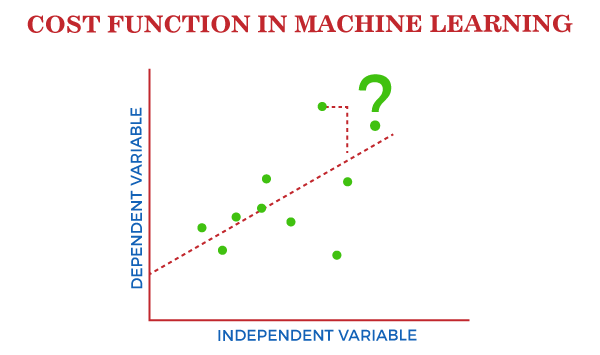
Chapter No. 5

**Concept of Cost Function:**

A Machine Learning model should have a very high level of accuracy in order to perform well with real-world applications. But how to calculate the accuracy of the model, i.e., how good or poor our model will perform in the real world? In such a case, the Cost function comes into existence. It is an important machine learning parameter to correctly estimate the model.



Cost function also plays a crucial role in understanding that how well your model estimates the relationship between the input and output parameters.

In this topic, we will explain the cost function in Machine Learning, Gradient descent, and types of cost functions.

## What is Cost Function?

***A cost function is an important parameter that determines how well a machine learning model performs for a given dataset.*** It calculates the difference between the expected value and predicted value and represents it as a single real number.

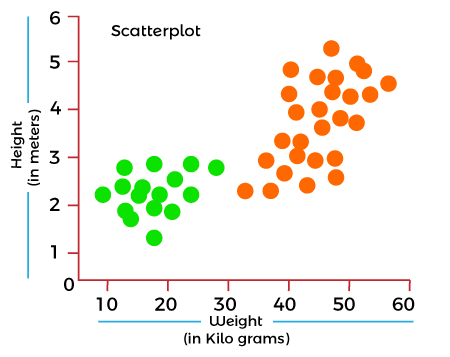
In machine learning, once we train our model, then we want to see how well our model is performing. Although there are various accuracy functions that tell you how your model is performing, but will not give insights to improve them. So, we need a function that can find when the model is most accurate by finding the spot between the undertrained and overtrained model.

In simple, "***Cost function is a measure of how wrong the model is in estimating the relationship between X(input) and Y(output) Parameter***." A cost function is sometimes also referred to as Loss function, and it can be estimated by iteratively running the model to compare estimated predictions against the known values of Y.

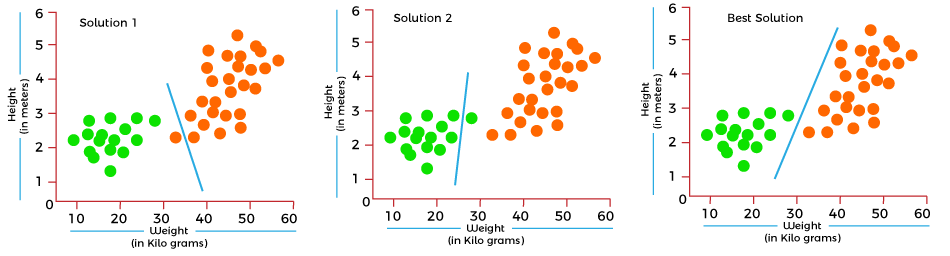
The main aim of each ML model is to determine parameters or weights that can minimize the cost function.

## Why use Cost Function?

While there are different accuracy parameters, then why do we need a Cost function for the Machine learning model. So, we can understand it with an example of the classification of data. Suppose we have a dataset that contains the height and weights of cats & dogs, and we need to classify them accordingly. If we plot the records using these two features, we will get a scatter plot as below:



In the above image, the green dots are cats, and the orange dots are dogs. Below are the three possible solutions for this classification problem.



In the above solutions, all three classifiers have high accuracy, but the third solution is the best because it correctly classifies each datapoint. The reason behind the best classification is that it is in mid between both the classes, not close or not far to any of them.

To get such results, we need a Cost function. It means for getting the optimal solution; we need a Cost function. It calculated the difference between the actual values and predicted values and measured how wrong was our model in the prediction. By minimizing the value of the cost function, we can get the optimal solution.

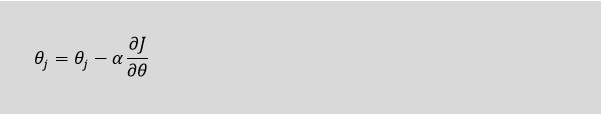
## Gradient Descent: Minimizing the cost function

As we discussed in the above section, the cost function tells how wrong your model is? And each machine learning model tries to minimize the cost function in order to give the best results. Here comes the role of Gradient descent.

"***Gradient Descent is an optimization algorithm which is used for optimizing the cost function or error in the model."*** It enables the models to take the gradient or direction to reduce the errors by reaching to least possible error. Here direction refers to how model parameters should be corrected to further reduce the cost function. The error in your model can be different at different points, and you have to find the quickest way to minimize it, to prevent resource wastage.

Gradient descent is an iterative process where the model gradually converges towards a minimum value, and if the model iterates further than this point, it produces little or zero changes in the loss. This point is known as convergence, and at this point, the error is least, and the cost function is optimized.

Below is the equation for gradient descent in linear regression:



In the gradient descent equation, alpha is known as the learning rate. This parameter decides how fast you should move down to the slope. For large alpha, take big steps, and for small alpha value, you need to take small steps.

Well, a cost function is something we want to minimize. For example, our cost function might be the sum of squared errors over the training set. Gradient descent is a method for finding the minimum of a function of multiple variables. So we can use gradient descent as a tool to minimize our cost function.

====================xxxxxxxx=========================

*Introduction to Natural Language Processing*



* Text processing has a direct application to Natural Language Processing, also known as NLP.
* NLP is aimed at processing the languages spoken or written by humans when they communicate with one another.
* This is different from the communication between a computer and a human where the communication is wither a computer program written by human or some gesture by human like clicking the mouse at some position.
* NLP tries to understand the natural language spoken by humans and classify it, analyses it as well if required respond to it.
* Python has a rich set of libraries which cater to the needs of NLP. The Natural Language Tool Kit (NLTK) is a suite of such libraries which provides the functionalities required for NLP.

Below are some applications which use NLP and indirectly python's NLTK.

## 1.Summarization

Many times, we need to get the summary of a news article, a movie plot or a big story. They are all written in human language and without NLP we have to rely on another human's interpretation and presentation of such summary to us. But with help of NLP we can write programs to use NLTK and summarize the long text with various parameters, like what is the percentage of text we want in the final output, choosing the positive and negative words for summarization etc. The online news feeds rely on such summarization techniques to present news insights.

## 2.Voice Based Tools

The voice-based tools like apples Siri or Amazon Alexa rely on NLP to understand the interaction mad with humans. They have a large training data set of words, sentences and grammar to interpret the question or command coming from a human and process it. Though it is about voice, indirectly it also gets translated to text and the resulting text form the voice is taken through the NLP system to produce result.

## 3.Information Extraction

Web scrapping is a common example of extracting data form the web pages using python code. Here it may not be strictly NLP based but it does involve text processing. For example, if we need to extract only the headers present in a html page, then we look for the h1 tag int he page structure and find a way to extract the text between only those tags. This need text processing program from python.

## 4.Spam Filtering

The spam in emails can be identified and eliminated by analysing the text in the subject line as well as in the content of the message. As the spam emails are usually sent in bulk to many recipients, even if their subjects and contents have little variation, that can be matched and tagged to mark them as spam Again it needs the use of the NLTK libraries.

## 5.Language Translation

Computerized language translation relies heavily on NLP. As more and more languages are used in the online platform, it becomes a necessity to automate the translation from one human language to another. This will involve programming to handle the vocabulary, grammar and context tagging of the languages involved in translation. Again, NLTK is used to handle such requirements.

## 6.Sentiment Analysis

To find out the overall reaction to the performance of a movie, we may have to read thousands of feedback posts from the audience. But that too can be automated by using the classification of positive an negative feedback through words and sentence analysis. And then measuring the frequency of positive and negative reviews to find the overall sentiment of the audience. This obviously needs the analysis of the human language written by the audience and NLTK is used heavily here for processing the text.

Natural language processing, or NLP, combines computational linguistics—rule-based modeling of human language—with statistical and machine learning models to enable computers and digital devices to recognize, understand and generate text and speech.

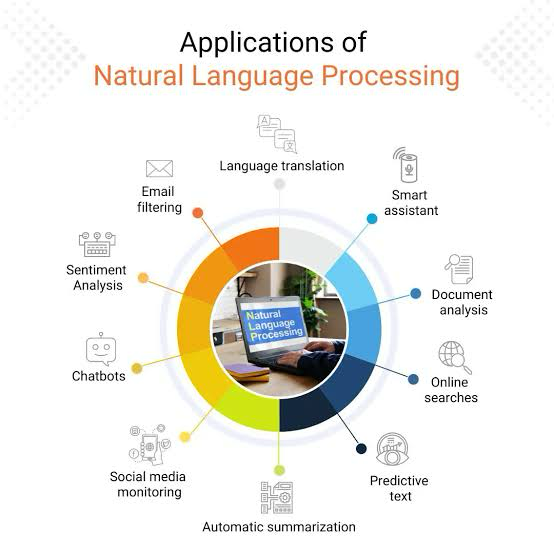
A branch of artificial intelligence (AI), NLP lies at the heart of applications and devices that can

* translate text from one language to another
* respond to typed or spoken commands
* recognize or authenticate users based on voice
* summarize large volumes of text
* assess the intent or sentiment of text or speech
* generate text or graphics or other content on demand

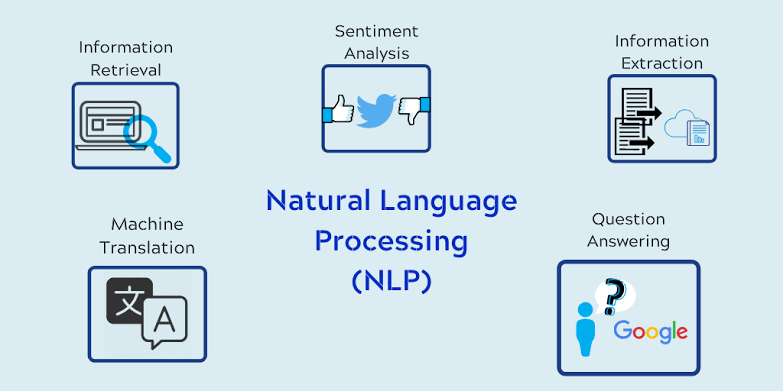
Steps in NLP:



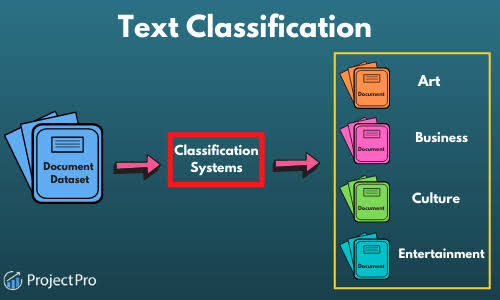
Applications of NLP



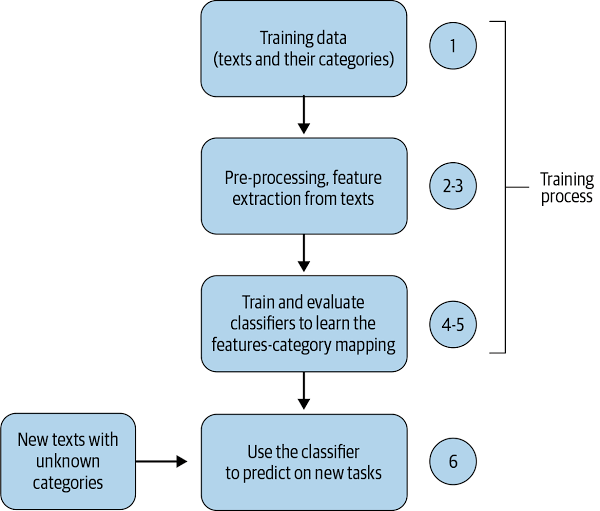
Applications of NLP



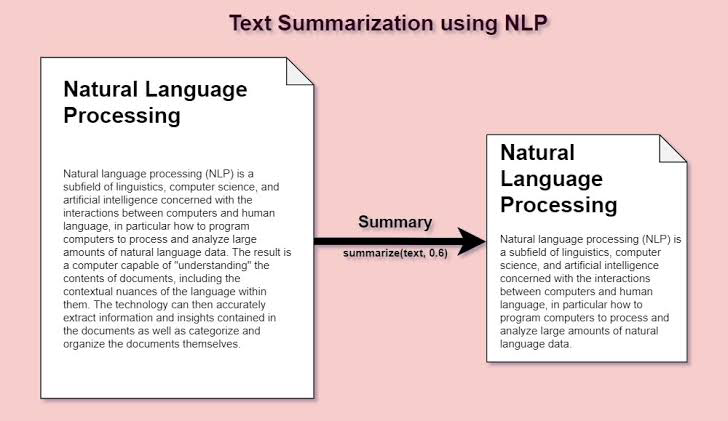
Concept of Text Classification:



**Text Classification Flowchart**



Text Summarization



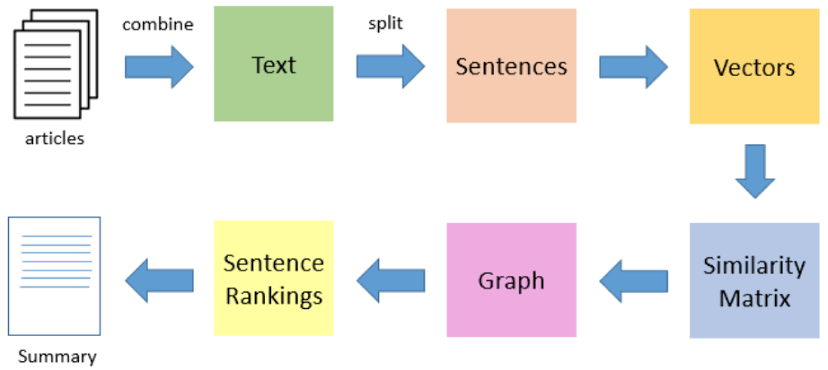


Fig: Text Summarization

Text summarisation NLP

Summarizing a text tract effectively requires understanding and condensing the main ideas into a shorter form without losing the essence. In Python, you can use the nltk library for basic natural language processing tasks and the gensim library for more sophisticated models like the Latent Semantic Analysis (LSA) or TextRank for summarization. Below is an example using gensim for summarizing a text.

pip install gensim

from gensim.summarization import summarize

# Sample text

text = """

Artificial intelligence (AI) refers to the simulation of human intelligence in machines that are programmed to think like humans and mimic their actions. The term may also be applied to any machine that exhibits traits associated with a human mind such as learning and problem-solving. The ideal characteristic of artificial intelligence is its ability to rationalize and take actions that have the best chance of achieving a specific goal. A subset of artificial intelligence is machine learning, which refers to the concept that computer programs can automatically learn from and adapt to new data without being assisted by humans. Deep learning techniques enable this automatic learning through the absorption of huge amounts of unstructured data such as text, images, or video.

"""

# Summarize text

summary = summarize(text, ratio=0.5) # Summarize to 50% of the original text's size

print("Original Text:\n", text)

print("\nSummary:\n", summary)

In this code, summarize function from gensim is used to condense the provided text based on a specified ratio. The ratio=0.5 parameter tells the function to reduce the original text to 50% of its size. You can adjust this ratio according to how short you want the summary to be.

It’s worth noting that summarization, especially on complex and nuanced texts, can be challenging, and the quality of the output might vary. Experimenting with different settings and summarization models (e.g., LSA, TextRank) can help achieve better results.

Keep in mind that the gensim summarization module works best with texts that are longer and well-structured. For very short texts or texts that don’t follow a coherent structure, the summarization might not be effective or could return unexpected results.

==============

Text classification involves categorizing text into predefined groups. It's a fundamental task in natural language processing (NLP), useful for spam filtering, sentiment analysis, tagging content, etc. A common approach uses machine learning algorithms. Here, I'll guide you through a simple text classification task using Python's `scikit-learn` library, which involves training a model to classify text into categories. We'll perform sentiment analysis as an example, classifying text into 'positive' or 'negative' sentiments.

First, ensure you have the necessary packages installed:

```bash

pip install scikit-learn numpy

```

Now, let's create a Python script for this task:

```python

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

from sklearn.feature\_extraction.text import CountVectorizer

from sklearn.naive\_bayes import MultinomialNB

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report

# Sample dataset

texts = ["I love this movie, it's amazing!", "Absolutely great movie, will see it again",

"Terrible movie, wasted my time", "The movie was bad, very disappointing",

"This movie made my day, so good!", "Not a good movie, I didn't like it",

"The plot was boring, didn't like the movie", "Fantastic movie, highly recommend watching it"]

labels = [1, 1, 0, 0, 1, 0, 0, 1] # 1 for positive sentiment, 0 for negative

# Split dataset into training and testing set

texts\_train, texts\_test, labels\_train, labels\_test = train\_test\_split(texts, labels, test\_size=0.25, random\_state=42)

# Text vectorization - converting text into numerical data

vectorizer = CountVectorizer()

X\_train = vectorizer.fit\_transform(texts\_train)

X\_test = vectorizer.transform(texts\_test)

# Train a Naive Bayes classifier

model = MultinomialNB()

model.fit(X\_train, labels\_train)

# Make predictions

predictions = model.predict(X\_test)

# Evaluate the model

print("Accuracy:", accuracy\_score(labels\_test, predictions))

print("\nConfusion Matrix:\n", confusion\_matrix(labels\_test, predictions))

print("\nClassification Report:\n", classification\_report(labels\_test, predictions))

```

In this script:

1. \*\*Data Preparation\*\*: We create a small dataset of texts (`texts`) and their corresponding labels (`labels`), where `1` represents a positive sentiment, and `0` represents a negative sentiment.

2. \*\*Training and Testing Split\*\*: The dataset is split into a training set and a testing set using `train\_test\_split`, with 75% of the data used for training and the rest for testing.

3. \*\*Vectorization\*\*: We convert the text data into numerical format using `CountVectorizer`, which transforms the text into a sparse matrix of token counts.

4. \*\*Model Training\*\*: We use the `MultinomialNB` (Multinomial Naive Bayes) classifier to train the model with the vectorized training data.

5. \*\*Prediction and Evaluation\*\*: The trained model is then used to predict the sentiment of the test data. The accuracy of the predictions, along with a confusion matrix and a classification report, are printed out to evaluate the model's performance.

This example provides a basic introduction to text classification. For more complex applications, consider exploring other feature extraction methods like TF-IDF, different machine learning models (e.g., Support Vector Machines, Logistic Regression, Neural Networks), and more sophisticated text preprocessing techniques.

=============xxxxxx================