MODULE 3

Syllabus Random Forest, Decision Tree, Normal and Binomial distributions, Time Series Analysis, Linear and Multiple Regression, Logistic Regression, Survival analysis, Analysis of Variance (One way ANOVA, Two way ANOVA), Time Series Analysis.

Note : Materials are taken from the following website

https://www.tutorialspoint.com/

If you need any clarification or found any error please let me know

Expected questions format : Describe xxxxx model. Give the R syntax of xxxxxx.

(No need to write program unless you are asked to do so)

Random Forest

In the random forest approach, a large number of decision trees are created. Every observation is fed into every decision tree. The most common outcome for each observation is used as the final output. A new observation is fed into all the trees and taking a majority vote for each classification model.

An error estimate is made for the cases which were not used while building the tree. That is called an **OOB (Out-of-bag)** error estimate which is mentioned as a percentage.

The R package **"randomForest"** is used to create random forests.

The package "randomForest" has the function **randomForest()** which is used to create and analyze random forests.

Syntax

The basic syntax for creating a random forest in R is −

```
randomForest(formula, data)
```

Following is the description of the parameters used −

- **formula** is a formula describing the predictor and response variables.
- **data** is the name of the data set used.

Input Data

We will use the R in-built data set named readingSkills to create a decision tree. It describes the score of someone's readingSkills if we know the variables "age","shoesize","score" and whether the person is a native speaker.

Here is the sample data.

```
# Load the party package. It will automatically load
other
# required packages.
library(party)

# Print some records from data set readingSkills.
print(head(readingSkills))
```

When we execute the above code, it produces the following result and chart −

```
  nativeSpeaker  age  shoeSize     score
1        yes      5   24.83189  32.29385
2        yes      6   25.95238  36.63105
3         no     11   30.42170  49.60593
4        yes      7   28.66450  40.28456
5        yes     11   31.88207  55.46085
6        yes     10   30.07843  52.83124
```

Example

We will use the **randomForest()** function to create the decision tree and see it's graph.

```
# Load the party package. It will automatically load
other
# required packages.
library(party)
library(randomForest)

# Create the forest.
output.forest <- randomForest(nativeSpeaker ~ age +
shoeSize + score,
           data = readingSkills)
```

```
# View the forest results.
print(output.forest)

# Importance of each predictor.
print(importance(fit,type = 2))
```

When we execute the above code, it produces the following result −

```
Call:
 randomForest(formula = nativeSpeaker ~ age + shoeSize + score,
             data = readingSkills)
            Type of random forest: classification
                Number of trees: 500
No. of variables tried at each split: 1

     OOB estimate of  error rate: 1%
Confusion matrix:
   no yes class.error
no  99   1       0.01
yes  1  99       0.01
      MeanDecreaseGini
age          13.95406
shoeSize        18.91006
score         56.73051
```

Conclusion

From the random forest shown above we can conclude that the shoesize and score are the important factors deciding if someone is a native speaker or not. Also the model has only 1% error which means we can predict with 99% accuracy.

Decision Tree

Decision tree is a graph to represent choices and their results in form of a tree. The nodes in the graph represent an event or choice and the edges of the graph represent the decision rules or conditions. It is mostly used in Machine Learning and Data Mining applications using R.

Examples of use of decision tress is − predicting an email as spam or not spam, predicting of a tumor is cancerous or predicting a loan as a good or bad credit risk based on the factors in each of these. Generally, a model is created with observed data also called training data. Then a set of validation data is used to verify and improve the model. R has packages which are used to create and visualize decision trees. For new set of predictor variable, we use this model to arrive at a decision on the category (yes/No, spam/not spam) of the data.

The R package **"party"** is used to create decision trees.

Install R Package

Use the below command in R console to install the package. You also have to install the dependent packages if any.

```
install.packages("party")
```

The package "party" has the function **ctree()** which is used to create and analyze decison tree.

Syntax

The basic syntax for creating a decision tree in R is −

```
ctree(formula, data)
```

Following is the description of the parameters used −

- **formula** is a formula describing the predictor and response variables.
- **data** is the name of the data set used.

Input Data

We will use the R in-built data set named **readingSkills** to create a decision tree. It describes the score of someone's readingSkills if we know the variables "age","shoesize","score" and whether the person is a native speaker or not.

Here is the sample data.

```
# Load the party package. It will automatically load
other
# dependent packages.
library(party)

# Print some records from data set readingSkills.
print(head(readingSkills))
```

When we execute the above code, it produces the following result and chart −

```
 nativeSpeaker  age  shoeSize     score
1        yes    5  24.83189  32.29385
2        yes    6  25.95238  36.63105
3         no   11  30.42170  49.60593
4        yes    7  28.66450  40.28456
5        yes   11  31.88207  55.46085
6        yes   10  30.07843  52.83124
Loading required package: methods
Loading required package: grid
...............................
...............................
```

Example

We will use the **ctree()** function to create the decision tree and see its graph.

```
# Load the party package. It will automatically load
other
# dependent packages.
library(party)

# Create the input data frame.
input.dat <- readingSkills[c(1:105),]

# Give the chart file a name.
png(file = "decision_tree.png")

# Create the tree.
  output.tree <- ctree(
  nativeSpeaker ~ age + shoeSize + score,
  data = input.dat)
```

```
# Plot the tree.
plot(output.tree)

# Save the file.
dev.off()
```

When we execute the above code, it produces the following result −

```
null device
      1
Loading required package: methods
Loading required package: grid
Loading required package: mvtnorm
Loading required package: modeltools
Loading required package: stats4
Loading required package: strucchange
Loading required package: zoo

Attaching package: ‘zoo’

The following objects are masked from ‘package:base’:

   as.Date, as.Date.numeric

Loading required package: sandwich
```

Time Series Analysis

Time Series Analysis in R is used to see how an object behaves over a period of time. In R Programming Language, it can be easily done by the **ts()** function with some parameters. Time series takes the data vector and each data is connected with a timestamp value as given by the user. This function is mostly used to learn and forecast the behavior of an asset in business for a period of time. For example, sales analysis of a company, inventory analysis, price analysis of a particular stock or market, population analysis, etc.

Syntax: *objectName <- ts(data, start, end, frequency)*

where,

- ***data** – represents the data vector*
- ***start** – represents the first observation in time series*
- ***end** – represents the last observation in time series*
- ***frequency** – represents number of observations per unit time. For example, frequency=1 for monthly data.*

Linear and multiple regression

Regression methods are used in different industries to understand which variables impact a given topic of interest.

For instance, Economists can use them to analyze the relationship between consumer spending and Gross Domestic Product (GDP) growth. Public health officials might want to understand the costs of individuals based on their historical information. In both cases, the focus is not on predicting individual scenarios but on getting an overview of the overall relationship.

Linear Regression

Regression analysis is a very widely used statistical tool to establish a relationship model between two variables. One of these variable is called predictor variable whose value is gathered through experiments. The other variable is called response variable whose value is derived from the predictor variable.

In Linear Regression these two variables are related through an equation, where exponent (power) of both these variables is 1. Mathematically a linear relationship represents a straight line when plotted as a graph. A non-linear relationship where the exponent of any variable is not equal to 1 creates a curve.

The general mathematical equation for a linear regression is −

$y = ax + b$

Following is the description of the parameters used −

- **y** is the response variable.
- **x** is the predictor variable.
- **a** and **b** are constants which are called the coefficients.

Steps to Establish a Regression

A simple example of regression is predicting weight of a person when his height is known. To do this we need to have the relationship between height and weight of a person.

The steps to create the relationship is −

- Carry out the experiment of gathering a sample of observed values of height and corresponding weight.
- Create a relationship model using the **lm()** functions in R.
- Find the coefficients from the model created and create the mathematical equation using these
- Get a summary of the relationship model to know the average error in prediction. Also called **residuals**.
- To predict the weight of new persons, use the **predict()** function in R.

Input Data

Below is the sample data representing the observations −

```
# Values of height
151, 174, 138, 186, 128, 136, 179, 163, 152, 131


# Values of weight.
63, 81, 56, 91, 47, 57, 76, 72, 62, 48
```

lm() Function

This function creates the relationship model between the predictor and the response variable.

Syntax

The basic syntax for **lm()** function in linear regression is −

```
lm(formula,data)
```

Following is the description of the parameters used −

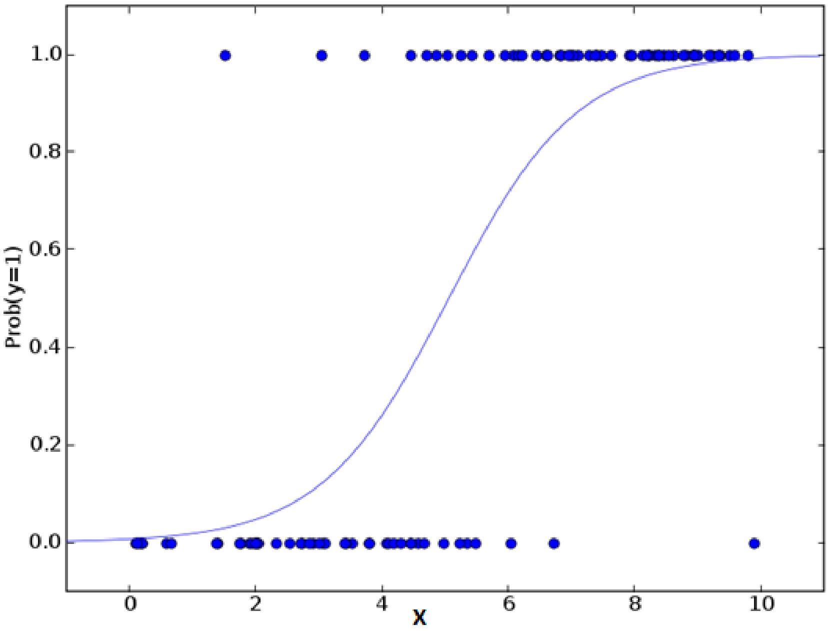
- **formula** is a symbol presenting the relation between x and y.
- **data** is the vector on which the formula will be applied.

```
x <- c(151, 174, 138, 186, 128, 136, 179, 163, 152, 131)
y <- c(63, 81, 56, 91, 47, 57, 76, 72, 62, 48)

# Apply the lm() function.
relation <- lm(y~x)

print(relation)
```

2. Logistic Regression

Don’t get confused by its name! It is a classification not a regression algorithm. It is used to estimate discrete values (Binary values like 0/1, yes/no, true/false) based on given set of independent variable(s). In simple words, it predicts the probability of occurrence of an event by fitting data to a logit function. Hence, it is also known as **logit regression**. Since, it predicts the probability, its output values lies between 0 and 1 (as expected).

Again, let us try and understand this through a simple example.



Let’s say your friend gives you a puzzle to solve. There are only 2 outcome scenarios – either you solve it or you don’t. Now imagine, that you are being given wide range of puzzles / quizzes in an attempt to understand which subjects you are good at. The outcome to this study would be something like this – if you are given a trignometry based tenth grade problem, you are 70% likely to solve it. On the other hand, if it is grade fifth history question, the probability of getting an answer is only 30%. This is what Logistic Regression provides you.

Coming to the math, the log odds of the outcome is modeled as a linear combination of the predictor variables.

Above, p is the probability of presence of the characteristic of interest. It chooses parameters that maximize the likelihood of observing the sample values rather than that minimize the sum of squared errors (like in ordinary regression).

Now, you may ask, why take a log? For the sake of simplicity, let's just say that this is one of the best mathematical way to replicate a step function. I can go in more details, but that will beat the purpose of this article.

There are many different steps that could be tried in order to improve the model:

- including interaction terms
- removing features
- regularization techniques
- using a non-linear model

The Multiple Linear Regression Assumptions

An important aspect when building a multiple linear regression model is to make sure that the following key assumptions are met.

- The ***residual values are normally distributed***. This can be checked by either using a normal probability plot or a **histogram**.
- There must be a l***inear relationship between the dependent and the independent variables***. This can be illustrated by scatterplots showing a linear or curvilinear relationship.
- Then, ***multicollinearity*** is another assumption, meaning that the independent variables are not highly correlated with each other. Multicollinearity makes it difficult to identify which variables better explain the dependent variable. This assumption is verified by computing a matrix of Pearson's bivariate correlations among all the independent variables. If there is no collinearity in the data, then all the values should be less than 0.8.
- The ***homoscedasticity*** assumes that the variance of the residual errors is similar across the value of each independent variable. One way of checking that is through a plot of the predicted values against the standardized residual values to see if the points are equally distributed across all the values of the independent variables.

In the next sections, we will cover some of these assumptions.

Multiple Linear Regression in R

Multiple regression is an extension of linear regression into relationship between more than two variables. In simple linear relation we have one predictor and one response variable, but in multiple regression we have more than one predictor variable and one response variable.

The general mathematical equation for multiple regression is −

```
y = a + b1x1 + b2x2 +...bnxn
```

Following is the description of the parameters used −

- **y** is the response variable.
- **a, b1, b2...bn** are the coefficients.
- **x1, x2, ...xn** are the predictor variables.

We create the regression model using the **lm()** function in R. The model determines the value of the coefficients using the input data. Next we can predict the value of the response variable for a given set of predictor variables using these coefficients.

lm() Function

lm() Function

This function creates the relationship model between the predictor and the response variable.

Syntax

The basic syntax for **lm()** function in multiple regression is −

```
lm(y ~ x1+x2+x3...,data)
```

Following is the description of the parameters used −

- **formula** is a symbol presenting the relation between the response variable and predictor variables.
- **data** is the vector on which the formula will be applied.

MODULE 4

Object-Oriented Programming in R

In R, we solve problems by dividing them into simpler functions and not objects. Hence, functional programming is generally used in R. OOP languages have a single object model in place. These models define the properties and behavior of the objects we create. R has a few different object-oriented programming systems or object models in place. These systems are:

1. **S3:** S3 is the first and the simplest OOP system in R. It does not have many restrictions and we can create objects by simply adding a class attribute to it. S3 classes don't have a formally defined structure. The methods defined in an S3 class belong to generic functions.
2. **S4:** S4 classes have a definitive structure to them. This means that different S4 classes have a similar structure. We use the `setClass()` function to create a new class and the `new()` function to create an object. Like S3 classes, methods in S4 classes belong to generic functions rather than the classes themselves.
3. **Reference classes(RC):** Reference classes in R are similar to object-oriented classes in other programming languages. They have all the features of S4 classes with an added environment. To create a reference class, we use the `setRefClass()` function. The methods in a reference class belong to the class itself.

While these three systems are in the base R packages, there are other packages on CRAN repository that provide more systems like the R6.

Generic Functions

Generic functions are a topic of much confusion in R. These functions are good examples of polymorphism. The most common example of a generic function is the `print()` function.

The function prints the arguments provided. The input argument can be a string, a numeric or even an object. It does not need a description of the object, its type or structure. How does it do that? The `print()` function is a collection of various print functions dedicated to different data types and data structures in R. When we use it, the function finds the type and class of the input object and calls the appropriate function to print it. We can see the different functions under the generic `print()` function by using the `method()` command. For example:

S3 Classes

S3 classes are the most basic object-oriented classes in R. They implement the polymorphism principle of OOP but not much else. To create an S3 class, we need to add a class attribute to an object.

Code:

```
emp <- list(name="ramesh",age="24",


department="sales",emp_id="00495")


class(emp) <- "employee"


emp
```

HYPOTHESIS TESTING

Hypothesis testing is a part of statistical analysis, where we test the assumptions made regarding a population parameter.

It is generally used when we were to compare:

- a single group with an external standard
- two or more groups with each other

Note: Don’t be confused between the terms Parameter and Satistic. **A Parameter is a number that describes the data from the *population* whereas, a *Statistic* is a number that describes the data from a *sample*.**

Before moving any further, it is important to know the terminology used.

2. Terminology used

Null Hypothesis: Null hypothesis is a statistical theory that suggests there is no statistical significance exists between the populations.

It is denoted by **H_0** and read as **H-naught**.

Alternative Hypothesis: An Alternative hypothesis suggests there is a significant difference between the population parameters. It could be greater or smaller. Basically, it is the contrast of the Null Hypothesis.

It is denoted by **H_a** or **H_1**.

Note: H_0 must always contain equality(=). H_a always contains difference(≠, >, <).

For example, if we were to test the equality of average means (μ) of two groups:

for a two-tailed test, we define H_0: $\mu_1 = \mu_2$ and H_a: $\mu_1 \neq \mu_2$

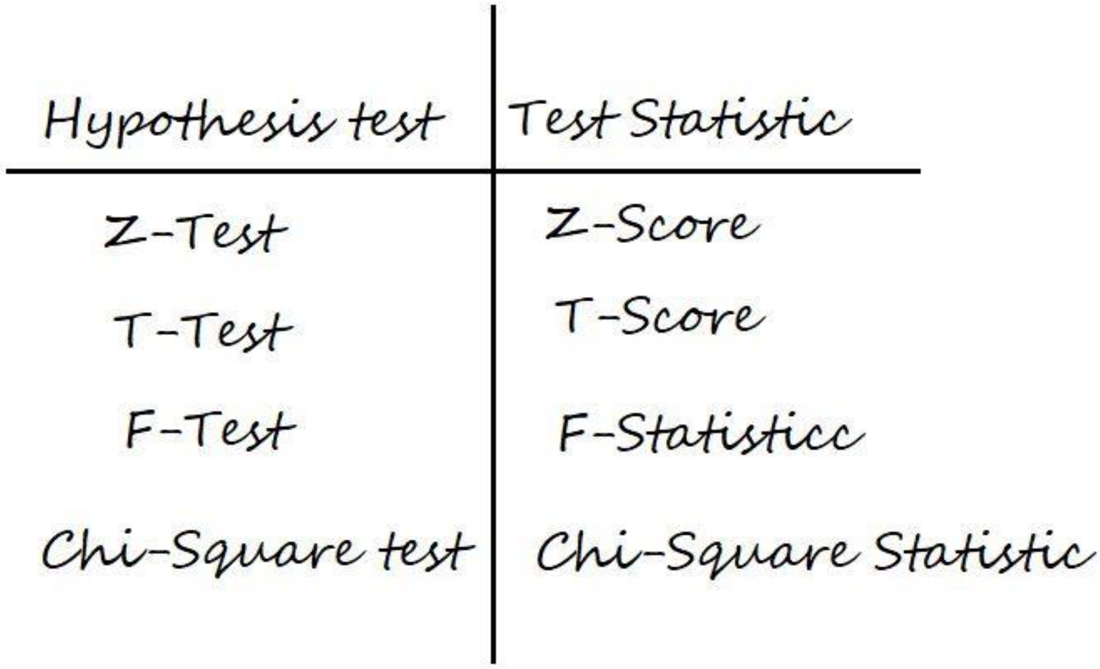
for a one-tailed test, we define H_0: $\mu_1 = \mu_2$ and H_a: $\mu_1 > \mu_2$ or H_a: $\mu_1 < \mu_2$

Level of significance: Denoted by **alpha** or **α**. It is a fixed probability of wrongly rejecting a True Null Hypothesis. For example, if α=5%, that means we are okay to take a 5% risk and conclude there exists a difference when there is no actual difference.

Critical Value: Denoted by **C** and it is a value in the distribution beyond which leads to the rejection of the Null Hypothesis. It is compared to the test statistic.

Test Statistic: It is denoted by **t** and is dependent on the test that we run. It is deciding factor to reject or accept Null Hypothesis.

The four main test statistics are given in the below table:



Hypothesis test	Test Statistic
Z-Test	Z-Score
T-Test	T-Score
F-Test	F-Statisticc
Chi-Square test	Chi-Square Statistic

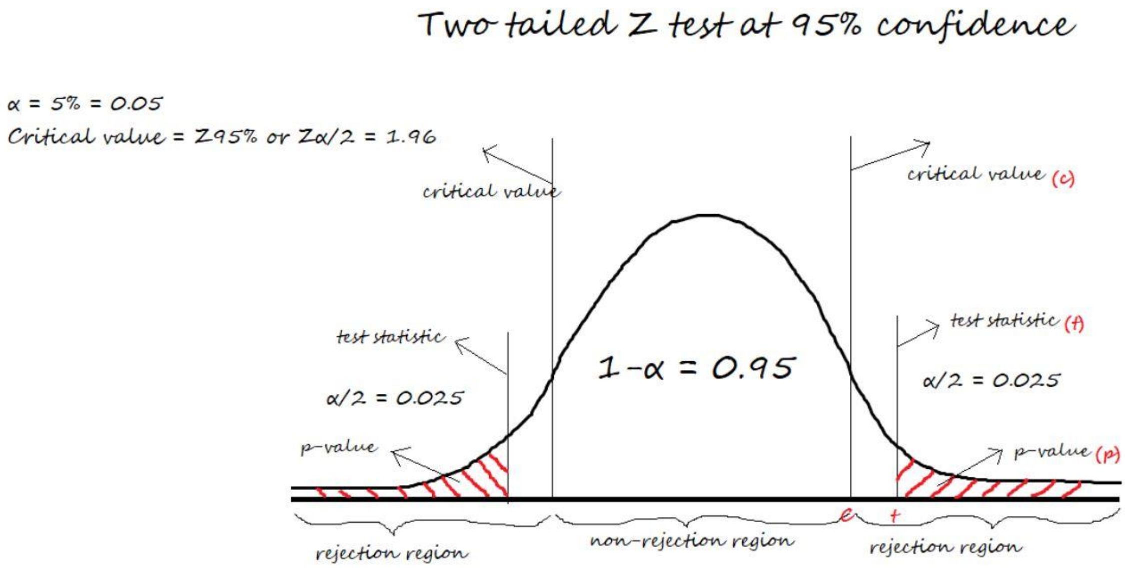
Types of Test Statistics (image by Author)

p-value: It is the proportion of samples (assuming the Null Hypothesis is true) that would be as extreme as the test statistic. It is denoted by the letter **p**.

Now, assume we are running a two-tailed Z-Test at 95% confidence. Then, the level of significance (α) = 5% = 0.05. Thus, we will have (1-α) = 0.95 proportion of data at the center, and α = 0.05 proportion will be equally shared to the two tails. Each tail will have (α/2) = 0.025 proportion of data.

The critical value i.e., $Z_{95\%}$ or $Z_{\alpha/2}$ = 1.96 is calculated from the Z-scores table.

Now, take a look at the below figure for a better understanding of critical value, test-statistic, and p-value.



Illustrating critical value, test-statistic, and p-value (Image by Author)

3. Steps of Hypothesis testing

For a given business problem,

1. Start with specifying Null and Alternative Hypotheses about a population parameter
2. Set the level of significance (α)
3. Collect Sample data and calculate the Test Statistic and P-value by running a Hypothesis test that well suits our data
4. Make Conclusion: Reject or Fail to Reject Null Hypothesis

4. Decision Rules

The two methods of concluding the Hypothesis test are using the Test-statistic value, p-value.

In both methods, we start assuming the Null Hypothesis to be true, and then we reject the Null hypothesis if we find enough evidence.

The decision rule for the Test-statistic method:

if test-statistic (t) > critical Value (C), we reject Null Hypothesis.
If test-statistic (t) ≤ critical value (C), we fail to reject Null Hypothesis.

The decision rule for the p-value method:

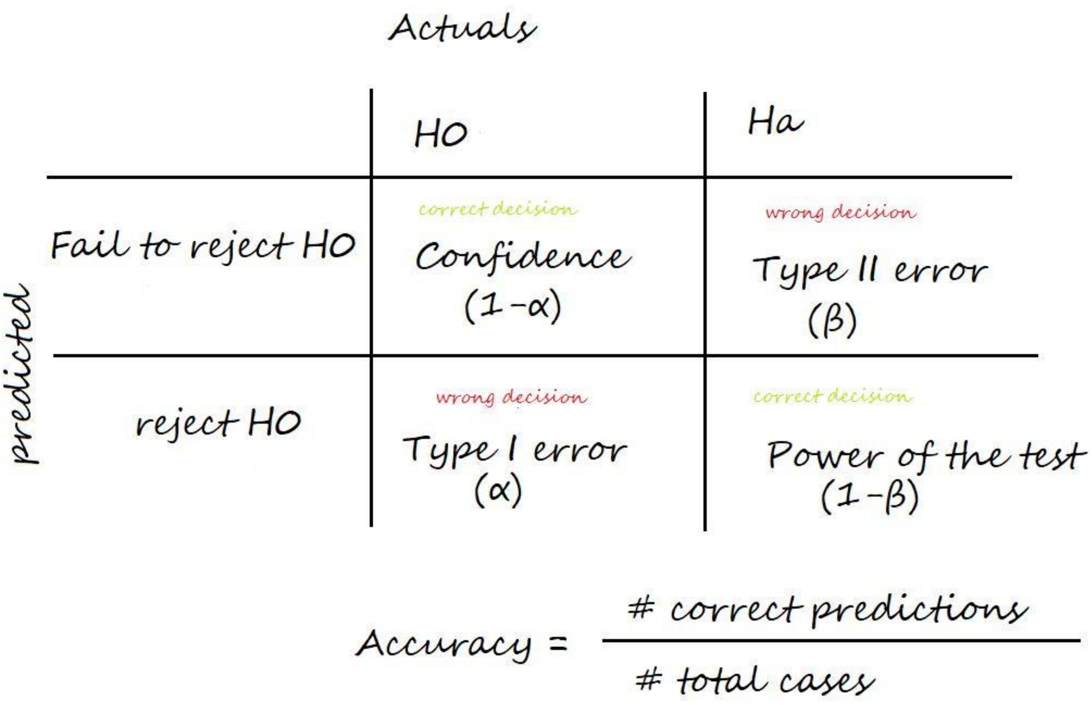
if p-value (p) > level of significance (α), we fail to reject Null Hypothesis
if p-value (p) ≤ level of significance (α), we reject Null Hypothesis

In easy terms, we say **P High, Null Fly** and **P low, Null go**.

5. Confusion Matrix in Hypothesis testing

To plot a confusion matrix, we can take actual values in columns and predicted values in rows or vice versa.

(I am illustrating by taking actuals in columns and predicted in rows.)



Actuals

predicted	HO	Ha
Fail to reject HO	correct decision Confidence $(1-\alpha)$	wrong decision Type II error (β)
reject HO	wrong decision Type I error (α)	correct decision Power of the test $(1-\beta)$

$$\text{Accuracy} = \frac{\#\ \text{correct predictions}}{\#\ \text{total cases}}$$

Confusion Matrix of Hypothesis Testing (image by Author).

Confidence: The probability of accepting a True Null Hypothesis. It is denoted as (1-α)

Power of test: The probability of rejecting a False Null Hypothesis i.e., the ability of the test to detect a difference. It is denoted as (1-β) and its value lies between 0 and 1.

Type I error: Occurs when we reject a True Null Hypothesis and is denoted as α.

Type II error: Occurs when we accept a False Null Hypothesis and is denoted as β.

Accuracy: Number of correct predictions / Total number of cases

The factors that affect the power of the test are sample size, population variability, and the confidence (α). Confidence and power of test are directly proportional. Increasing the confidence increases the power of the test.