Here's a brief summary of what you learned in this session:

1. Machine learning models can be classified into following two categories on the basis of learning algorithm:
   * **Supervised learning method:**Past data with labels is available to build the model
     + **Regression:**The output variable is continuous in nature
     + **Classification:**The output variable is categorical in nature
   * **Unsupervised learning method:**Past data with labels are not available
     + **Clustering:** No pre-defined notion of labels is there
2. Past data set is divided into two parts during supervised learning method:
   * **Training data**is used for the model to learn during modelling
   * **Testing data**is used by the trained model for prediction and model evaluation
3. Linear regression models can be classified into two types depending upon the number of independent variables:
   * **Simple linear regression:** When the number of independent variables is 1
   * **Multiple linear regression:** When the number of independent variables is more than 1
4. The equation of the best fit regression line Y = β₀ + β₁X can be found by minimising the cost function (RSS in this case, using the Ordinary Least Squares method) which is done using the following two methods:
   * **Differentiation**
   * **Gradient descent method**
5. The strength of a linear regression model is mainly explained by R²,  whereR² = 1 - (RSS / TSS)
   * **RSS:** Residual Sum of Squares
   * **TSS:** Total Sum of Squares

Introduction

**Introduction**

Welcome to the session on **Simple Linear Regression in Python**. So far, we have discussed the theory part of simple linear regression. Now, let’s move on to building a simple linear regression model in Python.

**In this session**

You will learn about the generic steps that are required to build a simple linear regression model. You will first read and visualise the dataset. Next, you will split the dataset into train and test sets. After that, you will build the model on the training data and draw inferences. We have used the dataset and example from the ISLR book. You will use the advertising dataset given in ISLR and analyse the relationship between 'TV advertising' and 'sales' using a simple linear regression model. You will learn to make a linear model using two different libraries: **statsmodels** and **SKLearn**.

But before you move on to the Python code, let's do a quick recap of what you have learnt so far.

**coefficient of determination**, in [statistics](https://www.britannica.com/science/statistics), *R*2 (or *r*2), a measure that assesses the ability of a [model](https://www.britannica.com/science/mathematical-model) to predict or explain an outcome in the linear [regression](https://www.britannica.com/topic/regression-statistics) setting. More specifically, *R*2 indicates the proportion of the [variance](https://www.britannica.com/topic/variance) in the dependent variable (*Y*) that is predicted or explained by linear regression and the predictor variable (*X*, also known as the independent variable).

In general, a high *R*2 value indicates that the model is a good fit for the data, although interpretations of fit depend on the [context](https://www.merriam-webster.com/dictionary/context) of analysis. An *R*2 of 0.35, for example, indicates that 35 percent of the variation in the outcome has been explained just by predicting the outcome using the covariates included in the model. That percentage might be a very high portion of variation to predict in a field such as the [social sciences](https://www.britannica.com/topic/social-science); in other fields, such as the [physical sciences](https://www.britannica.com/science/physical-science), one would expect *R*2 to be much closer to 100 percent. The theoretical minimum *R*2 is 0. However, since linear regression is based on the best possible fit, *R*2 will always be greater than zero, even when the predictor and outcome variables bear no relationship to one another.

*R*2 increases when a new predictor variable is added to the model, even if the new predictor is not associated with the outcome. To account for that effect, the adjusted *R*2 (typically [denoted](https://www.britannica.com/dictionary/denoted) with a bar over the *R* in *R*2) incorporates the same information as the usual *R*2 but then also penalizes for the number of predictor variables included in the model. As a result, *R*2 increases as new predictors are added to a multiple linear regression model, but the adjusted *R*2 increases only if the increase in *R*2 is greater than one would expect from chance alone. In such a model, the adjusted *R*2 is the most realistic estimate of the proportion of the variation that is predicted by the covariates included in the model.

When only one predictor is included in the model, the coefficient of determination is mathematically related to the [Pearson’s correlation coefficient](https://www.britannica.com/topic/Pearsons-correlation-coefficient), *r*. Squaring the [correlation](https://www.britannica.com/dictionary/correlation) coefficient results in the value of the coefficient of determination. The coefficient of determination can also be found with the following formula: *R*2 = *MSS*/*TSS* = (*TSS* − *RSS*)/*TSS*, where *MSS* is the model sum of squares (also known as *ESS*, or explained sum of squares), which is the sum of the squares of the prediction from the linear regression minus the mean for that variable; *TSS* is the total sum of squares associated with the outcome variable, which is the sum of the squares of the measurements minus their mean; and *RSS* is the residual sum of squares, which is the sum of the squares of the measurements minus the [prediction](https://www.britannica.com/dictionary/prediction) from the linear regression.

The coefficient of determination shows only association. As with [linear regression](https://www.britannica.com/topic/linear-regression), it is impossible to use *R*2 to determine whether one variable causes the other. In addition, the coefficient of determination shows only the magnitude of the association, not whether that association is statistically significant.

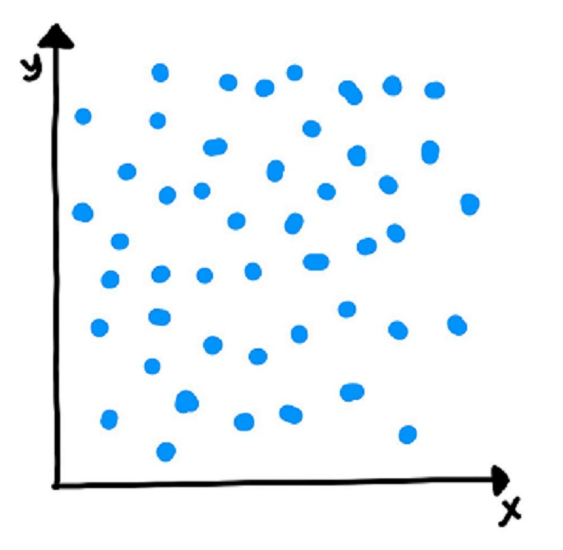
Hypothesis Testing in Linear Regression

**Hypothesis Testing in Linear Regression**

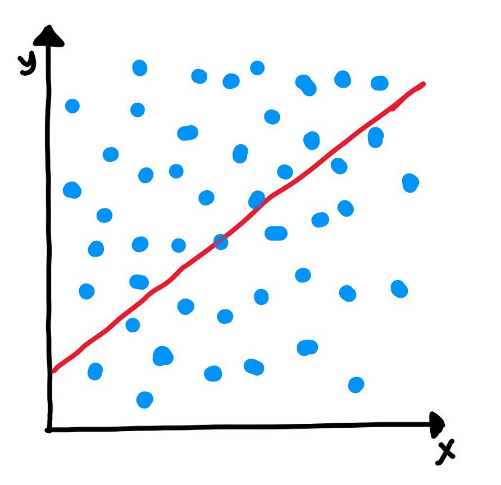
Before you move on to the model-building part, there is still one theoretical aspect left to be addressed: the significance of the derived beta coefficient. When you fit a straight line through the data, you'll obviously get the two parameters of the straight line, i.e. the intercept (β0) and the slope (β1). Now, while β0 is not of much importance right now, but there are a few aspects surrounding β1 which need to be checked and verified.

The first question we ask is, "Is the beta coefficient significant?" What does this mean?

Suppose you have a dataset for which the scatter plot looks like the following:



Now, if you run a linear regression on this dataset in Python, it will fit a line on the data which, say, looks like the following:



Now, you can clearly see that the data is randomly scattered and doesn't seem to follow a linear trend or any trend, in general. But Python will anyway fit a line through the data using the least squared method. But you can see that the fitted line is of no use in this case.

Hence, every time you perform a linear regression, you need to test whether the fitted line is a significant one or not or to simply put it, you need to test whether β1 is significant or not. And in comes the idea of Hypothesis Testing on β1. **Please note** that the following text will assume the knowledge of hypothesis testing, which was covered in one of the earlier modules. Please revisit the module on hypothesis testing in case you need to brush up.

You start by saying that β1 is not significant, i.e. there is no relationship between X and y.

So in order to perform the hypothesis test, we first propose the null hypothesis that β1 is 0. And the alternative hypothesis thus becomes β1 is not zero.

* **Null Hypothesis (H0): β1=0**
* **Alternate Hypothesis (HA): β1≠0**

Let's first discuss the implications of this hypothesis test. If you fail to reject the null hypothesis that would mean that β1 is zero which would simply mean that β1 is insignificant and of no use in the model. Similarly, if you reject the null hypothesis, it would mean that β1 is not zero and the line fitted is a significant one.

Now, how do you perform the hypothesis test?

Recall from your hypothesis testing module that you first used to compute the t-score (which is very similar to the Z-score) which is given by X−μs/√n

 where μ is the population mean and s is the sample standard deviation which when divided by √n

 is also known as standard error.

Using this, the t-score for ˆβ1

 comes out to be (since the null hypothesis is that β1 is equal to zero):

ˆβ1−0SE(ˆβ1)

Now, in order to perform the hypothesis test, you need to derive the p-value for the given beta. If you're hazy on what **p-value** is and how it is calculated, it is recommended that you revisit the segment on p-value. Please note that the formula of SE(β1) provided in the t-score above is out of scope of this course.

Let's do a quick recap of how do you calculate p-value anyway:

* Calculate the value of **t-score** for the mean point (in this case, zero, according to the Null hypothesis that we have stated) on the distribution
* Calculate the **p-value** from the cumulative probability for the given t-score using the t-table
* Make the decision on the basis of the p-value with respect to the given value of β (significance level)

Now, if the p-value turns out to be less than **0.05**,you can reject the null hypothesis and state that β1 is indeed significant.

Please note that all of the above steps will be performed automatically by the libraries we use Python which you'll learn in the very next segment.

Coming up

Now that you know how to determine whether your beta is significant or not, you'll start building the model in the next segment

In this session, you built a simple linear regression model in Python using the advertising dataset. You also saw some more theoretical aspects in between. Here's a brief of what you learnt in this session.

1. A quick recap of simple linear regression
2. Assumptions of simple linear regression
   * Linear relationship between X and y.
   * Normal distribution of error terms.
   * Independence of error terms.
   * Constant variance of error terms.
3. Hypothesis testing in linear regression
   * To determine the significance of beta coefficients.
   * H0:β1=0;HA:β1≠0.
   * T-test on the beta coefficient.
   * t score=
4. Building a linear model
   * OLS (Ordinary Least Squares) method in statsmodels to fit a line.
   * Summary statistics
     + F-statistic, R-squared, coefficients and their p-values.
5. Residual Analysis
   * Histogram of the error terms to check normality.
   * Plot of the error terms with X or y to check independence.
6. Predictions
   * Making predictions on the test set using the 'predict()' function.
7. Linear Regression using SKLearn
   * A second package apart from statsmodels for linear regression.
   * A more hassle-free package to just fit a line without any inferences.

Rahim has also answered some common doubts surrounding linear regression. This part has also been included in the notebook provided to you at the beginning of the session.

Coming Up

In the next session, you will move from simple linear regression to multiple linear regression, wherein you will use multiple independent variables to explain a single dependent variable.

Multicollinearity refers to the phenomenon of having related predictor variables in the input dataset. In simple terms, in a model which has been built using several independent variables, some of these variables might be interrelated, due to which the presence of that variable in the model is redundant. You drop some of these related independent variables as a way of dealing with multicollinearity.

Multicollinearity affects:

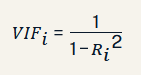
* **Interpretation**:
  + Does “change in Y, when all others are held constant” apply?
* **Inference**:
  + Coefficients swing wildly, signs can invert
  + p-values are, therefore, not reliable

Multicollinearity is, thus, a big issue when you are trying to interpret the model. It is essential to detect and deal with the multicollinearity present in the model.

You saw two basic ways of dealing with multicollinearity

1. Looking at **pairwise correlations**
   * Looking at the correlation between different pairs of independent variables
2. Checking the **Variance Inflation Factor**(VIF)
   * Sometimes pairwise correlations aren't enough
   * Instead of just one variable, the independent variable might depend upon a combination of other variables
   * VIF calculates how well one independent variable is explained by all the other independent variables combined

The VIF is given by:



where *'i'* refers to the i-th variable which is being represented as a linear combination of rest of the independent variables. You'll see VIF in action during the Python demonstration on multiple linear regression.

The common heuristic we follow for the VIF values is:

**> 10:** Definitely high VIF value and the variable should be eliminated.

**> 5:** Can be okay, but it is worth inspecting.

**< 5:**Good VIF value. No need to eliminate this variable.

But once you have detected the multicollinearity present in the dataset, how exactly do you deal with it?

Some methods that can be used to deal with multicollinearity are:

1. **Dropping variables**
   * Drop the variable which is highly correlated with others
   * Pick the business interpretable variable
2. **Create new variable**using the interactions of the older variables
   * Add interaction features, i.e. features derived using some of the original features
3. **Variable transformations**
   * Principal Component Analysis (covered in a later module)

**Coming up**

In the next segment, you will learn to handle the categorical variables present in the dataset.

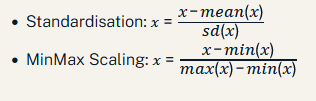
**Additional Reading**

* [Partial Least Squares (PLS)](https://support.minitab.com/en-us/minitab/18/help-and-how-to/modeling-statistics/regression/supporting-topics/partial-least-squares-regression/what-is-partial-least-squares-regression/)

Before you proceed further, spend some time answering the question next.

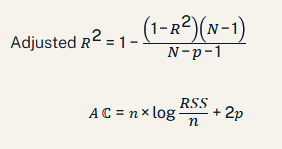
It is important to note that **scaling just affects the coefficients** and none of the other parameters like t-statistic, F-statistic, p-values, R-squared, etc.

There are two major methods to scale the variables, i.e. standardisation and MinMax scaling. Standardisation basically brings all of the data into a standard normal distribution with mean zero and standard deviation one. MinMax scaling, on the other hand, brings all of the data in the range of 0 and 1. The formulae in the background used for each of these methods are as given below:



Now, for the assessment, you have a lot of new considerations to make. Besides, selecting the best model to obtain decent predictions becomes quite subjective. You need to maintain a balance between **keeping the model simple** and **explaining the highest variance** (which means that you would want to keep as many variables as possible). This can be done using the key idea that a model can be penalised for keeping a large number of predictor variables.

Hence, there are two new parameters that come into picture:



Here, n is the sample size meaning the number of rows you'd have in the dataset and p is the number of predictor variables.

Coming up

Adjusted R2 adjusts the value of R2 such that a model with a larger number of variables is penalized. In the next segment, Rahim will talk about feature selection.

**Additional Reading :**

The following links provide a detail study on AIC and other parameters used in automatic feature selection :

Lower the value of AIC is better model.

* [AIC](https://en.wikipedia.org/wiki/Akaike_information_criterion)
* [BIC](https://en.wikipedia.org/wiki/Bayesian_information_criterion)
* [Mallows' CP](https://en.wikipedia.org/wiki/Mallows%27s_Cp)

Before you proceed further, spend some time answering the question next.

Feature Selection - Part 1

The one crucial aspect of multiple linear regression that remains to be discussed is feature selection. When building a multiple linear regression model, you might have quite a few potential predictor variables; selecting just the right ones becomes an extremely important exercise.

Let’s see how you can **select the optimal features** for building a good model.

To get the optimal model, you can always try all the possible combinations of independent variables and see which model fits the best. But this method is obviously, time-consuming and infeasible. Hence, you need some other method to get a decent model. This is where manual feature elimination comes in, where you:

1. Build the model with all the features
2. Drop the features that are least helpful in prediction (high p-value)
3. Drop the features that are redundant (using correlations and VIF)
4. Rebuild model and repeat

Note that, the second and third steps go hand in hand and the choice of which features to eliminate first is very subjective. You'll see this during the hands-on demonstration of multiple linear regression in Python in the next session.

**Feature Selection - Part 2**

Now, manual feature elimination might work when you have a relatively low number of potential predictor variables, say, ten or even twenty. But it is not a practical approach once you have a large number of features, say 100. In such a case, you automate the feature selection (or elimination) process. Let's see how.

You need to combine the manual and the automated approaches in order to get an optimal model relevant to the business. Hence, you first do an automated elimination (coarse tuning), and when you have a small set of potential variables left to work with, you can use your expertise and subjectivity to eliminate a few other features (fine tuning).

Manual feature elimination:

* Build model
* Drop features that are helpful in prediction (high p-value)
* Drop features that are redundant (using correlations, VIF)

VIF=1 / (1− (correlations R2)2 )

* Rebuild model and repeat

Automated approach:

* Top n features: RFE (Recursive Feature Elimination)
* Forward / Backward / Stepwise selection: Based on AIC
* Regularization (Lasso)

Balanced approach: Use a combination of automated (coarse tuning) + manual (fine tuning) selection.  
Coming up

In the next segment, let’s summarise what you have learnt in this session.  
  
Before you proceed further, spend some time answering the question next.

Summary

**Summary**  
  
Here’s a brief summary of what you learned in this session:

1. When one variable might not be enough
   * A lot of variance isn’t explained by just one feature
   * Inaccurate predictions
2. Formulation of MLR: MLR helps us to understand how much will the dependent variable change when we change the independent variables.
3. New considerations to be made when moving from SLR to MLR
   * Overfitting - When the model becomes complex and gives very good results in training data and fails in the testing data.
   * Multicollinearity - To identify if there is any dependency within the pool of independent variables to remove redundancy.
   * Feature selection - Out of the pool of many features what features are considered to be most important. We drop the redundant features and those features that are not helpful in prediction.
4. Dealing with categorical variables
   * Dummy variables - USed when there are fewer levels. You learnt about it using the marital status example.
5. Feature Scaling
   * Standardisation - Method used to make sure that **data** is internally consistent.
   * MinMax scaling - Method used to make sure that **data** is internally consistent.
   * Scaling for categorical variables - Categorical variables cannot used as they are, so they are converted to numeric format.
6. Model Assessment and Comparison
   * Adjusted R-squared - The **adjusted R**-**squared** value increases only if the new term improves the model more than would be expected by chance.
   * AIC, BIC - Various types of criteria used for automatic feature selection
7. Feature Selection
   * Manual feature selection - A very tedious task in order to select the correct set of features.
   * Automated feature selection - The three step process is involved.
     + Select top 'n' features
     + Forward/backward/Stepwise selection based on AIC
     + Regularization
   * Finding a balance between the two - A balance of both manual and automatic feature selection is required to attain the features.
8. You are given a multiple linear regression model: Y=β0+β1x1+β2x2+β3x3
9. Recall that the null hypothesis states that the variable is insignificant. Thus, if we fail to reject the null hypothesis, you can say that the predictor is insignificant.
10. For e.g. if you fail to reject null hypothesis for x1, you can say that x1 is insignificant. This would also imply that the coefficient for x1 i.e., β1 = 0.
11. In other words, the null hypothesis tests if the predictor's coefficient, i.e βi = 0. If the null hypothesis is rejected then βi≠0.

Multicollinearity:

Detection: VIF

Handling: Drop features, Create new features, and other methods like PCA, PLS etc.

Model Comparison:

Can’t compare R squared for models with different number of features

Use measures that penalize high #features: Adjusted R2, AIC, BIC etc.

Feature selection:

Manual approach: p-value, VIF; can be tedious and impractical for large #features

Automated approach: Top N features (RFE), forward/backward/stepwise selection using AIC, BIC etc.

Mixed approach: Automated coarse selection then manual fine tuning.