than 30
Base Model, Condition number: 14793.16
Model 1, Condition number: 14949.83
Model 2, Condition number: 13155.98
Model 3, Condition number: 12774.1

Model Performance

Adjusted R Square, ideal is close to 1
Base Model, Adjusted R Square : 0.68
Model 1, Adjusted R Square : 0.68
Model 2, Adjusted R Square : 0.63
Model 3, Adjusted R Square : 0.62

Significance of Independent Variables

Probability(F Stat), less than 0.05 suggests independent variables are important
Base Model, Probability of F Statistics : 0.0
Model 1, Probability of F Statistics : 0.0
Model 2, Probability of F Statistics : 0.0
Model 3, Probability of F Statistics : 0.0

Homoscedasticity of Error

Durbin Watson, ideal is between 1 to 2
Base Model, Durbin Watson Value : 0.78
Model 1, Durbin Watson Value : 0.78
Model 2, Durbin Watson Value : 2.1
Model 3, Durbin Watson Value : 2.07

Accuracy of the Model

MAPE should be close to 0
Base Model, will not have MAPE value because we train the data on full dataset
Model 1, Mean Absolute Percentage Error : 52.39
Model 2, Mean Absolute Percentage Error : 50.5
Model 3, Mean Absolute Percentage Error : 50.33