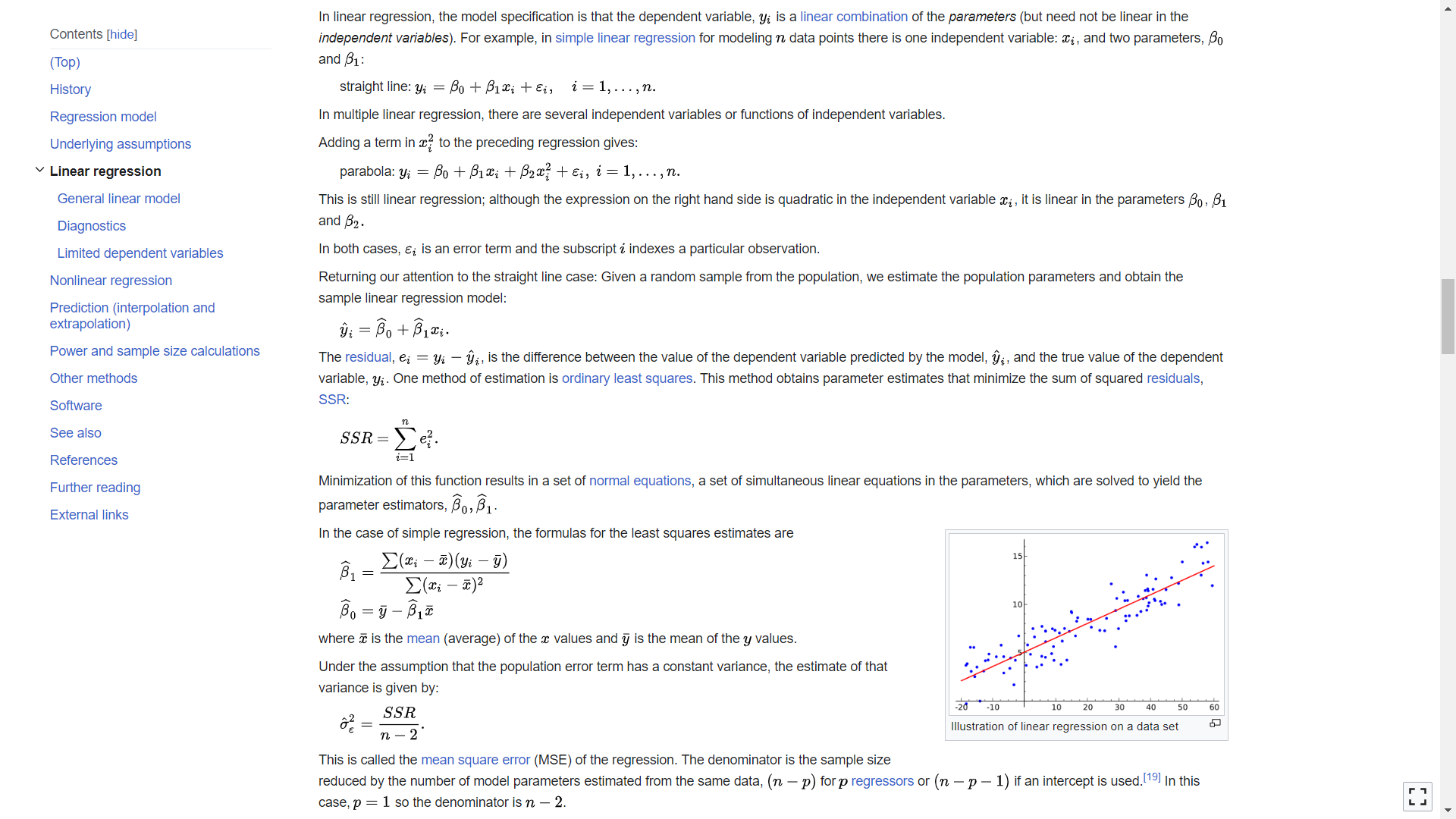
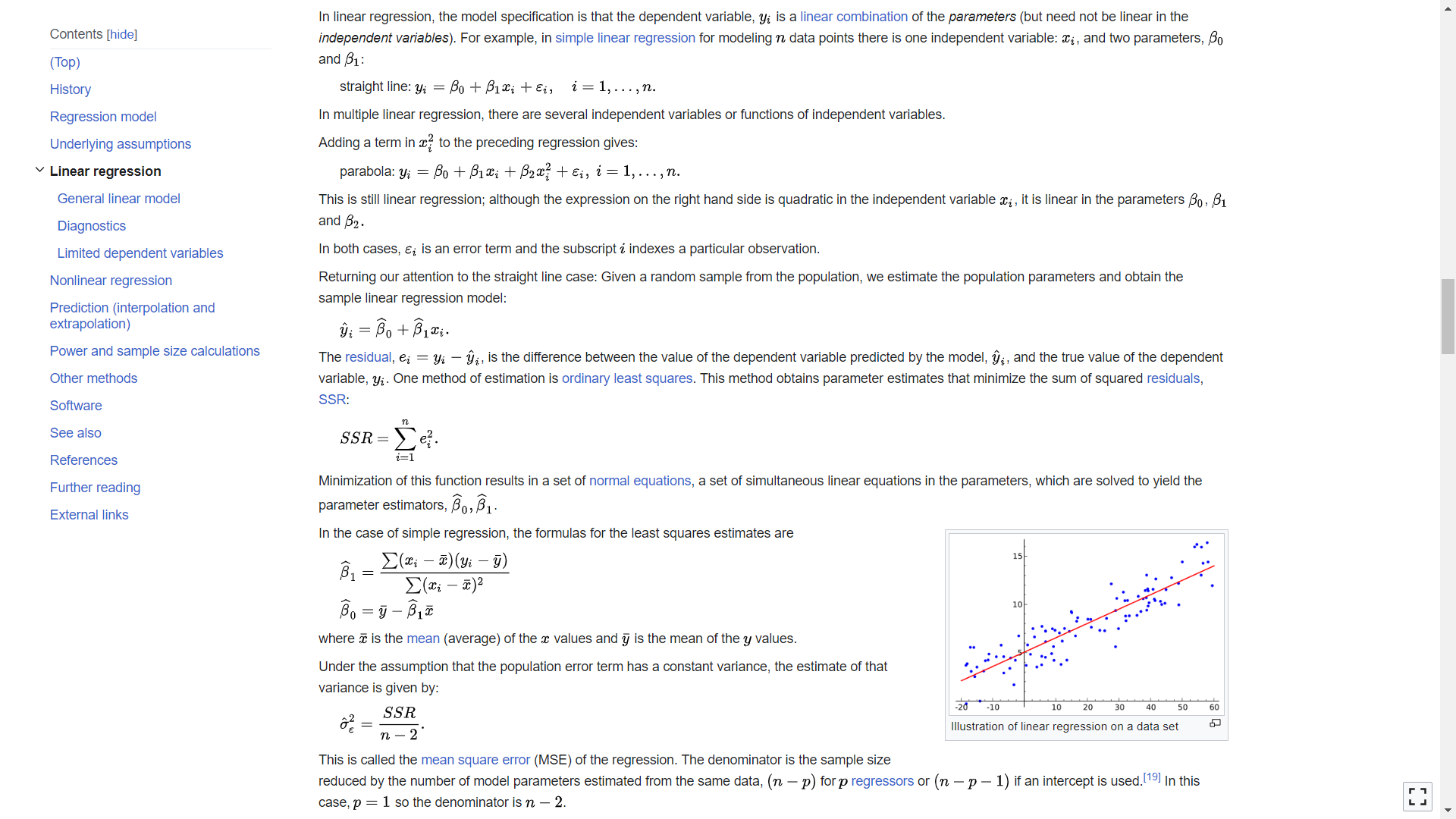
**Linear Regression**

**Regression** is a statistical technique that relates a dependent variable to one or more independent (explanatory) variables. A regression model is able to show whether changes observed in the dependent variable are associated with changes in one or more of the explanatory variables.

****

****

The OLS estimator is identical to the [maximum likelihood estimator](https://en.wikipedia.org/wiki/Maximum_likelihood_estimator) (MLE) under the normality assumption for the error terms. In statistics, **maximum likelihood estimation** (**MLE**) is a method of estimating the parameters of an assumed probability distribution, given some observed data. This is achieved by maximizing a likelihood function so that, under the assumed statistical model, the observed data is most probable. The [point](https://en.wikipedia.org/wiki/Point_estimate) in the [parameter space](https://en.wikipedia.org/wiki/Parameter_space) that maximizes the likelihood function is called the maximum likelihood estimate.

**In the below guide, OLS, MLE and how they are linked has been explained in detail.**

<https://www.analyticsvidhya.com/blog/2023/01/a-comprehensive-guide-to-ols-regression-part-1/>

<https://www.google.com/url?q=https://www.analyticsvidhya.com/blog/2021/09/maximum-likelihood-estimation-a-comprehensive-guide/&sa=D&source=editors&ust=1686385561265457&usg=AOvVaw0erIRR6j7DCMFvPWOLyufC>

<https://openclassrooms.com/en/courses/5873596-design-effective-statistical-models-to-understand-your-data/6233001-appreciate-ordinary-least-square-and-maximum-likelihood-estimation#:~:text=The%20ordinary%20least%20square%20minimizes,a%20model%20and%20its%20parameters.>

## **Assumptions of Linear Regression**

We can divide the basic assumptions of linear regression into two categories based on whether the assumptions are about the explanatory variables (i.e. features) or the residuals.

Assumptions about the explanatory variables (features):

* Linearity
* No multicollinearity

Assumptions about the error terms (residuals):

* Gaussian distribution
* Homoskedasticity(having the same scatter/variance)
* No autocorrelation

Tapping the below, we can know in detail about them.

<https://blog.quantinsti.com/linear-regression-assumptions-limitations/>

Below file contains all the assumptions checking.

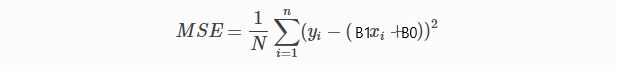
<https://github.com/avanish123456789/Machine-Learning/blob/main/stress%20prediction.ipynb>

## **Cost Function for Linear Regression**

The [cost function](https://www.analyticsvidhya.com/blog/2021/03/data-science-101-introduction-to-cost-function/) helps to work out the optimal values for B0 and B1, which provides the best fit line for the data points.

In Linear Regression, generally **Mean Squared Error (MSE)** cost function is used, which is the average of squared error that occurred between the **ypredicted** and **yi**.

We calculate MSE using simple linear equation y=mx+b:

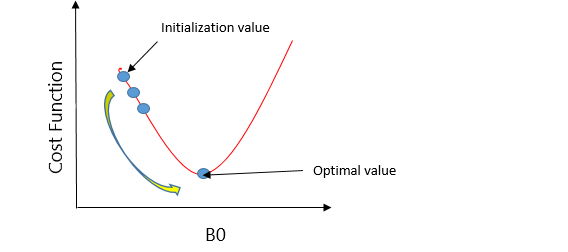


Using the MSE function, we’ll update the values of B0 and B1 such that the MSE value settles at the minima.  These parameters can be determined using the gradient descent method such that the value for the cost function is minimum.

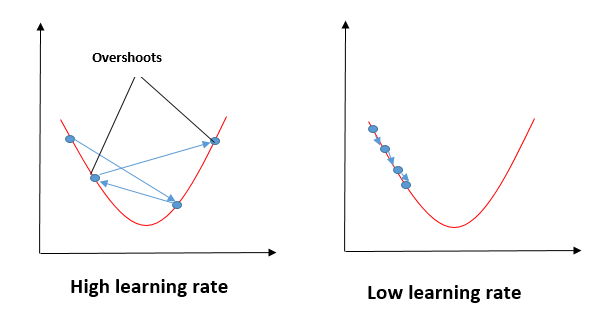
**Gradient Descent for Linear Regression**

Gradient Descent is one of the optimization algorithms that optimize the cost function(objective function) to reach the optimal minimal solution. To find the optimum solution we need to reduce the cost function(MSE) for all data points. This is done by updating the values of B0 and B1 iteratively until we get an optimal solution.

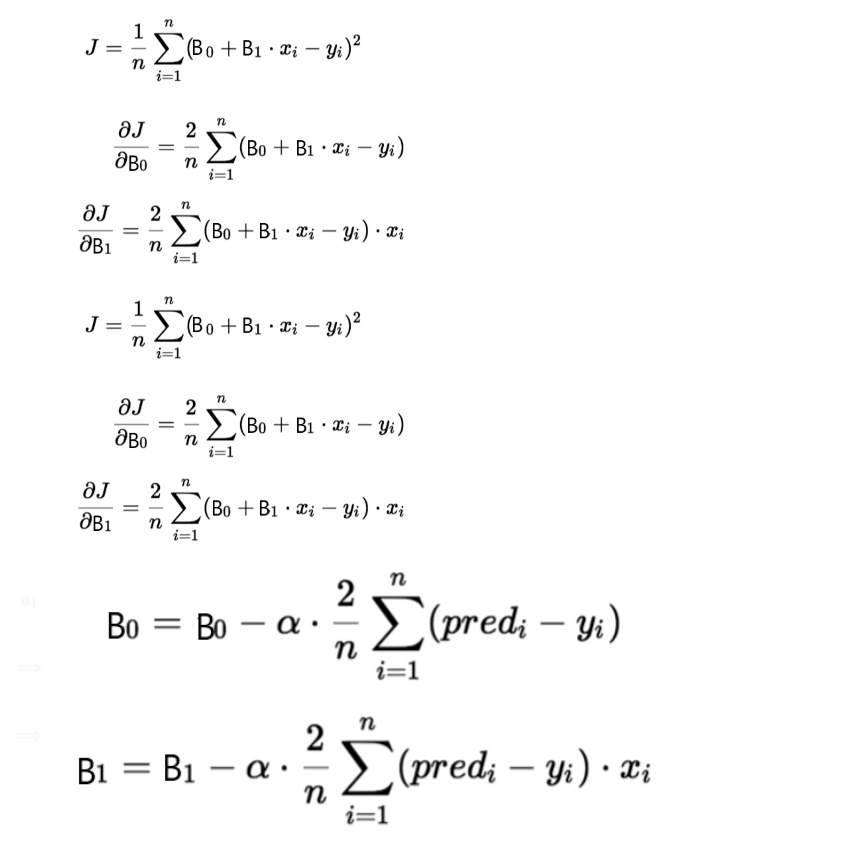
A regression model optimizes the gradient descent algorithm to update the coefficients of the line by reducing the cost function by randomly selecting coefficient values and then iteratively updating the values to reach the minimum cost function.



Let’s take an example to understand this. Imagine a U-shaped pit. And you are standing at the uppermost point in the pit, and your motive is to reach the bottom of the pit. Suppose there is a treasure at the bottom of the pit, and you can only take a discrete number of steps to reach the bottom. If you opted to take one step at a time, you would get to the bottom of the pit in the end but, this would take a longer time. If you decide to take larger steps each time, you may achieve the bottom sooner but, there’s a probability that you could overshoot the bottom of the pit and not even near the bottom. In the gradient descent algorithm, the number of steps you’re taking can be considered as the **learning rate**, and this decides how fast the algorithm **converges** to the minima.



To update B0 and B1, we take gradients from the cost function. To find these gradients, we take partial derivatives for B0 and B1.



We need to minimize the cost function J. One of the ways to achieve this is to apply the batch gradient descent algorithm. In batch gradient descent, the values are updated in each iteration. (Last two equations shows the updating of values)

The partial derivates are the gradients, and they are used to update the values of B0 and B1. Alpha is the learning rate.

## **Evaluation Metrics for Linear Regression**

The strength of any linear regression model can be assessed using various evaluation metrics. These evaluation metrics usually provide a measure of how well the observed outputs are being generated by the model.

The most used metrics are,

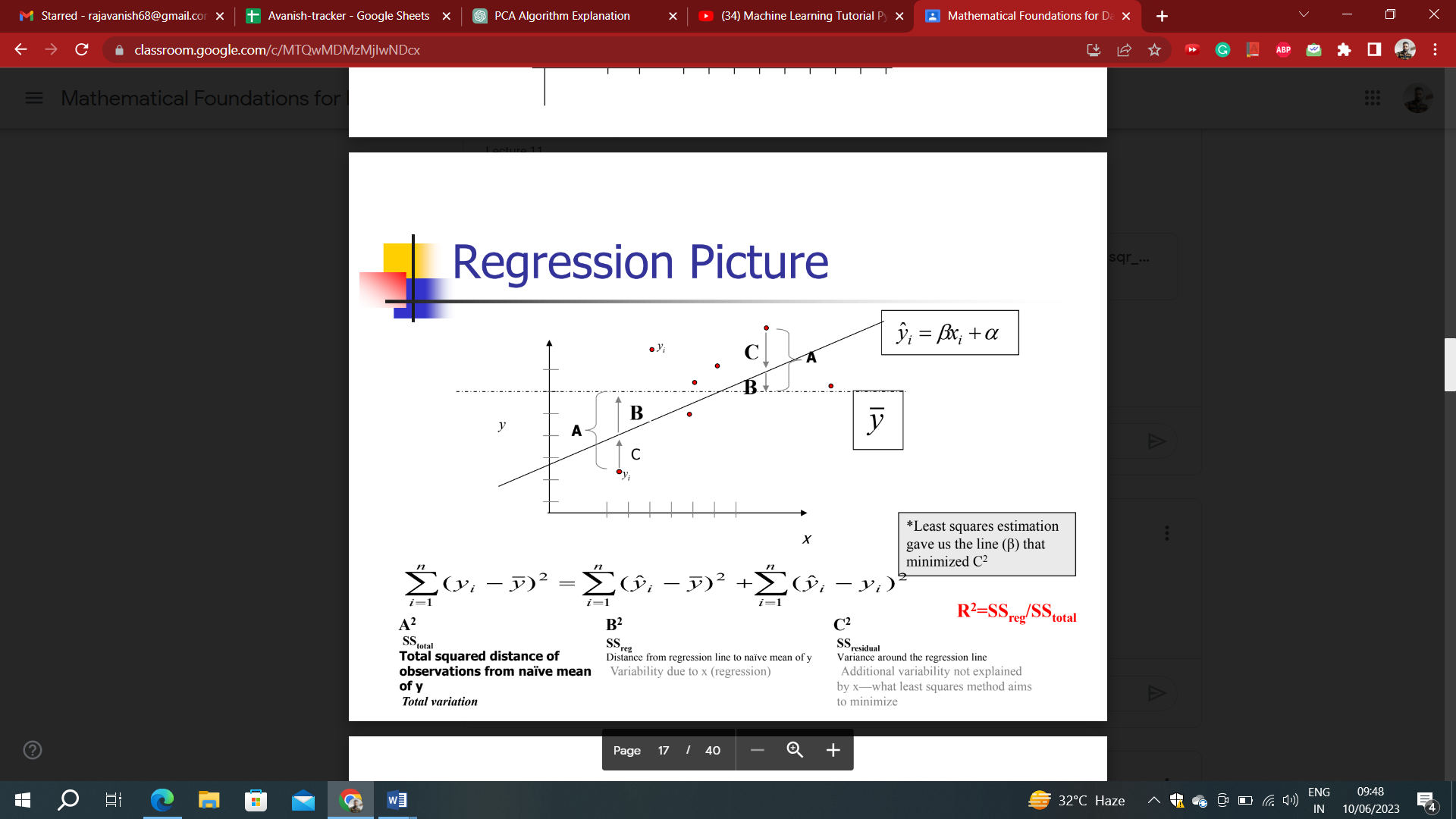
1. Coefficient of Determination or R-Squared (R2)
2. Root Mean Squared Error (RSME) and Residual Standard Error (RSE)

### **1. Coefficient of Determination or R-Squared (R2)**

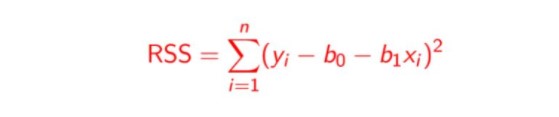
R-Squared is a number that explains the amount of variation that is explained/captured by the developed model. It always ranges between 0 & 1 . Overall, the higher the value of R-squared, the better the model fits the data.

Mathematically it can be represented as,

**R2 = 1 – ( RSS/TSS )**

****

* **Residual sum of Squares (RSS)** is defined as the sum of squares of the residual for each data point in the plot/data. It is the measure of the difference between the expected and the actual observed output.

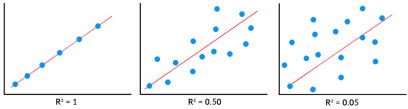


* **Total Sum of Squares (TSS)** is defined as the sum of errors of the data points from the mean of the response variable. Mathematically TSS is,

Total Sum of Squares

where y hat is the mean of the sample data points.

The significance of R-squared is shown by the following figures,



**Adjusted R squared:**

Adjusted R-squared (or adjusted R2) is a modified version of the R-squared (R2) statistic used in regression analysis. R-squared represents the proportion of the variance in the dependent variable that is explained by the independent variables in a regression model. It measures the goodness-of-fit of the model.

The adjusted R-squared takes into account the number of predictors or independent variables in the model and adjusts the R2 value accordingly. It is designed to prevent the overestimation of the model's explanatory power that can occur when additional predictors are added, even if they do not significantly improve the model's performance.

The adjusted R-squared is calculated using the formula:

Adjusted R-squared = 1 - [(1 - R2) \* (n - 1) / (n - k - 1)]

Where:

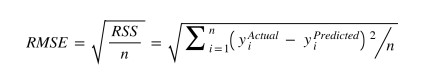
* R2 is the ordinary R-squared,
* n is the number of observations,
* k is the number of predictors (independent variables) in the model.

The adjusted R-squared penalizes the inclusion of unnecessary predictors and generally yields a lower value than the R-squared. It provides a more conservative measure of the model's fit and helps in comparing models with different numbers of predictors. A higher adjusted R-squared indicates a better fit, as it accounts for the trade-off between model complexity (number of predictors) and explanatory power.

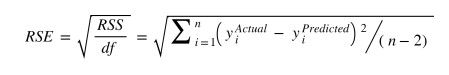
R-squared provides an overall measure of the model's fit, but it does not necessarily indicate the quality or significance of the independent variables. It can be influenced by the number of predictors in the model, and adding more predictors tends to increase the R-squared value, even if they have little or no meaningful relationship with the dependent variable. Therefore, it is important to consider other evaluation metrics and statistical tests to assess the significance and reliability of the model's coefficients.

### **2. Root Mean Squared Error**

The Root Mean Squared Error is the square root of the variance of the residuals. It specifies the absolute fit of the model to the data i.e. how close the observed data points are to the predicted values. Mathematically it can be represented as,



To make this estimate unbiased, one has to divide the sum of the squared residuals by the **degrees of freedom** rather than the total number of data points in the model. This term is then called the **Residual Standard Error(RSE)**. Mathematically it can be represented as,



R-squared is a better measure than RSME. Because the value of Root Mean Squared Error depends on the units of the variables (i.e. it is not a normalized measure), it can change with the change in the unit of the variables.

## **Hypothesis in Linear Regression**

Once you have fitted a straight line on the data, you need to ask, “**Is this straight line a significant fit for the data?**” Or “I**s the** **beta coefficient explain the variance in the data plotted?”**And here comes the idea of **hypothesis testing** on the beta coefficient. The Null and Alternate hypotheses in this case are:

H0: B1 = 0

HA: B1 **≠** 0

To test this hypothesis we use a **t-test,**test statistics for the beta coefficient is given by,

### **Assessing the model fit**

Some other parameters to assess a model are:

1. **t statistic:** It is used to determine the p-value and hence, helps in determining whether the coefficient is significant or not
2. **F statistic**: It is used to assess whether the overall model fit is significant or not. Generally, the higher the value of the F-statistic, the more significant a model turns out to be.

## **Multiple Linear Regression**

Multiple linear regression is a technique to understand the relationship between a *single*dependent variable and *multiple*independent variables.

The formulation for multiple linear regression is also similar to simple linear regression with

the small change that instead of having one beta variable, you will now have betas for all the variables used. The formula is given as:

 Y = B0 + B1X1 + B2X2 + … + BpXp + **ε**

## **Considerations of Multiple Linear Regression**

All the four assumptions made for Simple Linear Regression still hold true for Multiple Linear Regression along with a few new additional assumptions.

1. **Overfitting**: When more and more variables are added to a model, the model may become far too complex and usually ends up memorizing all the data points in the training set. This phenomenon is known as the overfitting of a model. This usually leads to high training accuracy and very low test accuracy.
2. **Multicollinearity**: It is the phenomenon where a model with several independent variables, may have some variables interrelated.
3. **Feature Selection:** With more variables present, selecting the optimal set of predictors from the pool of given features (many of which might be redundant) becomes an important task for building a relevant and better model.

## **Overfitting and Underfitting in Linear Regression**

There have always been situations where a model performs well on training data but not on the test data. While training models on a dataset, overfitting, and underfitting are the most common problems faced by people.

Before understanding overfitting and underfitting one must know about bias and variance.

**Bias:**

Bias is a measure to determine how accurate is the model likely to be on future unseen data. Complex models, assuming there is enough training data available, can do predictions accurately. Whereas the models that are too naive, are very likely to perform badly with respect to predictions. Simply, Bias is errors made by training data.

Generally, linear algorithms have a high bias which makes them fast to learn and easier to understand but in general, are less flexible. Implying lower predictive performance on complex problems that fail to meet the expected outcomes.

**Variance:**

Variance is the sensitivity of the model towards training data, that is it quantifies how much the model will react when input data is changed.

Ideally, the model shouldn’t change too much from one training dataset to the next training data, which will mean that the algorithm is good at picking out the hidden underlying patterns between the inputs and the output variables.

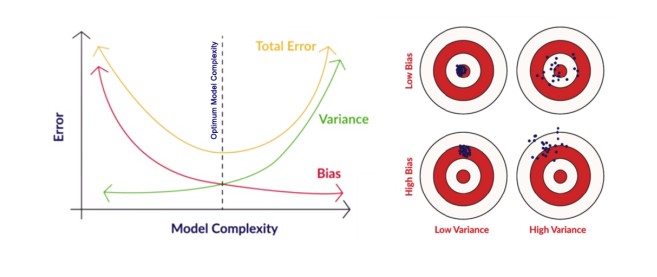
Ideally, a model should have lower variance which means that the model doesn’t change drastically after changing the training data(it is generalizable). Having higher variance will make a model change drastically even on a small change in the training dataset.

Let’s understand what is a bias-variance tradeoff is.

## **Bias Variance Tradeoff:**

The aim of any supervised machine learning algorithm is to achieve low bias and low variance as it is more robust. So that the algorithm should achieve better performance.

There is no escape from the relationship between bias and variance in machine learning.



There is an inverse relationship between bias and variance,

* An increase in bias will decrease the variance.
* An increase in the variance will decrease the bias.

There is a trade-off that plays between these two concepts and the algorithms must find a balance between bias and variance.

As a matter of fact, one cannot calculate the real bias and variance error terms because we do not know the actual underlying target function.

Now coming to the overfitting and underfitting.

## **Overfitting:**

When a model learns each and every pattern and noise in the data to such extent that it affects the performance of the model on the unseen future dataset, it is referred to as ***overfitting***. The model fits the data so well that it interprets noise as patterns in the data.

When a model has low bias and higher variance it ends up memorizing the data and causing overfitting. Overfitting causes the model to become specific rather than generic. This usually leads to high training accuracy and very low test accuracy.

Detecting overfitting is useful, but it doesn’t solve the actual problem. There are several ways to prevent overfitting, which are stated below:

* Cross-validation
* If the training data is too small to train add more relevant and clean data.
* If the training data is too large, do some feature selection and remove unnecessary features.
* Regularization

## **Underfitting:**

Underfitting is not often discussed as often as overfitting is discussed. When the model fails to learn from the training dataset and is also not able to generalize the test dataset, is referred to as ***underfitting***. This type of problem can be very easily detected by the performance metrics.

When a model has high bias and low variance it ends up not generalizing the data and causing underfitting. It is unable to find the hidden underlying patterns from the data. This usually leads to low training accuracy and very low test accuracy. The ways to prevent underfitting are stated below,

* Increase the model complexity
* Increase the number of features in the training data
* Remove noise from the data.

## **Hands-on Coding: Linear Regression Model**

This is the section where you’ll find out how to perform the regression in Python. We will use ***Advertising sales channel prediction data. You can access the data***[***here***](https://www.kaggle.com/ashydv/sales-prediction-simple-linear-regression/notebook)***.***

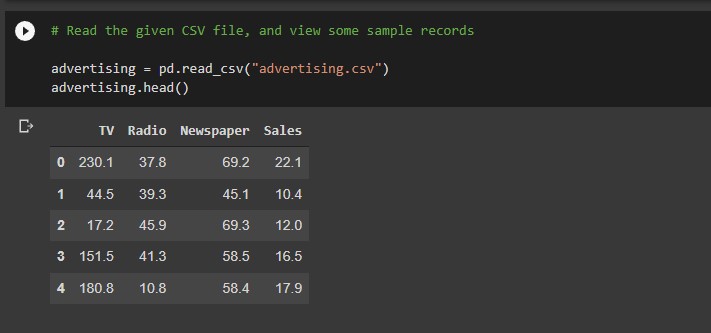
### **Step 1: Importing Python Libraries**

Import pandas as pd

Import numpy as np

Import matplotlib.pyplot as plt

### **Step 2: Loading the Dataset**



### **Step 3: Visualization**

Let us plot the scatter plot for target variable vs. predictor variables in a single plot to get the intuition. Also, plotting a heatmap for all the variables,

#Importing seaborn library for visualizations

import seaborn as sns

#to plot all the scatterplots in a single plot

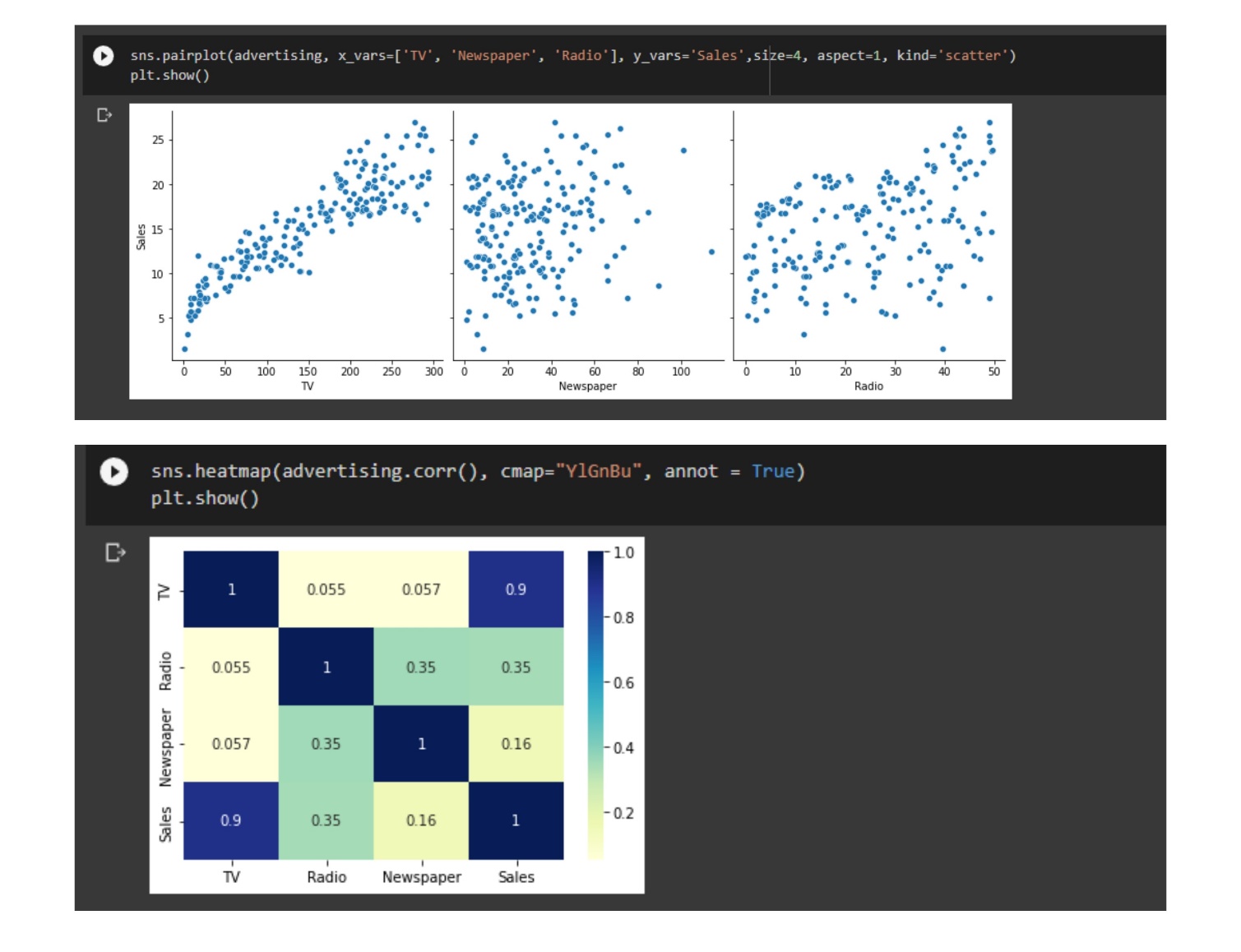
sns.pairplot(advertising, x\_vars=[ 'TV', ' Newspaper.,'Radio' ], y\_vars = 'Sales', size = 4, kind = 'scatter' )

plt.show()

#To plot heatmap to find out correlations

sns.heamap( advertising.corr(), cmap = 'YlGnBl', annot = True )

plt.show()



From the scatterplot and the heatmap, we can observe that ‘Sales’ and ‘TV’ have a higher correlation as compared to others because it shows a linear pattern in the scatterplot as well as giving 0.9 correlation.

You can go ahead and play with the visualizations and can find out interesting insights from the data.

### **Step 4: Performing Simple Linear Regression**

Here, as the TV and Sales have a higher correlation we will perform the simple linear regression for these variables.

We can use sklearn or statsmodels to apply linear regression. So we will go ahead with **statmodels**.

We first assign the feature variable, `TV`, during this case, to the variable `X` and the response variable, `Sales`, to the variable `y`.

X = advertising[ 'TV' ]

y = advertising[ 'Sales' ]

And after assigning the variables you need to split our variable into training and testing sets. You’ll perform this by importing train\_test\_split from the sklearn.model\_selection library. It is usually a good practice to keep 70% of the data in your train dataset and the rest 30% in your test dataset.

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split( X, y, train\_size = 0.7, test\_size = 0.3, random\_state = 100 )

In this way, you can split the data into train and test sets.

One can check the shapes of train and test sets with the following code,

print( X\_train.shape )

print( X\_test.shape )

print( y\_train.shape )

print( y\_test.shape )

importing statmodels library to perform linear regression

import statsmodels.api as sm

By default, the statsmodels library fits a line on the dataset which passes through the origin. But in order to have an intercept, you need to manually use the add\_constant attribute of statsmodels. And once you’ve added the constant to your X\_train dataset, you can go ahead and fit a regression line using the OLS (Ordinary Least Squares) the attribute of statsmodels as shown below,

# Add a constant to get an intercept

X\_train\_sm = sm.add\_constant(X\_train)

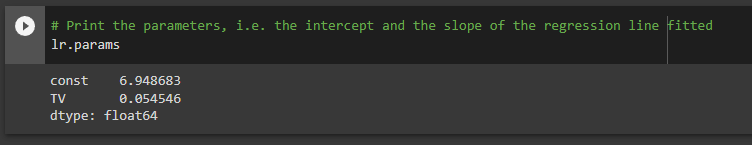
# Fit the regression line using 'OLS'

lr = sm.OLS(y\_train, X\_train\_sm).fit()

One can see the values of betas using the following code,

# Print the parameters,i.e. intercept and slope of the regression line obtained

lr.params

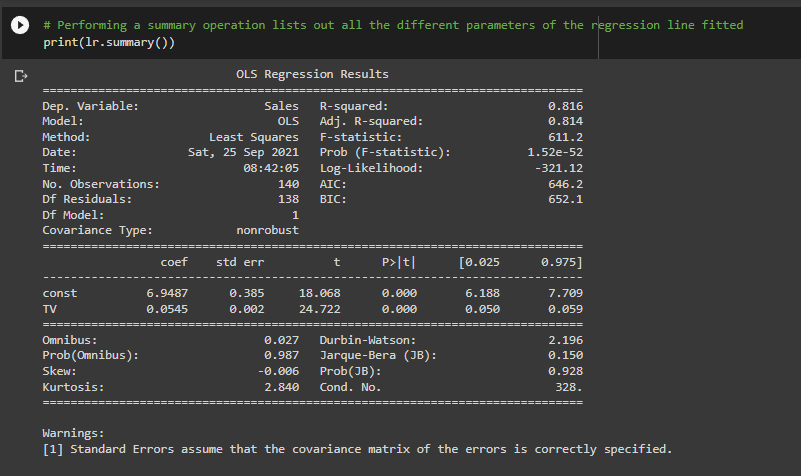


Here, 6.948 is the intercept, and 0.0545 is a slope for the variable TV.

Now, let’s see the evaluation metrics for this linear regression operation. You can simply view the summary using the following code.

#Performing a summary operation lists out all different parameters of the regression line fitted

print(lr.summary())



As you can see, this code gives you a brief summary of the linear regression. Here are some key statistics from the summary:

1. The **coefficient**for TV is 0.054, with a very low p-value. The coefficient is statistically significant. So the association is not purely by chance.
2. **R – squared** is 0.816 Meaning that 81.6% of the variance in `Sales` is explained by `TV`. This is a decent R-squared value.
3. **F-statistics** has a very low p-value(practically low). Meaning that the model fit is statistically significant, and the explained variance isn’t purely by chance.

### **Step 5: Performing predictions on the test set**

Now that you have simply fitted a regression line on your train dataset, it is time to make some predictions on the test data. For this, you first need to add a constant to the X\_test data like you did for X\_train and then you can simply go on and predict the y values corresponding to X\_test using the predict the attribute of the fitted regression line.

# Add a constant to X\_test

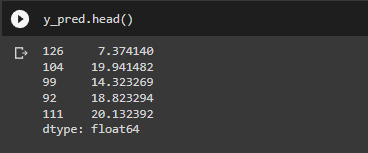
X\_test\_sm = sm.add\_constant(X\_test)

# Predict the y values corresponding to X\_test\_sm

y\_pred = lr.predict(X\_test\_sm)

You can see the predicted values with the following code,

y\_pred.head()



To check how well the values are predicted on the test data we will check some evaluation metrics using sklearn library.

#Imporitng libraries

from sklearn.metrics import mean\_squared\_error

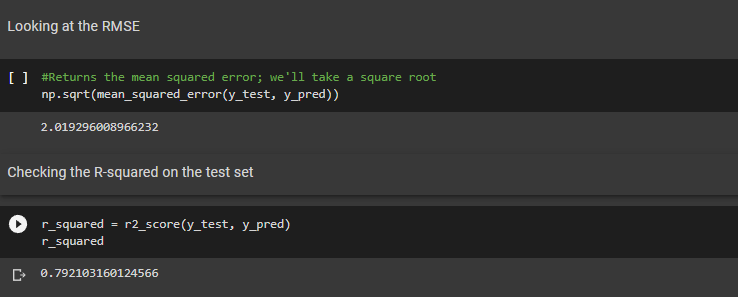
from sklearn.metrics import r2\_score

#RMSE value

print( "RMSE: ",np.sqrt( mean\_squared\_error( y\_test, y\_pred ) )

#R-squared value

print( "R-squared: ",r2\_score( y\_test, y\_pred ) )



We are getting a decent score for both train and test sets.

Apart from `statsmodels`, there is another package namely `sklearn` that can be used to perform linear regression. We will use the `linear\_model` library from `sklearn` to build the model. Since we have already performed a train-test split, we don’t need to do it again.

There’s one small step that we need to add, though. When there’s only a single feature, we need to add an additional column in order for the linear regression fit to be performed successfully. Code is given below,

X\_train\_lm = X\_train\_lm.values.reshape(-1,1)

X\_test\_lm = X\_test\_lm.values.reshape(-1,1)

One can check the change in the shape of the above data frames.

print(X\_train\_lm.shape)

print(X\_train\_lm.shape)

To fit the model, write the below code,

from sklearn.linear\_model import LinearRegression

#Representing LinearRegression as lr (creating LinearRegression object)

lr = LinearRegression()

#Fit the model using lr.fit()

lr.fit( X\_train\_lm , y\_train\_lm )

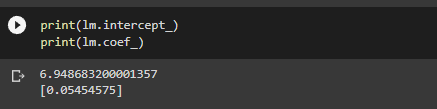
You can get the intercept and slope values with sklearn using the following code,

#get intercept

print( lr.intercept\_ )

#get slope

print( lr.coef\_ )



This is how we can perform the simple linear regression.

Top of Form

**How does linear regression vary when datasets are large and small**

Linear regression can exhibit different behaviors when applied to large and small datasets. Here are some ways in which linear regression can vary based on the dataset size:

1. Model Stability:
   * Large datasets generally provide more representative samples of the underlying population, resulting in more stable model estimates.
   * With a larger dataset, the estimates of the regression coefficients tend to be more reliable and less sensitive to individual data points or noise.
   * In contrast, small datasets may not capture the full diversity of the population, making the estimates more susceptible to variations and outliers.
2. Overfitting:
   * Overfitting occurs when a model captures noise or idiosyncrasies in the training data, leading to poor generalization to new data.
   * In large datasets, the risk of overfitting is typically lower because the ample amount of data helps to generalize patterns and reduce the influence of noise.
   * In small datasets, there is a higher risk of overfitting as the model may attempt to fit the noise or specific patterns that may not be representative of the true underlying relationship.
3. Precision of Parameter Estimates:
   * Large datasets allow for more precise estimation of the regression coefficients. The standard errors associated with the coefficient estimates tend to be smaller, providing more reliable measures of their precision.
   * Small datasets, on the other hand, often result in larger standard errors and wider confidence intervals, making it more challenging to draw precise inferences about the parameters.
4. Statistical Power:
   * Large datasets generally provide greater statistical power, enabling the detection of smaller, more subtle effects in the data.
   * With a small dataset, the statistical power is reduced, and it may be more challenging to identify significant relationships or accurately estimate the magnitude of the effects.
5. Computational Efficiency:
   * Linear regression is computationally efficient and can handle large datasets without significant computational burden.
   * In small datasets, the computational efficiency of linear regression is not a major concern, as the calculations can be performed relatively quickly even with limited computational resources.

It's important to note that while linear regression can be applied to both small and large datasets, the size of the dataset can influence the stability, generalization, precision, and statistical power of the model. As the dataset size increases, linear regression tends to benefit from more reliable estimates, reduced overfitting, greater precision, and improved statistical power.

Top of Form

**Linear Regression sensitivity to outlier**

Linear regression is known to be sensitive to outliers, which are extreme data points that deviate significantly from the majority of the data. Here's how outliers can affect linear regression:

1. Influence on Parameter Estimates:
   * Outliers can have a substantial impact on the estimated regression coefficients (slope and intercept) in linear regression.
   * Since linear regression aims to minimize the sum of squared residuals, outliers with large residuals can disproportionately influence the parameter estimates.
   * Outliers can result in biased coefficient estimates, leading to a poor representation of the true underlying relationship between the variables.
2. Distortion of the Line of Best Fit:
   * Outliers can significantly affect the line of best fit in linear regression, altering its slope and intercept.
   * If outliers have extreme x-values, they can exert a strong leverage effect and pull the line of best fit closer to or away from them.
   * Outliers with large y-values can also have a substantial impact on the slope of the line, causing it to be steeper or flatter than it should be.
3. Violation of Assumptions:
   * Linear regression assumes that the residuals (differences between observed and predicted values) are normally distributed with constant variance (homoscedasticity) and that the relationship between the variables is linear.
   * Outliers can violate these assumptions by introducing nonlinearity, heteroscedasticity (unequal variances), or non-normality in the residuals.
   * These violations can affect the accuracy and reliability of the regression model, compromising the validity of the statistical inferences.
4. Influence on Predictions:
   * Outliers can have a substantial impact on the predictions made by linear regression.
   * Since outliers can lead to significant changes in the estimated regression line, they can result in inaccurate predictions for new observations, particularly if those new observations are similar to the outliers.

To mitigate the influence of outliers in linear regression, several approaches can be considered:

* Outlier detection: Identify and remove outliers using statistical techniques such as the Z-score, modified Z-score, or robust methods like the Median Absolute Deviation (MAD).
* Data transformation: Apply data transformations, such as log transformation or winsorization, to reduce the impact of outliers on the regression analysis.
* Robust regression: Utilize robust regression techniques like RANSAC (RANdom SAmple Consensus) or Theil-Sen regression, which are less sensitive to outliers.
* Non-linear regression: If outliers are due to non-linear relationships, consider using non-linear regression models that can capture more complex patterns in the data.

It's important to handle outliers appropriately to ensure that the linear regression model is not unduly influenced by extreme values, leading to more accurate and reliable results.

Top of Form

Top of Form

**Effect of missing values on Linear Regression**

Missing values in the data can have various effects on linear regression. Here are some effects of missing values on linear regression:

1. Reduction in Sample Size:
   * Missing values reduce the effective sample size available for analysis. The observations with missing values may need to be excluded from the analysis, resulting in a smaller sample size.
   * A smaller sample size can decrease the precision and reliability of the estimated regression coefficients and other statistical measures.
2. Bias in Parameter Estimates:
   * If the missingness is related to the dependent variable or independent variables, excluding observations with missing values can introduce bias in the estimated regression coefficients.
   * This bias can occur if the missingness is associated with the relationship between the variables being studied.
3. Loss of Information:
   * Missing values lead to a loss of information, as the values that are missing may contain valuable insights or patterns.
   * The missingness may be related to specific patterns or relationships in the data, and excluding those observations can lead to an incomplete understanding of the underlying phenomenon.
4. Assumption Violation:
   * Linear regression assumes that the data are missing completely at random (MCAR), meaning that the probability of missingness is unrelated to both observed and unobserved variables.
   * If the missingness is related to the variables being analyzed, such as a specific subgroup having more missing values, the assumption of MCAR is violated.
   * Violation of the MCAR assumption can lead to biased parameter estimates and inaccurate inference.
5. Imputation of Missing Values:
   * To handle missing values, imputation techniques can be employed to fill in the missing values with estimated values.
   * Imputation can introduce uncertainty and potential bias in the regression analysis, as the imputed values are approximations.
   * The choice of imputation method can impact the results and should be carefully considered based on the characteristics of the data and the research question.

To address the effect of missing values in linear regression, it is important to handle missing data appropriately. Some common approaches include:

* Complete Case Analysis: Exclude observations with missing values from the analysis. However, this can result in reduced sample size and potential bias.
* Imputation: Fill in the missing values using imputation techniques such as mean imputation, median imputation, or multiple imputation. Imputation should be performed carefully, considering the nature of the missing data and the specific requirements of the analysis.
* Advanced Methods: Advanced techniques like full information maximum likelihood (FIML) or expectation-maximization (EM) algorithms can be used to estimate parameters while accounting for missing values.

It is crucial to carefully consider the nature and extent of missing values and select an appropriate method to handle them, ensuring the integrity and validity of the linear regression analysis.

Top of Form

**Effect of correlation on Linear Regression**

Correlation among variables can have several effects on linear regression:

1. Multicollinearity:
   * Correlation among independent variables (predictors) can lead to multicollinearity, which occurs when two or more predictors are highly correlated with each other.
   * Multicollinearity can cause challenges in linear regression by making it difficult to determine the individual effects of correlated predictors on the dependent variable.
   * In the presence of multicollinearity, the estimated regression coefficients can become unstable, and their interpretation becomes problematic.
2. Influence on Coefficient Estimates:
   * Correlation between an independent variable and the dependent variable can affect the estimated regression coefficients.
   * When two variables are positively correlated, an increase in one variable is often associated with an increase in the other variable. In such cases, the estimated coefficient for one variable may be larger than it would be in the absence of correlation.
   * Conversely, when two variables are negatively correlated, an increase in one variable is often associated with a decrease in the other variable. This negative correlation can lead to a smaller estimated coefficient for one variable compared to its true effect.
3. Precision of Parameter Estimates:
   * Correlation among predictors can inflate the standard errors of the estimated regression coefficients, reducing their precision.
   * Larger standard errors indicate greater uncertainty in the coefficient estimates, making it more challenging to draw reliable conclusions about the relationships between variables.
4. Change in Significance:
   * Correlated predictors can impact the significance of individual predictors in the regression model.
   * When two correlated predictors are included in the model, their individual significance may decrease. This occurs because some of the information captured by one predictor is already accounted for by the other predictor.
5. Influence on Prediction Accuracy:
   * Correlation among predictors can affect the accuracy of predictions made by the linear regression model.
   * If highly correlated predictors are included in the model, they may introduce redundancy and noise, potentially leading to less accurate predictions.

Handling Correlation in Linear Regression:

* Feature Selection: Consider removing one of the correlated predictors from the model to mitigate multicollinearity. If required, use VIF(**Variance Inflation Factor**).
* Feature Transformation: Apply dimensionality reduction techniques such as principal component analysis (PCA) to transform the correlated predictors into a new set of uncorrelated variables.
* Regularization: Use regularization methods like ridge regression or LASSO (Least Absolute Shrinkage and Selection Operator) to shrink or eliminate the coefficients of correlated predictors.
* Domain Knowledge: Rely on domain knowledge to determine which predictors are most relevant and include only those in the model.

Addressing the impact of correlation in linear regression is essential for obtaining reliable and interpretable results. By understanding the correlation structure among variables and employing appropriate techniques, the potential challenges posed by correlation can be mitigated, resulting in more accurate and robust regression analyses.

Top of Form

**Feature Engineering, Feature Selection and Feature Importance for Linear Regression algorithm**

Feature engineering, feature selection, and feature importance techniques can be applied in conjunction with linear regression to improve its performance and interpretability. Here's how these techniques can be used with linear regression:

1. Feature Engineering:
   * Feature engineering involves creating new features or transforming existing features to better represent the underlying patterns in the data.
   * Domain knowledge and understanding of the problem can guide the creation of relevant features that capture important information.
   * Feature engineering techniques like scaling, polynomial features, logarithmic transformations, interaction terms, or binning can be applied to enhance the representation of the data.
   * Feature engineering can help linear regression by providing more informative features and capturing non-linear relationships.
2. Feature Selection:
   * Feature selection involves identifying and selecting the most relevant subset of features from the original feature set.
   * Including irrelevant or redundant features in the model can introduce noise and increase the complexity of the model without adding significant predictive power.
   * Techniques like univariate feature selection (e.g., based on statistical tests like F-test or p-value), stepwise selection, or regularization methods (e.g., L1 regularization) can be applied to select the most informative features for linear regression. Best to use cross validation for feature selection.
   * Selecting the model with the highest value of R-squared is not a correct approach as the value of R-squared shall always increase whenever a new feature is taken for consideration even if the feature is unrelated to the response.
   * The alternative is to use **adjusted R-squared** which penalises the model complexity (to control overfitting), but this again generally [under-penalizes complexity](http://scott.fortmann-roe.com/docs/MeasuringError.html).
   * a better approach to feature selection is **Cross-validation.** It provides a more reliable way to choose which of the created models will best **generalise** as it better estimates of out-of-sample error. An advantage is that the cross-validation method can be applied to any machine learning model and the scikit-learn package provides extensive functionality for that.
   * Feature selection can help simplify the model, improve interpretability, and reduce the risk of overfitting.
3. Feature Importance:
   * Feature importance refers to quantifying the influence or contribution of each feature in the prediction made by the model.
   * Feature importance techniques can provide insights into the relative importance of different features for linear regression.
   * One common approach is to examine the magnitudes of the estimated regression coefficients. Larger coefficients generally indicate more influential features.
   * Additionally, techniques like forward or backward variable selection, or model performance metrics such as R-squared or mean squared error, can be used to assess the importance of features.
   * It's important to note that feature importance in linear regression is typically based on the linear relationship assumption and may not capture complex interactions or non-linear effects.

By applying these techniques, you can enhance the performance and interpretability of linear regression by focusing on the most relevant and informative features. However, it's crucial to note that feature engineering, feature selection, and feature importance techniques should be used judiciously, considering the specific characteristics of the dataset and the underlying assumptions of linear regression.

Top of Form

**Overfitting handling in Linear Regression**

Handling overfitting is crucial in linear regression to ensure that your model generalizes well to new, unseen data. Overfitting occurs when a model learns noise and random fluctuations in the training data rather than the underlying true relationship between the variables. Here are some strategies to handle overfitting in linear regression:

1. **Feature Selection:**
   * Choose relevant features: Include only the most relevant features in your model. Removing irrelevant or redundant features can help prevent the model from fitting noise in the data.
   * Use domain knowledge: Leverage your understanding of the problem domain to select features that are likely to have a meaningful impact on the target variable.
2. **Regularization Techniques:**
   * Ridge Regression (L2 regularization): This adds a penalty term to the linear regression cost function, discouraging the model from assigning overly large coefficients to any one feature.
   * Lasso Regression (L1 regularization): Similar to Ridge, but it encourages some coefficients to become exactly zero, effectively performing feature selection as well.
   * Elastic Net Regression: A combination of Ridge and Lasso regularization, offering a balance between the two approaches.
3. **Cross-Validation:**
   * Utilize techniques like k-fold cross-validation to assess how well your model generalizes to new data. Cross-validation helps you understand the model's performance on unseen data and can guide you in choosing hyperparameters that prevent overfitting.
4. **Early Stopping:**
   * If you're using an iterative optimization algorithm to fit your linear regression model, such as gradient descent, monitor the model's performance on a validation set. Stop training once the validation performance starts to degrade, preventing the model from fitting noise in the training data.
5. **Data Augmentation:**
   * Generating additional training examples by adding variations or noise to the existing data can help the model learn to generalize better.
6. **Reduce Model Complexity:**
   * Limit the degree of polynomial features: In polynomial regression, avoid using very high-degree polynomials, as they can lead to overfitting.
   * Reduce the number of interactions: If you're including interaction terms between features, be cautious not to include too many, as this can increase model complexity.
7. **Collect More Data:**
   * Increasing the size of your training dataset can help the model capture the underlying patterns better and reduce the risk of overfitting.
8. **Regularization Strength Tuning:**
   * For techniques like Ridge and Lasso regression, the strength of regularization is controlled by hyperparameters. Tuning these hyperparameters using techniques like grid search or random search can help find the right balance between fitting the data and preventing overfitting.
9. **Model Selection:**
   * Consider using different algorithms: If linear regression is prone to overfitting for your problem, you might explore other algorithms that are naturally less prone to overfitting, such as decision trees or random forests.

Remember that a combination of these strategies may be needed, and the appropriate approach can depend on the specific characteristics of your data and problem domain. Regular validation and testing of your model's performance on new data are essential to ensure that overfitting is being effectively handled.