1. What exactly is a feature? Give an example to illustrate your point.

**What are Product Features? Definition, Examples and Types**

**Product features are attributes or function that adds value to the customer base of the product**. It may consist of features, functional components, or performance enhancements. **Product features are the essence of what makes a product valuable and appealing to customers.**

Products are made up of their features, and product managers are aware that **adding the proper features can increase adoption rates and provide a positive return on investment.** From innovative functionalities to user-friendly designs, each feature plays a crucial role in defining the product’s utility and attractiveness in the market.

If the right research and prioritization are done, it might be easier to balance consumer pleasure and product vision. In this article, we will learn what is Product Feature, how to define product features, its components, types, examples and some tips to Prioritize the features of a Product.

## What is Product Feature?

A**product feature is an attribute or function that adds value to the customer base of the product**. It may consist of features, functional components, or performance enhancements. Product feature refers to a specific characteristic or functionality of a product that provides value to the customer.**It’s what distinguishes the product from others in the market and addresses particular needs or desires of the target audience**. One of the most important aspects of a product manager’s job is defining and prioritizing items to build.

## How to Define Product Features

“**A feature is something your product has or is**,” according to Dan Shewan, “usually functionality offered by a software program that enables users to do something.”

This **definition is supported by the features listed above for Google Docs and the MacBook Pro. When features are utilized in this manner, they both explain your product and aid in the decision-making process for potential buyers.** To determine whether Product board can assist you with your specific [product management](https://www.geeksforgeeks.org/product-management/) problems, you can review its feature list.

## ****What are Product Features in Marketing?****

**Product features, as used in marketing, are the particular attributes or features of a product that set it apart from the competition and add value for buyers**. These characteristics demonstrate the product’s capabilities and value proposition. They help customers understand the benefits of the product and why it’s better than alternatives. They are the essential distinctions between items in the same market that make companies stand out from their competitors.

## Why are Product Features Important?

**Product Features** are important, Because **they provide clients with the value and functionality they require to complete their activities**. Features contribute to improving the customer experience, which raises revenue and reduces churn while increasing customer retention rates. **Product Features give**[**customer Satisfaction**](https://www.geeksforgeeks.org/customer-satisfaction-surveys-meaning-importance-and-survey-questions/)**and it is good for our product’s growth.** When a product has useful features that meet customer needs, it leads to happier customers. For example, if a software application has a user-friendly interface and helpful tools, customers are more likely to enjoy using it.

product features are the backbone of what makes a product attractive, useful, and successful in the market. They’re essential for both attracting customers initially and keeping them satisfied over time.

## What are Features and Benefits of a Product?

When assessing a product, features and benefits are crucial factors to take into account.

### Features

**Development and features of a product are greatly influenced by marketing.** Developing features for your **product that customers desire and that your rivals lack will make it simpler for a marketing team to craft sales pitches that highlight the benefits of the new features.** Your feature might not be in line with what your customers require if your marketing team is unable to explain why it would benefit them or why it will solve their problem.

### Benefits

**Benefits and features are distinct from one another, although they are related. A particular part of the product is called a feature. The advantage that users will experience while utilizing the function is in the interim**. Consider Slack as an example. The ability to message coworkers instantly is a feature of the product. Team members may work on projects more quickly by utilizing this feature since it makes communication simple. Instead of purchasing communication tools, people choose to purchase efficiently.

## Product Features Vs Benefits

Here are the following differences between Features and Benefits:

| Features | ****Benefits**** |
| --- | --- |
| Features defines different aspects of the product. It can be technical or descriptive. | Benefits refers to the improved life when they use it. |
| **Features are the physical characteristics of your product** features are what you are selling. | **Benefits are why the customers is there**people buy good feelings. |
| Simply **features tell customers what.** | Simply **benefits tell customers why.** |
| Feature **refers to what your product or service is, what it includes**, how it was made. | Benefits **refers to how customers’ lives are improved**when they use it. |
| We can say a feature is what something is. | We can say a benefit is what something does. |
| It is obtained from fact about a product or service. | It is obtained when the feature meets the need. |
| Features have almost **no emotional effect on user’s prospect.** | **Benefits have almost emotional effect** on user’s prospect. |
| Features gives rise to advantage. | Advantage gives rise to the benefits. |

## Product Features Vs Epics, User Stories, and Requirements

Let Us learn Product Epics, User Stories, and Requirements along with Product Features:

| **Term** | **Meaning** |
| --- | --- |
| Epic | An **epic is like a big project or goal for making something new, like a game or app.** It’s too big to tackle all at once, so we break it into smaller tasks to work on step by step. |
| Feature | A **feature is something a product can do to help users**. It’s like a special function or capability that makes the product useful or interesting. |
| User Story | A user story is like a short story about what a user wants to do with a product, told from the user’s perspective. It helps developers understand what users need and guides the creation of features. |
| Requirement | **Requirements are like detailed instructions that explain exactly what a product needs to do and how it should work**. They include specific details and criteria that must be met for the product to be successful. |

## Product feature Components

The secret to a successful feature specification is consistency. Since most product teams conform to a predetermined framework, every feature’s specifics and approach are clear. The specifics may differ, but the following is a summary of the elements that should be included in a product feature:

| **Feature name** | **A title that describes the functionality you want to develop** |
| --- | --- |
| **Overview** | The functionality of the feature is explained, and you can contribute any background information that would help the team. |
| **Timing** | When do you expect to release the updated user experience |
| **Status** | * Not started * On track * At risk |
| **Team** | [Product manager](https://www.geeksforgeeks.org/roles-and-responsibilities-of-product-manager/), development team, designers, QA, etc. |
| **Strategic alignment** | An explanation of how this advances the objectives of the product and business, and why this feature is being developed now. |
| **Who it benefits** | Who will benefit from the feature — link to any personas you have. |
| **User challenge** | The problem the user is trying to solve and ways they may be attempting to solve it. |
| **Value score** | The value estimate for the feature. |
| **Design / UX** | Link to ongoing design experiments or mockups |
| **Impacted functionality** | Note any additional functionality that this feature would impact. |
| **Open questions** | Any queries the group may have for non-functional groups or one another. |

## Types of Products Features

A **new product will come with a variety of feature sets**. You may set yourself apart from the competition by observing what features your rivals’ offerings lack. Here are five different product feature types to think about.

### Function

In most cases, **functionality refers to what or how a person may utilise a product.**The idea is to think of ways a product might relieve a user’s pain and assist in completing a task. Certain characteristics are common, however certain goods can develop additional useful features.

### Experience

**Experience is about a product’s intangible qualities.** Even though they might not be as necessary as functionality, they can help make consumers’ experiences memorable and even win them over as devoted patrons. A seamless onboarding procedure or receiving 5-star customer assistance are a couple of instances.

### Quality

Qualities are associated with tangible and intangible characteristics. Another **benchmark for a product’s perceived value is its quality.** For instance, you could be prepared to pay more for clothing from luxury brands like Gucci because you anticipate high-quality materials from them.

### Design

**Another kind of product element that is solely aesthetic is design.** One approach to set oneself apart from competitors producing identical things is through design or style. Potential customers may be won over by a phone case’s appearance, so it’s critical to take these circumstances into account while designing a product.

### Added Value

Added Value The focus of added value product feature types is on the inclusions of a product. Customers who **believe they are getting more features for the same price may place a higher value on your goods**. Some online retailers may achieve this by including gifts with orders. SaaS products may provide free product setup.

## Product Features Example

Here are the following examples of Product Feature:

### Wave

Wave **provides tools to streamline invoicing and bookkeeping**. The option to link a bank account to the accounting software is one of the functions. **End users gain from the fact that there is no effort required on their part because their cash flow is automatically updated in real-time.** Owners of businesses can save time by not having to monitor credit card usage. Therefore, it should come as no surprise that the marketing slogan is “Make tax time a breeze.”

### Duolingo

A **language-learning software is called Duolingo. Personalised lessons created using AI and language science are among its salient characteristics.** The ability to learn a new language at their own speed and ability level is the product’s benefit for users. This translates to “Language courses that efficiently teach reading, listening, and speaking skills” in the marketing pitch with ease.

### Wix

With many capabilities, **Wix is a drag-and-drop website builder. The online scheduler is one of these capabilities.**Using templates to add an online calendar to a website is a great way for website owners to increase revenue. “Let clients easily book appointments online” is the feature’s marketing tagline.

### Headspace

You might or might not have anticipated a partnership with singer John Legend, but people frequently utilise meditation applications like Headspace to aid with sleep. John Legend reads a series of audio stories on Headspace that are intended to help people fall asleep. By utilising this option, listeners can enhance the quality of their sleep. The lovely tagline “Fall in love with sleep and practise self-love” is used to talk about this benefit.

### Starbucks

**Starbucks is not limited to merely drinks. Check out EarthSleeves. Drinkers are able to hold their hot beverages without feeling the heat because the sleeves fit closely around the cup.** It serves as both an experience and an additional value element because it is also environmentally sustainable.

## Tips to Prioritize the Features of a Product

No matter how big or little your business is, there are always a lot of different viewpoints and suggestions for the next feature to add to your product. In order to optimise the worth of suggested concepts, a prioritisation procedure needs to be put in place.

It’s a difficult process, though, because a lot of stakeholders will want to know that the correct product feature is being developed to guarantee profitability. These are six suggestions for selecting a product feature that complements your company.

### Remember your business goals

A right product manager will ensure that new **features align with the broader business strategy and the**[**product plan**](https://www.geeksforgeeks.org/product-planning-introduction-purpose-importance-and-steps/). This might involve making sure that all parties are in agreement and reminding stakeholders of the direction of a product. It will be simpler to gather solid product ideas and win over key stakeholders if you do this.

### Ensure ideas are complete

Ideas can come from a variety of sources. You can get ideas from every team in your business, conduct concept tests, or get feedback from your clients. Whatever the source, it’s critical to make sure concepts are communicated clearly. For example, a feature can be proposed without a justification for its necessity. Finding out how a feature would improve a product overall may require following up with that individual.

### Merge duplicate ideas

People will frequently offer ideas that are similar to one another, so you might wish to merge them to shorten your list of original ideas. However, you should proceed with caution. It is not desirable to combine similar concepts and lose specific specifics. Subsequent concepts are grouped underneath the main idea. This allows you to retain valuable information that may be used to assess the need for a certain product feature.

### Use a scoring system

It’s time to start assigning a rank to each concept. **Product managers ought to take into account a minimum of two metrics: impact and effort.** Idea A, for instance, might have a significant impact, but it might also be more expensive and yield a lower return on investment than Idea B. It will be simpler to reject or accept ideas if the impact and effort of each product idea are thoroughly understood and ranked.

### Select an idea

You ought to have a better idea of which product features will benefit the organisation the most when you’ve finished the grading system. Occasionally, difficult choices must be taken, particularly when stakeholders disagree on the best course of action. Nevertheless, in order for the team to perform concept testing and determine precisely what has to be built, a call for promoting an idea to a product feature needs to be made.

### Provide feedback to stakeholders

Those who submit suggestions will be curious as to why they were turned down. It is crucial to explain this, as it fosters communication and increases internal openness in an organization. You may clarify why an idea was turned down and then confirm the direction and placement of the product by using the scoring system.

1. What are the various circumstances in which feature construction is required?

An alternative or supplemental approach to facilitate the detection and modeling of interactions is to apply feature construction (see Fig. 1), also known as constructive induction or feature extraction. Feature construction methods, e.g. principle component analysis or linear discriminant analysis [68], define new features as a function of two or more other features [74]. This subset of constructed features can be added to the original feature space, or analyzed in its place (achieving dimensionality reduction). A common side effect of most any feature construction method is that the original features are no longer recognizable, leading to challenges in downstream model interpretability.

One feature construction method geared specifically towards capturing feature interactions is multifactor dimensionality reduction (MDR) [87]. Another more general example is polynomial feature construction that is able to detect multiplicative interactions [106]. These approaches attempt to combine individual features that may be interacting and construct a single feature that can be more easily identified as relevant using any simple feature selection or induction method. There are many possible feature construction approaches to chose from and some can be quite computationally expensive. Notably, applying feature construction does not necessarily preclude the need for feature selection. Thus, assuming that a feature selection and modeling approach has been chosen that is sensitive to a target interaction dimensionality (e.g. 2-way or 3-way), it may be most efficient to skip feature construction, particularly if downstream model interpretation is critical. While feature construction certainly has its own utility, further discussion is outside the scope of this review.

1. Describe how nominal variables are encoded.

### Encoding nominal categories (without assuming any order)

OneHotEncoder is an alternative encoder that prevents the downstream models to make a false assumption about the ordering of categories. For a given feature, it creates as many new columns as there are possible categories. For a given sample, the value of the column corresponding to the category is set to 1 while all the columns of the other categories are set to 0.

We can encode a single feature (e.g. "education") to illustrate how the encoding works.

from sklearn.preprocessing import OneHotEncoder

encoder = OneHotEncoder(sparse\_output=False).set\_output(transform="pandas")

education\_encoded = encoder.fit\_transform(education\_column)

education\_encoded

|  | **education\_ 10th** | **education\_ 11th** | **education\_ 12th** | **education\_ 1st-4th** | **education\_ 5th-6th** | **education\_ 7th-8th** | **education\_ 9th** | **education\_ Assoc-acdm** | **education\_ Assoc-voc** | **education\_ Bachelors** | **education\_ Doctorate** | **education\_ HS-grad** | **education\_ Masters** | **education\_ Preschool** | **education\_ Prof-school** | **education\_ Some-college** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **1** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **2** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **3** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 |
| **4** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **48837** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **48838** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **48839** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **48840** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **48841** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 |

48842 rows × 16 columns

Note

sparse\_output=False is used in the OneHotEncoder for didactic purposes, namely easier visualization of the data.

Sparse matrices are efficient data structures when most of your matrix elements are zero. They won’t be covered in detail in this course. If you want more details about them, you can look at [this](https://scipy-lectures.org/advanced/scipy_sparse/introduction.html#why-sparse-matrices).

We see that encoding a single feature gives a dataframe full of zeros and ones. Each category (unique value) became a column; the encoding returned, for each sample, a 1 to specify which category it belongs to.

Let’s apply this encoding on the full dataset.

print(f"The dataset is composed of {data\_categorical.shape[1]} features")

data\_categorical

The dataset is composed of 8 features

|  | **workclass** | **education** | **marital-status** | **occupation** | **relationship** | **race** | **sex** | **native-country** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | Private | 11th | Never-married | Machine-op-inspct | Own-child | Black | Male | United-States |
| **1** | Private | HS-grad | Married-civ-spouse | Farming-fishing | Husband | White | Male | United-States |
| **2** | Local-gov | Assoc-acdm | Married-civ-spouse | Protective-serv | Husband | White | Male | United-States |
| **3** | Private | Some-college | Married-civ-spouse | Machine-op-inspct | Husband | Black | Male | United-States |
| **4** | ? | Some-college | Never-married | ? | Own-child | White | Female | United-States |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... |
| **48837** | Private | Assoc-acdm | Married-civ-spouse | Tech-support | Wife | White | Female | United-States |
| **48838** | Private | HS-grad | Married-civ-spouse | Machine-op-inspct | Husband | White | Male | United-States |
| **48839** | Private | HS-grad | Widowed | Adm-clerical | Unmarried | White | Female | United-States |
| **48840** | Private | HS-grad | Never-married | Adm-clerical | Own-child | White | Male | United-States |
| **48841** | Self-emp-inc | HS-grad | Married-civ-spouse | Exec-managerial | Wife | White | Female | United-States |

48842 rows × 8 columns

data\_encoded = encoder.fit\_transform(data\_categorical)

data\_encoded[:5]

|  | **workclass\_ ?** | **workclass\_ Federal-gov** | **workclass\_ Local-gov** | **workclass\_ Never-worked** | **workclass\_ Private** | **workclass\_ Self-emp-inc** | **workclass\_ Self-emp-not-inc** | **workclass\_ State-gov** | **workclass\_ Without-pay** | **education\_ 10th** | **...** | **native-country\_ Portugal** | **native-country\_ Puerto-Rico** | **native-country\_ Scotland** | **native-country\_ South** | **native-country\_ Taiwan** | **native-country\_ Thailand** | **native-country\_ Trinadad&Tobago** | **native-country\_ United-States** | **native-country\_ Vietnam** | **native-country\_ Yugoslavia** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 |
| **1** | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 |
| **2** | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 |
| **3** | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 |
| **4** | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 |

5 rows × 102 columns

print(f"The encoded dataset contains {data\_encoded.shape[1]} features")

The encoded dataset contains 102 features

Look at how the "workclass" variable of the 3 first records has been encoded and compare this to the original string representation.

The number of features after the encoding is more than 10 times larger than in the original data because some variables such as occupation and native-country have many possible categories.

1. Describe how numeric features are converted to categorical features.

# How to convert Categorical features to Numerical Features in Python?

**Last Updated :**26 Jan, 2022

It’s difficult to create machine learning models that can’t have features that have categorical values, such models cannot function. categorical variables have string-type values. thus we have to convert string values to numbers. This can be accomplished by creating new features based on the categories and setting values to them. In this article, we are going to see how to convert Categorical features to Numerical Features in Python

## Stepwise Implementation

### Step 1: Import the necessary packages and modules

* Python3

|  |
| --- |
| # import packages and modules  import numpy as np  import pandas as pd  from sklearn import preprocessing |

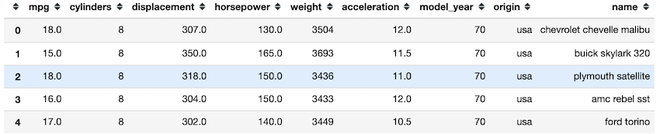
### Step 2: Import the CSV file

We will use the pandas [read\_csv()](https://www.geeksforgeeks.org/python-read-csv-using-pandas-read_csv/) method to import the CSV file. To view and download the CSV file used click [here](https://media.geeksforgeeks.org/wp-content/cdn-uploads/20220123233310/cluster_mpg.csv).

* Python3

|  |
| --- |
| # import the CSV file  df = pd.read\_csv('cluster\_mpg.csv')  print(df.head()) |

**Output:**



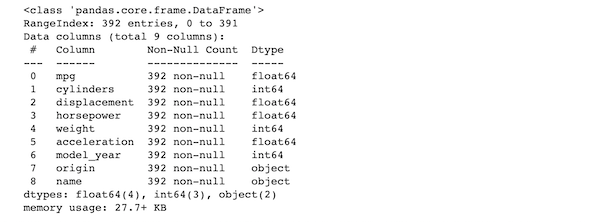
### Step 3: Get all features with categorical values

We use [df.info()](https://www.geeksforgeeks.org/python-pandas-dataframe-info/) to find categorical features. Categorical features have Dtype as “object”.

* Python3

|  |
| --- |
| df.info() |

**Output:**



In the given database columns “origin” and “name” is object type.

### Step 4: Convert string values of origin column to numerical values

We will fit the “origin” column using preprocessing.LabelEncoder().fit() method.

* Python3

|  |
| --- |
| label\_encoder = preprocessing.LabelEncoder()  label\_encoder.fit(df["origin"]) |

### Step 5: Get the unique values out of the categorical features

We will use label\_encoder.classes\_ attribute for this purpose.

*classes\_:ndarray of shape (n\_classes,)*

*Holds the label for each class.*

* Python3

|  |
| --- |
| # finding the unique classes  print(list(label\_encoder.classes\_))  print() |

**Output**

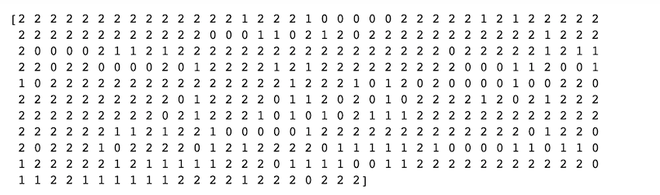
['europe', 'japan', 'usa']

### Step 6: Transforming the categorical values

* Python3

|  |
| --- |
| # values after transforming the categorical column.  print(label\_encoder.transform(df["origin"])) |

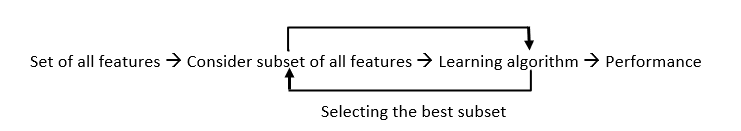
**Output:**



1. Describe the feature selection wrapper approach. State the advantages and disadvantages of this approach?

**Wrapper methods:**

Wrapper methods, also referred to as greedy algorithms train the algorithm by using a subset of features in an iterative manner. Based on the conclusions made from training in prior to the model, addition and removal of features takes place. Stopping criteria for selecting the best subset are usually pre-defined by the person training the model such as when the performance of the model decreases or a specific number of features has been achieved. The main advantage of wrapper methods over the filter methods is that they provide an optimal set of features for training the model, thus resulting in better accuracy than the filter methods but are computationally more expensive.



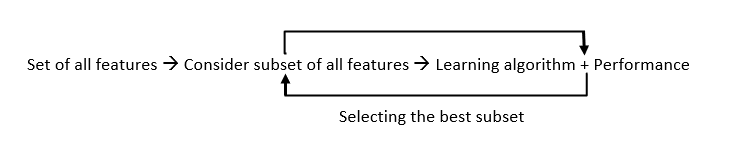
*Wrapper Methods Implementation*

Some techniques used are:

* **Forward selection –**This method is an iterative approach where we initially start with an empty set of features and keep adding a feature which best improves our model after each iteration. The stopping criterion is till the addition of a new variable does not improve the performance of the model.
* **Backward elimination –** This method is also an iterative approach where we initially start with all features and after each iteration, we remove the least significant feature. The stopping criterion is till no improvement in the performance of the model is observed after the feature is removed.
* **Bi-directional elimination –** This method uses both forward selection and backward elimination technique simultaneously to reach one unique solution.
* **Exhaustive selection –** This technique is considered as the brute force approach for the evaluation of feature subsets. It creates all possible subsets and builds a learning algorithm for each subset and selects the subset whose model’s performance is best.
* **Recursive elimination –** This greedy optimization method selects features by recursively considering the smaller and smaller set of features. The estimator is trained on an initial set of features and their importance is obtained using feature\_importance\_attribute. The least important features are then removed from the current set of features till we are left with the required number of features.

**Embedded methods:**

In embedded methods, the feature selection algorithm is blended as part of the learning algorithm, thus having its own built-in feature selection methods. Embedded methods encounter the drawbacks of filter and wrapper methods and merge their advantages. These methods are faster like those of filter methods and more accurate than the filter methods and take into consideration a combination of features as well.



*Embedded Methods Implementation*

Some techniques used are:

* **Regularization –** This method adds a penalty to different parameters of the machine learning model to avoid over-fitting of the model. This approach of feature selection uses Lasso (L1 regularization) and Elastic nets (L1 and L2 regularization). The penalty is applied over the coefficients, thus bringing down some coefficients to zero. The features having zero coefficient can be removed from the dataset.
* **Tree-based methods –**These methods such as Random Forest, Gradient Boosting provides us feature importance as a way to select features as well. Feature importance tells us which features are more important in making an impact on the target feature.

**Conclusion:**

Apart from the methods discussed above, there are many other methods of feature selection. Using hybrid methods for feature selection can offer a selection of best advantages from other methods, leading to reduce in the disadvantages of the algorithms. These models can provide greater accuracy and performance when compared to other methods. Dimensionality reduction techniques such as Principal Component Analysis (PCA), Heuristic Search Algorithms, etc. don’t work in the way as to feature selection techniques but can help us to reduce the number of features.

Feature selection is a wide, complicated field and a lot of studies has already been made to figure out the best methods. It depends on the machine learning engineer to combine and innovate approaches, test them and then see what works best for the given problem.

1. When is a feature considered irrelevant? What can be said to quantify it?

**Feature Subset Selection Process**

**Feature Selection** is the most critical pre-processing activity in any machine learning process. It intends to select a subset of attributes or features that makes the most meaningful contribution to a machine learning activity. In order to understand it, let us consider a small example i.e. **Predict the weight of students based on the past information about similar students**, which is captured inside a ‘Student Weight’ data set. The data set has 04 features like **Roll Number, Age, Height & Weight.**Roll Number has no effect on the weight of the students, so we eliminate this feature. So now the new data set will be having only 03 features. This subset of the data set is expected to give better results than the full set.

| **Age** | **Height** | **Weight** |
| --- | --- | --- |
| 12 | 1.1 | 23 |
| 11 | 1.05 | 21.6 |
| 13 | 1.2 | 24.7 |
| 11 | 1.07 | 21.3 |
| 14 | 1.24 | 25.2 |
| 12 | 1.12 | 23.4 |

**The above data set is a reduced dataset.** Before proceeding further, we should look at the fact why we have reduced the dimensionality of the above dataset OR what are the issues in **High Dimensional Data?**

**High Dimensional**refers to the high number of variables or attributes or features present in certain data sets, more so in the domains like DNA analysis, geographic information system (GIS), etc.  It may have sometimes hundreds or thousands of dimensions which is not good from the machine learning aspect because it may be a big challenge for any ML algorithm to handle that. On the other hand, a high quantity of computational and a high amount of time will be required. Also, a model built on an extremely high number of features may be very difficult to understand. **For these reasons, it is necessary to take a subset of the features instead of the full set.** So we can deduce that the objectives of feature selection are:

1. Having a faster and more cost-effective (less need for computational resources) learning model
2. Having a better understanding of the underlying model that generates the data.
3. Improving the efficacy of the learning model.

**Main Factors Affecting Feature Selection**

**a. Feature Relevance:** In the case of supervised learning, the input data set (which is the training data set), has a class label attached. A model is inducted based on the training data set — so that the inducted model can assign class labels to new, unlabeled data. Each of the predictor variables, ie expected to contribute information to decide the value of the class label. In case of a variable is not contributing any information, it is said to be irrelevant. In case the information contribution for prediction is very little, the variable is said to be weakly relevant. The remaining variables, which make a significant contribution to the prediction task are said to be strongly relevant variables.

In the case of unsupervised learning, there is no training data set or labelled data. Grouping of similar data instances are done and the similarity of data instances are evaluated based on the value of different variables. Certain variables do not contribute any useful information for deciding the similarity of dissimilar data instances. Hence, those variable makes no significant contribution to the grouping process. These variables are marked as irrelevant variables in the context of the unsupervised machine learning task.

We can understand the concept by taking a real-world example: At the start of the article, we took a random dataset of the student. In that, Roll Number doesn’t contribute any significant information in predicting what the Weight of a student would be. Similarly, if we are trying to group together students with similar academic capabilities, *Roll No* can really not contribute any information. So, in the context of grouping students with similar academic merit, the variable *Roll No* is quite irrelevant.  Any feature which is irrelevant in the context of a machine learning task is a candidate for rejection when we are selecting a subset of features.

**b. Feature Redundancy:** A feature may contribute to information that is similar to the information contributed by one or more features. For example, in the Student Data-set, both the features **Age & Height** contribute similar information. This is because, with an increase in age, weight is expected to increase. Similarly, with the increase in Height also weight is expected to increase. So, in context to that problem, Age and Height contribute similar information. In other words, irrespective of whether the feature **Height** is present or not, the learning model will give the same results. In this kind of situation where one feature is similar to another feature, **the feature is said to be potentially redundant** in the context of a machine learning problem.

All features having potential redundancy are candidates for rejection in the final feature subset. Only a few representative features out of a set of potentially redundant features are considered for being a part of the final feature subset. So in short, the main objective of feature selection is to remove all features which are irrelevant and take a representative subset of the features which are potentially redundant. This leads to a meaningful feature subset in the context of a specific learning task.

**The measure of feature relevance and redundancy**

**a. Measures of Feature Relevance:**In the case of supervised learning, **mutual information** is considered as a good measure of information contribution of a feature to decide the value of the class label. That is why it is a good indicator of the relevance of a feature with respect to the class variable. The higher the value of mutual information of a feature, the more relevant is that feature. Mutual information can be calculated as follows:

**Where, marginal entropy of the class,**(

**Marginal entropy of the feature** ‘x’,

And **K** = number of classes, **C** = class variable, **f** = feature set that take discrete values. In the case of unsupervised learning, there is no class variable. Hence, feature-to-class mutual information cannot be used to measure the information contribution of the features. In the case of unsupervised learning, the entropy of the set of features without one feature at a time is calculated for all features. Then the features are ranked in descending order of information gain from a feature and the top percentage (value of beta is a design parameter of the algorithm) of features are selected as relevant features. The entropy of a feature f is calculated using Shannon’s formula below:

is used only for features that take the discrete values. For continuous features, it should be replaced by discretization performed first to estimate the probabilities p(f=x).

**b. Measures of Feature Redundancy:** There are multiple measures of similarity of information contribution, the main ones are:

* Correlation-based Measures
* Distance-based Measures
* Other coefficient-based Measure

**1. Correlation Based Similarity Measure**

Correlation is a measure of linear dependency between two random variables. Pearson’s product correlation coefficient is one of the most popular and accepted measures correlation between two random variables. For two random feature variables F1 and F2 , the Pearson coefficient is defined as:

Correlation value ranges between +1 and -1. A correlation of 1 (+/-) indicates perfect correlation. In case the correlation is zero, then the features seem to have no linear relationship. Generally for all feature selection problems, a threshold value is adopted to decide whether two features have adequate similarity or not.

**2. Distance-Based Similarity Measure**

The most common distance measure is the **Euclidean distance,** which, between two features F1 and F2 are calculated as:

Where the features represent an n-dimensional dataset. Let us consider that the dataset has two features, Subjects (F1) and marks (F2) under consideration. The Euclidean distance between the two features will be calculated like this:

| **Subjects (F1)** | **Marks (F2)** | **(F1-F2)** | **(F1-F2)2** |
| --- | --- | --- | --- |
| 2 | 6 | -4 | 16 |
| 3 | 5.5 | -2.5 | 6.25 |
| 6 | 4 | 2 | 4 |
| 7 | 2.5 | 4.5 | 20.25 |
| 8 | 3 | 5 | 25 |
| 6 | 5.5 | 0.5 | 0.25 |
| 6 | 7 | -1 | 1 |
| 7 | 6 | 1 | 1 |
| 8 | 6 | 2 | 4 |
| 9 | 7 | 2 | 4 |

A more generalized form of the Euclidean distance is the **Minkowski Distance,** measured as

Minkowski distance takes the form of Euclidean distance (also called **L2 norm**) where r = 2. At r=1, it takes the form of **Manhattan** distance (also called **L1 norm**) :

**3. Other Similarity Measures**

**Jaccard index/coefficient** is used as a measure of dissimilarity between two features is complementary of Jaccard Index. For two features having binary values, Jaccard Index is measured as:

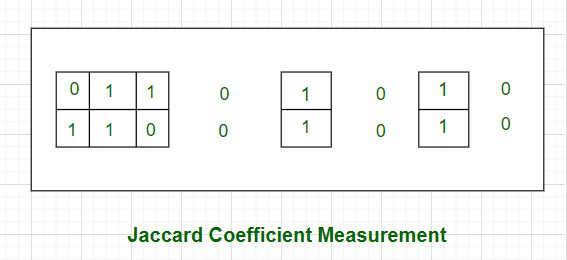
Where   = number of cases when both the feature have value 1,

=  number of cases where the feature 1 has value 0 and feature 2 has value 1,

=  the number of cases where feature 1 has value 1 and feature 2 has value 0.

Jaccard distance:

Let us take an example to understand it better. Consider two features, F1 and F2 having values (0, 1, 1, 0, 1, 0, 1, 0) and (1, 1, 0, 0, 1, 0, 0, 0).



As shown in the above picture, the cases where both the values are 0 have been left out without border- as an indication of the fact that they will be excluded in the calculation of the Jaccard coefficient.

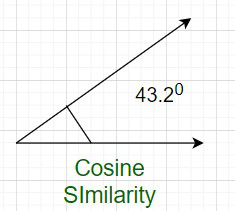
Jaccard coefficient of F1 and F2 , J =

**Therefore, Jaccard Distance between those two features is dj = (1 – 0.4) = 0.6**

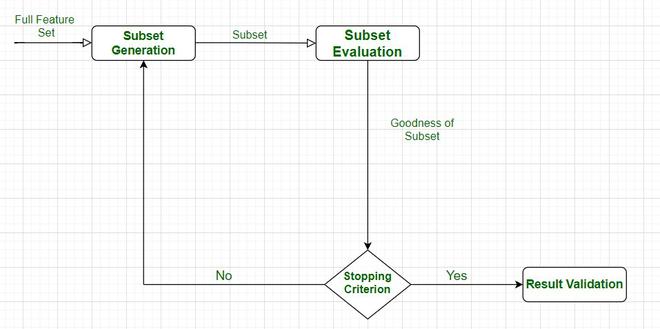
**Note:** One more measure of similarity using similarity coefficient calculation is **Cosine Similarity**. For the sake of understanding, let u stake an example of the **text classification problem.** The text needs to be first transformed into features with a word token being a feature and the number of times the word occurs in a document comes as a value in each row. There are thousands of features in such a text dataset. However, the data set is sparse in nature as only a few words do appear in a document and hence in a row of the data set. So each row has very few non-zero values. However, the non-zero values can be anything integer value as the same word may occur any number of times. Also, considering the sparsity of the dataset, the 0-0 matches need to be ignored. **Cosine similarity** which is one of the most popular measures in text classification is calculated as:

Where, x.y is the vector dot product of x and y  = and So let’s calculate the cosine similarity of x and y, where x = (2,4,0,0,2,1,3,0,0) and y = (2,1,0,0,3,2,1,0,1). In this case, dot product of x and y will be **x.y = 2\*2 + 4\*1 + 0\*0 + 0\*0 + 2\*3 + 1\*2 + 3\*1 + 0\*0 + 0\*1 = 19.**

**Cosine Similarity** measures the angle between x and y vectors. Hence, if cosine similarity has a value of 1, the angles between x and y is 0 degrees which means x and y are the same except for the magnitude. If the cosine similarity is 0, the angle between x and y is 900. Hence, they do not share any similarity. In the case of the above example, the angle comes out to be 43.20.



**Even after all these steps, there are some few more steps.**You can understand it by the following flowchart:



***Feature Selection Process***

After the successful completion of this cycle, we get the desired features, and we have finally tested them also.

The feature subset selection process involves identifying and selecting a subset of relevant features from a given dataset. It aims to improve model performance, reduce overfitting, and enhance interpretability. Here is a general outline of the feature subset selection process:

**Data Preparation:**

Clean the data: Handle missing values, outliers, and data inconsistencies.  
Encode categorical variables: Convert categorical features into numerical representations, such as one-hot encoding or label encoding.  
Normalize or standardize numerical features: Scale numerical features to a common range to avoid bias.

**Feature Ranking/Scoring:**

Select an appropriate feature scoring or ranking method based on the nature of the data and the problem you are addressing. Common scoring methods include correlation coefficient, information gain, chi-square test, mutual information, or statistical tests like t-test or ANOVA.  
Calculate the score or rank for each feature based on its relationship with the target variable.

**Feature Selection Techniques:**

Filter-based methods: Select features based on their individual scores or rankings. Set a threshold and select features above that threshold.  
Wrapper-based methods: Use a machine learning model with different feature subsets and evaluate their performance to determine the optimal subset. Techniques like forward selection, backward elimination, or recursive feature elimination fall under this category.  
Embedded methods: Incorporate feature selection as part of the model training process. Algorithms like Lasso (L1 regularization) and Ridge (L2 regularization) can automatically select relevant features during model training.

**Evaluate Subset Performance:**

Split the dataset into training and validation sets.  
Train a machine learning model using the selected subset of features.  
Evaluate the model’s performance metrics (e.g., accuracy, precision, recall, F1-score, or area under the ROC curve) on the validation set.

If the performance is not satisfactory, go back to the feature selection step and try different techniques or adjust the parameters

.  
**Iterative Refinement:**

Iterate through steps 2-4, trying different feature scoring methods, selection techniques, thresholds, or algorithms to find the optimal feature subset.

Utilize cross-validation techniques, such as k-fold cross-validation, to obtain more robust estimates of model performance and feature relevance.

Final Model Training and Testing:

Once the optimal feature subset is selected, train the final machine learning model on the entire training dataset using the selected features.  
Evaluate the model’s performance on an independent test dataset to assess its generalization ability.

**Interpretability and Validation:**

Analyze the selected features and their relationship with the target variable to gain insights into the problem domain.

Validate the selected features on new, unseen data to ensure their robustness and effectiveness.

Remember that the feature subset selection process is iterative and may require experimenting with different techniques, thresholds, and evaluation metrics to find the most suitable subset of features for your specific problem.

1. When is a function considered redundant? What criteria are used to identify features that could be redundant?

# The Problem of Redundancy in Database

**Redundancy** means having multiple copies of the same data in the database. This problem arises when a database is not normalized. Suppose a table of student details attributes is: student ID, student name, college name, college rank, and course opted.

| **Student\_ID** | **Name** | **Contact** | **College** | **Course** | **Rank** |
| --- | --- | --- | --- | --- | --- |
| 100 | Himanshu | 7300934851 | GEU | B.Tech | 1 |
| 101 | Ankit | 7900734858 | GEU | B.Tech | 1 |
| 102 | Ayush | 7300936759 | GEU | B.Tech | 1 |
| 103 | Ravi | 7300901556 | GEU | B.Tech | 1 |

It can be observed that values of attribute college name, college rank, and course are being repeated which can lead to problems. Problems caused due to redundancy are:

* [Insertion anomaly](https://www.geeksforgeeks.org/anomalies-in-relational-model/)
* [Deletion anomaly](https://www.geeksforgeeks.org/anomalies-in-relational-model/)
* [Updation anomaly](https://www.geeksforgeeks.org/anomalies-in-relational-model/)

### ****Insertion Anomaly****

If a student detail has to be inserted whose course is not being decided yet then insertion will not be possible till the time course is decided for the student.

| **Student\_ID** | **Name** | **Contact** | **College** | **Course** | **Rank** |
| --- | --- | --- | --- | --- | --- |
| 100 | Himanshu | 7300934851 | GEU |  | 1 |

This problem happens when the insertion of a data record is not possible without adding some additional unrelated data to the record.

### ****Deletion Anomaly****

If the details of students in this table are deleted then the details of the college will also get deleted which should not occur by common sense. This anomaly happens when the deletion of a data record results in losing some unrelated information that was stored as part of the record that was deleted from a table.

It is not possible to delete some information without losing some other information in the table as well.

### ****Updation Anomaly****

Suppose the rank of the college changes then changes will have to be all over the database which will be time-consuming and computationally costly.

| **Student\_ID** | **Name** | **Contact** | **College** | **Course** | **Rank** |
| --- | --- | --- | --- | --- | --- |
| 100 | Himanshu | 7300934851 | GEU | B.Tech | 1 |
| 101 | Ankit | 7900734858 | GEU | B.Tech | 1 |
| 102 | Ayush | 7300936759 | GEU | B.Tech | 1 |
| 103 | Ravi | 7300901556 | GEU | B.Tech | 1 |

All places should be updated, If updation does not occur at all places then the database will be in an inconsistent state.

Redundancy in a database occurs when the same data is stored in multiple places. Redundancy can cause various problems such as data inconsistencies, higher storage requirements, and slower data retrieval.

## Problems Caused Due to Redundancy

* **Data Inconsistency:**Redundancy can lead to data inconsistencies, where the same data is stored in multiple locations, and changes to one copy of the data are not reflected in the other copies. This can result in incorrect data being used in decision-making processes and can lead to errors and inconsistencies in the data.
* **Storage Requirements:**Redundancy increases the storage requirements of a database. If the same data is stored in multiple places, more storage space is required to store the data. This can lead to higher costs and slower data retrieval.
* **Update Anomalies:**Redundancy can lead to update anomalies, where changes made to one copy of the data are not reflected in the other copies. This can result in incorrect data being used in decision-making processes and can lead to errors and inconsistencies in the data.
* **Performance Issues:**Redundancy can also lead to performance issues, as the database must spend more time updating multiple copies of the same data. This can lead to slower data retrieval and slower overall performance of the database.
* **Security Issues:** Redundancy can also create security issues, as multiple copies of the same data can be accessed and manipulated by unauthorized users. This can lead to data breaches and compromise the [confidentiality, integrity, and availability of the data.](https://www.geeksforgeeks.org/the-cia-triad-in-cryptography/)
* **Maintenance Complexity:**Redundancy can increase the complexity of database maintenance, as multiple copies of the same data must be updated and synchronized. This can make it more difficult to troubleshoot and resolve issues and can require more time and resources to maintain the database.
* **Data Duplication:** Redundancy can lead to data duplication, where the same data is stored in multiple locations, resulting in wasted storage space and increased maintenance complexity. This can also lead to confusion and errors, as different copies of the data may have different values or be out of sync.
* **Data Integrity:** Redundancy can also compromise data integrity, as changes made to one copy of the data may not be reflected in the other copies. This can result in inconsistencies and errors and can make it difficult to ensure that the data is accurate and up-to-date.
* **Usability Issues:** Redundancy can also create usability issues, as users may have difficulty accessing the correct version of the data or may be confused by inconsistencies and errors. This can lead to frustration and decreased productivity, as users spend more time searching for the correct data or correcting errors.

To prevent redundancy in a database, normalization techniques can be used. Normalization is the process of organizing data in a database to eliminate redundancy and improve data integrity. [Normalization](https://www.geeksforgeeks.org/introduction-of-database-normalization/) involves breaking down a larger table into smaller tables and establishing relationships between them. This reduces redundancy and makes the database more efficient and reliable.

## Advantages of Redundant Data

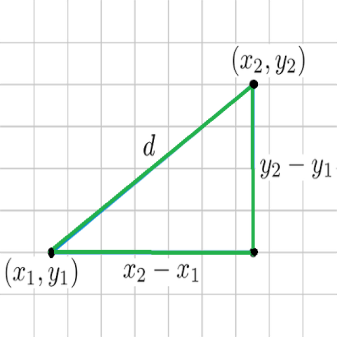
* **Enhanced Query Performance:**By eliminating the need for intricate joins, redundancy helps expedite data retrieval.
* **Offline Access:**In offline circumstances, redundant copies allow data access even in the absence of continuous connectivity.
* **Increased Availability:** Redundancy helps to increase fault tolerance, which makes data accessible even in the event of server failures.

## Disadvantages of Redundant Data

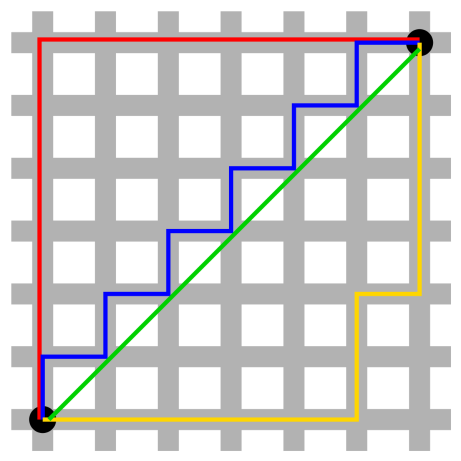
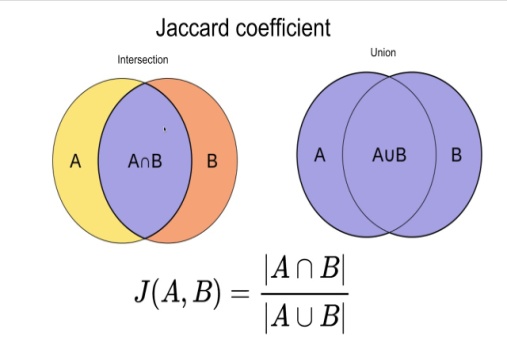
* **Increased storage requirements:**Redundant data takes up additional storage space within the database, which can increase costs and slow down performance.
* **Inconsistency:**If the same data is stored in multiple places within the database, there is a risk that updates or changes made to one copy of the data may not be reflected in other copies, leading to inconsistency and potentially incorrect results.
* **Difficulty in maintenance:**With redundant data, it becomes more difficult to maintain the accuracy and consistency of the data. It requires more effort and resources to ensure that all copies of the data are updated correctly.
* **Increased risk of errors:**When data is redundant, there is a greater risk of errors in the [database](https://www.geeksforgeeks.org/what-is-database/). For example, if the same data is stored in multiple tables, there is a risk of inconsistencies between the tables.
* **Reduced flexibility:**Redundancy can reduce the flexibility of the database. For example, if a change needs to be made to a particular piece of data, it may need to be updated in multiple places, which can be time-consuming and error-prone.

1. What are the various distance measurements used to determine feature similarity?

**Measures of Distance in Data Mining**

**Clustering** consists of grouping certain objects that are similar to each other, it can be used to decide if two items are similar or dissimilar in their properties. In a [Data Mining](https://www.geeksforgeeks.org/data-mining/) sense, the similarity measure is a distance with dimensions describing object features. That means if the distance among two data points is **small** then there is a **high** degree of similarity among the objects and vice versa. The similarity is **subjective** and depends heavily on the context and application. For example, similarity among vegetables can be determined from their taste, size, colour etc. Most clustering approaches use distance measures to assess the similarities or differences between a pair of objects, the most popular distance measures used are: **1. Euclidean Distance:** Euclidean distance is considered the traditional metric for problems with geometry. It can be simply explained as the **ordinary distance** between two points. It is one of the most used algorithms in the cluster analysis. One of the algorithms that use this formula would be **K-mean**. Mathematically it computes the **root of squared differences** between the coordinates between two objects. 

**Figure –** Euclidean Distance

**2. Manhattan Distance:** This determines the absolute difference among the pair of the coordinates. Suppose we have two points P and Q to determine the distance between these points we simply have to calculate the perpendicular distance of the points from X-Axis and Y-Axis. In a plane with P at coordinate (x1, y1) and Q at (x2, y2). Manhattan distance between P and Q = |x1 – x2| + |y1 – y2|Here the total distance of the **Red** line gives the Manhattan distance between both the points. **3. Jaccard Index:** The Jaccard distance measures the similarity of the two data set items as the **intersection** of those items divided by the **union** of the data items.

**Figure –** Jaccard Index

**4. Minkowski distance:** It is the **generalized** form of the Euclidean and Manhattan Distance Measure. In an**N-dimensional space**, a point is represented as,

(x1, x2, ..., xN)

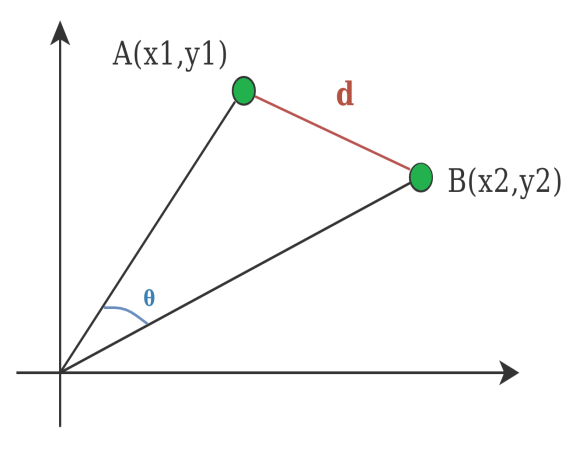
Consider two points P1 and P2:

**P1:** (X1, X2, ..., XN)

**P2:** (Y1, Y2, ..., YN)

Then, the Minkowski distance between P1 and P2 is given as:

* When **p = 2**, Minkowski distance is same as the **Euclidean** distance.
* When**p = 1**, Minkowski distance is same as the **Manhattan** distance.

**5. Cosine Index:** Cosine distance measure for clustering determines the **cosine** of the angle between two vectors given by the following formula. Here (**theta**) gives the angle between two vectors and A, B are n-dimensional vectors.

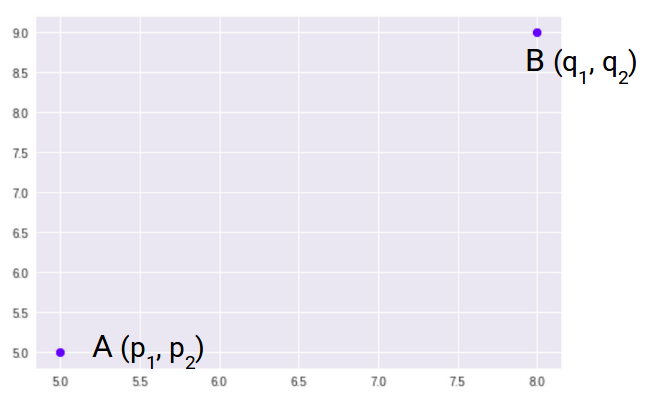
**Figure –** Cosine Distance

1. State difference between Euclidean and Manhattan distances?

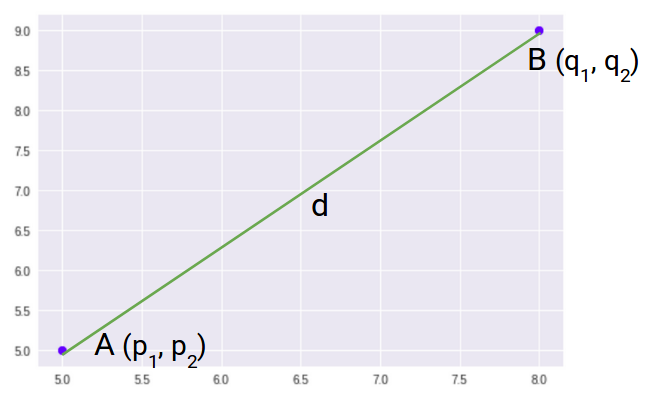
## 1. Euclidean Distance

Euclidean Distance represents the shortest distance between two points.

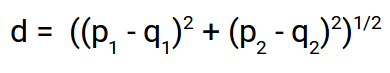
Most machine learning algorithms including K-Means use this distance metric to measure the similarity between observations. Let’s say we have two points as shown below:



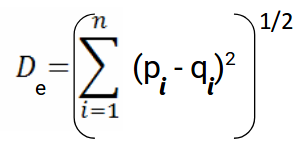
So, the Euclidean Distance between these two points A and B will be:



Here’s the formula for Euclidean Distance:



We use this formula when we are dealing with 2 dimensions. We can generalize this for an n-dimensional space as:



Where,

* n = number of dimensions
* pi, qi = data points

Let’s code Euclidean Distance in [Python 2](https://courses.analyticsvidhya.com/courses/introduction-to-data-science?utm_source=blog&utm_medium=4-types-of-distance-metrics-in-machine-learning). This will give you a better understanding of how this distance metric works.

We will first import the required libraries. I will be using the SciPy library that contains pre-written codes for most of the distance functions used in Python:

|  | **# importing the library** |
| --- | --- |
|  | from scipy.spatial import distance |
|  |  |
|  | # defining the points |
|  | point\_1 = (1, 2, 3) |
|  | point\_2 = (4, 5, 6) |
|  | point\_1, point\_2 |

[view raw](https://gist.github.com/PulkitS01/fadcf66d95184e98946483172cb03080/raw/09139fca133bc790c249374822a2170b79fc3acb/library_and_dataset.py)[library\_and\_dataset.py](https://gist.github.com/PulkitS01/fadcf66d95184e98946483172cb03080#file-library_and_dataset-py) hosted with by [GitHub 1](https://github.com/)

[dataset for distance functions in machine learning](https://cdn.analyticsvidhya.com/wp-content/uploads/2020/02/Screenshot-from-2020-02-18-12-33-34.png)

These are the two sample points which we will be using to calculate the different distance functions. Let’s now calculate the Euclidean Distance between these two points:

|  | **# computing the euclidean distance** |
| --- | --- |
|  | euclidean\_distance = distance.euclidean(point\_1, point\_2) |
|  | print(‘Euclidean Distance b/w’, point\_1, ‘and’, point\_2, 'is: ', euclidean\_distance) |

[view raw](https://gist.github.com/PulkitS01/fadcf66d95184e98946483172cb03080/raw/09139fca133bc790c249374822a2170b79fc3acb/euclidean_distance.py)[euclidean\_distance.py 1](https://gist.github.com/PulkitS01/fadcf66d95184e98946483172cb03080#file-euclidean_distance-py) hosted with by [GitHub 1](https://github.com/)

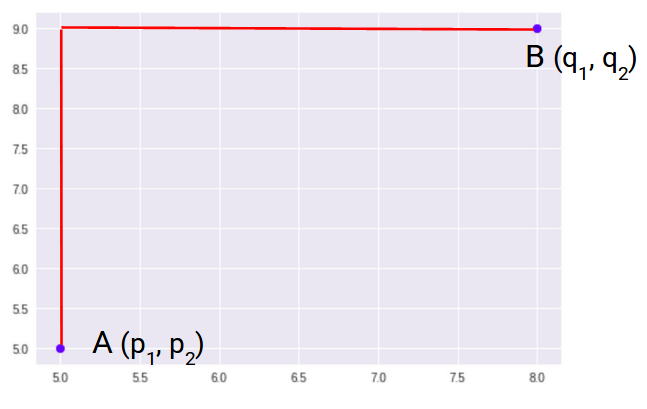
[[euclidean distance in python](https://cdn.analyticsvidhya.com/wp-content/uploads/2020/02/Screenshot-from-2020-02-18-12-35-24.png) 2](https://cdn.analyticsvidhya.com/wp-content/uploads/2020/02/Screenshot-from-2020-02-18-12-35-24.png)

This is how we can calculate the Euclidean Distance between two points in Python. Let’s now understand the second distance metric, Manhattan Distance.

## 2. Manhattan Distance

Manhattan Distance is the sum of absolute differences between points across all the dimensions.

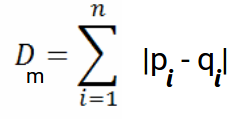
We can represent Manhattan Distance as:



Since the above representation is 2 dimensional, to calculate Manhattan Distance, we will take the sum of absolute distances in both the x and y directions. So, the Manhattan distance in a 2-dimensional space is given as:

manhattan distance formula

And the generalized formula for an n-dimensional space is given as:



Where,

* n = number of dimensions
* pi, qi = data points

Now, we will calculate the Manhattan Distance between the two points:

|  | **# computing the manhattan distance** |
| --- | --- |
|  | manhattan\_distance = distance.cityblock(point\_1, point\_2) |
|  | print(‘Manhattan Distance b/w’, point\_1, ‘and’, point\_2, 'is: ', manhattan\_distance) |

[view raw](https://gist.github.com/PulkitS01/fadcf66d95184e98946483172cb03080/raw/09139fca133bc790c249374822a2170b79fc3acb/manhattan_distance.py)[manhattan\_distance.py 2](https://gist.github.com/PulkitS01/fadcf66d95184e98946483172cb03080#file-manhattan_distance-py) hosted with by [GitHub 1](https://github.com/)

[[manhattan distance in python](https://cdn.analyticsvidhya.com/wp-content/uploads/2020/02/Screenshot-from-2020-02-18-12-38-12.png) 1](https://cdn.analyticsvidhya.com/wp-content/uploads/2020/02/Screenshot-from-2020-02-18-12-38-12.png)

Note that **Manhattan Distance is also known as city block distance.** SciPy has a function called cityblock that returns the Manhattan Distance between two points.

Let’s now look at the next distance metric – Minkowski Distance.

1. Distinguish between feature transformation and feature selection.

## ****Difference Feature Selection and Feature Extraction Methods****

Feature selection and feature extraction methods have their advantages and disadvantages, depending on the nature of the data and the task at hand.

|  | **Feature Selection** | **Feature Extraction** |
| --- | --- | --- |
| 1. | Selects a subset of relevant features from the original set of features. | Extracts a new set of features that are more informative and compact. |
| 2. | Reduces the dimensionality of the feature space and simplifies the model. | Captures the essential information from the original features and represents it in a lower-dimensional feature space. |
| 3. | Can be categorized into filter, wrapper, and embedded methods. | Can be categorized into linear and nonlinear methods. |
| 4. | Requires domain knowledge and feature engineering. | Can be applied to raw data without feature engineering. |
| 5. | Can improve the model’s interpretability and reduce overfitting. | Can improve the model performance and handle nonlinear relationships. |
| 6. | May lose some information and introduce bias if the wrong features are selected. | May introduce some noise and redundancy if the extracted features are not informative. |

11. Make brief notes on any two of the following:

1.SVD (Standard Variable Diameter)

# Singular Value Decomposition (SVD)

The Singular Value Decomposition (SVD) of a matrix is a factorization of that matrix into three matrices. It has some interesting algebraic properties and conveys important geometrical and theoretical insights about linear transformations. It also has some important applications in data science. In this article, I will try to explain the mathematical intuition behind SVD and its geometrical meaning.

**Mathematics behind SVD:**

The SVD of  mxn matrix A is given by the formula

where:

* U:  mxm matrix of the orthonormal eigenvectors of .
* VT: transpose of a nxn matrix containing the orthonormal eigenvectors of.
* : diagonal matrix with r elements equal to the root of the positive eigenvalues of AAᵀ or Aᵀ A (both matrics have the same positive eigenvalues anyway).

#### Examples

* Find the SVD for the matrix A =
* To calculate the SVD, First, we need to compute the singular values by finding eigenvalues of AA^{T}.
* The characteristic equation for the above matrix is:

so our singular values are:

* Now we find the right singular vectors i.e orthonormal set of eigenvectors of ATA. The eigenvalues of ATA are 25, 9, and 0, and since ATA is symmetric we know that the eigenvectors will be orthogonal.

For

which can be row-reduces to :

A unit vector in the direction of it is:

Similarly, for \lambda = 9, the eigenvector is:

For the 3rd eigenvector, we could use the property that it is perpendicular to v1 and v2 such that:

Solving the above equation to generate the third eigenvector

Now, we calculate U using the formula u\_i = \frac{1}{\sigma} A v\_i and this gives U =. Hence, our final SVD equation becomes:

### Applications

* **Calculation of Pseudo-inverse:**Pseudo inverse or Moore-Penrose inverse is the generalization of the matrix inverse that may not be invertible (such as low-rank matrices). If the matrix is invertible then its inverse will be equal to Pseudo inverse but pseudo inverse exists for the matrix that is not invertible. It is denoted by A+.

The above equation gives the pseudo-inverse.

**Solving a set of Homogeneous Linear Equation (Mx =b):**if b=0,  calculate SVD and take any column of VT associated with a singular value (in W) equal to 0.

From the Pseudo-inverse, we know that

Hence,

* **Rank, Range, and Null space:**
  + The rank of matrix M can be calculated from SVD by the number of nonzero singular values.
  + The range of matrix M is The left singular vectors of U corresponding to the non-zero singular values.
  + The null space of matrix M is The right singular vectors of V corresponding to the zeroed singular values.
* **Curve Fitting Problem:**Singular value decomposition can be used to minimize the least square error. It uses the pseudo inverse to approximate it.
* Besides the above application, singular value decomposition and pseudo-inverse can also be used in Digital signal processing and image processing

### Implementation:

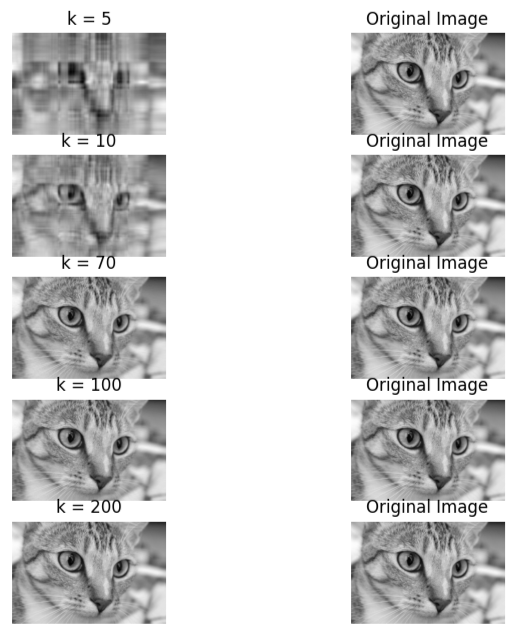
In this code, we will try to calculate the Singular value decomposition using Numpy and Scipy.  We will be calculating SVD, and also performing pseudo-inverse. In the end, we can apply SVD for compressing the image

* Python3

|  |
| --- |
| # Imports   from skimage.color import rgb2gray  from skimage import data  import matplotlib.pyplot as plt  import numpy as np  from scipy.linalg import svd   """  Singular Value Decomposition  """  # define a matrix  X = np.array([[3, 3, 2], [2, 3, -2]])  print(X)  # perform SVD  U, singular, V\_transpose = svd(X)  # print different components  print("U: ", U)  print("Singular array", singular)  print("V^{T}", V\_transpose)  """  Calculate Pseudo inverse  """  # inverse of singular matrix is just the reciprocal of each element  singular\_inv = 1.0 / singular  # create m x n matrix of zeroes and put singular values in it  s\_inv = np.zeros(X.shape)  s\_inv[0][0] = singular\_inv[0]  s\_inv[1][1] = singular\_inv[1]  # calculate pseudoinverse  M = np.dot(np.dot(V\_transpose.T, s\_inv.T), U.T)  print(M)   """  SVD on image compression  """  cat = data.chelsea()  plt.imshow(cat)  # convert to grayscale  gray\_cat = rgb2gray(cat)  # calculate the SVD and plot the image  U, S, V\_T = svd(gray\_cat, full\_matrices=False)  S = np.diag(S)  fig, ax = plt.subplots(5, 2, figsize=(8, 20))  curr\_fig = 0  for r in [5, 10, 70, 100, 200]:      cat\_approx = U[:, :r] @ S[0:r, :r] @ V\_T[:r, :]      ax[curr\_fig][0].imshow(cat\_approx, cmap='gray')      ax[curr\_fig][0].set\_title("k = "+str(r))      ax[curr\_fig, 0].axis('off')      ax[curr\_fig][1].set\_title("Original Image")      ax[curr\_fig][1].imshow(gray\_cat, cmap='gray')      ax[curr\_fig, 1].axis('off')      curr\_fig += 1  plt.show() |

**Output:**

[[ 3 3 2]  
 [ 2 3 -2]]  
---------------------------  
U: [[-0.7815437 -0.6238505]  
 [-0.6238505 0.7815437]]  
---------------------------  
Singular array [5.54801894 2.86696457]  
---------------------------  
V^{T} [[-0.64749817 -0.7599438 -0.05684667]  
 [-0.10759258 0.16501062 -0.9804057 ]  
 [-0.75443354 0.62869461 0.18860838]]  
--------------------------  
# Inverse   
array([[ 0.11462451, 0.04347826],  
 [ 0.07114625, 0.13043478],  
 [ 0.22134387, -0.26086957]])  
---------------------------



*Original vs SVD k-image*

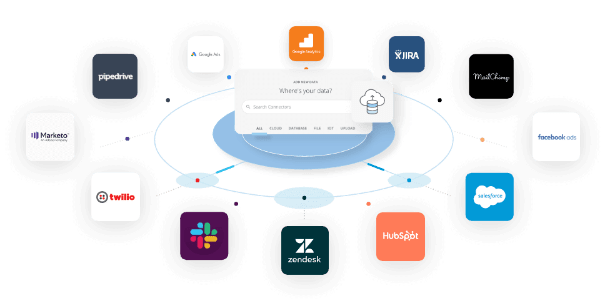
2. Collection of features using a hybrid approach

# What is hybrid machine learning?

[Forrester reports](https://www.intel.com/content/www/us/en/analytics/advanced-analytics-and-ai-infographic.html) that, “98 percent of organizations said that analytics are important to driving business priorities, yet fewer than 40 percent of workloads are leveraging advanced analytics or [artificial intelligence](https://www.domo.com/glossary/what-is-artificial-intelligence/).”

Automation and AI become even more important when you consider the exponential rate at which companies are producing data in need of continuous analysis. AI powered by [machine learning (ML)](https://www.domo.com/glossary/what-is-machine-learning/) will be critical to managing future insights for data scientists. ML uses algorithms and statistical models to identify patterns, mine data, and apply labels across different datasets. These models learn from the data as they go and will help data scientists develop increasingly sophisticated and accurate predictions.

With the right [BI and ML tools](https://www.domo.com/data-science) in place, companies will be able to extract even greater insights from their data.



### Defining hybrid machine learning

Most learning algorithms used in ML are really good at completing one task or working with one dataset. While helpful and infinitely better than doing it manually, these algorithms won’t help you realize the full potential of AI across all of your data.

That’s where hybrid machine learning (HML) comes in. Multiple simple algorithms work together to complement and augment each other. Together they can solve problems that alone they were not designed to solve.

Within HML there are various types of techniques that interact with the data in different ways. Which technique you use depends on the problems you’re trying to solve, the technical expertise available, and the tools you’re using.

Here are some types of hybrid machine learning.

#### ****Semi-supervised learning****

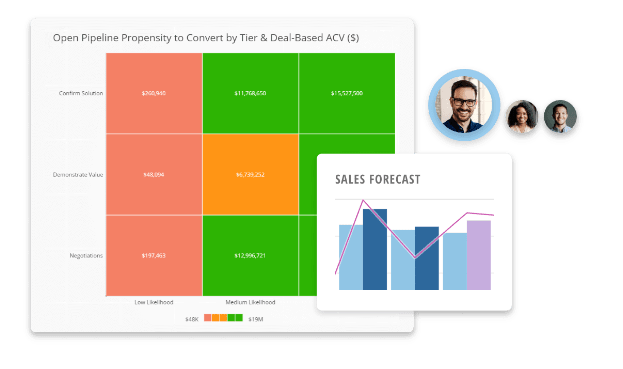
In semi-supervised learning, you provide the algorithm with a small set of labelled data. Then, you give it a much larger set of unlabeled data and put it to work. This type of algorithm is helpful when you need (or have) to start with a smaller batch of data upfront. It learns from all the data, not just the labelled data, and helps you organize it.

This form of HML is especially helpful with data that changes over time. For example, use it to track things like the cost of supplies. As costs change, it will impact production and forecasts. You can use this method of HML in your [inventory and supply chain management](https://www.domo.com/learn/guide/use-data-to-optimize-manufacturing-supply-chain/) to forecast future costs.

Or, it can be useful to track brand sentiment for customer retention. Track how current customers are engaging with or discussing your brand on social media, and use it to develop targeted mitigation strategies when customers fall below a designated threshold.

Often, you can use semi-supervised learning in tandem with unsupervised and supervised learning methods. These additional models can help with grouping and training on unlabeled data.

In business, it is common to be in situations without a lot of labeled data. Semi-supervised learning reduces the upfront costs and burdens with organizing that data and allows you to work with a dynamic dataset and start working much faster.



#### ****Self-supervised learning****

A self-supervised learning model combines unsupervised and supervised learning problems, then applies a supervised learning algorithm. You can create the model for the algorithm to follow, and it begins applying that to unlabeled data.

This type of learning is commonly used on unlabeled images and defines actions that can be taken on those images—like rotating them, identifying color or grayscale, or distinguishing between real and fake photos.

This HML method is trained using supervised learning and applied to problems that are generally solved with unsupervised learning. It’s helpful for analyzing things like photos, that will have a lot of context about the image that may not be initially machine readable. However, it is important to only use it in use cases where unsupervised learning is helpful. Identifying features about the image—like color, size, or orientation—can be performed with unsupervised models. However, it won’t be effective at identifying data contained within the images, such as what the picture is about.

#### ****Multi-instance learning****

Multi-instance learning is a method where you are labeling groups or collections of data, rather than the individual members of the group. This is a helpful method when you’re working with large sets of similar data and have a lot of duplicates.

This method uses supervised learning models to identify labels for groups of data. You train the models to recognize attributes of a few pieces of data within a group, and then it predicts labels for future groups based on attributes of some of the data within the new groups.

### Tools to support hybrid machine learning

The point of incorporating ML into your data science and analytics processes is to allow you to begin looking forward with your data. Rather than relying on datasets that have been cleaned and organized, you’ll be able to quickly group and label data in real-time for the most accurate analysis and forecasts.

When considering how to manage this data, you’ll need a few key features in your [business intelligence tools](https://www.domo.com/why-domo) to support this type of advanced analysis. Your tool will need to support:

**Integration from all your data sources**  
You’ll need one place to manage your data and train your ML models. Find a tool that will allow for[easy integration of all your data sources](https://www.domo.com/data-integration).

**Real-time analysis**  
Many of the HML models mentioned here function best as they’re learning from new data. Find tools that will support [real-time ingestion and analysis](https://www.domo.com/platform/leverage-the-cloud), and then will push that data out to workers who can use it to improve performance right then.

**Automatic decisions**  
Find a tool that will support [automatic decisions](https://www.domo.com/glossary/what-is-automated-machine-learning/) for your team, with [alerts and notifications](https://www.domo.com/business-intelligence/features/alerts) for when your data passes specific thresholds.

No matter your industry, your data will continue to [play an increasingly important role in how you do business](https://www.domo.com/learn/webinar/level-up-your-analytics-strategy-with-augmented-bi/). Incorporating hybrid-machine learning techniques will be one of the best ways you’ll be able to create tools that will allow you to get value from your data now and as your business grows.

3. The width of the silhouette

**Silhouette Algorithm to determine the optimal value of k**

One of the fundamental steps of an unsupervised learning algorithm is to determine the number of clusters into which the data may be divided. The silhouette algorithm is one of the many algorithms to determine the optimal number of clusters for an unsupervised learning technique. In the Silhouette algorithm, we assume that the data has already been clustered into k clusters by a clustering technique(Typically [K-Means Clustering technique](https://www.geeksforgeeks.org/k-means-clustering-introduction/)). Then for each data point, we define the following:- C(i) -The cluster assigned to the ith data point |C(i)| – The number of data points in the cluster assigned to the ith data point a(i) – It gives a measure of how well assigned the ith data point is to it’s clusterb(i) – It is defined as the average dissimilarity to the closest cluster which is not it’s clusterThe silhouette coefficient s(i) is given by:- We determine the average silhouette for each value of k and for the value of k which has the **maximum value of s(i)** is considered the optimal number of clusters for the unsupervised learning algorithm. Let us consider the following data:-

| **S.No** | **X1** | **X2** |
| --- | --- | --- |
| 1. | -7.36 | 6.37 |
| 2. | 3.08 | -6.78 |
| 3. | 5.03 | -8.31 |
| 4. | -1.93 | -0.92 |
| 5. | -8.86 | 6.60 |

We now iterate the values of k from 2 to 5. We assume that no practical data exists for which all the data points can be optimally clustered into 1 cluster. We construct the following tables for each value of k:- **k = 2**

| **S.No** | **a(i)** | **b(i)** | **s(i)** |
| --- | --- | --- | --- |
| 1. | 5.31 | 14.1 | 0.62 |
| 2. | 2.47 | 13.15 | 0.81 |
| 3. | 2.47 | 14.97 | 0.84 |
| 4. | 9.66 | 8.93 | -0.076 |
| 5. | 5.88 | 19.16 | 0.69 |

**Average value of s(i) = 0.58** **k = 3**

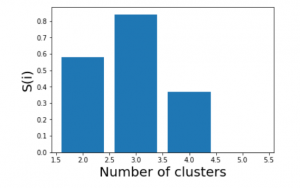
| **S.No** | **a(i)** | **b(i)** | **s(i)** |
| --- | --- | --- | --- |
| 1. | 1.52 | 9.09 | 0.83 |
| 2. | 2.47 | 7.71 | 0.68 |
| 3. | 2.47 | 10.15 | 0.76 |
| 4. | 0 | 7.71 | 1 |
| 5. | 1.52 | 17.93 | 0.92 |

**Average value of s(i) = 0.84** **k = 4**

| **S.No** | **a(i)** | **b(i)** | **s(i)** |
| --- | --- | --- | --- |
| 1. | 1.52 | 9.09 | 0.83 |
| 2. | infinite | 2.47 | 0 |
| 3. | infinite | 2.47 | 0 |
| 4. | infinite | 7.71 | 0 |
| 5. | 1.52 | 10.23 | 0.85 |

**Average value of s(i) = 0.37** **k = 5**

| **S.No** | **a(i)** | **b(i)** | **s(i)** |
| --- | --- | --- | --- |
| 1. | infinite | 1.52 | 0 |
| 2. | infinite | 2.47 | 0 |
| 3. | infinite | 2.47 | 0 |
| 4. | infinite | 7.71 | 0 |
| 5. | infinite | 1.52 | 0 |

**Average value of s(i) = 0**We see that the highest value of s(i) exists for k = 3. Therefore we conclude that the optimal number of clusters for the given data is 3.

1. Receiver operating characteristic curve

# AUC ROC Curve in Machine Learning

One important aspect of [Machine Learning](https://www.geeksforgeeks.org/machine-learning/) is model evaluation. You need to have some mechanism to evaluate your model. This is where these performance metrics come into the picture they give us a sense of how good a model is. If you are familiar with some of the basics of [Machine Learning](https://www.geeksforgeeks.org/machine-learning/) then you must have come across some of these metrics, like accuracy, precision, recall, auc-roc, etc., which are generally used for classification tasks. In this article, we will explore in depth one such metric, which is the AUC-ROC curve.

**Table of Content**

* [What is the AUC-ROC curve?](https://www.geeksforgeeks.org/auc-roc-curve/#what-is-the-aucroc-curve)
* [Key terms used in AUC and ROC Curve](https://www.geeksforgeeks.org/auc-roc-curve/#key-terms-used-in-auc-and-roc-curve)
* [Relationship between Sensitivity, Specificity, FPR, and Threshold.](https://www.geeksforgeeks.org/auc-roc-curve/#relationship-between-sensitivity-specificity-fpr-and-threshold)
* [How does AUC-ROC work?](https://www.geeksforgeeks.org/auc-roc-curve/#how-does-aucroc-work)
* [When should we use the AUC-ROC evaluation metric?](https://www.geeksforgeeks.org/auc-roc-curve/#when-should-we-use-the-aucroc-evaluation-metric)
* [Speculating the performance of the model](https://www.geeksforgeeks.org/auc-roc-curve/#speculating-the-performance-of-the-model)
* [Understanding the AUC-ROC Curve](https://www.geeksforgeeks.org/auc-roc-curve/#understanding-the-aucroc-curve)
* [Implementation using two different models](https://www.geeksforgeeks.org/auc-roc-curve/#implementation-using-two-different-models)
* [How to use ROC-AUC for a multi-class model?](https://www.geeksforgeeks.org/auc-roc-curve/#how-to-use-rocauc-for-a-multiclass-model)
* [FAQs for AUC ROC Curve in Machine Learning](https://www.geeksforgeeks.org/auc-roc-curve/#faqs-for-auc-roc-curve-in-machine-learning)

## What is the AUC-ROC curve?

The AUC-ROC curve, or Area Under the Receiver Operating Characteristic curve, is a graphical representation of the performance of a binary classification model at various classification thresholds. It is commonly used in machine learning to assess the ability of a model to distinguish between two classes, typically the positive class (e.g., presence of a disease) and the negative class (e.g., absence of a disease).

Let’s first understand the meaning of the two terms **ROC**and**AUC**.

* **ROC**: Receiver Operating Characteristics
* **AUC**: Area Under Curve

### ****Receiver Operating Characteristics (ROC) Curve****

ROC stands for Receiver Operating Characteristics, and the ROC curve is the graphical representation of the effectiveness of the binary classification model. It plots the true positive rate (TPR) vs the false positive rate (FPR) at different classification thresholds.

### Area Under Curve****(AUC) Curve:****

AUC stands for the Area Under the Curve, and the AUC curve represents the area under the ROC curve. It measures the overall performance of the binary classification model. As both TPR and FPR range between 0 to 1, So, the area will always lie between 0 and 1, and A greater value of AUC denotes better model performance. Our main goal is to maximize this area in order to have the highest TPR and lowest FPR at the given threshold. The AUC measures the probability that the model will assign a randomly chosen positive instance a higher predicted probability compared to a randomly chosen negative instance.

 It represents the [probability](https://www.geeksforgeeks.org/probability-gq/) with which our model can distinguish between the two classes present in our target.

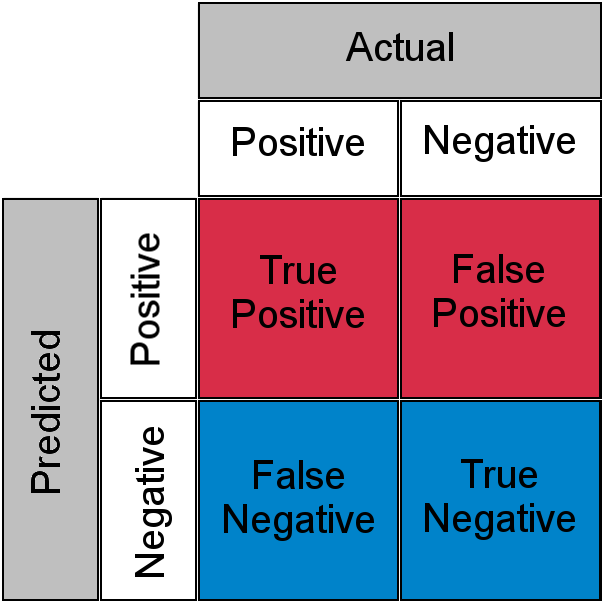
Key terms used in AUC and ROC Curve

### 1. TPR and FPR

This is the most common definition that you would have encountered when you would Google AUC-ROC. Basically, the ROC curve is a graph that shows the performance of a classification model at all possible thresholds( threshold is a particular value beyond which you say a point belongs to a particular class). The curve is plotted between two parameters

* **TPR** – True Positive Rate
* **FPR** – False Positive Rate

Before understanding, TPR and FPR let us quickly look at the [confusion matrix](https://www.geeksforgeeks.org/confusion-matrix-machine-learning/).



*Confusion Matrix for a Classification Task*

* **True Positive**: Actual Positive and Predicted as Positive
* **True Negative**: Actual Negative and Predicted as Negative
* **False Positive(Type I Error)**: Actual Negative but predicted as Positive
* **False Negative(Type II Error)**: Actual Positive but predicted as Negative

In simple terms, you can call False Positive a **false alarm** and False Negative a **miss**. Now let us look at what TPR and FPR are.

### 2. Sensitivity / True Positive Rate / Recall

Basically, TPR/Recall/Sensitivity is the ratio of positive examples that are correctly identified.  It represents the ability of the model to correctly identify positive instances and is calculated as follows:

Sensitivity/Recall/TPR measures the proportion of actual positive instances that are correctly identified by the model as positive.

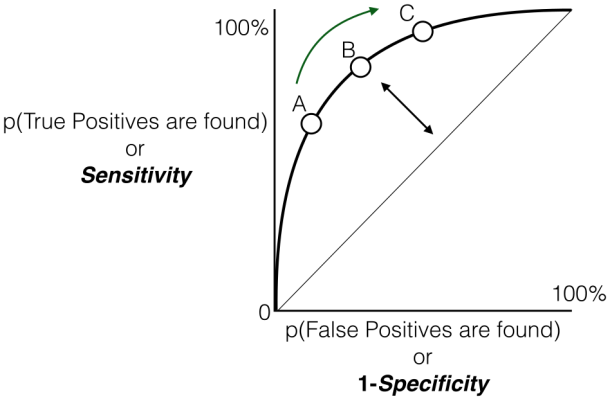
### 3. False Positive Rate

FPR is the ratio of negative examples that are incorrectly classified.

### 4. Specificity

Specificity measures the proportion of actual negative instances that are correctly identified by the model as negative. It represents the ability of the model to correctly identify negative instances

And as said earlier ROC is nothing but the plot between TPR and FPR across all possible thresholds and AUC is the entire area beneath this ROC curve.



*Sensitivity versus False Positive Rate plot*

## Relationship between Sensitivity, Specificity, FPR, and Threshold****.****

### ****Sensitivity and Specificity:****

* **Inverse Relationship:**  sensitivity and specificity have an inverse relationship. When one increases, the other tends to decrease. This reflects the inherent trade-off between true positive and true negative rates.
* **Tuning via Threshold:** By adjusting the threshold value, we can control the balance between sensitivity and specificity. Lower thresholds lead to higher sensitivity (more true positives) at the expense of specificity (more false positives). Conversely, raising the threshold boosts specificity (fewer false positives) but sacrifices sensitivity (more false negatives).

### ****Threshold and False Positive Rate (FPR):****

* **FPR and Specificity Connection:** False Positive Rate (FPR) is simply the complement of specificity (FPR = 1 – specificity). This signifies the direct relationship between them: higher specificity translates to lower FPR, and vice versa.
* **FPR Changes with TPR:** Similarly, as you observed, the True Positive Rate (TPR) and FPR are also linked. An increase in TPR (more true positives) generally leads to a rise in FPR (more false positives). Conversely, a drop in TPR (fewer true positives) results in a decline in FPR (fewer false positives)

## How does AUC-ROC work?

We looked at the geometric interpretation, but I guess it is still not enough in developing the intuition behind what 0.75 AUC actually means, now let us look at AUC-ROC from a probabilistic point of view. Let us first talk about what AUC does and later we will build our understanding on top of this

***AUC measures how well a model is able to distinguish between classes.***

An AUC of 0.75 would actually mean that let’s say we take two data points belonging to separate classes then there is a 75% chance the model would be able to segregate them or rank order them correctly i.e positive point has a higher prediction probability than the negative class. (assuming a higher prediction probability means the point would ideally belong to the positive class). Here is a small example to make things more clear.

| **Index** | **Class** | **Probability** |
| --- | --- | --- |
| **P1** | 1 | 0.95 |
| **P2** | 1 | 0.90 |
| **P3** | 0 | 0.85 |
| **P4** | 0 | 0.81 |
| **P5** | 1 | 0.78 |
| **P6** | 0 | 0.70 |

Here we have 6 points where P1, P2, and P5 belong to class 1 and P3, P4, and P6 belong to class 0 and we’re corresponding predicted probabilities in the Probability column, as we said if we take two points belonging to separate classes then what is the probability that model rank orders them correctly.

We will take all possible pairs such that one point belongs to class 1 and the other belongs to class 0, we will have a total of 9 such pairs below are all of these 9 possible pairs.

| **Pair** | **isCorrect** |
| --- | --- |
| **(P1,P3)** | Yes |
| **(P1,P4)** | Yes |
| **(P1,P6)** | Yes |
| **(P2,P3)** | Yes |
| **(P2,P4)** | Yes |
| **(P2,P6)** | Yes |
| **(P3,P5)** | No |
| **(P4,P5)** | No |
| **(P5,P6)** | Yes |

Here column is Correct tells if the mentioned pair is correctly rank-ordered based on the predicted probability i.e class 1 point has a higher probability than class 0 point, in 7 out of these 9 possible pairs class 1 is ranked higher than class 0, or we can say that there is a 77% chance that if you pick a pair of points belonging to separate classes the model would be able to distinguish them correctly. Now, I think you might have a bit of intuition behind this AUC number, just to clear up any further doubts let’s validate it using Scikit learns AUC-ROC implementation.

* Python3

|  |
| --- |
| import numpy as np  from sklearn .metrics import roc\_auc\_score    y\_true = [1, 1, 0, 0, 1, 0]  y\_pred = [0.95, 0.90, 0.85, 0.81, 0.78, 0.70]  auc = np.round(roc\_auc\_score(y\_true, y\_pred), 3)  print("Auc for our sample data is {}".format(auc)) |

**Output:**

AUC for our sample data is 0.778

## When should we use the AUC-ROC evaluation metric?

There are some areas where using ROC-AUC might not be ideal. In cases where the dataset is highly imbalanced, **the ROC curve can give an overly optimistic assessment of the model’s performance**. This optimism bias arises because the ROC curve’s false positive rate (FPR) can become very small when the number of actual negatives is large.

Looking at the FPR formula,

**we observe**,

* The Negative class is in the majority, the denominator of FPR is dominated by True Negatives, because of which FPR becomes less sensitive to changes in predictions related to the minority class (positive class).
* ROC curves may be appropriate when the cost of False Positives and False Negatives is balanced and the dataset is not heavily imbalanced.

In those case, [Precision-Recall Curves](https://www.geeksforgeeks.org/precision-recall-curve-ml/)can be used which provide an alternative evaluation metric that is more suitable for imbalanced datasets, focusing on the performance of the classifier with respect to the positive (minority) class.

## Speculating the performance of the model

* A high AUC (close to 1) indicates excellent discriminative power. This means the model is effective in distinguishing between the two classes, and its predictions are reliable.
* A low AUC (close to 0) suggests poor performance. In this case, the model struggles to differentiate between the positive and negative classes, and its predictions may not be trustworthy.
* AUC around 0.5 implies that the model is essentially making random guesses. It shows no ability to separate the classes, indicating that the model is not learning any meaningful patterns from the data.

## Understanding the AUC-ROC Curve

In an ROC curve, the x-axis typically represents the False Positive Rate (FPR), and the y-axis represents the True Positive Rate (TPR), also known as Sensitivity or Recall. So, a higher x-axis value (towards the right) on the ROC curve does indicate a higher False Positive Rate, and a higher y-axis value (towards the top) indicates a higher True Positive Rate.The ROC curve is a graphical representation of the trade-off between true positive rate and false positive rate at various thresholds. It shows the performance of a classification model at different classification thresholds. The AUC (Area Under the Curve) is a summary measure of the ROC curve performance.The choice of the threshold depends on the specific requirements of the problem you’re trying to solve and the trade-off between false positives and false negatives that is acceptable in your context.

* If you want to prioritize reducing false positives (minimizing the chances of labeling something as positive when it’s not), you might choose a threshold that results in a lower false positive rate.
* If you want to prioritize increasing true positives (capturing as many actual positives as possible), you might choose a threshold that results in a higher true positive rate.

Let’s consider an example to illustrate how ROC curves are generated for different [thresholds](https://www.geeksforgeeks.org/decision-threshold-in-machine-learning/)and how a particular threshold corresponds to a confusion matrix. Suppose we have a [binary classification problem](https://www.geeksforgeeks.org/basic-concept-classification-data-mining/) with a model predicting whether an email is spam (positive) or not spam (negative).

Let us consider the hypothetical data,

True Labels: [1, 0, 1, 0, 1, 1, 0, 0, 1, 0]

Predicted Probabilities: [0.8, 0.3, 0.6, 0.2, 0.7, 0.9, 0.4, 0.1, 0.75, 0.55]

### ****Case 1: Threshold = 0.5****

| **True Labels** | **Predicted Probabilities** | **Predicted Labels (if Threshold = 0.5)** |
| --- | --- | --- |
| 1 | 0.8 | 1 |
| 0 | 0.3 | 0 |
| 1 | 0.6 | 1 |
| 0 | 0.2 | 0 |
| 1 | 0.7 | 1 |
| 1 | 0.9 | 1 |
| 0 | 0.4 | 0 |
| 0 | 0.1 | 0 |
| 1 | 0.75 | 1 |
| 0 | 0.55 | 1 |

#### Confusion matrix based on above predictions

|  | **Prediction = 0** | **Prediction = 1** |
| --- | --- | --- |
| **Actual = 0** | TP=4 | FN=1 |
| **Actual = 1** | FP=0 | TN=5 |

Accordingly,

* **True Positive Rate (TPR)**:  
  Proportion of actual positives correctly identified by the classifier is
* **False Positive Rate (FPR)**:  
  Proportion of actual negatives incorrectly classified as positives

So, at the threshold of 0.5:

* True Positive Rate (Sensitivity): 0.8
* False Positive Rate: 0

The interpretation is that the model, at this threshold, correctly identifies 80% of actual positives (TPR) but incorrectly classifies 0% of actual negatives as positives (FPR).

Accordingly for different thresholds we will get ,

### ****Case 2: Threshold = 0.7****

| **True Labels** | **Predicted Probabilities** | **Predicted Labels (if Threshold = 0.7)** |
| --- | --- | --- |
| 1 | 0.8 | 1 |
| 0 | 0.3 | 0 |
| 1 | 0.6 | 0 |
| 0 | 0.2 | 0 |
| 1 | 0.7 | 0 |
| 1 | 0.9 | 1 |
| 0 | 0.4 | 0 |
| 0 | 0.1 | 0 |
| 1 | 0.75 | 1 |
| 0 | 0.55 | 0 |

#### Confusion matrix based on above predictions

|  | **Prediction = 0** | **Prediction = 1** |
| --- | --- | --- |
| **Actual = 0** | TP=5 | FN=0 |
| **Actual = 1** | FP=2 | TN=3 |

Accordingly,

* **True Positive Rate (TPR)**:  
  Proportion of actual positives correctly identified by the classifier is
* **False Positive Rate (FPR)**:  
  Proportion of actual negatives incorrectly classified as positives

### ****Case 3: Threshold = 0.4****

| **True Labels** | **Predicted Probabilities** | **Predicted Labels (if Threshold = 0.4)** |
| --- | --- | --- |
| 1 | 0.8 | 1 |
| 0 | 0.3 | 0 |
| 1 | 0.6 | 1 |
| 0 | 0.2 | 0 |
| 1 | 0.7 | 1 |
| 1 | 0.9 | 1 |
| 0 | 0.4 | 0 |
| 0 | 0.1 | 0 |
| 1 | 0.75 | 1 |
| 0 | 0.55 | 1 |

#### Confusion matrix based on above predictions

|  | **Prediction = 0** | **Prediction = 1** |
| --- | --- | --- |
| **Actual = 0** | TP=4 | FN=1 |
| **Actual = 1** | FP=0 | TN=5 |

Accordingly,

* **True Positive Rate (TPR)**:  
  Proportion of actual positives correctly identified by the classifier is
* **False Positive Rate (FPR)**:  
  Proportion of actual negatives incorrectly classified as positives

### ****Case 4: Threshold = 0.2****

| **True Labels** | **Predicted Probabilities** | **Predicted Labels (if Threshold = 0.2)** |
| --- | --- | --- |
| 1 | 0.8 | 1 |
| 0 | 0.3 | 1 |
| 1 | 0.6 | 1 |
| 0 | 0.2 | 0 |
| 1 | 0.7 | 1 |
| 1 | 0.9 | 1 |
| 0 | 0.4 | 1 |
| 0 | 0.1 | 0 |
| 1 | 0.75 | 1 |
| 0 | 0.55 | 1 |

#### Confusion matrix based on above predictions

|  | **Prediction = 0** | **Prediction = 1** |
| --- | --- | --- |
| **Actual = 0** | TP=2 | FN=3 |
| **Actual = 1** | FP=0 | TN=5 |

Accordingly,

* **True Positive Rate (TPR)**:  
  Proportion of actual positives correctly identified by the classifier is
* **False Positive Rate (FPR)**:  
  Proportion of actual negatives incorrectly classified as positives

### ****Case 5: Threshold = 0.85****

| **True Labels** | **Predicted Probabilities** | **Predicted Labels (if Threshold = 0.85)** |
| --- | --- | --- |
| 1 | 0.8 | 0 |
| 0 | 0.3 | 0 |
| 1 | 0.6 | 0 |
| 0 | 0.2 | 0 |
| 1 | 0.7 | 0 |
| 1 | 0.9 | 1 |
| 0 | 0.4 | 0 |
| 0 | 0.1 | 0 |
| 1 | 0.75 | 0 |
| 0 | 0.55 | 0 |

#### Confusion matrix based on above predictions

|  | **Prediction = 0** | **Prediction = 1** |
| --- | --- | --- |
| **Actual = 0** | TP=5 | FN=0 |
| **Actual = 1** | FP=4 | TN=1 |

Accordingly,

* **True Positive Rate (TPR)**:  
  Proportion of actual positives correctly identified by the classifier is
* **False Positive Rate (FPR)**:  
  Proportion of actual negatives incorrectly classified as positives

Based on the above result, we will plot the ROC curve

* Python3

|  |
| --- |
| true\_positive\_rate = [0.4, 0.8,  0.8, 1.0, 1]  false\_positive\_rate = [0, 0,  0, 0.2, 0.8]   plt.plot(false\_positive\_rate, true\_positive\_rate, 'o-', label='ROC')  plt.plot([0, 1], [0, 1], '--', color='grey', label='Worst case')  plt.xlabel('False Positive Rate')  plt.ylabel('True Positive Rate')  plt.title('ROC Curve')  plt.legend()  plt.show() |

**Output:**

From the graph it is implied that:

* The gray dashed line represents the “Worst case” scenario, where the model’s predictions i.e TPR are FPR are same. This diagonal line is considered the worst-case scenario, indicating an equal likelihood of false positives and false negatives.
* As points deviate from the random guess line towards the upper-left corner, the model’s performance improves.
* The Area Under the Curve (AUC) is a quantitative measure of the model’s discriminative ability. A higher AUC value, closer to 1.0, indicates superior performance. The best possible AUC value is 1.0, corresponding to a model that achieves 100% sensitivity and 100% specificity.

In all, the Receiver Operating Characteristic (ROC) curve serves as a graphical representation of the trade-off between a binary classification model’s True Positive Rate (sensitivity) and False Positive Rate at various decision thresholds. As the curve gracefully ascends towards the upper-left corner, it signifies the model’s commendable ability to discriminate between positive and negative instances across a range of confidence thresholds. This upward trajectory indicates an improved performance, with higher sensitivity achieved while minimizing false positives. The annotated thresholds, denoted as A, B, C, D, and E, offer valuable insights into the model’s dynamic behavior at different confidence levels.

## Implementation using two different models

#### Installing Libraries

* Python3

|  |
| --- |
| import numpy as np  import pandas as pd  import matplotlib.pyplot as plt  from sklearn.datasets import make\_classification  from sklearn.model\_selection import train\_test\_split  from sklearn.linear\_model import LogisticRegression  from sklearn.ensemble import RandomForestClassifier  from sklearn.metrics import roc\_curve, auc |

In order to train the[Random Forest](https://www.geeksforgeeks.org/random-forest-classifier-using-scikit-learn/) and [Logistic Regression](https://www.geeksforgeeks.org/understanding-logistic-regression/) models and to present their ROC curves with AUC scores, the algorithm creates artificial binary classification data.

#### Generating data and splitting data

* Python3

|  |
| --- |
| # Generate synthetic data for demonstration  X, y = make\_classification(      n\_samples=1000, n\_features=20, n\_classes=2, random\_state=42)    # Split the data into training and testing sets  X\_train, X\_test, y\_train, y\_test = train\_test\_split(      X, y, test\_size=0.2, random\_state=42) |

Using an 80-20 split ratio, the algorithm creates artificial binary classification data with 20 features, divides it into training and testing sets, and assigns a random seed to ensure reproducibility.

#### Training the different models

* Python3

|  |
| --- |
| # Train two different models  logistic\_model = LogisticRegression(random\_state=42)  logistic\_model.fit(X\_train, y\_train)    random\_forest\_model = RandomForestClassifier(n\_estimators=100, random\_state=42)  random\_forest\_model.fit(X\_train, y\_train) |

Using a fixed random seed to ensure repeatability, the method initializes and trains a logistic regression model on the training set. In a similar manner, it uses the training data and the same random seed to initialize and train a Random Forest model with 100 trees.

#### Predictions

* Python3

|  |
| --- |
| # Generate predictions  y\_pred\_logistic = logistic\_model.predict\_proba(X\_test)[:, 1]  y\_pred\_rf = random\_forest\_model.predict\_proba(X\_test)[:, 1] |

Using the test data and a trained [Logistic Regression](https://www.geeksforgeeks.org/ml-logistic-regression-using-python/) model, the code predicts the positive class’s probability. In a similar manner, using the test data, it uses the trained Random Forest model to produce projected probabilities for the positive class.

#### Creating a dataframe

* Python3

|  |
| --- |
| # Create a DataFrame  test\_df = pd.DataFrame(      {'True': y\_test, 'Logistic': y\_pred\_logistic, 'RandomForest': y\_pred\_rf}) |

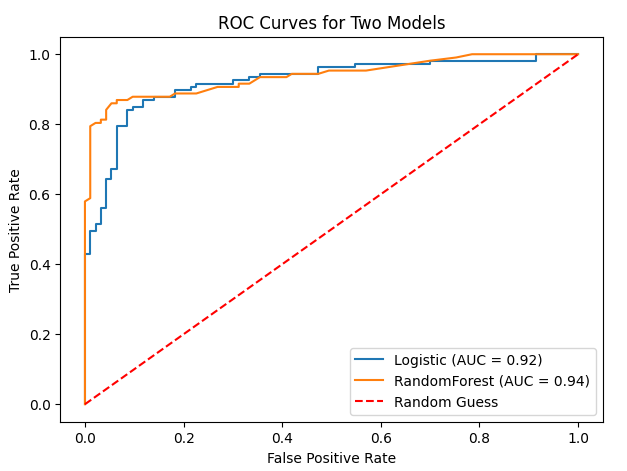
Using the test data, the code creates a DataFrame called test\_df with columns labeled “True,” “Logistic,” and “RandomForest,” adding true labels and predicted probabilities from the Random Forest and Logistic Regression models.

#### Plot the ROC Curve for the models

* Python3

|  |
| --- |
| # Plot ROC curve for each model  plt.figure(figsize=(7, 5))  for model in ['Logistic', 'RandomForest']:      fpr, tpr, \_ = roc\_curve(test\_df['True'], test\_df[model])      roc\_auc = auc(fpr, tpr)      plt.plot(fpr, tpr, label=f'{model} (AUC = {roc\_auc:.2f})')  # Plot random guess line  plt.plot([0, 1], [0, 1], 'r--', label='Random Guess')  # Set labels and title  plt.xlabel('False Positive Rate')  plt.ylabel('True Positive Rate')  plt.title('ROC Curves for Two Models')  plt.legend()  plt.show() |

**Output:**



The code generates a plot with 8 by 6 inch figures. It computes the AUC and ROC curve for each model (Random Forest and Logistic Regression), then plots the ROC curve. The [ROC curve](https://www.geeksforgeeks.org/calculate-roc-auc-for-classification-algorithm-such-as-random-forest/) for random guessing is also represented by a red dashed line, and labels, a title, and a legend are set for visualization.

## How to use ROC-AUC for a multi-class model?

For a multi-class setting, we can simply use one vs all methodology and you will have one ROC curve for each class. Let’s say you have four classes A, B, C, and D then there would be ROC curves and corresponding AUC values for all the four classes, i.e. once A would be one class and B, C, and D combined would be the others class, similarly, B is one class and A, C, and D combined as others class, etc.

The general steps for using AUC-ROC in the context of a multiclass classification model are:

#### ****One-vs-All Methodology:****

* For each class in your multiclass problem, treat it as the positive class while combining all other classes into the negative class.
* Train the binary classifier for each class against the rest of the classes.

#### Calculate AUC-ROC for Each Class:

* Here we plot the ROC curve for the given class against the rest.
* Plot the ROC curves for each class on the same graph. Each curve represents the discrimination performance of the model for a specific class.
* Examine the AUC scores for each class. A higher AUC score indicates better discrimination for that particular class.

### Implementation of AUC-ROC in Multiclass Classification

#### Importing Libraries

* Python3

|  |
| --- |
| import numpy as np  import matplotlib.pyplot as plt  from sklearn.datasets import make\_classification  from sklearn.model\_selection import train\_test\_split  from sklearn.preprocessing import label\_binarize  from sklearn.multiclass import OneVsRestClassifier  from sklearn.linear\_model import LogisticRegression  from sklearn.ensemble import RandomForestClassifier  from sklearn.metrics import roc\_curve, auc  from itertools import cycle |

The program creates artificial multiclass data, divides it into training and testing sets, and then uses the [One-vs-Restclassifier](https://www.geeksforgeeks.org/one-vs-rest-strategy-for-multi-class-classification/)technique to train classifiers for both Random Forest and Logistic Regression. Lastly, it plots the two models’ multiclass ROC curves to demonstrate how well they discriminate between various classes.

#### Generating Data and splitting

* Python3

|  |
| --- |
| # Generate synthetic multiclass data  X, y = make\_classification(      n\_samples=1000, n\_features=20, n\_classes=3, n\_informative=10, random\_state=42)  # Binarize the labels  y\_bin = label\_binarize(y, classes=np.unique(y))  # Split the data into training and testing sets  X\_train, X\_test, y\_train, y\_test = train\_test\_split(      X, y\_bin, test\_size=0.2, random\_state=42) |

Three classes and twenty features make up the synthetic multiclass data produced by the code. After label binarization, the data is divided into training and testing sets in an 80-20 ratio.

#### Training Models

* Python3

|  |
| --- |
| # Train two different multiclass models  logistic\_model = OneVsRestClassifier(LogisticRegression(random\_state=42))  logistic\_model.fit(X\_train, y\_train)    rf\_model = OneVsRestClassifier(      RandomForestClassifier(n\_estimators=100, random\_state=42))  rf\_model.fit(X\_train, y\_train) |

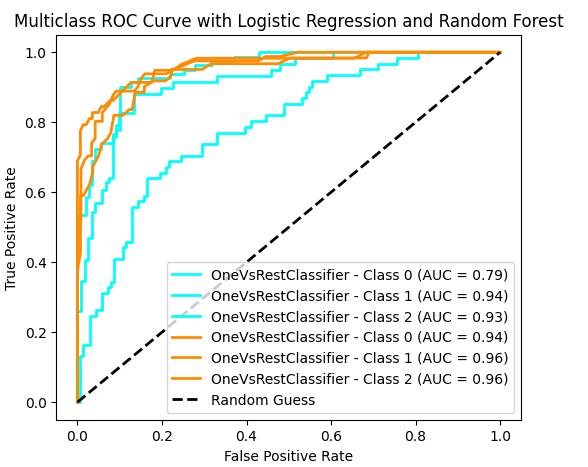
The program trains two multiclass models: a Random Forest model with 100 estimators and a Logistic Regression model with the [One-vs-Rest approach](https://www.geeksforgeeks.org/plot-multinomial-and-one-vs-rest-logistic-regression-in-scikit-learn/). With the training set of data, both models are fitted.

#### Plotting the AUC-ROC Curve

* Python3

|  |
| --- |
| # Compute ROC curve and ROC area for each class  fpr = dict()  tpr = dict()  roc\_auc = dict()  models = [logistic\_model, rf\_model]  plt.figure(figsize=(6, 5))  colors = cycle(['aqua', 'darkorange'])   for model, color in zip(models, colors):      for i in range(model.classes\_.shape[0]):          fpr[i], tpr[i], \_ = roc\_curve(              y\_test[:, i], model.predict\_proba(X\_test)[:, i])          roc\_auc[i] = auc(fpr[i], tpr[i])          plt.plot(fpr[i], tpr[i], color=color, lw=2,                   label=f'{model.\_\_class\_\_.\_\_name\_\_} - Class {i} (AUC = {roc\_auc[i]:.2f})')   # Plot random guess line  plt.plot([0, 1], [0, 1], 'k--', lw=2, label='Random Guess')   # Set labels and title  plt.xlabel('False Positive Rate')  plt.ylabel('True Positive Rate')  plt.title('Multiclass ROC Curve with Logistic Regression and Random Forest')  plt.legend(loc="lower right")  plt.show() |

**Output:**



The Random Forest and Logistic Regression models’ ROC curves and AUC scores are calculated by the code for each class. The multiclass ROC curves are then plotted, showing the discrimination performance of each class and featuring a line that represents random guessing. The resulting plot offers a graphic evaluation of the models’ classification performance.