**Machine Learning with Python**

**Week 1 – Introduction to Machine Learning**

Machine learning is the subfield of computer science that gives computers the ability to learn without being explicitly programmed.

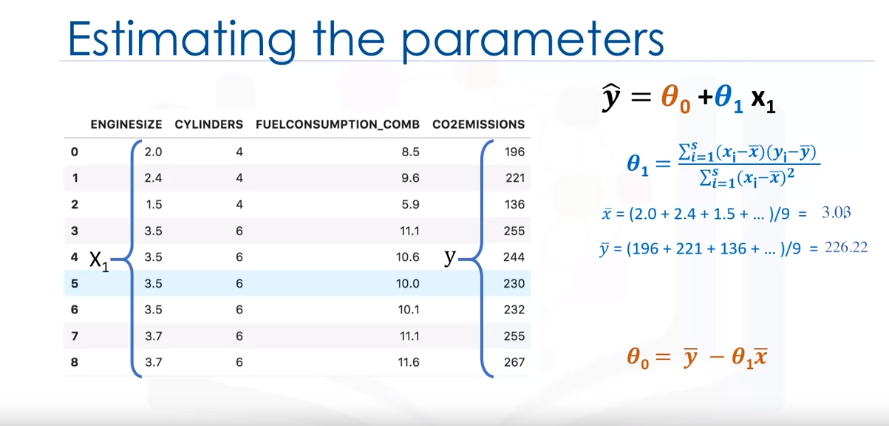
**Week 2 – Supervised Learning – Linear Regression**

**Simple Linear Regression:**

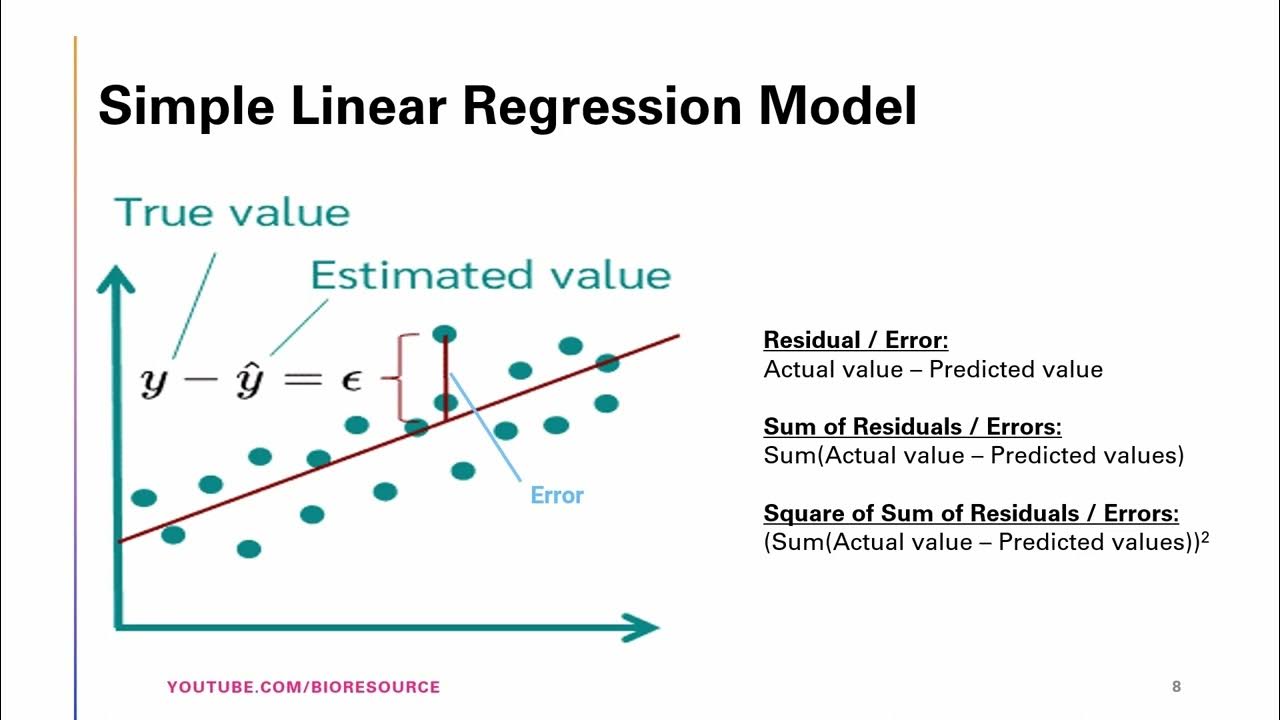
**Yhat = a + bx**

**Residual Error = Yactual - YPredicted**

The mean of all residual errors shows how poorly the line fits with the whole data set. Mathematically it can be shown by the equation Mean Squared Error, shown as MSE. Our objective is to find a line where the meaning of all these errors is minimized. In other words, the mean error of the prediction using the fit line should be minimized. Let's reword it more technically. The objective of linear regression is to minimize this MSE equation and to minimize it, we should find the best parameters; a and b.



As mentioned before, **Coefficient** and **Intercept** in the simple linear regression, are the parameters of the fit line. Given that it is a simple linear regression, with only 2 parameters, and knowing that the parameters are the intercept and slope of the line, sklearn can estimate them directly from our data. Notice that all of the data must be available to traverse and calculate the parameters

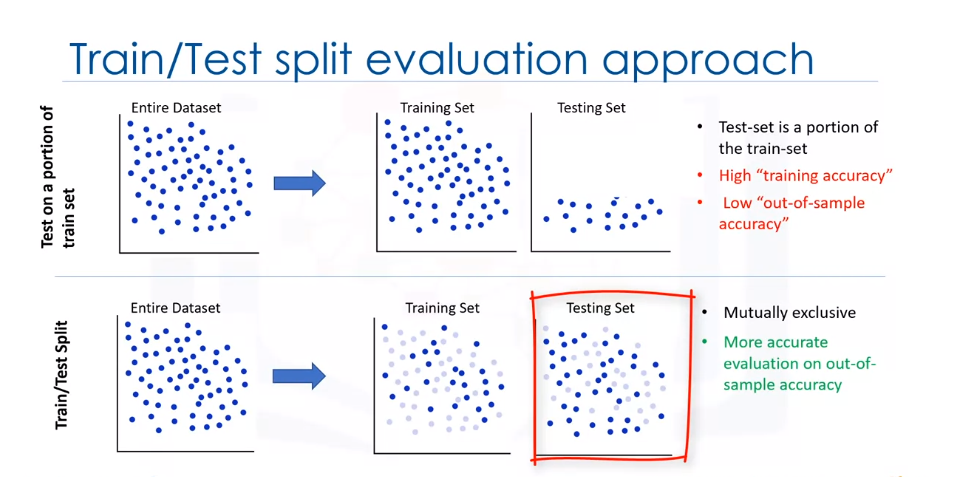


**Training Accuracy:**

Training accuracy is the percentage of correct predictions that the model makes when using the test dataset. However, a high training accuracy isn't necessarily a good thing. For instance, having a high training accuracy may result in an over-fit the data. This means that the model is overly trained to the dataset, which may capture noise and produce a non-generalized model.

**Out-of-sample Accuracy:**

Out-of-sample accuracy is the percentage of correct predictions that the model makes on data that the model has not been trained on. Doing a train and test on the same dataset will most likely have low out-of-sample accuracy due to the likelihood of being over-fit. It's important that our models have high out-of-sample accuracy because the purpose of our model is, of course, to make correct predictions on unknown data.



**Creating train and test dataset**

Train/Test Split involves splitting the dataset into training and testing sets that are mutually exclusive. After which, you train with the training set and test with the testing set. This will provide a more accurate evaluation on out-of-sample accuracy because the testing dataset is not part of the dataset that have been used to train the model. Therefore, it gives us a better understanding of how well our model generalizes on new data.

This means that we know the outcome of each data point in the testing dataset, making it great to test with! Since this data has not been used to train the model, the model has no knowledge of the outcome of these data points. So, in essence, it is truly an out-of-sample testing.

Let's split our dataset into train and test sets. 80% of the entire dataset will be used for training and 20% for testing. We create a mask to select random rows using **np.random.rand()** function:

**Evaluation Metrics in Regression Models:**

Error: It is the measure of how far the data point of Y is from the fitted regression line (Yhat).

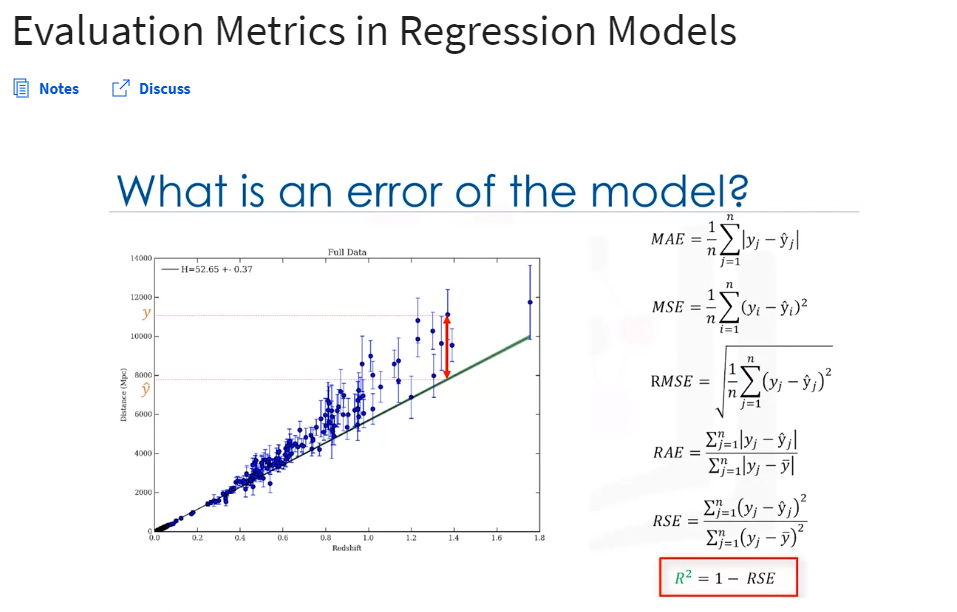
We compare the actual values and predicted values to calculate the accuracy of a regression model. Evaluation metrics provide a key role in the development of a model, as it provides insight to areas that require improvement.

There are different model evaluation metrics, lets use MSE here to calculate the accuracy of our model based on the test set:

* Mean Absolute Error: It is the mean of the absolute value of the errors. This is the easiest of the metrics to understand since it’s just average error.
* Mean Squared Error (MSE): Mean Squared Error (MSE) is the mean of the squared error. It’s more popular than Mean Absolute Error because the focus is geared more towards large errors. This is due to the squared term exponentially increasing larger errors in comparison to smaller ones.
* Root Mean Squared Error (RMSE).
* R-squared is not an error, but rather a popular metric to measure the performance of your regression model. It represents how close the data points are to the fitted regression line. The higher the R-squared value, the better the model fits your data. The best possible score is 1.0 and it can be negative (because the model can be arbitrarily worse).

MAE(Mean Absolute Error):

MSE(Mean Squared Error):

RMSE(Root Mean Squared Error):

R-Squared:

Adjusted R-squared:

RAE(Relative Absolute Error):

RSE(Relative Squared Error):

**Multiple Linear Regression/Model**

In reality, there are multiple variables that impact the co2emission(dependent variable). When more than one independent variable is present, the process is called multiple linear regression. An example of multiple linear regression is predicting co2emission using the features (independent variables) FUELCONSUMPTION\_COMB, EngineSize and Cylinders of cars and more. The good thing here is that multiple linear regression model is the extension of the simple linear regression model.

There can be two methods to optimize the parameters of the multiple linear regression model:

1. Ordinary Least Square method (takes a long time to optimize the parameters vector if the data is too large (data >=10k). It is a good practice to go with OLS if data is less than < 10K.
2. Optimization Approach: it is used for the larger set of data. Technique can be Gradient Descent, Stochastic Gradient Descent, Newton’s Method, etc.

As mentioned before, **Coefficient** and **Intercept** are the parameters of the fitted line. Given that it is a multiple linear regression model with 3 parameters and that the parameters are the intercept and coefficients of the hyperplane, sklearn can estimate them from our data. Scikit-learn uses plain Ordinary Least Squares method to solve this problem.

**Ordinary Least Squares (OLS)**

OLS is a method for estimating the unknown parameters in a linear regression model. OLS chooses the parameters of a linear function of a set of explanatory variables by minimizing the sum of the squares of the differences between the target dependent variable and those predicted by the linear function. In other words, it tries to minimize the sum of squared errors (SSE) or mean squared error (MSE) between the target variable (y) and our predicted output (^y�^) over all samples in the dataset.

OLS can find the best parameters using of the following methods:

* Solving the model parameters analytically using closed-form equations
* Using an optimization algorithm (Gradient Descent, Stochastic Gradient Descent, Newton’s Method, etc.)

**Explained variance regression score:**   
Let ^y�^ be the estimated target output, y the corresponding (correct) target output, and Var be the Variance (the square of the standard deviation). Then the explained variance is estimated as follows:

explainedVariance(y, ^y)=1−Var{y−^y} / Var{y}

The best possible score is 1.0, the lower values are worse.

**---------Explained Variance is equal to the R-squared.-----------**

**Week 3 – Supervised Learning - Classification**

It can be thought of as a means of categorizing or classifying some unknown items into a discrete set of classes.

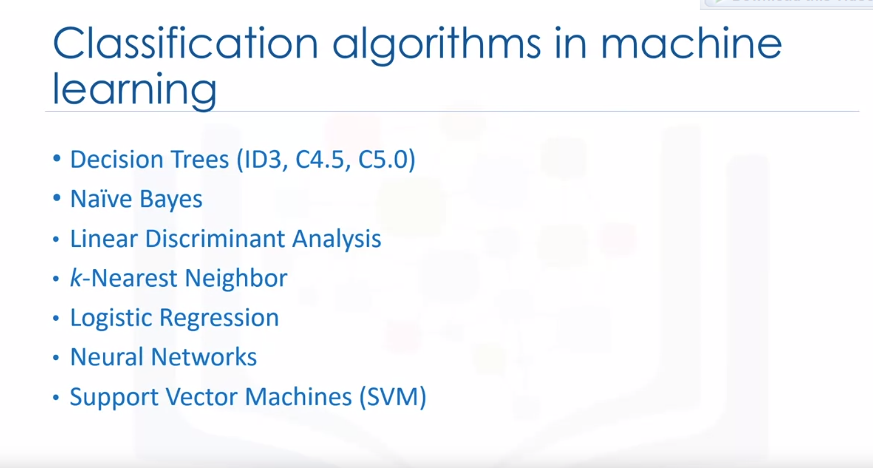
Target Attribute/variable is a categorical variable.

Classification determines the class label for an unlabeled test case.

Data classification has several applications in a wide variety of industries.

Essentially, many problems can be expressed as associations between feature and target variables, especially when labelled data is available.

This provides a broad range of applicability for classification. For example, classification can be used for email filtering, speech recognition, handwriting recognition, biometric identification, document classification and much more.

Here we have the types of classification algorithms and machine learning.

However, there are many types of classification algorithms though. We will probably cover these algorithms later after completing the most important ones first.

**KNN (K-Nearest Neighbors)**

K-nearest neighbors (KNN) is a type of supervised learning algorithm which is used for both regression and classification purposes, but mostly it is used for classification problem. Given a dataset with different classes, KNN tries to predict the correct class of test data by calculating the distance between the test data and all the training points. It then selects the k points which are closest to the test data. Once the points are selected, the algorithm calculates the probability (in the case of classification) of the test point belonging to the classes of the k training points, and the class with the highest probability is selected. In the case of a regression problem, the predicted value is the mean of the k selected training points.

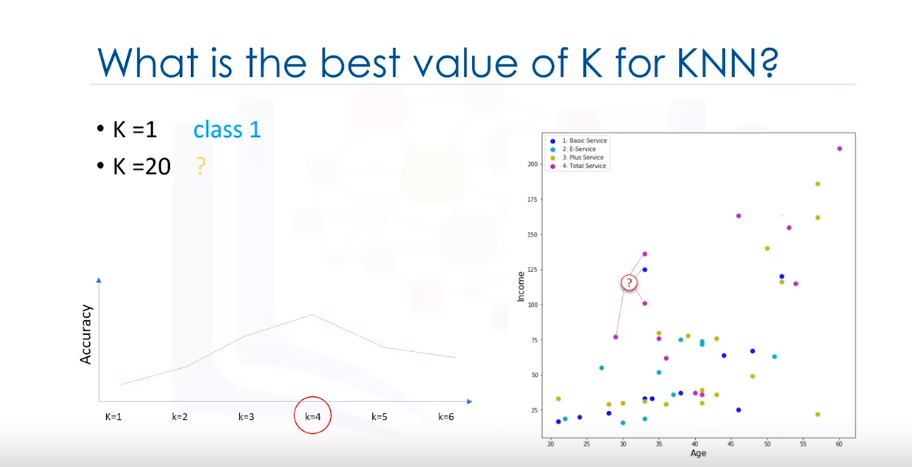
K and K-Nearest Neighbors is the number of nearest neighbors to examine. It is supposed to be specified by the user.

A low value of K causes a highly complex model as well, which might result in overfitting of the model.

It means the prediction process is not generalized enough to be used for out-of-sample cases.

Out-of-sample data is data that is outside of the data set used to train the model.

Now, on the opposite side of the spectrum, if we choose a very high value of K such as K equals 20, then the model becomes overly generalized.

In order to find the best value of k, the general solution is to reserve a part of your data for testing the accuracy of the model, then we choose k-class (like k-1, or k-2 and so on) and then use the training data to calculate the optimal value of k which gives the highest accuracy among other k-values. In other words, choose k=1 first, and then use the training part for modeling and calculate the accuracy of prediction using all samples in your test set. Repeat this process increasing the K and see which K is best for your model.

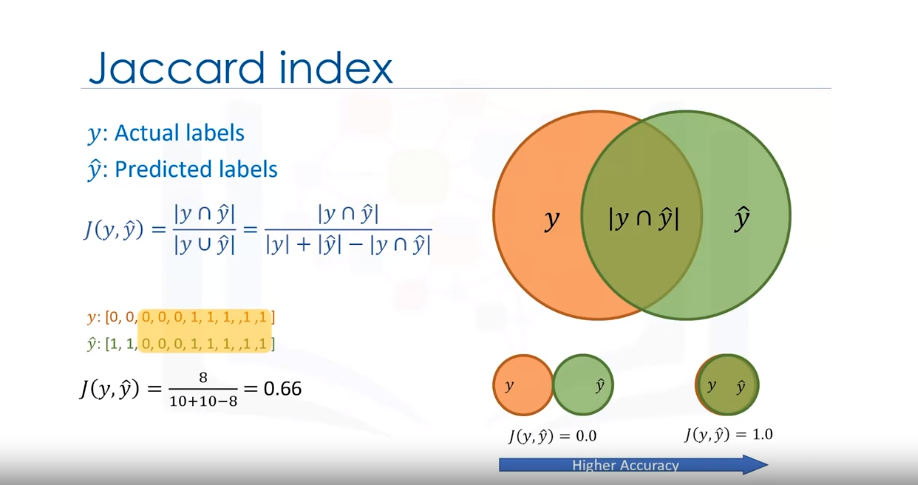


**How to evaluate the best classification model?**

**Evaluation Metrics or Accuracy Metrics:**

Evaluation metrics provide a key role in building the best machine learning model.

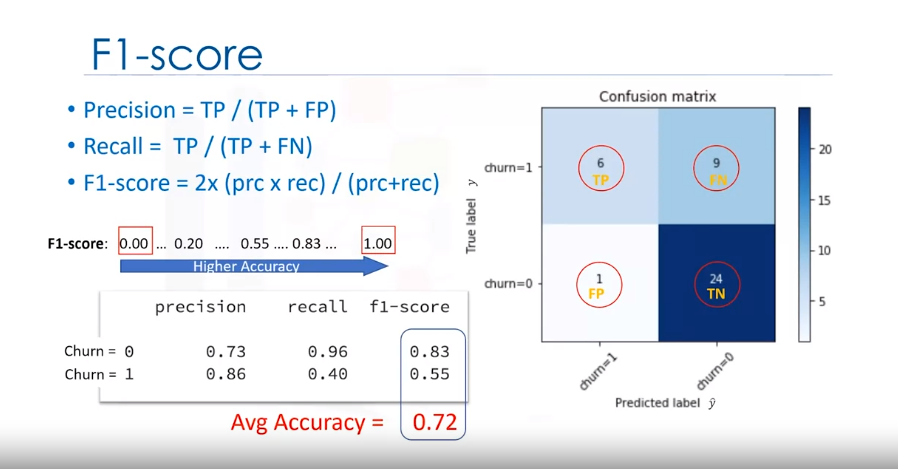
**Jaccard Index:**



**F1-Score:**

This matrix shows the corrected and wrong predictions, in comparison with the actual labels.

Each confusion matrix row shows the Actual/True labels in the test set, and the columns show the predicted labels by classifier.



A good thing about the confusion matrix is that it shows the model’s ability to correctly predict or separate the classes.

In the specific case of a binary classifier, such as this example, we can interpret these

numbers as the count of true positives, false negatives, true negatives, and false positives.

Based on the count of each section, we can calculate the precision and recall of each

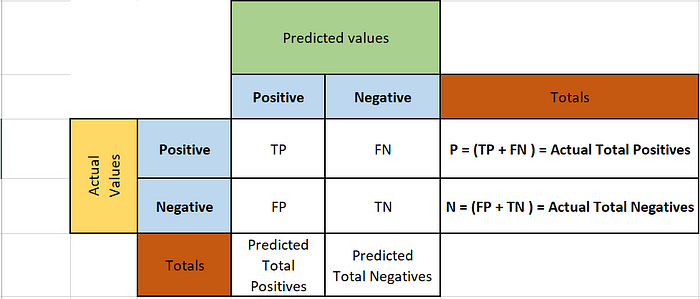
label.

Precision is a measure of the accuracy, provided that a class label has been predicted.

It is defined by:

**Precision = True Positives / (True Positives + False Positives).**

**Recall = True Positives / (True Positives + False Negatives)**



**Log Loss:**

It measures the performance of a classifier where the predicted output is a probability of value between 0 and 1.

The output of a classifier is the probability of a class label, instead of the label. For example, in logistic regression, the output can be the probability of customer churn, i.e., yes (or equals to 1).

This probability is a value between 0 and 1. Logarithmic loss (also known as Log loss) measures the performance of a classifier where the predicted output is a probability value between 0 and 1.

**Logloss = - 1 / n ∑ ( Yactual x log(Ypredicted) + (1 – Yactual) x log(1 – Ypredicted) )**

LogLoss measures how far each prediction is, from the actual label.

So, lower the logloss is, the better the model is.

**Most important points and summary:**

**K-Nearest Neighbors** is a supervised learning algorithm. Where the data is 'trained' with data points corresponding to their classification. To predict the class of a given data point, it takes into account the classes of the 'K' nearest data points and chooses the class in which the majority of the 'K' nearest data points belong to as the predicted class.

### **Train Test Split**

Out of Sample Accuracy is the percentage of correct predictions that the model makes on data that the model has NOT been trained on. Doing a train and test on the same dataset will most likely have low out-of-sample accuracy, due to the likelihood of our model overfitting.

It is important that our models have a high, out-of-sample accuracy, because the purpose of any model, of course, is to make correct predictions on unknown data. So how can we improve out-of-sample accuracy? One way is to use an evaluation approach called Train/Test Split. Train/Test Split involves splitting the dataset into training and testing sets respectively, which are mutually exclusive. After which, you train with the training set and test with the testing set.

This will provide a more accurate evaluation on out-of-sample accuracy because the testing dataset is not part of the dataset that has been used to train the model. It is more realistic for the real world problems.

**Data Standardization gives the data zero mean and unit variance, it is good practice, especially for algorithms such as KNN which is based on the distance of data points:**

### Accuracy evaluation

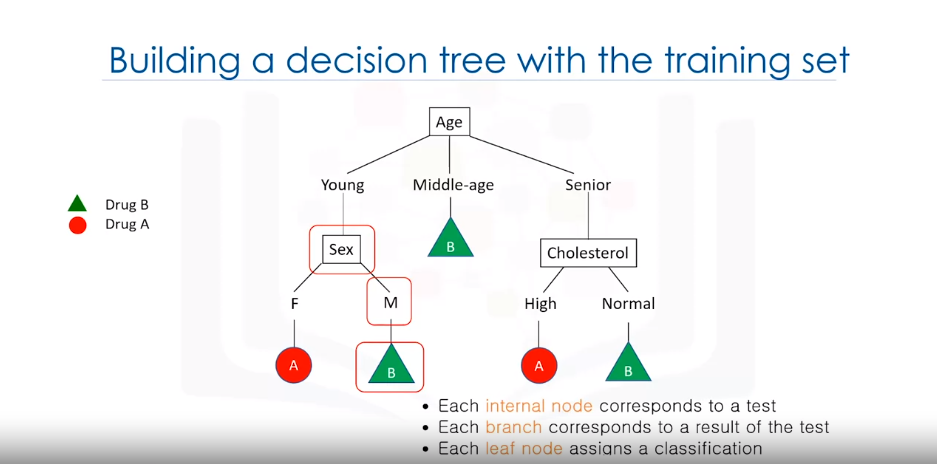
In multilabel classification, **accuracy classification score** is a function that computes subset accuracy. This function is equal to the jaccard\_score function. Essentially, it calculates how closely the actual labels and predicted labels are matched in the test set.

#### What about other K?

K in KNN, is the number of nearest neighbors to examine. It is supposed to be specified by the user. So, how can we choose right value for K? The general solution is to reserve a part of your data for testing the accuracy of the model. Then choose k =1, use the training part for modeling, and calculate the accuracy of prediction using all samples in your test set. Repeat this process, increasing the k, and see which k is the best for your model.

We can calculate the accuracy of KNN for different values of k.

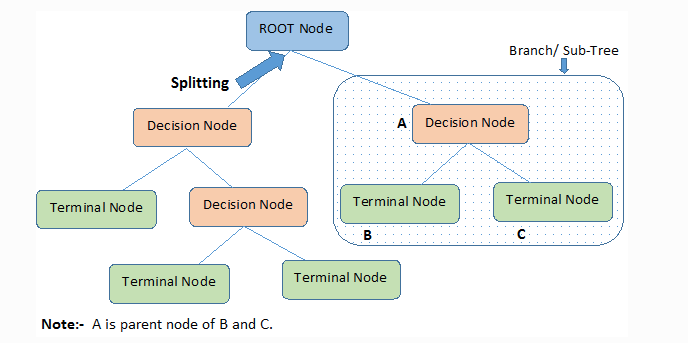
**Decision Tree**

The basic intuition behind a decision tree is to map out all the possible decision paths in the form of a tree.

Decision trees are built using recursive partitioning to classify the data.

**Recursive Partitioning:**

Recursive partitioning is a statistical method for multivariable analysis. Recursive partitioning creates a decision tree that strives to correctly classify members of the population by splitting it into sub-populations based on several dichotomous independent variables

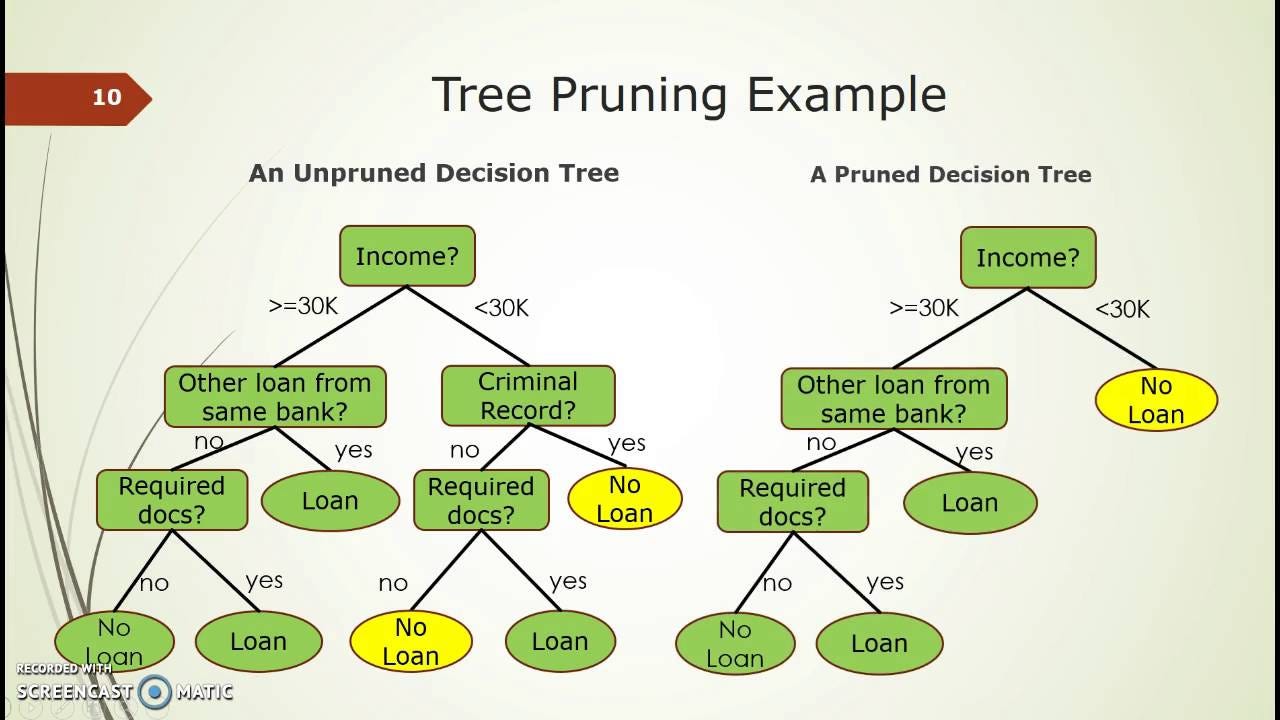


A diagram of a decision tree

Description automatically generated

Let's say we have 14 patients in our data set, the algorithm chooses the most predictive feature to split the data on. What is important in making a decision tree, is to determine which attribute is the best or more predictive to split data based on the feature.

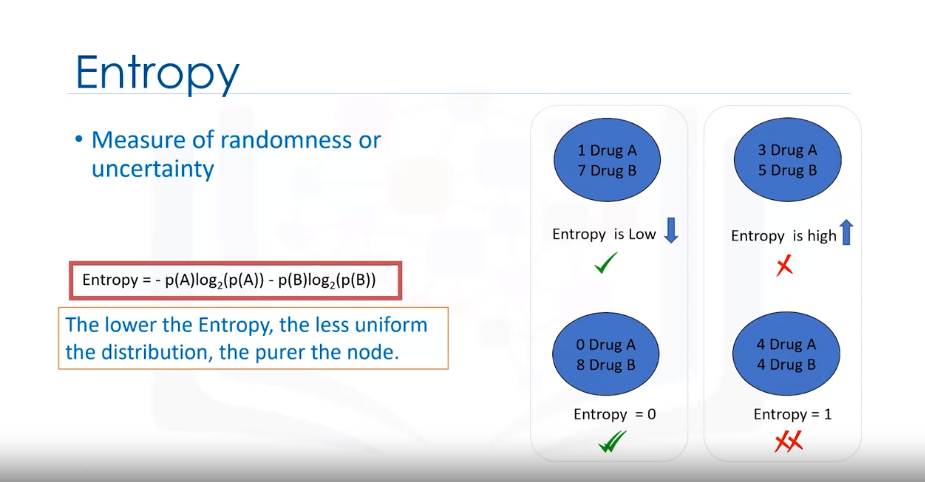
EXAMPLE:



**Entropy (Decision Tree):**

It is a measure of randomness or uncertainty. The lower the entropy, the less uniform the distribution, the purer the node.

The entropy is used to calculate the homogeneity of the samples in that node. If the samples are completely homogeneous, the entropy is zero and if the samples are equally divided it has an entropy of one.

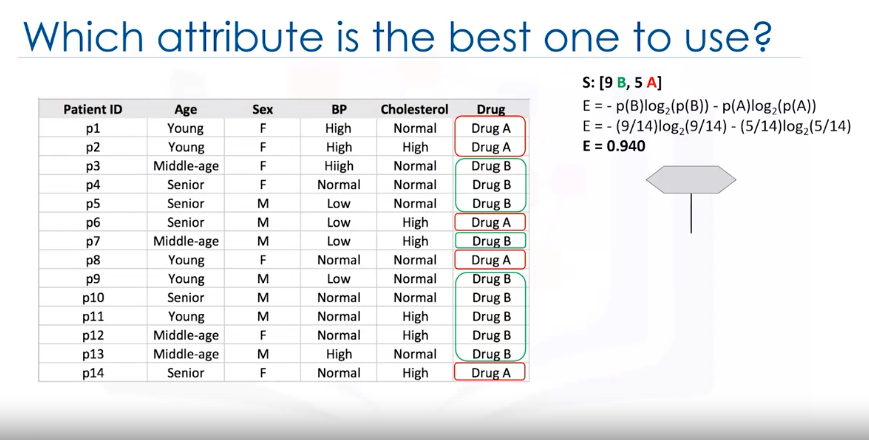


**But remember you don’t have to calculate entropy of attribute manually since the python libraries easily calculate it for you.**

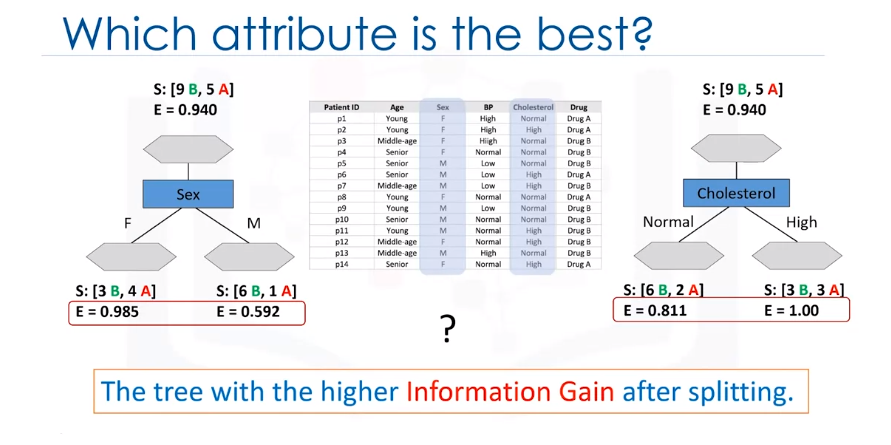
**How to find the best attribute for Decision Tree Model?**

Step 1 - Before spitting the data and before considering the best attribute to build the decision tree model, you need to calculate the total occurrences of the target attribute classes ( like in the example below):

As an example, let's calculate the entropy of the data set before splitting it. We have nine occurrences of drug B and five of drug A. You can embed these numbers into the entropy formula to calculate the impurity of the target attribute before splitting it. In this case, it is 0.94.



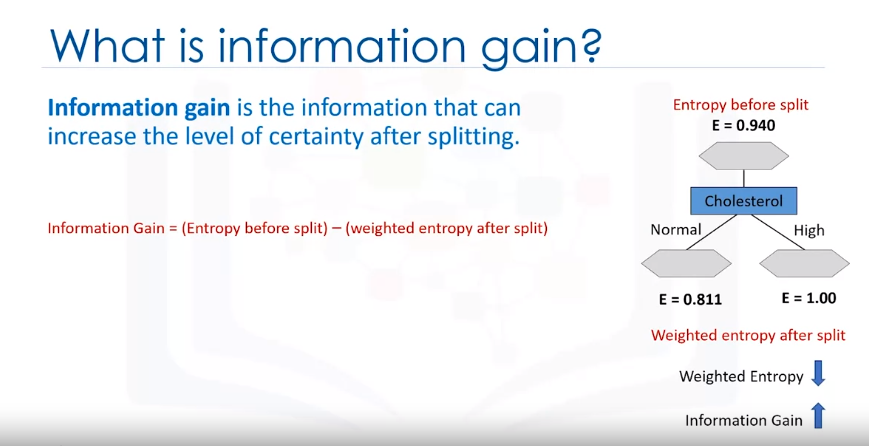
Step 2 – after calculating the entropy of node, you need to select each attribute and calculate entropies with respect to each decision node.

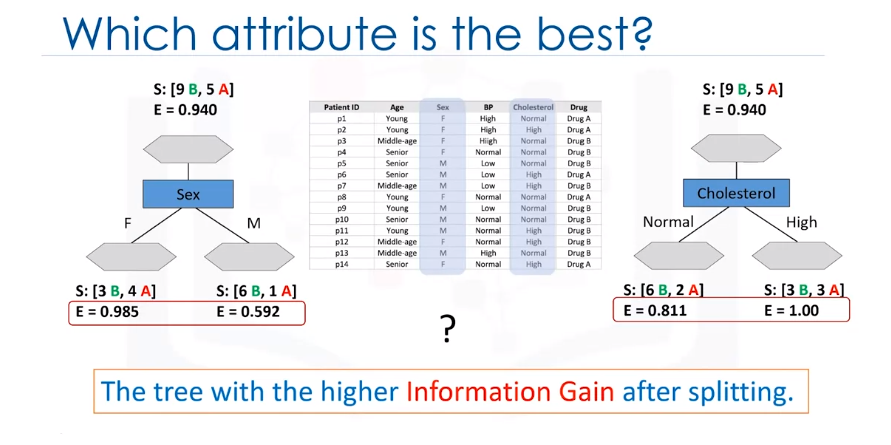
Example:

**Information Gain:**

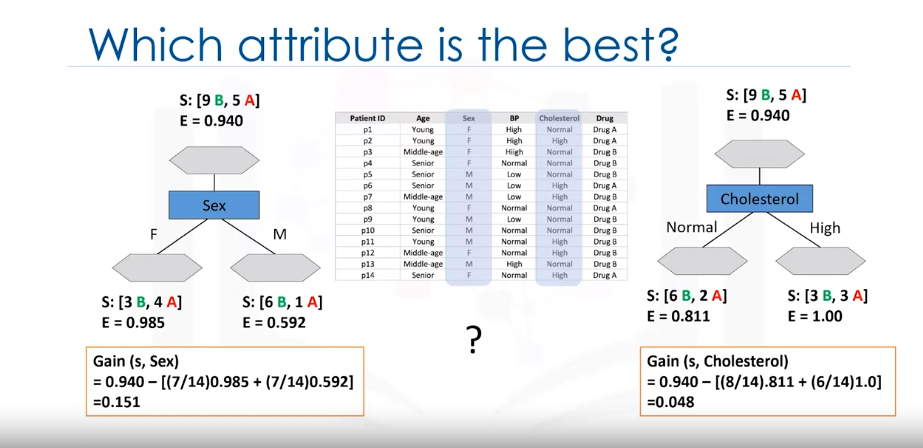
Information gain is the information that can increase the level of certainty after splitting.

It is the entropy of a tree before the split minus the weighted entropy after the split by an attribute. We can think of information gain and entropy as opposites. As entropy or the amount of randomness decreases, the information gain or amount of certainty increases and vice versa.





As you could see, we will consider the entropy over the distribution of samples falling under each leaf node and we'll take a weighted average of that entropy weighted by the proportion of samples falling under that leave.



Well, as mentioned, the tree with the higher information gain after splitting, this means the **sex attribute.**

Simply put, an attribute with higher INFORMATION GAIN is selected for the Decision Tree Prediction Model.

Now, what is the next attribute after branching by the sex attribute? Well, as you can guess, we should repeat the process for each branch and test each of the other attributes to continue to reach the most pure leaves. This is the way you build a decision tree.

**IMPORTANT POINTS AND SUMMARY:**

<https://www.youtube.com/watch?v=gdkUs4fNIHc>