ACKNOWLEDGMENT

We would like to express our deepest gratitude to Dr. Suchit Purohit, our project supervisor, whose unwavering support and expertise were instrumental in the successful completion of this Speech Emotion Recognition project. Her guidance, insightful feedback, and encouragement throughout the project have been invaluable. We are also thankful to Department of computer Science Faculties and all lab stab members for providing the necessary resources and facilities that enabled us to conduct this research effectively. We also wish to express our heartfelt gratitude to Almighty for her blessings, guidance, and grace, which have sustained us and provided clarity throughout this project. Additionally, We extend our sincere appreciation to our family and friends for their unwavering support and encouragement.

Table of Contentss

ABSTRACT

In this project, we explore the recognition of speech emotions using a combination of machine learning and deep learning techniques. The study employs traditional machine learning classifiers including Support Vector Machines (SVM), k-Nearest Neighbors (kNN), Decision Trees, and Random Forests, alongside the deep learning approach of Long Short-Term Memory (LSTM) networks. Our aim is to compare the performance of these methods in accurately identifying emotions from speech signals. The dataset used consists of labeled audio recordings representing various emotional states. Preprocessing steps include feature extraction and normalization to ensure robust model training. The results indicate that while traditional machine learning models provide a solid baseline, the LSTM network significantly improves the recognition accuracy by effectively capturing temporal dependencies in speech data. This research demonstrates the potential of combining both traditional and deep learning approaches to enhance the accuracy and reliability of speech emotion recognition systems.

Chapter-1. Introduction to Speech Emotion Recognition (SER)

1.1 Basic introduction of SER?

Emotions: are an integral part of human communication, which allows us to convey feelings, intentions, and nuances beyond words.

The process of identifying and analyzing emotion conveyed in spoken language is called Speech Emotion Recognition(SER).

For example ,Imagine a world where machines can understand our feelings just by listening to what we speak. To make this possible using advanced algorithms to detect and interpret emotional content in audio recordings.

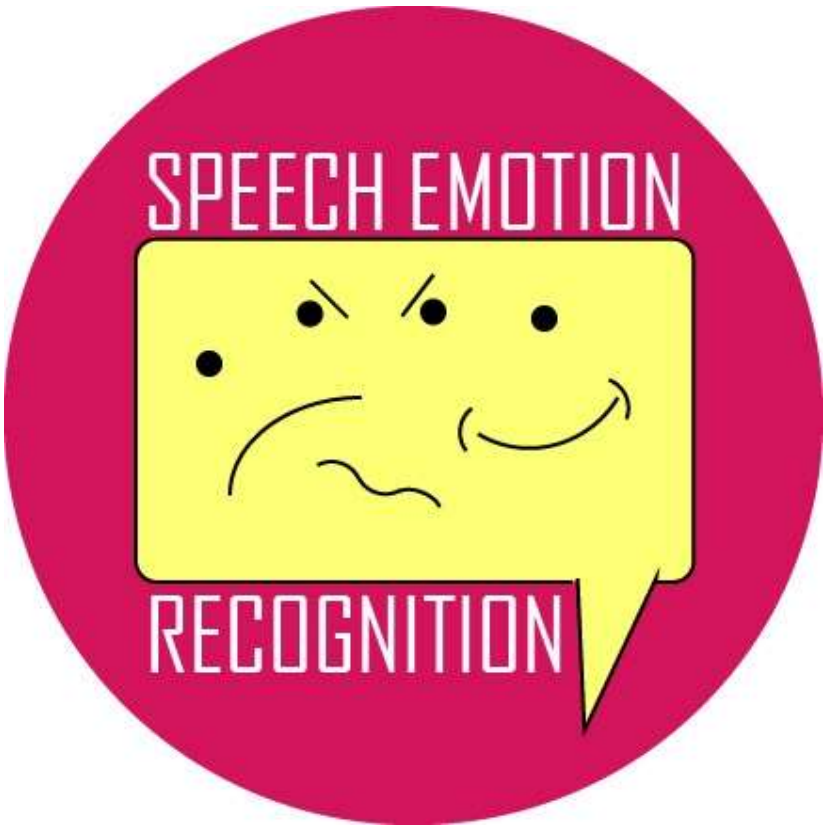


Figure 1 SER

1.1.1.How does it work ?

Speech Patterns: SER analysis of various speech features, which includes Prosody, Pitch, Rhythm, and many other things

Prosody: Changes in pitch, rhythm, and intonation.

Pitch: The frequency of the voice.

Rhythm: Patterns of speech.

Emotional States: SER aims to determine the emotional state of a speaker. Like happiness, anger, sadness, frustration, and many other emotions of a human being. By understanding these emotions with the help of advanced algorithms SER enhances human and computer interaction.

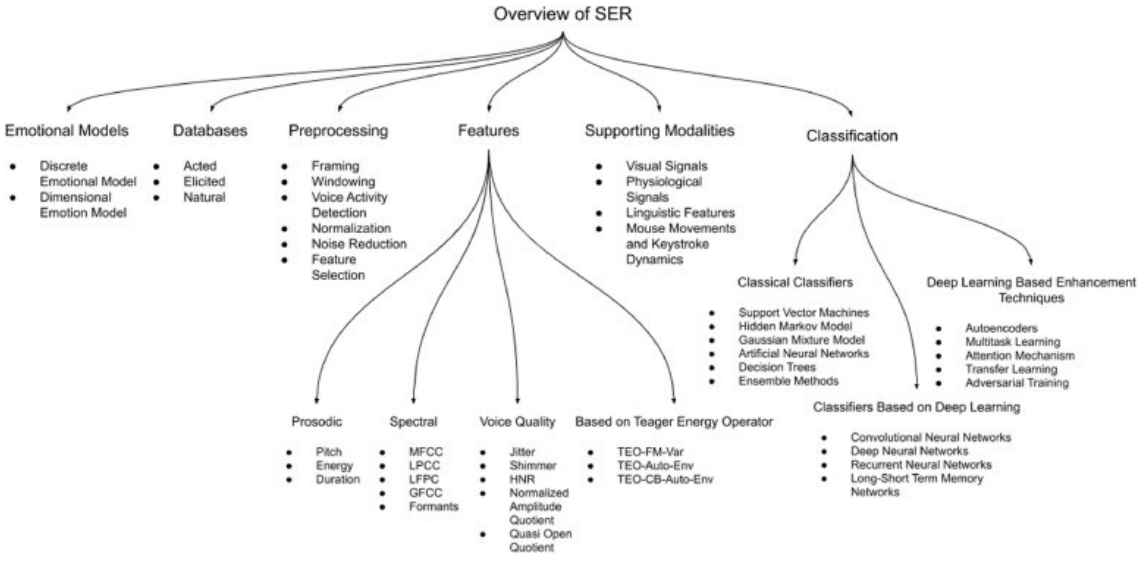


Figure 2 Overviewssss

1.1.2 Real-world application where SER works

1) Intelligent Tutoring Systems(ITS): it helps tutors identify or understand students' saturation points, confusion, and boredom during lectures. It adapts content and teaching styles for better learning outcomes.

2) Lie Detection: Emotions and expressions play a crucial role in detecting Lies spoken by humans. SER can help law enforcement agencies capture the person who's guilty of a crime based on their Speech.

3) In-car Emotion Recognition: Detecting persons emotion(anger, tiredness, stress, etc) while driving to prevent accidents. Imagine a car that senses a driver's emotions alerts the driver prevents accidents and enhances road safety.

4) Customer care: Call centers use SER to understand customer emotions, prioritize dissatisfied callers, and improve overall satisfaction.

5) Robots and Smart Homes: SER enables more natural interaction with robots and detects distress, fear, and other emotions in smart homes.

6) Gender Inequity and Hate Speech Detection: SER can identify acoustic cues related to gender discrimination and hate speech. And there are many other places where SER is applied for the betterment of humans and to enhance human-computer relations.

7) Healthcare: SER assists in psychological wellness evaluation, stress monitoring, and therapy sessions.

8) Social media analysis: Analyzing emotions in social media content helps tailor marketing strategies and user experiences.

9) Intelligent systems: SER is crucial for virtual assistants, chatbots, and personalized recommendations.

1.1.3 Advancements of SER in recent years

As the importance of human-computer interaction(HCI) continues to strengthen. In the field of deep learning, numerous models have found their application in the realm of Speech Emotion recognition(SER), leading to significant advancements in recent years. However, effectively

recognizing and processing human emotions through computational systems remains a complex and formidable challenge.

1) Deep learning techniques: Recently there have increasingly adopted deep learning models such as Deep Neural Networks(DNNs) and Convolutional Neural Networks(CNNs)

2) Emotional Databases: The availability of appropriate emotional datasets has expanded. These datasets contain labeled speech samples representing various emotions.

3) Machine learning and deep learning classifiers: Apply suitable classifiers including both traditional Machine Learning(ml) algorithms and deep learning architectures.

1.2 Challenges face while building the SER system

we face many challenges while building the SER system including problems regarding Datasets:

1) Existing datasets: Some datasets are free and some of them are paid datasets for SER, they all have limitations. These datasets often lack diversity in terms of languages, speakers, genders, ages, and dialects. Training models on limited data can lead to overconfidence(overfitting) and future failures.

There is no standardized set of labels for human emotions. Different datasets use slightly different emotion categories, making it challenging to define a global standard

2) Curating a Datasets: collecting and labeling training examples directly is one approach. however, ongoing data collection beyond the initial dataset version is crucial.

crowdsources can be a cost-effective way to label data, but it still incurs expenses per label. Alternately we can use an already trained model to create labels for our audio data.

3) Training a Model: we have sufficient data, and on that data, we train a classifier that maps from a speech signal to emotion labels. Feature extraction, model architecture, and hyperparameter tunning are crucial steps. And balancing performance and ethical considerations remains imperative as the field evolves.

Addressing these challenges leads to a more accurate and empathetic SER system that enhances human-computer interaction.

1.3 Ongoing research efforts to improve SER

There is much ongoing research giving efforts to improve the SER system some of them are:

1) Comprehensive surveys: Researchers conduct comprehensive surveys to analyze existing studies, datasets, feature extraction methods, and classification techniques.

these surveys provide insights into the state of the art and guide future research.

2) Deep learning models: Deep Learning approaches, including Attention Models and Convolutional Neural Networks(CNNs), have gained prominence in SER. These models automatically learn relevant features from row audio data, improving recognition accuracy.

3) Emotional speech datasets: The availability of diverse and well-labeled emotional speech datasets is crucial for training and evaluating SER systems.

4) Feature Extraction: Novel feature extraction methods are being explored to capture emotional cues effectively. According to updated data, Researchers focus on robustness against noise and variations in speech quality.

5) Classification techniques: Both traditional machine learning algorithms and deep learning architectures are used for emotion classification and recognition. Comparative analyses of different approaches help identify the most effective methods.

1.4 How we contribute to improving SER research

Contributing to SER research is a valuable endeavor. some of how we can contribute to enhancing the SER system:

1) Graph-Based Approaches: Explore novel methods like graph theory for classifying emotionally-colored speech signals. Extract statistical information from time series data to create unique feature sets.

2) Survey Existing Work: Comprehensive surveys and literature reviews on SER to understand the landscape, existing challenges, and recent advancements

3) Feature Engineering: Contributing by designing or improving new features. Feature selection significantly impacts SER performance.

4) Dataset creation and Annotation: Diverse datasets with labels and emotional speech samples. Ensures representation across languages, cultures, and demographics.

5) Collaborate with Researchers: join research communities, attend conferences, and engage in discussions. collaborates with the experts to exchange ideas and insights.

6) Open-Source contribution: contributes to open-source SER libraries, tools, or frameworks. Enhance accessibility and reproducibility.

Every contribution, big or small, contributes to advancing SER and its real-world applications.

1.5 Programming languages are commonly used in SER Research

In SER research, several programming languages are commonly used. some of them are:

1) python: Python is a widely used language in the Computer field, it is widely adopted for data preparation, feature extraction, and classification tasks in SER. It extracts speech features and implements machine learning models, including deep learning frameworks.

2) Other languages: While Python dominates, other languages like Java, c++, and MATLAB have also been used for SER research. Research chooses languages based on their familiarity, libraries, and performance requirements.

The choice of languages depends on the specific research context and tools available to researches.

1.6 Structure of Report

Chapter 2: Tools & Technologies

Here, we will talk about every tool and technology used in the project.

Programming Language: Python

Libraries: Scikit-learn, Pandas, NumPy, Matplotlib, Seaborn

Platforms: Jupyter Notebook, Visual Studio Code

Framework: Streamlit

Chapter 3: Background Principle

In this chapter, we'll discussed about Machine Learning and Deep Learning which we used in this project.

Machine Learning Algorithms Used:

Decision Trees: A non-parametric supervised learning method.

Random Forest: An ensemble method using multiple decision trees.

Support Vector Machine (SVM): A classification method that finds the optimal hyperplane.

K-Nearest Neighbors (KNN): A simple, instance-based learning algorithm

Chapter 4: Literature Survey

An extensive review of existing literature revealed various methodologies and algorithms used for Speech Emotion Recognition(SER). Previous studies have highlighted the effectiveness of ensemble methods and neural networks in improving prediction accuracy.

Chapter 5: Methodology

This chapter provides a high-level overview of the entire development process of model building, outlining the steps involved in achieving the project's goals.

Chapter 6: Dataset

In our project, we working with two obvious datasets: the Toronto Affecting Speech Set (TESS) and The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS).

TESS contains a different range of excitements containing Anger, Disgust, Fear, Happiness, Noncommittal, and Friendly Surprise. This dataset encompasses a total of 2800 visual and audio entertainment transmitted via radio waves files, contribution a rich variety of sentimental verbalizations for study.

TESS: Toronto emotional speech set (TESS) (kaggle.com)

Chapter 7: Implementation

In this chapter, we discussed everything about the models which were implemented using Python and the Scikit-learn library. Each algorithm was trained on the preprocessed data, and hyperparameters of the best model were optimized using GridSearchCV.

Chapter 8: Result and Conclusion

Here, we will focus on evaluating the performance of the developed model, comparison of the model, and live evaluation on the deployed website.

Chapter 9: References

Chapter-2 Tools & Technologies

2.1 Importance of Tools & Technologies

Importance of Tools & Technologies for Speech Emotion Recognition using SVM and LSTM with MFCC:

Utilizing Support Vector Machines (SVM) and Long Short-Term Memory (LSTM) networks in conjunction with Mel-Frequency Cepstral Coefficients (MFCC) for speech emotion recognition holds significant importance due to several reasons:

1. Robustness: SVMs are known for their robustness in handling high-dimensional data and their ability to effectively classify data points into different classes, making them suitable for emotion classification tasks. Similarly, LSTM networks are capable of capturing temporal dependencies in sequential data, allowing them to model the dynamic nature of emotional expression in speech.

2. Feature Representation : MFCCs are widely used in speech processing due to their effectiveness in capturing the spectral characteristics of speech signals. By extracting MFCCs as features, the model can focus on relevant information for emotion recognition, enhancing classification performance.

3. Interpretability: SVMs provide a clear decision boundary between classes, making it easier to interpret the reasoning behind classification decisions. This can be particularly valuable in applications where understanding the underlying factors contributing to emotion recognition is crucial.

4. Memory Retention: LSTM networks excel in retaining long-term dependencies in sequential data, allowing them to capture subtle variations in speech patterns that contribute to emotional expression. This memory retention capability is essential for accurately modeling the temporal dynamics of emotions in speech.

5. Generalization: By leveraging the capabilities of SVM and LSTM models with MFCC features, the system can generalize well to unseen data, enabling reliable emotion recognition across diverse speech samples and contexts.

In summary, the combination of SVM and LSTM models with MFCC features provides a powerful framework for speech emotion recognition, offering robustness, interpretability, memory retention, and generalization capabilities essential for accurate and effective emotion classification.

2.2 Tools & Technologies Used

To streamline the process of speech emotion recognition using different classifiers, we can leverage various libraries and modules as follows:

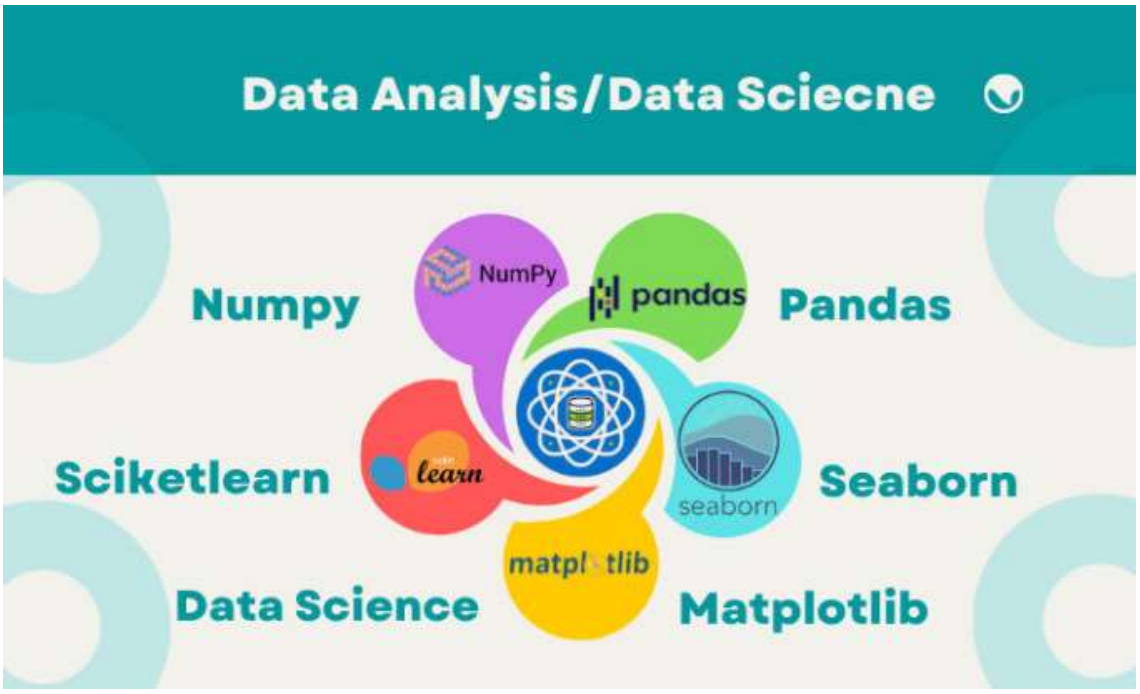


Figure 3 Tools

Numpy :

NumPy will be used for efficient numerical computations, such as handling arrays and matrices, which are essential for processing and manipulating the audio data and features extracted from it.

Pandas :

Pandas will assist in data manipulation and analysis. It will be used for organizing the dataset, managing labels, and performing data cleaning and transformations required for preparing the data for machine learning models.

Matplotlib :

Matplotlib will be utilized for visualizing the data, such as plotting the distribution of different emotions, the waveform of audio signals, and the performance metrics of the classifiers.

Seaborn :

Seaborn will be used to create more attractive and informative statistical graphics. It can help in visualizing correlations between features, the distribution of features, and enhancing the aesthetic appeal of plots created with Matplotlib.

Scikit-learn (sklearn) :

Scikit-learn will be the main library for implementing machine learning models. It will be used for feature extraction, data splitting, training different classifiers (like SVM, Random Forest, etc.), model evaluation, and performance comparison.

Librosa :

Librosa will be critical for audio processing. It will be used for loading audio files, extracting features like MFCC (Mel-Frequency Cepstral Coefficients), chroma features, spectral contrast, and other relevant audio features needed for emotion recognition.

IPython.display :

IPython.display will be useful for displaying rich media within Jupyter notebooks, such as audio playback for verification purposes and displaying interactive widgets to control the flow of the notebook.

SpeechRecognition :

SpeechRecognition will be used for converting spoken audio into text, which can be helpful for additional text-based features or for verifying the content of the audio files used in the dataset.

pyttsx3 :

Pyttsx3 will be used for generating synthetic speech from text. This can be useful for creating custom audio datasets or for creating audio outputs as part of a user interface for demonstrating the emotion recognition system.

Workflow Summary

1. Data Preparation : Use Pandas and Numpy to load, clean, and preprocess the audio dataset.

2. Feature Extraction : Use Librosa to extract relevant audio features.

3. Exploratory Data Analysis : Use Matplotlib and Seaborn to visualize the distribution and relationships of features.

4. Model Training and Evaluation : Use Scikit-learn to train various classifiers on the extracted features and evaluate their performance.

5. Rich Media Display : Use IPython.display to interactively display results and audio within Jupyter notebooks.

6. Speech Recognition and Synthesis : Use SpeechRecognition to verify audio content and pyttsx3 to generate synthetic speech for demonstrations or dataset augmentation.

By using these libraries and modules effectively, we can simplify the workflow for speech emotion recognition and ensure a streamlined and efficient process.

Chapter 3 Background Principle

3.1 Data Preprocessing :

Data preprocessing is the process of transforming raw data into an understandable format. It is also an important step in data mining as we cannot work with raw data. The quality of the data should be checked before applying machine learning or data mining algorithms.

Why is Data preprocessing important?

Preprocessing of data is mainly to check the data quality. The quality can be checked by the following

- **Accuracy**: To check whether the data entered is correct or not.
- **Completeness**: To check whether the data is available or not recorded.
- **Consistency:** To check whether the same data is kept in all the places that do or do not match.
- **Timeliness**: The data should be updated correctly.
- **Believability**: The data should be trustable.
- **Interpretability**: The understandability of the data.

Major Tasks in Data Preprocessing:

1. Data integration
2. Data reduction
3. Data cleaning
4. Data-transformation

Figure 4Data-Preprocessing

3.1.1 Data cleaning:

Data cleaning is the process of removing incorrect data, incomplete data, and inaccurate data from the datasets, and it also replaces the missing values. There are some techniques for data cleaning

Handling missing values:

- Standard values like “Not Available” or “NA” can be used to replace the missing values.
- Missing values can also be filled manually but it is not recommended when that dataset is big.
- The attribute’s mean value can be used to replace the missing value when the data is normally distributed whereas in the case of non-normal distribution median value of the attribute can be used.

- While using regression or decision tree algorithms the missing value can be replaced by the most probable value.

Noisy:

Noisy generally means random error or containing unnecessary data points. Here are some of the methods to handle noisy data.

- **Binning**: This method is to smooth or handle noisy data. First, the data is sorted and then the sorted values are separated and stored in the form of bins. There are three methods for smoothing data in the bin. **Smoothing by bin mean method**: In this method, the values in the bin are replaced by the mean value of the bin; **Smoothing by bin median**: In this method, the values in the bin are replaced by the median value; **Smoothing by bin boundary**: In this method, the using minimum and maximum values of the bin values are taken and the values are replaced by the closest boundary value.
- **Regression**: This is used to smooth the data and will help to handle data when unnecessary data is present. For the analysis, purpose regression helps to decide the variable that is suitable for our analysis.
- **Clustering**: This is used for finding the outliers and also in grouping the data. Clustering is generally used in unsupervised learning.

3.1.2 Data integration:

The process of combining multiple sources into a single dataset. The Data integration process is one of the main components of data management. There are some problems to be considered during data integration.

- **Schema integration**: Integrates metadata(a set of data that describes other data) from different sources.
- **Entity identification problem:** Identifying entities from multiple databases. For example, the system or the user should know the student _ID of one database and the student name of another database belonging to the same entity.
- **Detecting and resolving data value concepts**: The data taken from different databases while merging may differ. The attribute values from one database may differ from

another database. For example, the date format may differ like “MM/DD/YYYY” or “DD/MM/YYYY”.

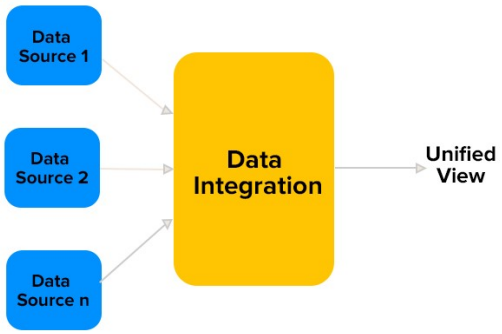


Figure 5 Data Integration

3.1.3 Data Transformation :

The change made in the format or the structure of the data is called data transformation. This step can be simple or complex based on the requirements. There are some methods for data transformation.

- **Smoothing**: With the help of algorithms, we can remove noise from the dataset and help in knowing the important features of the dataset. By smoothing we can find even a simple change that helps in prediction.
- **Aggregation**: In this method, the data is stored and presented in the form of a summary. The data set which is from multiple sources is integrated into with data analysis description. This is an important step since the accuracy of the data depends on the quantity and quality of the data. When the quality and the quantity of the data are good the results are more relevant.
- **Discretization**: The continuous data here is split into intervals. Discretization reduces the data size. For example, rather than specifying the class time, we can set an interval like (3 pm-5 pm, 6 pm-8 pm).
- **Normalization:** It is the method of scaling the data so that it can be represented in a smaller range. Example ranging from -1.0 to 1.0.

Figure 6 Data Transformation

3.1.4 Data reduction:

This process helps in the reduction of the volume of the data which makes the analysis easier yet produces the same or almost the same result. This reduction also helps to reduce storage space. There are some of the techniques in data reduction are Dimensionality reduction, Numerosity reduction, and Data compression.

- **Dimensionality reduction:** This process is necessary for real-world applications as the data size is big. In this process, the reduction of random variables or attributes is done so that the dimensionality of the data set can be reduced. Combining and merging the attributes of the data without losing its original characteristics. This also helps in the reduction of storage space and computation time is reduced. When the data is highly dimensional the problem called “Curse of Dimensionality” occurs.
- **Numerosity Reduction:** In this method, the representation of the data is made smaller by reducing the volume. There will not be any loss of data in this reduction.
- **Data compression:** The compressed form of data is called data compression. This compression can be lossless or lossy. When there is no loss of information during compression it is called lossless compression. Whereas lossy compression reduces information but removes only the unnecessary information.

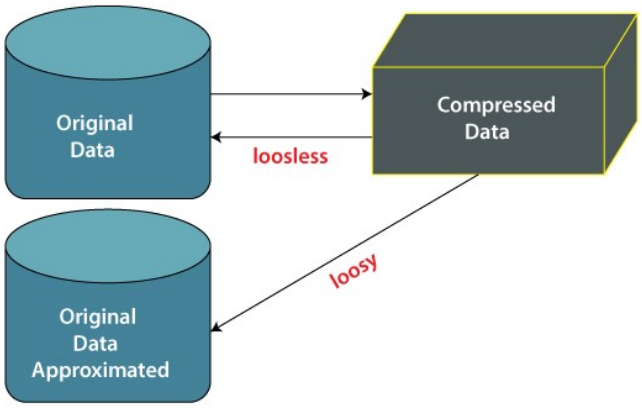


Figure 7 Data reduction

Feature Selection and Feature Extraction :

Feature selection is a way of selecting the subset of the most relevant features from the original features set by removing the redundant, irrelevant, or noisy features.

A feature is an attribute that has an impact on a problem or is useful for the problem, and choosing the important features for the model is known as feature selection. Each machine learning process depends on feature engineering, which mainly contains two processes; which are Feature Selection and Feature Extraction.

Although feature selection and extraction processes may have the same objective, both are completely different from each other. The main difference between them is that feature selection is about selecting the subset of the original feature set, whereas feature extraction creates new features. Feature selection is a way of reducing the input variable for the model by using only relevant data in order to reduce overfitting in the model.

Below are some benefits of using feature selection in machine learning:

- It helps in avoiding the curse of dimensionality.
- It helps in the simplification of the model so that it can be easily interpreted by the researchers.
- It reduces the training time.
- It reduces overfitting hence enhancing the generalization

Feature Selection Techniques

There are mainly two types of Feature Selection techniques, which are:

- **Supervised Feature Selection technique** Supervised Feature selection techniques consider the target variable and can be used for the labelled dataset.
- **Unsupervised Feature Selection technique** Unsupervised Feature selection techniques ignore the target variable and can be used for the unlabelled dataset.

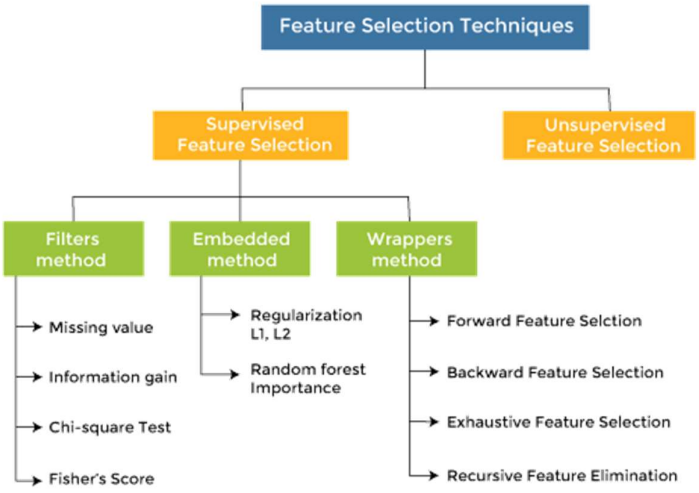


Figure 8 Feature Selection

Feature Extraction :

Feature extraction in machine learning involves transforming or selecting raw data into a format suitable for training models. Its goal is to identify relevant aspects of the data that improve model accuracy and efficiency while reducing computational complexity.

Several techniques are commonly used in feature extraction:

1. Principal Component Analysis (PCA) :

   PCA reduces dimensionality by transforming data into a lower-dimensional space while preserving as much variance as possible. It is effective for handling high-dimensional data and removing correlated features.

2. Feature Selection :

   This involves selecting a subset of the most relevant features for training the model. Methods include Univariate Selection, Recursive Feature Elimination (RFE), and Feature Importance from tree-based models like Random Forests.

3. Manual Feature Engineering :

   Creating new features based on domain knowledge or insights about the data can improve model performance. Examples include generating interaction terms or aggregating data over time windows.

4. Dimensionality Reduction Techniques :
   Apart from PCA, methods like t-Distributed Stochastic Neighbor Embedding (t-SNE) and Linear Discriminant Analysis (LDA) are used to visualize or transform data into lower-dimensional spaces while preserving important information.

5. Transformations :

   Applying mathematical transformations such as scaling (e.g., Min-Max scaling, Standardization) or encoding categorical variables into numerical form (e.g., one-hot encoding) prepares data for modeling.

Mel-Frequency Cepstral Coefficients (MFCC) :

Mel-Frequency Cepstral Coefficients (MFCCs) are a crucial feature used in audio signal processing, particularly in tasks involving speech and speaker recognition, as well as emotion

classification from speech. MFCCs provide a compact representation of the power spectrum of an audio signal, capturing important characteristics that relate to how humans perceive sound.

1. Cepstrum :

   The cepstrum is the result of taking the inverse Fourier transform of the logarithm of the estimated spectrum of a signal. It allows for the separation of the spectral envelope from the fine structure in a signal, which is useful for various audio processing tasks.

2. Mel Scale :

   The Mel scale is a perceptual scale of pitches judged by listeners to be equal in distance from one another. It is a linear frequency spacing below 1000 Hz and a logarithmic spacing above 1000 Hz. This scale more accurately reflects the human ear's response to different frequencies.

Calculation Steps :

1. Pre-emphasis :

   Apply a pre-emphasis filter to the audio signal to amplify high frequencies. This step helps balance the frequency spectrum and compensates for the natural decrease in energy at higher frequencies.

2. Framing :

   Divide the audio signal into small overlapping frames. This is necessary because the properties of speech signals change over time, and analyzing short frames allows capturing these variations.

3. Windowing :
   Apply a window function (typically a Hamming window) to each frame to minimize signal discontinuities at the edges. Windowing reduces spectral leakage in the Fourier transform.

4. Fast Fourier Transform (FFT) :

Perform FFT on each windowed frame to convert the time-domain signal into the frequency domain. This step reveals the frequency components of the signal within each frame.

5. Mel Filter Bank :

Pass the power spectrum of each frame through a set of Mel filter banks. Each filter in the bank focuses on a specific range of frequencies, emphasizing frequencies that are more critical to human hearing.

6. Logarithm :

Compute the logarithm of the filter bank outputs. This step converts the power spectrum to a logarithmic scale, simulating the human ear's perception of sound intensity.

7. Discrete Cosine Transform (DCT) :

Apply DCT to the logarithm of the Mel filter bank outputs. The result is a set of coefficients known as MFCCs. The DCT helps to decorrelate the filter bank coefficients and concentrate the signal information into a few coefficients.

MFCC Representation

Typically, the first 12-13 MFCCs are used to represent the spectral properties of the signal. These coefficients capture the broad spectral shape, which is crucial for distinguishing different speech sounds and emotions.

Applications

MFCCs are extensively used in:

- Speech Recognition : Identifying spoken words or phrases.

- Speaker Recognition : Identifying or verifying a speaker's identity.

- Emotion Classification : Determining the emotional state of a speaker from their speech.

By transforming the audio signal into a compact representation that aligns well with human auditory perception, MFCCs provide a powerful tool for various audio and speech processing applications.

3.2 Machine learning :

Machine learning (ML) is a subset of artificial intelligence (AI) that involves training algorithms to recognize patterns and make decisions based on data.

Machine Learning is the field of study that gives computers the capability to learn without being explicitly programmed. ML is one of the most exciting technologies that one would have ever come across. As is evident from the name, it gives the computer that makes it more similar to humans: The ability to learn.

3.2.1 Types of Machine Learning :

There are several types of machine learning, each with special characteristics and applications. Some of the main types of machine learning algorithms are as follows:

1. Supervised Machine Learning
2. Unsupervised Machine Learning
3. Semi-Supervised Machine Learning
4. Reinforcement Learning

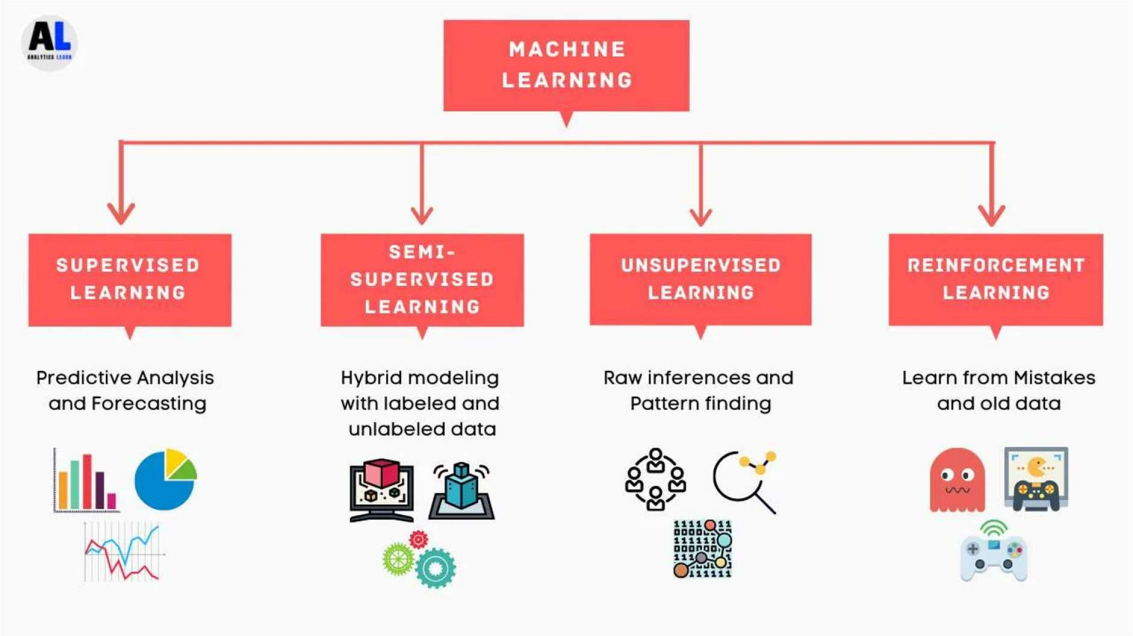


Figure 9 Machine learning

1. Supervised Machine Learning :

Supervised Machine Learning is defined as when a model gets trained on a **"Labelled Dataset"**. Labeled datasets have both input and output parameters. In **Supervised Learning** algorithms learn to map points between inputs and correct outputs. It has both training and validation datasets labeled.

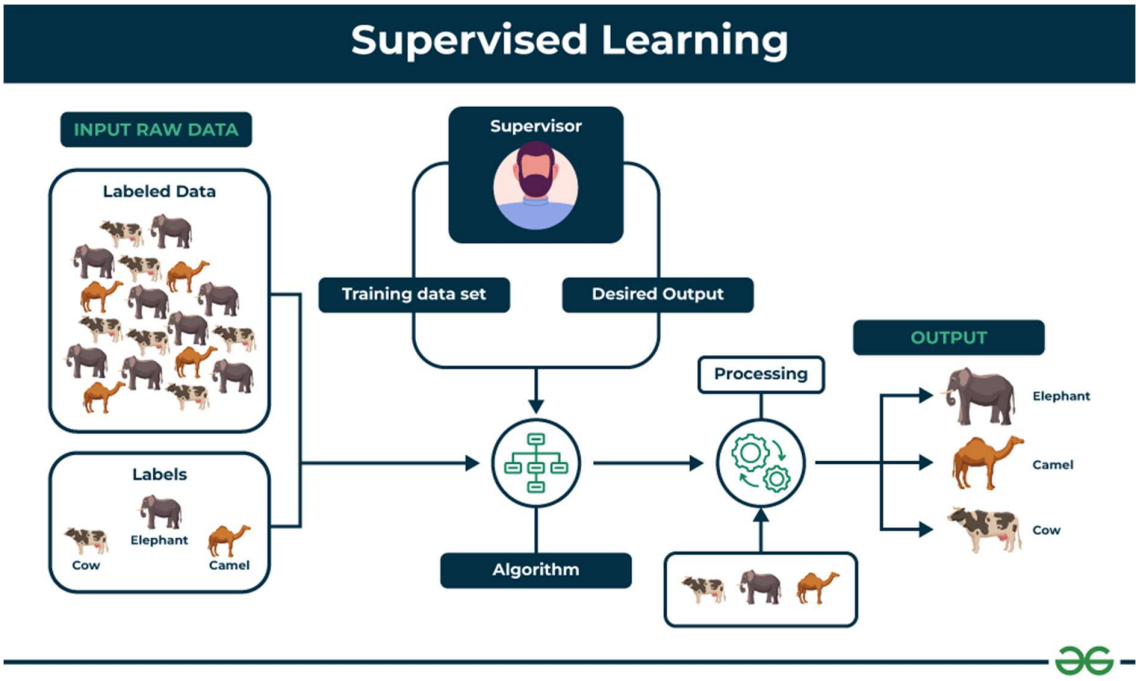


Figure 10 Supervised Learning

There are two main categories of supervised learning that are mentioned below:

- Classification
- Regression

Classification :

Classification deals with predicting **categorical** target variables, which represent discrete classes or labels. For instance, classifying emails as spam or not spam, true/false, male/female, yes/no, first/second/third, etc

Regression:

Regression, on the other hand, deals with predicting **continuous** target variables, which represent numerical values. For example, predicting the price of a house based on its size, location, and amenities, or forecasting the sales of a product, etc.

2. Unsupervised Machine Learning

Unsupervised Learning Unsupervised learning is a type of machine learning technique in which an algorithm discovers patterns and relationships using unlabeled data. Unlike supervised learning, unsupervised learning doesn't involve providing the algorithm with labeled target outputs. The primary goal of Unsupervised learning is often to discover hidden patterns, similarities, or clusters within the data, which can then be used for various purposes, such as data exploration, visualization, dimensionality reduction, and more.

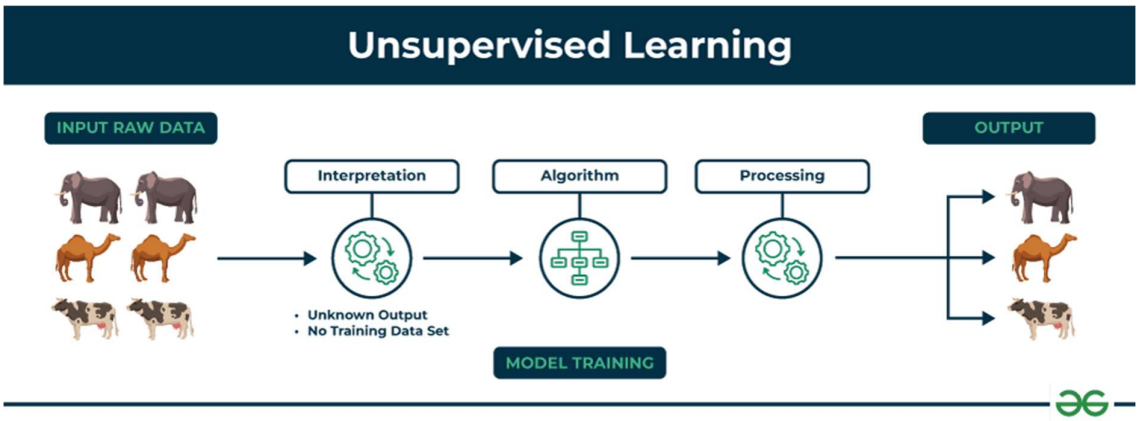


Figure 11 Unsupervised learning

There are two main categories of unsupervised learning that are mentioned below:

- Clustering
- Association

Clustering :

Clustering is the process of grouping data points into clusters based on their similarity. This technique is useful for identifying patterns and relationships in data without the need for labelled examples.

Association :

Association rule learning is a technique for discovering relationships between items in a dataset. It identifies rules that indicate the presence of one item implies the presence of another item with a specific probability.

3.2.2 Classification :

- Classification is a process of categorizing data or objects into predefined classes or categories based on their features or attributes.
- Machine Learning classification is a type of supervised learning technique where an algorithm is trained on a labeled dataset to predict the class or category of new, unseen data.
- The main objective of classification machine learning is to build a model that can accurately assign a label or category to a new observation based on its features.
- For example, a classification model might be trained on a dataset of images labeled as either dogs or cats and then used to predict the class of new, unseen images of dogs or cats based on their features such as color, texture, and shape.

Classification Types

There are two main classification types in machine learning:

- **Binary Classification**

  In binary classification, the goal is to classify the input into one of two classes or categories. Example – On the basis of the given health conditions of a person, we have to determine whether the person has a certain disease or not.

Multiclass Classification

In multi-class classification, the goal is to classify the input into one of several classes or categories. For Example – On the basis of data about different species of flowers, we have to determine which species our observation belongs to.

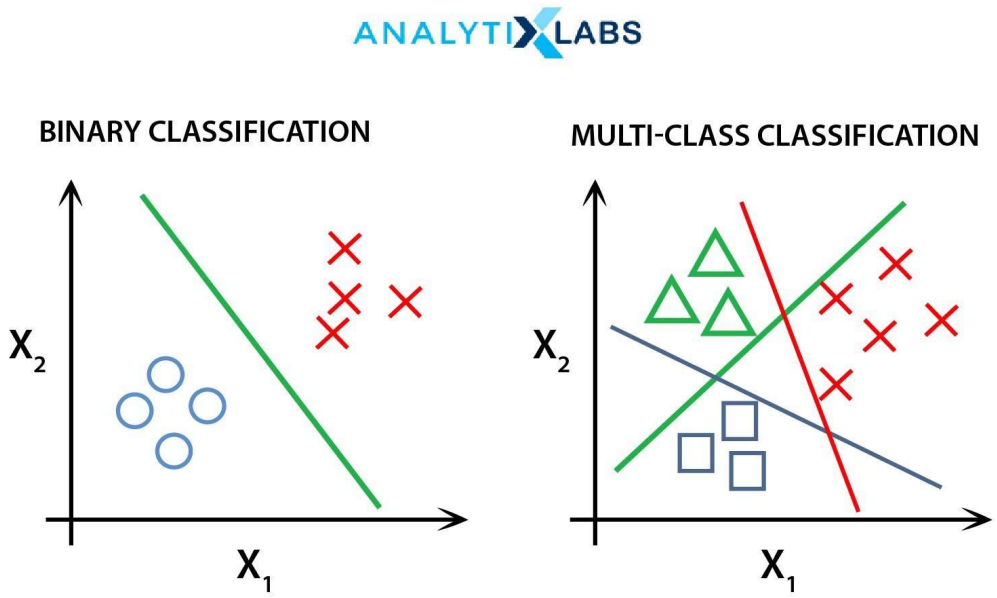


Figure 12 Types of classification

KNN [K-NEAREST NEIGHBORS] :

- The K-Nearest Neighbors (KNN) algorithm is a supervised machine learning method employed to tackle classification and regression problems.
- The K-Nearest Neighbors (KNN) algorithm operates on the principle of similarity, where it predicts the label or value of a new data point by considering the labels or values of its K nearest neighbors in the training dataset.

STEPS :

1. Choose the value of K.
2. Measure the distance between the test point and training data points using any distance metrics.
3. Now sort the data points according to the distance choose the First K datapoint's label/target and perform aggregation.
4. Aggregation :

   For classification: Take the mode of the target class of the first k samples

For regression: Take the mean of target class values.

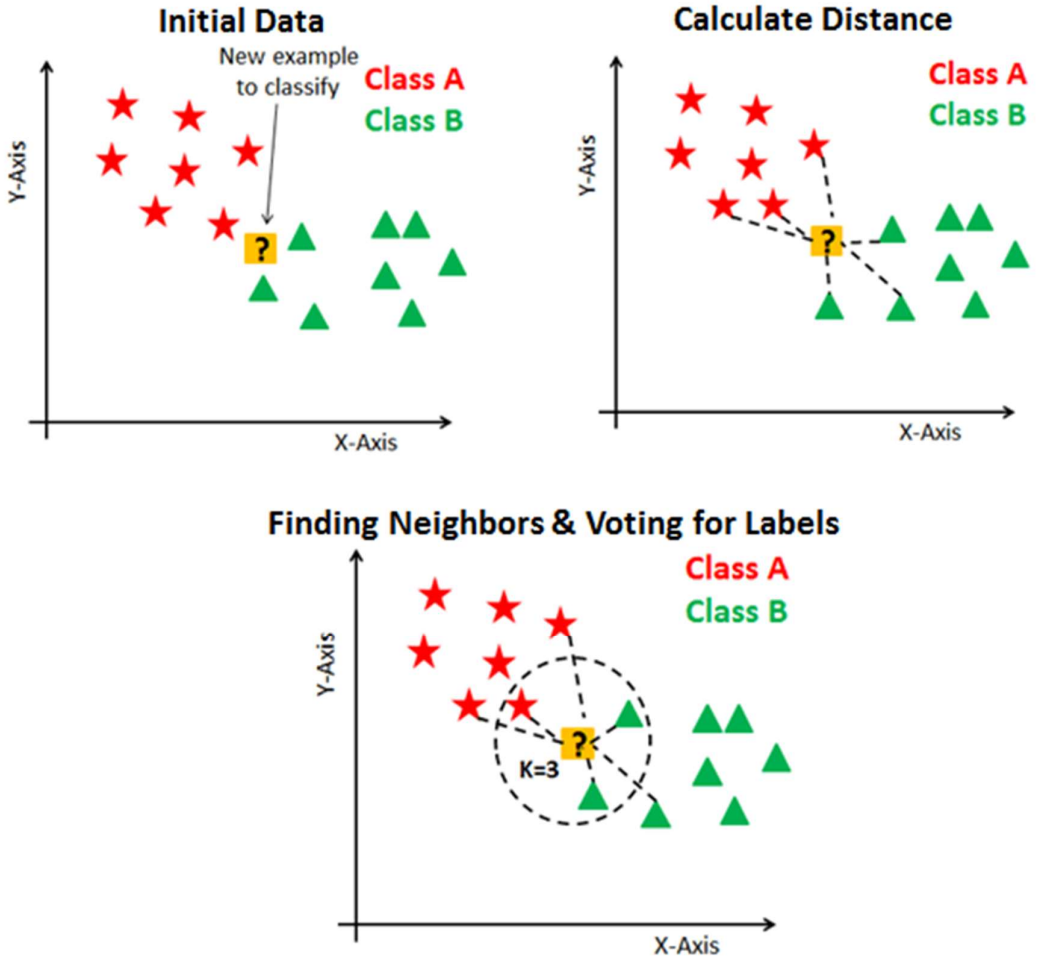


Figure 13 KNN

DECISION TREE :

- A decision tree is a flow chart-like tree structure where each internal mode denotes the feature, branches denote the rules and the leaf node denotes the result of the algorithm. It is used for classification as well as for regression.

- Decision tree uses the tree representation to solve the problem in which each leaf node corresponds to a class label and attributes are represented on the internal node of the tree. We can represent any Boolean function on discrete attributes using the decision tree.

How does it work?

1. Begin the tree with the root node, say S which contains the whole dataset.
2. Find the best attribute in the dataset using the Attribute selection measure Information gain/entropy / gini index/chi-square.
3. Divide the S into subsets say s1,s2,….sn. That contains possible values for the best attributes
4. Generate the decision tree node, which contains the best attribute
5. Recursively make new decision trees using the subsets of the dataset created in step 3

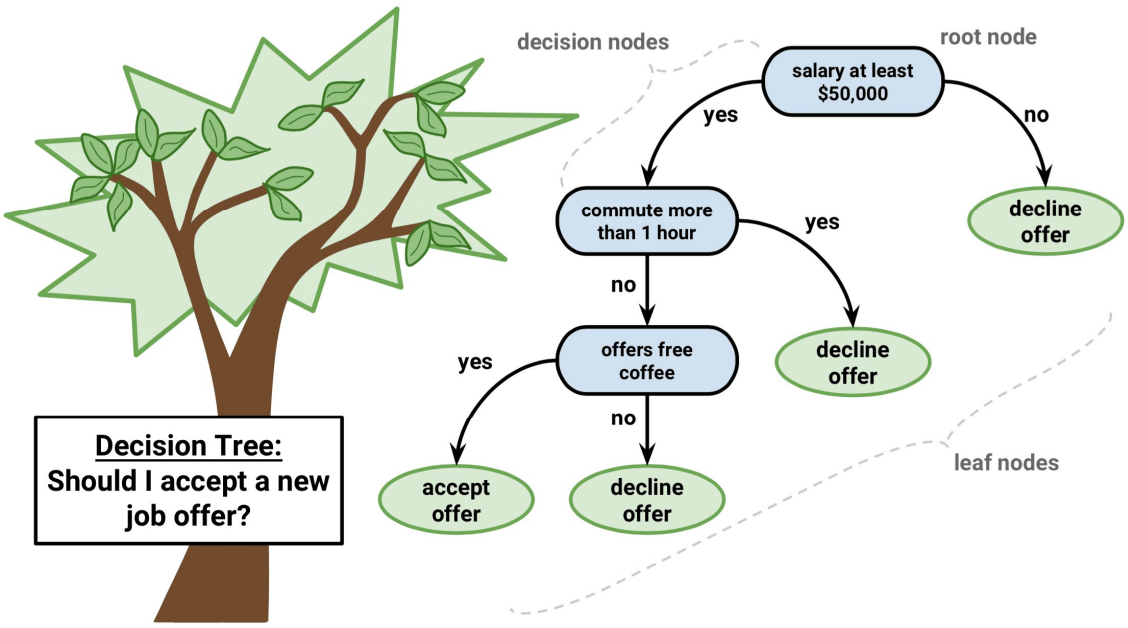


Figure 14 Decision Tree

The major challenge is the identification of the attribute for the root node at each level. This process is known as attribute selection. We have two popular attribute selection measures:

1. **Information Gain**
2. **Gini Index**

Impurity Criterion

Gini Index

$$I_G = 1 - \sum_{j=1}^{c} p_j^2$$

p_j: proportion of the samples that belongs to class c for a particular node

Entropy

$$I_H = -\sum_{j=1}^{c} p_j log_2(p_j)$$

p_j: proportion of the samples that belongs to class c for a particular node.

*This is the the definition of entropy for all non-empty classes ($p \neq 0$). The entropy is 0 if all samples at a node belong to the same class.

Figure 15 Gini & Entropy

$Values(Outlook) = Sunny, Overcast, Rain$
$S = [9+, 5-]$
$S_{\text{sunny}} = [2+, 3-]$
$S_{\text{overcast}} = [4+, 0-]$
$S_{\text{rain}} = [3+, 2-]$

$$G(S, Outlook) = Entropy(S) - (\frac{5}{14} Entropy(S_{\text{sunny}}) + \frac{4}{14} Entropy(S_{\text{overcast}}) + \frac{5}{14} Entropy(S_{\text{rain}})) \quad (1.10)$$

$$Entropy(S_{\text{sunny}}) = -(\frac{2}{5} log_2 \frac{2}{5} + \frac{3}{5} log_2 \frac{3}{5}) = 0.971 \quad (1.11)$$

$$Entropy(S_{\text{overcast}}) = -(\frac{4}{4} log_2 \frac{4}{4} + \frac{0}{4} log_2 \frac{0}{4}) = 0 \quad (1.12)$$

$$Entropy(S_{\text{rain}}) = -(\frac{3}{5} log_2 \frac{3}{5} + \frac{2}{5} log_2 \frac{2}{5}) = 0.971 \quad (1.13)$$

Put the values of eqtn 1.11, 1.12, 1.13 in 1.10.

Figure 16 Entropy

SVM [SUPPORT VECTOR MACHINE] :

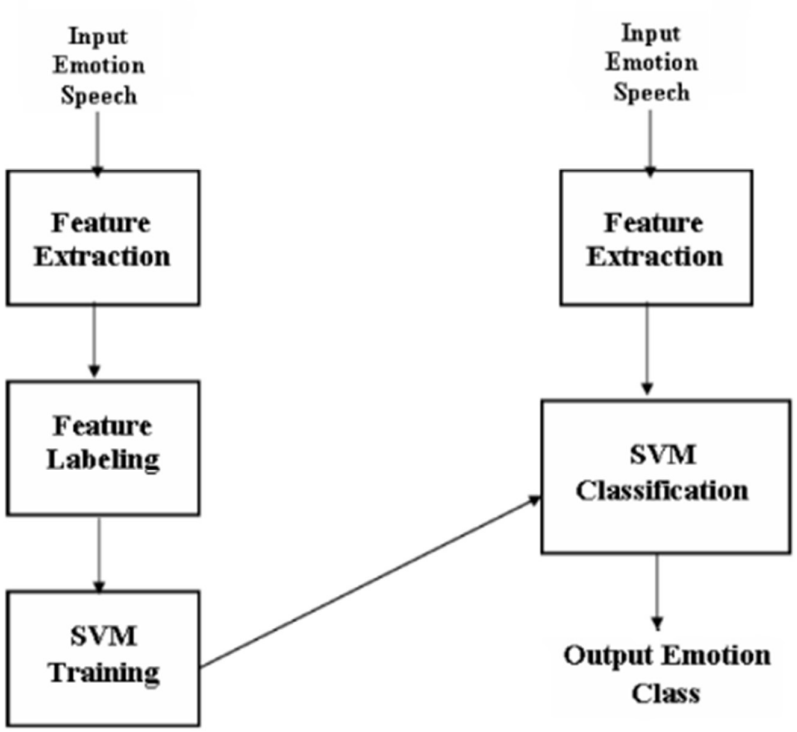


Figure 17 SVM

- Support Vector Machine (SVM) is a supervised machine learning algorithm used for both classification and regression. It uses a hyperplane to separate two classes, the class above the hyperplane is a positive class and the class below the hyperplane is said to be a negative class.
- Our main objective in svm is to maximize the margin of the hyperplane and then that hyperplane is said to be the best/optimal hyperplane.

Hyperplane and Support Vectors in the SVM algorithm:

Hyperplane:

There can be multiple lines/decision boundaries to segregate the classes in n-dimensional space, but we need to find out the best decision boundary that helps to classify the data points. This best boundary is known as the hyperplane of SVM.

The dimensions of the hyperplane depend on the features present in the dataset, which means if there are 2 features (as shown in the image), then the hyperplane will be a straight line. And if there are 3 features, then the hyperplane will be a 2-dimension plane.

We always create a hyperplane that has a maximum margin, which means the maximum distance between the data points.

Support Vectors:

The data points or vectors that are the closest to the hyperplane and which affect the position of the hyperplane are termed as Support Vector. Since these vectors support the hyperplane, hence called a Support vector.

Types Of Support Vector Machine

a. Linear SVM

Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed linearly separable data, and classifier is used called as Linear SVM classifier.

The working of the SVM algorithm can be understood by using an example. Suppose we have a dataset that has two tags (green and blue), and the dataset has two features x1 and x2. We want a classifier that can classify the pair(x1, x2) of coordinates in either green or blue. Consider the below image:

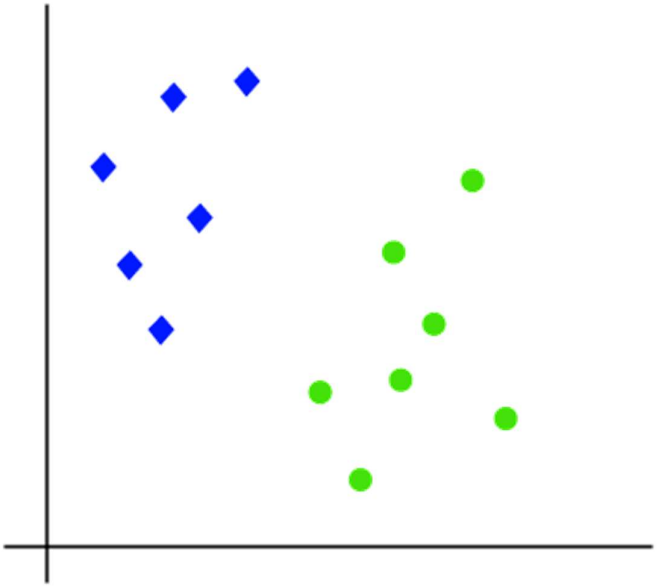


Figure 18 Linear SVM

So as it is 2-d space so by just using a straight line, we can easily separate these two classes. But there can be multiple lines that can separate these classes. Consider the below image:

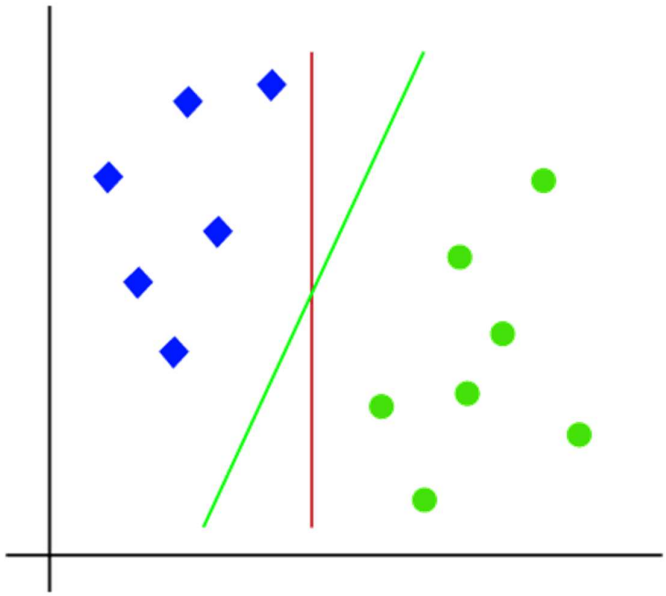


Figure 19 Non-Linear SVM

Hence, the SVM algorithm helps to find the best line or decision boundary; this best boundary or region is called as a **hyperplane**. SVM algorithm finds the closest point of the lines from both the classes. These points are called support vectors. The distance between the vectors and the hyperplane is called as **margin**. The goal of SVM is to maximize this margin. The **hyperplane** with the maximum margin is called the **optimal hyperplane**.

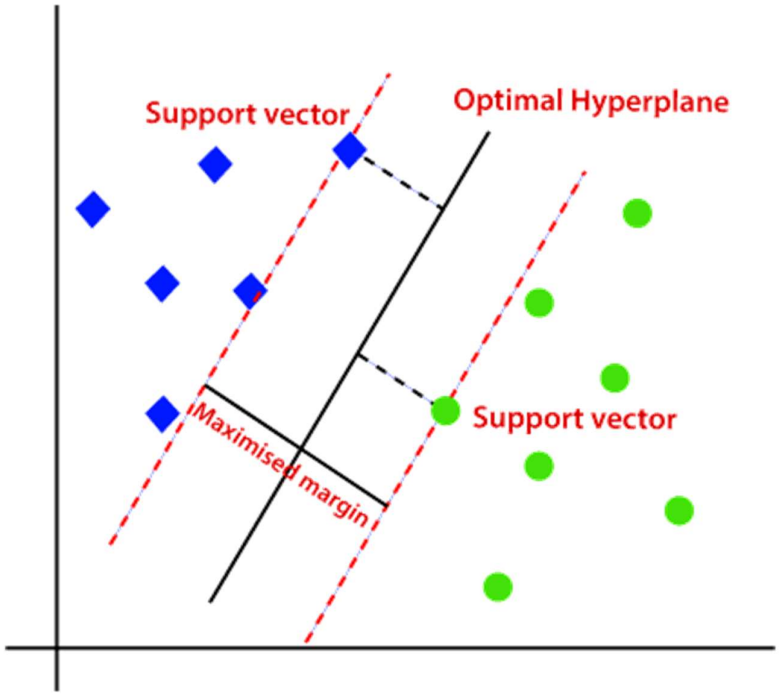


Figure 20 Hyper plane

b. Non-Linear SVM

If data is linearly arranged, then we can separate it by using a straight line, but for non-linear data, we cannot draw a single straight line. Consider the below image:

So to separate these data points, we need to add one more dimension. For linear data, we have used two dimensions x and y, so for non-linear data, we will add a third dimension z. It can be calculated as:

$$z=x^2+y^2$$

By adding the third dimension, the sample space will become as below image :

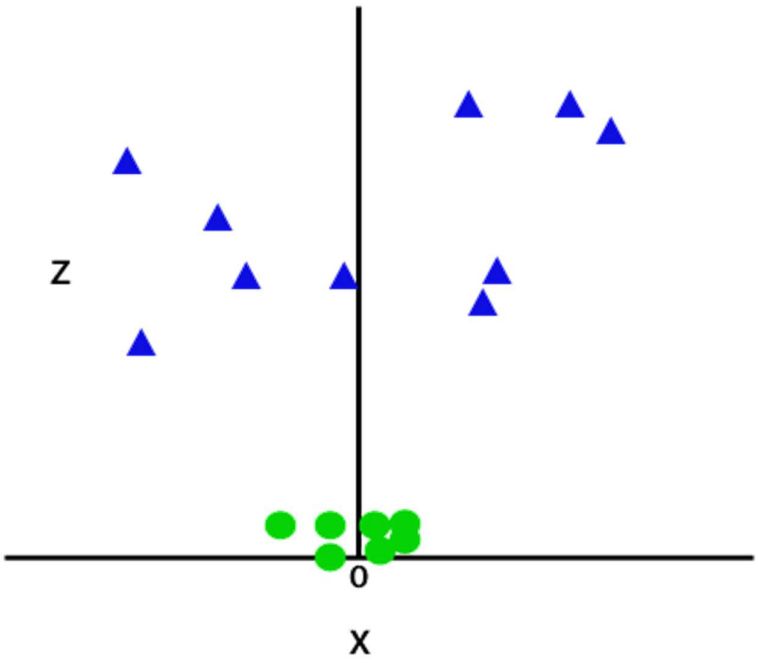


Figure 21 Hyper plane-2

So now, SVM will divide the datasets into classes in the following way. Consider the below image:

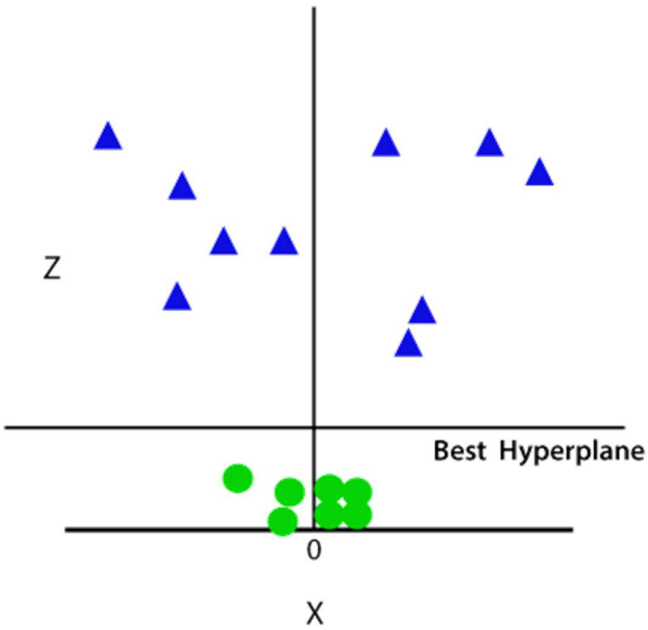


Figure 22 Hyper plane-3

Since we are in 3-D Space, hence it is looking like a plane parallel to the x-axis. If we convert it in 2d space with z=1, then it will become as:

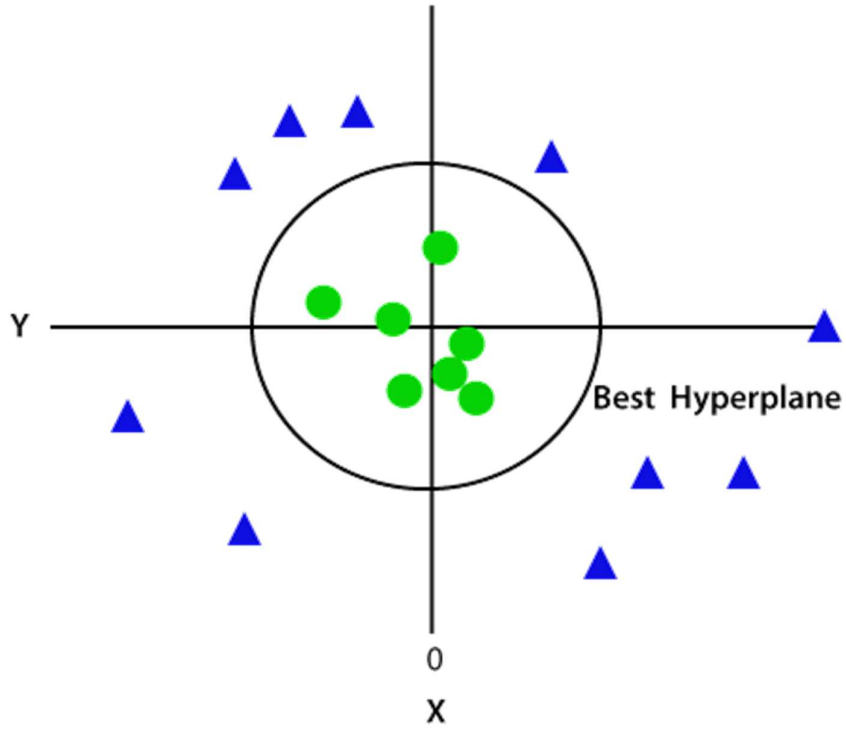


Figure 23 Hyper plane-4

Hence we get a circumference of radius 1 in the case of non-linear data.

Hyperparameters

Hyperparameters play a key role in machine learning algorithms such as Support Vector Machines (SVM). We begin with an introduction to hyperparameters and then focus specifically on SVM hyperparameters.

Introduction to hyperparameters:

Hyperparameters are parameters that are set before the learning process begins. They control aspects of the learning algorithm and have a significant effect on model performance. Unlike model parameters (such as weights and biases), which are learned during training, hyperparameters are set by a data scientist or machine learning engineer. The goal is to optimize these hyperparameters to improve the ability of the model to generalize to new, unseen data.

Kernel trick in SVM :

The SVM kernel is a function that takes low-dimensional input space and transforms it into higher-dimensional space, ie it converts nonseparable problems to separable problems. It is mostly useful in non-linear separation problems.

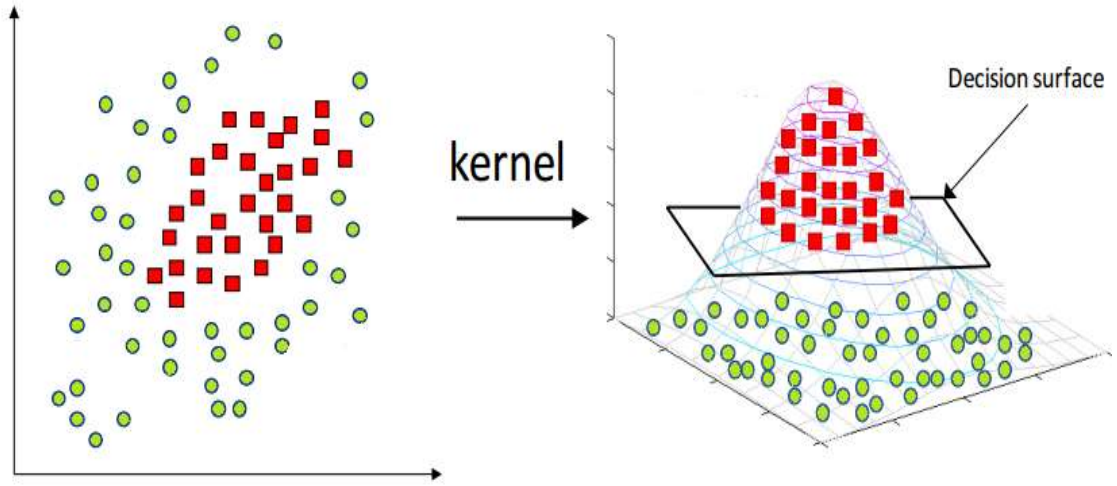


Figure 24 Kernel-Trick

Some of the useful kernels of SVM:

$$\text{Linear}: K(w,b) = w^T x + b$$
$$\text{Polynomial}: K(w,x) = (\gamma w^T x + b)^N$$
$$\text{Gaussian RBF}: K(w,x) = \exp(-\gamma ||x_i - x_j||^n$$
$$\text{Sigmoid}: K(x_i, x_j) = \tanh(\alpha x_i^T x_j + b)$$

Hyperparameters in SVM

Support Vector Machines has several hyperparameters that can be adjusted to adjust the behavior of the model. Here's a detailed look at each hyperparameter:

a. Type of kernel (`kernel`):

SVM uses various kernel functions to transform the input space into a higher dimensional feature space where the data points can be linearly separated. Common core types include:

- Linear: \(K(x, x') = x^T x' \) - Suitable for linearly separable data.

- Polynomial: \(K(x, x') = (1 + x^T x')^d \) - Controlled by `degree` parameter.

- Radial basis function (RBF): \(K(x, x') = \exp(-\gamma \|x - x'\|^2) \) - Controlled by `gamma` parameter.

- Sigmoid: \(K(x, x') = \tanh(\gamma x^T x' + r) \) - Controlled by parameters `gamma` and `coef0`.

b. Regularization parameter ("C"):

"C" controls the trade-off between maximizing the margin and minimizing the classification error in the training data. A large "C" penalizes the model more for misclassifications, aiming to fit the training data as closely as possible (potentially leading to overfitting). Conversely, a lowercase "C" emphasizes a wider margin and focuses more on generalization.

c. Kernel coefficient (`gamma`):

This parameter defines the influence of one training example. It determines the range of the kernel function and affects the fluidity of the decision boundary. A high "gamma" value means that the points must be very close to be considered similar, resulting in a more complex decision boundary (potentially offset). Conversely, a low `gamma' value means a smoother decision boundary.

Tuning hyperparameters:

Hyperparameter tuning techniques can be used to determine the best set of hyperparameters for an SVM model. The most common methods include:

a. Search in the grid:

Evaluates all possible combinations of hyperparameter values within a predefined grid of values. This method is exhaustive but can be computationally expensive.

b. Random Search:

Randomly samples combinations of hyperparameters from predefined distributions. It is less computationally intensive than a grid search and can sometimes discover better combinations faster.

c. Bayesian optimization:

Uses probabilistic models to predict which hyperparameter values are likely to lead to the best performance. This method efficiently explores the hyperparameter space and can quickly converge to optimal values.

Hyperparameter tuning typically involves splitting the dataset into training, validation, and test sets. The training set is used to train the model with different hyperparameter configurations, the validation set is used to evaluate the performance of the model and select the best hyperparameters, and the test set is used to assess the performance of the final model.

By fine-tuning SVM hyperparameters, you can optimize model performance, improve generalization ability, and make the model robust to unseen data. Experimentation with different hyperparameter settings is necessary to find the optimal configuration for a particular data set and task.

Challenges

Support Vector Machines (SVMs) are powerful supervised learning models used for classification and regression tasks. Here are the main advantages and disadvantages of SVM:

Pros:

1. Efficient in high-dimensional spaces SVM performs well even when the number of dimensions is larger than the number of samples. This makes it suitable for tasks such as text classification or image recognition where the feature space can be very large.

2. Robust to outliers: SVMs are effective in handling outliers due to the use of a margin that is inherently less affected by individual data points that are far from the decision boundary.

3. Memory efficient: SVMs use a subset of training points (support vectors) in the decision function, so they are memory efficient especially when working with large datasets.

4. Versatile kernels: SVMs can model non-linear decision boundaries using different kernel functions (e.g. polynomial, radial basis function), allowing flexibility in capturing complex relationships in data.

5. Global optimum: The SVM (convex optimization) objective function leads to a unique solution, so it typically avoids local minima and finds the global optimum.

6. Regularization: SVMs have a regularization parameter (C) that helps prevent overfitting by penalizing large coefficients, thereby promoting a simpler model.

Cons:

1. Computationally intensive: Training SVMs on large datasets can be time-consuming. The time complexity is roughly O(n^2) to O(n^3) for training, making it impractical for very large datasets.

2. Sensitivity to feature scaling: SVMs are sensitive to feature scaling. Before using SVM, it is essential to properly scale the features to avoid biased results.

3. Kernel selection: Choosing an appropriate kernel and tuning its parameters (such as regularization parameter C and kernel parameters) can be challenging and requires domain knowledge or experimentation.

4. Memory consumption: While SVMs are memory efficient during testing (because they only use support vectors), they can consume a significant amount of memory during training, especially for large datasets.

5. Limited interpretability: The SVM decision function can be difficult to interpret directly, especially when non-linear kernels are used, which can make the model less transparent compared to simpler models such as logistic regression.

6. Binary Classification: SVM is inherently a binary classifier. For multi-class classification tasks, techniques such as one-against-rest or one-against-one are commonly used, which can increase complexity.

Overall, SVMs are effective for many machine learning tasks, especially when working with small to medium-sized datasets with complex decision boundaries. However, they have limitations, especially with regard to scalability and the need for careful parameter tuning.

Important Parameters in Kernelized SVC:

The Kernel: The kernel, is selected based on the type of data and also the type of transformation. By default, the kernel is the Radial Basis Function Kernel (RBF).

Gamma: This parameter decides how far the influence of a single training example reaches during transformation, which in turn affects how tightly the decision boundaries end up surrounding points in the input space. If there is a small value of gamma, points farther apart are considered similar. So more points are grouped and have smoother decision boundaries (may be less accurate). Larger values of gamma cause points to be closer together (may cause overfitting).

The 'C' parameter: This parameter controls the amount of regularization applied to the data. Large values of C mean low regularization which in turn causes the training data to fit very well (may cause overfitting). Lower values of C mean higher regularization which causes the model to be more tolerant of errors (may lead to lower accuracy)

RANDOM FOREST :

- Random Forest algorithm is a powerful tree-learning technique in Machine Learning. It works by creating several Decision Trees during the training phase.

- Each tree is constructed using a random subset of the data set to measure a random subset of features in each partition. This randomness introduces variability among individual trees, reducing the risk of overfitting and improving overall prediction performance.
- In prediction, the algorithm aggregates the results of all trees, either by voting (for classification tasks) or by averaging (for regression tasks) This collaborative decision-making process, supported by multiple trees with their insights, provides an example of stable and precise results.
- Random forests are widely used for classification and regression functions, which are known for their ability to handle complex data, reduce overfitting, and provide reliable forecasts in different environments.

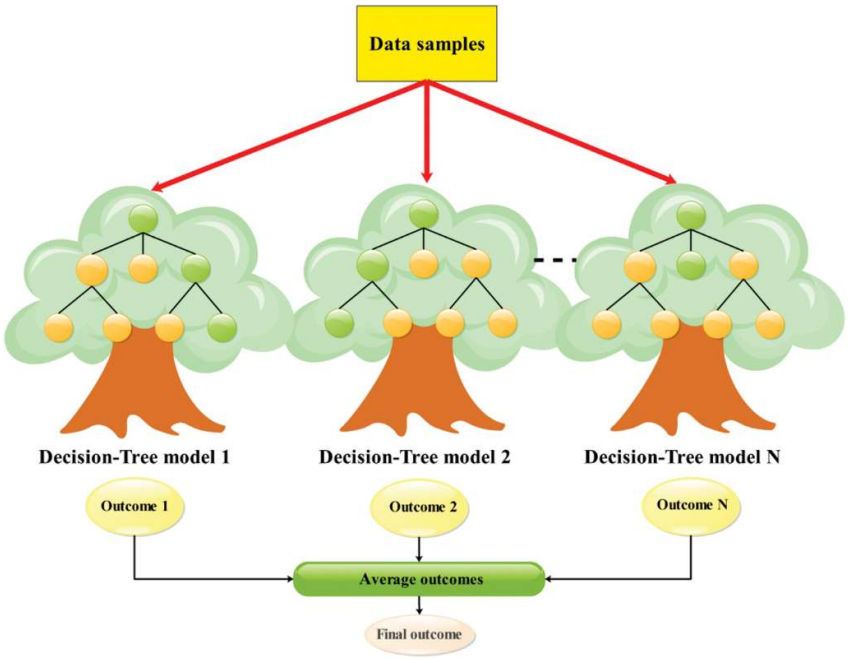


Figure 25 Random Forest

- Random Forest employs a technique called bootstrap aggregating or bagging, where each decision tree is trained on a random subset of the training data with replacement.
- During prediction, Random Forest combines the predictions from individual decision trees through a voting mechanism (for classification) or averaging (for regression).

It is the collection of the decision tree in which the decision trees get trained through bootstrapping of the dataset.

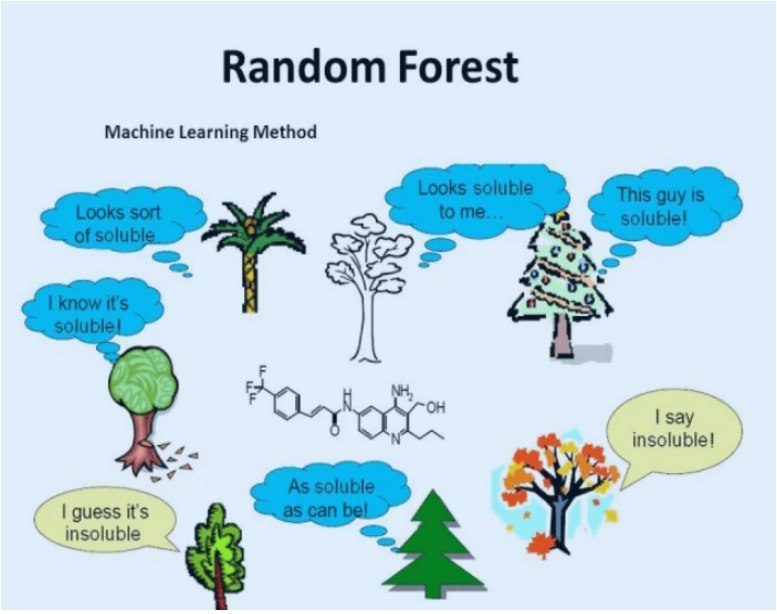


Figure 26 Random Forest -2

Why use Random Forest?

- It takes less training time as compared to other algorithms.
- It predicts output with high accuracy, even for the large dataset it runs efficiently.
- It can also maintain accuracy when a large proportion of data is missing.

How does the algorithm work?

It works in four steps:

1. Select random samples from a given dataset.
2. Construct a decision tree for each sample and get a prediction result from each decision tree.
3. Perform a vote for each predicted result.

Select the prediction result with the most votes as the final prediction

NAÏVE BAYES :

- Naïve Bayes algorithm is used for classification problems. It is highly used in text classification. In text classification tasks, data contains high dimensions (as each word represents one feature in the data).
- It is used in spam filtering, sentiment detection, rating classification, etc. The advantage of using naïve Bayes is its speed. It is fast and making predictions is easy with high dimensions of data.
- This model predicts the probability of an instance belonging to a class with a given set of feature values. It is a probabilistic classifier. This is because it assumes that one feature in the model is independent of the existence of another feature.
- In other words, each feature contributes to the predictions with no relation between each other. In the real world, this condition is rare. It uses Bayes theorem in the algorithm for training and prediction

Bayes' Theorem

- Bayes' Theorem finds the probability of an event occurring given the probability of another event that has already occurred. Bayes' theorem is stated mathematically as the following equation:

$$P(A|B)=P(B|A)P(A)P(B)P(A|B)=P(B)P(B|A)P(A)$$

where A and B are events and P(B) ≠ 0

- Now, with regard to our dataset, we can apply Bayes' theorem in the following way:

$$P(y|X)=P(X|y)P(y)P(X)P(y|X)=P(X)P(X|y)P(y)$$

where y is the class variable and X is a dependent feature vector (of size n)

$$X=(x1,x2,x3,…..,xn)X=(x1,x2,x3,…..,xn)$$

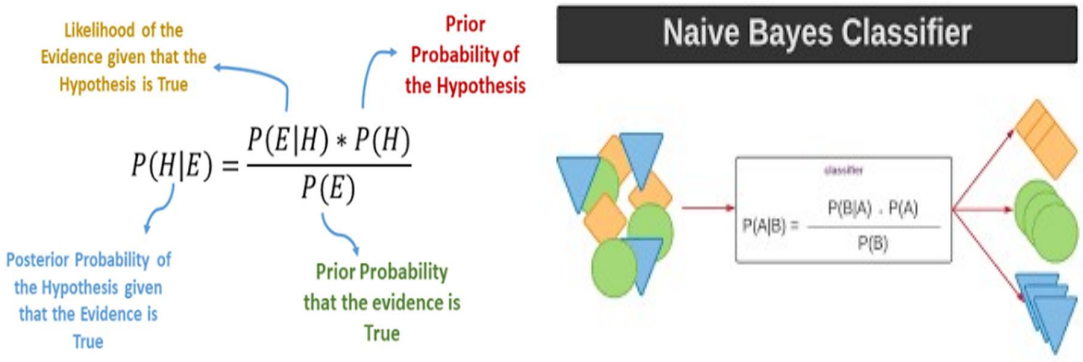


Figure 27 Naive Bayes

Types of Naive Bayes Model

There are three types of Naive Bayes Model:

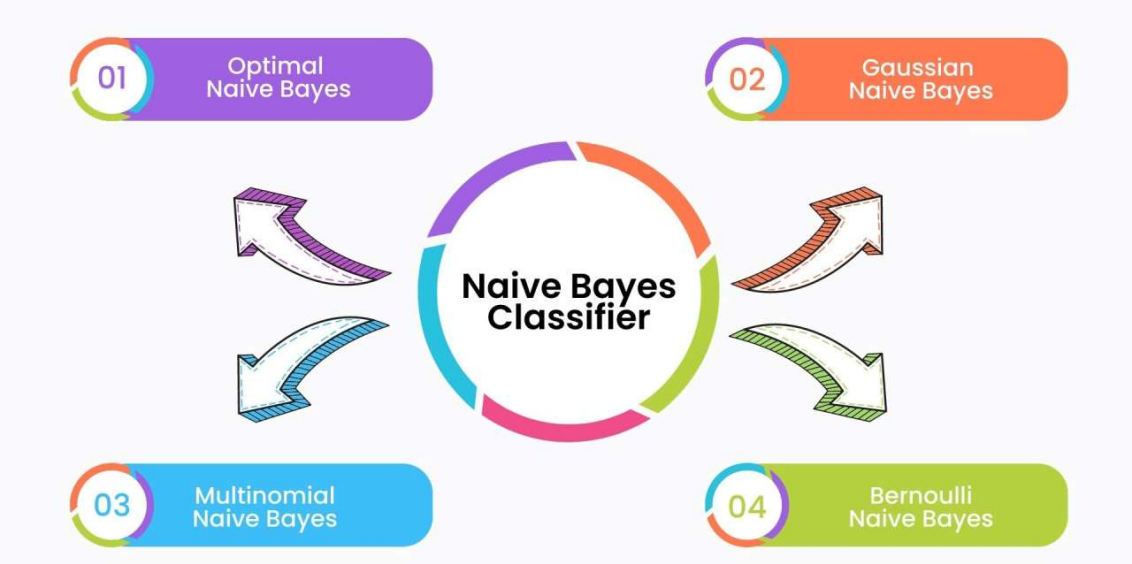


Figure 28 Types of Naive Bayes

ENSEMBLE LEARNING :

- Ensemble learning is a machine learning technique that enhances accuracy and resilience in forecasting by merging predictions from multiple models. It aims to mitigate errors or biases that may exist in individual models by leveraging the collective intelligence of the ensemble.

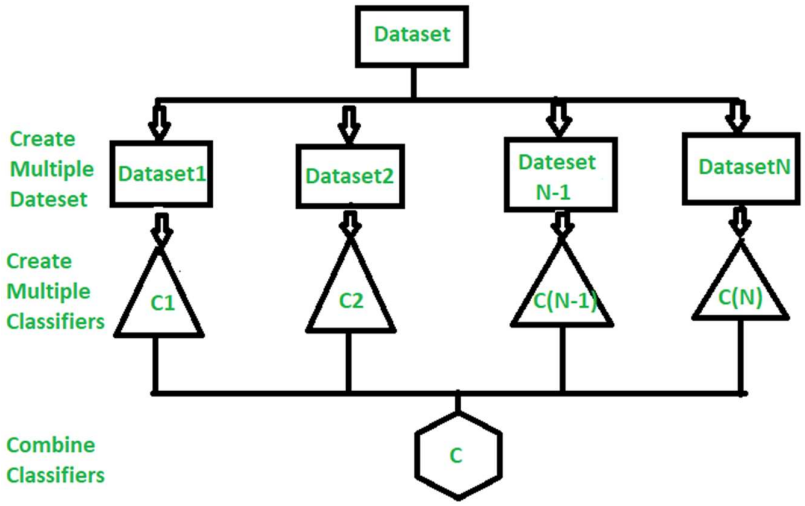


Figure 29 Ensemble Learning

Ensemble algorithms or methods can be divided into two groups:

- Sequential ensemble methods where the base learners are generated sequentially (e.g. AdaBoost). The basic motivation of sequential methods is to exploit the dependence between the base learners. The overall performance can be boosted by weighing previously mislabeled examples with higher weight.

- Parallel ensemble methods where the base learners are generated in parallel (e.g. Random Forest). The basic motivation of parallel methods is to exploit independence between the base learners since the error can be reduced dramatically by averaging.

BAGGING :

Bagging is an ensemble learning technique, it is the process of combining the result of multiple models (with the same algorithm) to get a generalized result.

- **BOOTSTRAPPING :**

It is a sampling technique in which we create subsets of observations from the original Dataset, with replacement

- **AGGREGATION :**

Classification: Mode of all the predicted classes.

Regression: Average of all the predicted values.

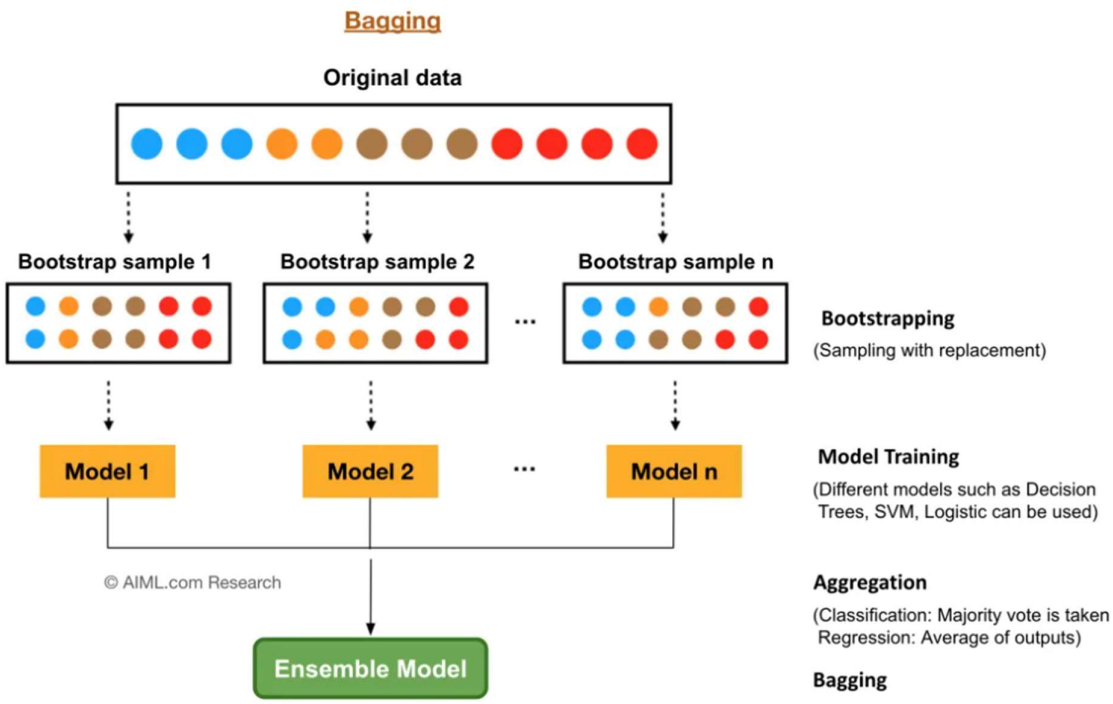


Figure 30 Bagging

BOOSTING :

- Boosting is an ensemble modeling technique that attempts to build a strong classifier from the number of weak classifiers. It is done by building a model by using weak models in series.
- Firstly, a model is built from the training data. Then the second model is built which tries to correct the errors present in the first model.
- This procedure is continued and models are added until either the complete training data set is predicted correctly or the maximum number of models are added.

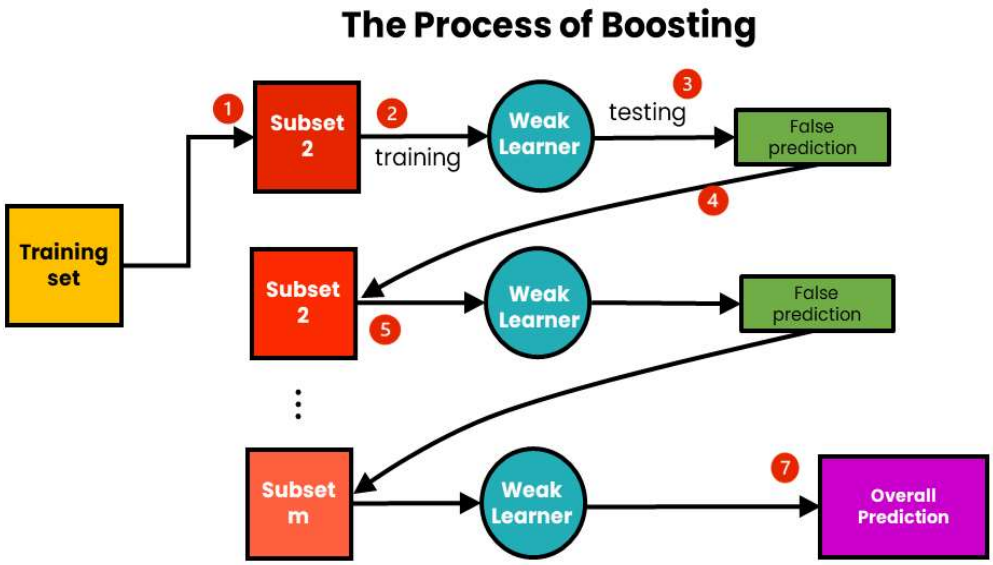


Figure 31 Boostingss

***How Boosting works* :**

- Here D1,D2,D3 are decision stumps(decision tree model with Depth=1). D4 is the final model trained from misclassified as well as Classified points of weak learners.
- **Types of Boosting Algorithms** :
    1. **Adaboost**
    2. **Gradient Boosting**
    3. **XGBoost**

- It is a parallel ensemble method in which several weak learners learn from the mistakes/misclassifications/errors of previous weak learners. The next weak learner tries to avoid the previous mistakes and in this way, it works to get an optimal model.

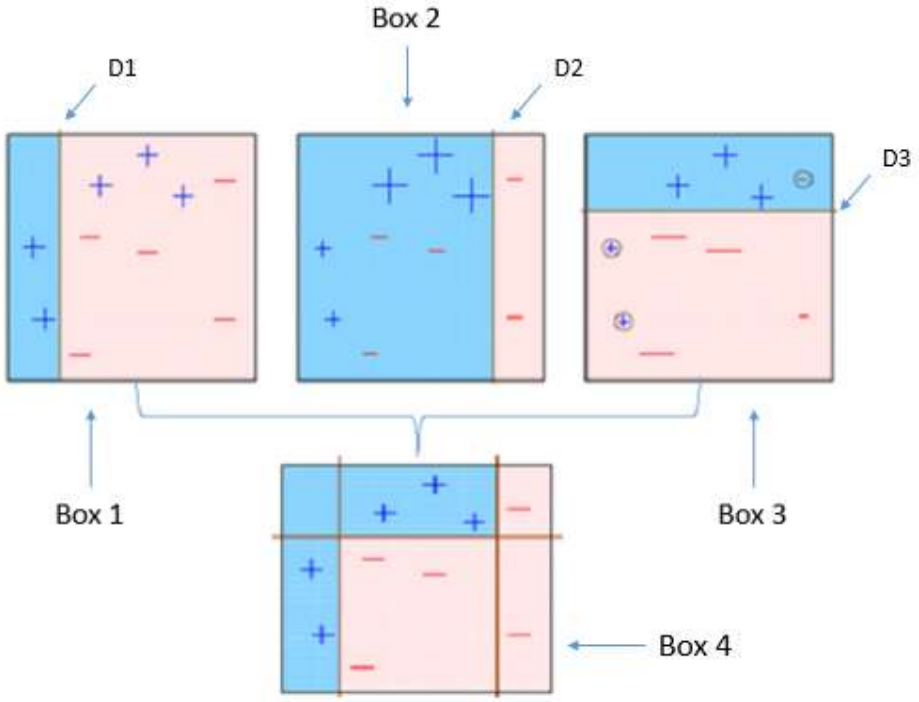


Figure 32 Boosting work

ADABOOST :

- AdaBoost is a boosting algorithm that also works on the principle of the stagewise addition method where multiple weak learners are used for getting strong learners.

- The value of the alpha parameter, in this case, will be indirectly proportional to the error of the weak learners**.**

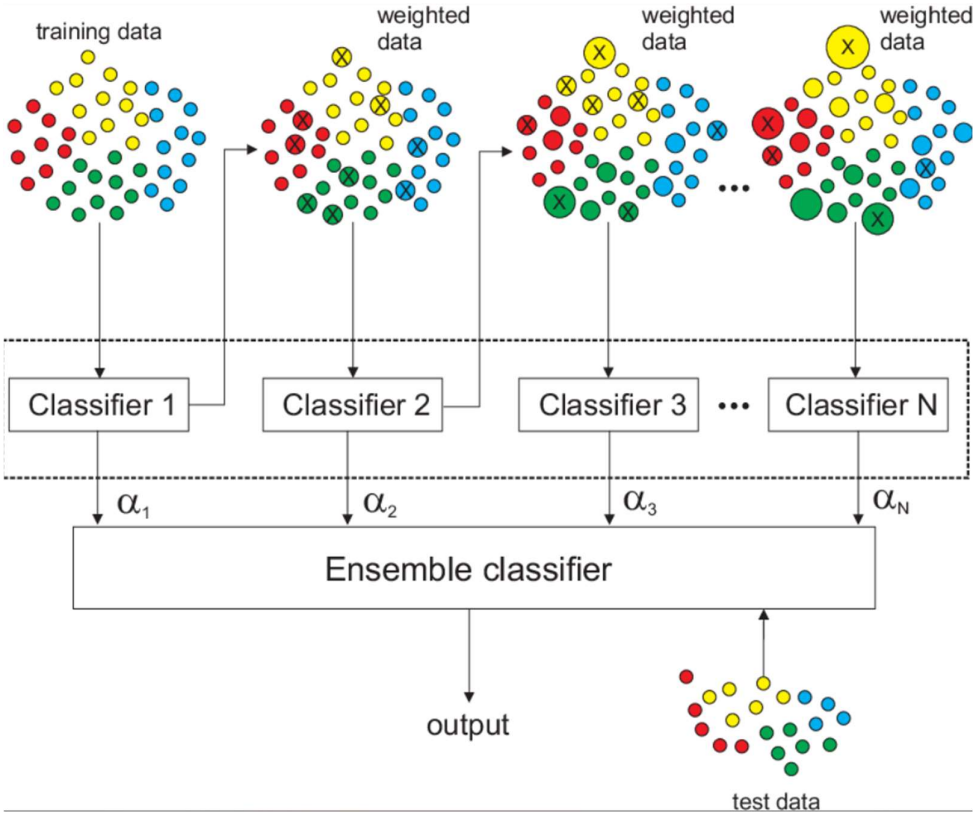


Figure 33 Adaboost

How Adaboost works :

1. Assign initial weights (1/n) to each datapoint.
2. Train a weak learner on the dataset and calculate the error and Alpha

   ERROR = sum of weights of misclassified points

3. Now assign the new weights to all points:

   for misclassified point : $new_{wt} = old_{wt} . e^{\alpha}$

   for classified point : $new_{wt} = old_{wt} . e^{-\alpha}$

4. Normalize the weights in such a way that $\sum \boldsymbol{wts} = \mathbf{1}$.

5. Now make a range according to weights and perform upsampling using new weights.

6. Repeat step 2 again for new weak learners.

- In this manner in the last we have n models and for any unknown pt, we ask each model and we give importance to the answer of each model according to their value of $\boldsymbol{\alpha}$.

$$\mathbf{H(x)=\alpha_1 h(x1) + \alpha_2 h(x2) + \alpha_3 h(x3) . . . . . + \alpha_n h(xn)}$$

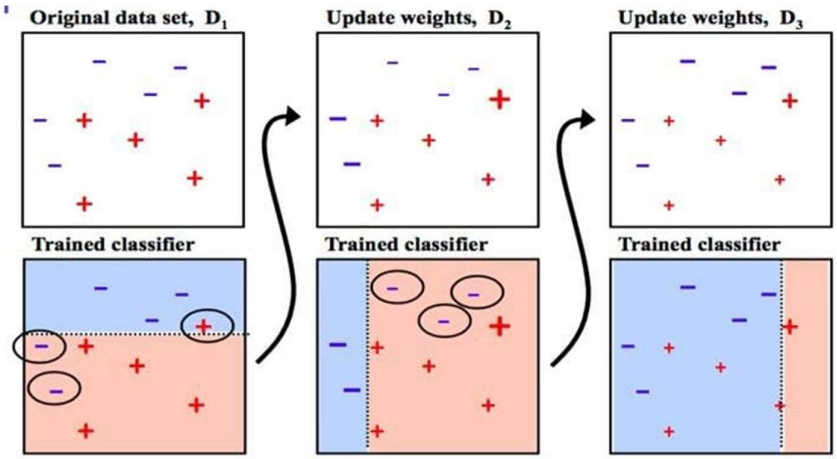


Figure 34 Adaboost work

3.3 Deep Learning

Introduction :

In the fast-evolving era of artificial intelligence, Deep Learning stands as a cornerstone technology, revolutionizing how machines understand, learn, and interact with complex data. At its essence, Deep Learning AI mimics the intricate neural networks of the human brain, enabling computers to autonomously discover patterns and make decisions from vast amounts of unstructured data. This transformative field has propelled breakthroughs across various domains, from computer vision and natural language processing to healthcare diagnostics and autonomous driving.

As we dive into this introductory exploration of Deep Learning, we uncover its foundational principles, applications, and the underlying mechanisms that empower machines to achieve human-like cognitive abilities. This article serves as a gateway into understanding how Deep Learning is reshaping industries, pushing the boundaries of what's possible in AI, and paving the way for a future where intelligent systems can perceive, comprehend, and innovate autonomously.

What is Deep Learning?

The definition of Deep learning is that it is the branch of machine learning that is based on artificial neural network architecture. An artificial neural network or ANN uses layers of interconnected nodes called neurons that work together to process and learn from the input data.

In a fully connected Deep neural network, there is an input layer and one or more hidden layers connected one after the other. Each neuron receives input from the previous layer neurons or the input layer. The output of one neuron becomes the input to other neurons in the next layer of the network, and this process continues until the final layer produces the output of the network. The layers of the neural network transform the input data through a series of nonlinear transformations, allowing the network to learn complex representations of the input data.

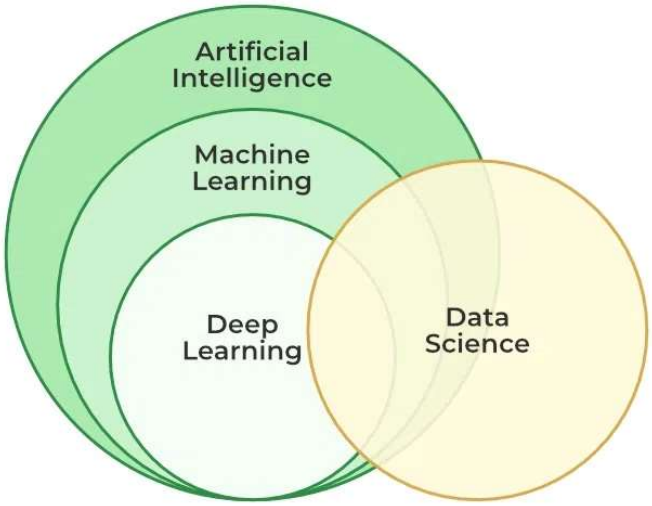


Figure 35 Deep Learning

Today Deep learning AI has become one of the most popular and visible areas of machine learning, due to its success in various applications, such as computer vision, natural language processing, and Reinforcement learning.

Deep learning AI can be used for supervised, unsupervised as well as reinforcement machine learning. it uses a variety of ways to process these.

- **Supervised Machine Learning:** Supervised machine learning is the machine learning technique in which the neural network learns to make predictions or classify data based on the labeled datasets. Here we input both input features along with the target variables. the neural network learns to make predictions based on the cost or error that comes from the difference between the predicted and the actual target, this process is known as backpropagation. Deep learning algorithms like Convolutional neural networks, Recurrent neural networks are used for many supervised tasks like image classifications and recognization, sentiment analysis, language translations, etc.

- **Unsupervised Machine Learning:** Unsupervised machine learning is the machine learning technique in which the neural network learns to discover the patterns or to cluster the dataset based on unlabeled datasets. Here there are no

target variables. while the machine has to self-determined the hidden patterns or relationships within the datasets. Deep learning algorithms like autoencoders and generative models are used for unsupervised tasks like clustering, dimensionality reduction, and anomaly detection.

- **Reinforcement Machine Learning**: Reinforcement Machine Learning is the machine learning technique in which an agent learns to make decisions in an environment to maximize a reward signal. The agent interacts with the environment by taking action and observing the resulting rewards. Deep learning can be used to learn policies, or a set of actions, that maximizes the cumulative reward over time. Deep reinforcement learning algorithms like Deep Q networks and Deep Deterministic Policy Gradient (DDPG) are used to reinforce tasks like robotics and game playing etc.

Artificial neural networks:

Artificial neural networks are built on the principles of the structure and operation of human neurons. It is also known as neural networks or neural nets. An artificial neural network's input layer, which is the first layer, receives input from external sources and passes it on to the hidden layer, which is the second layer. Each neuron in the hidden layer gets information from the neurons in the previous layer, computes the weighted total, and then transfers it to the neurons in the next layer. These connections are weighted, which means that the impacts of the inputs from the preceding layer are more or less optimized by giving each input a distinct weight. These weights are then adjusted during the training process to enhance the performance of the model.

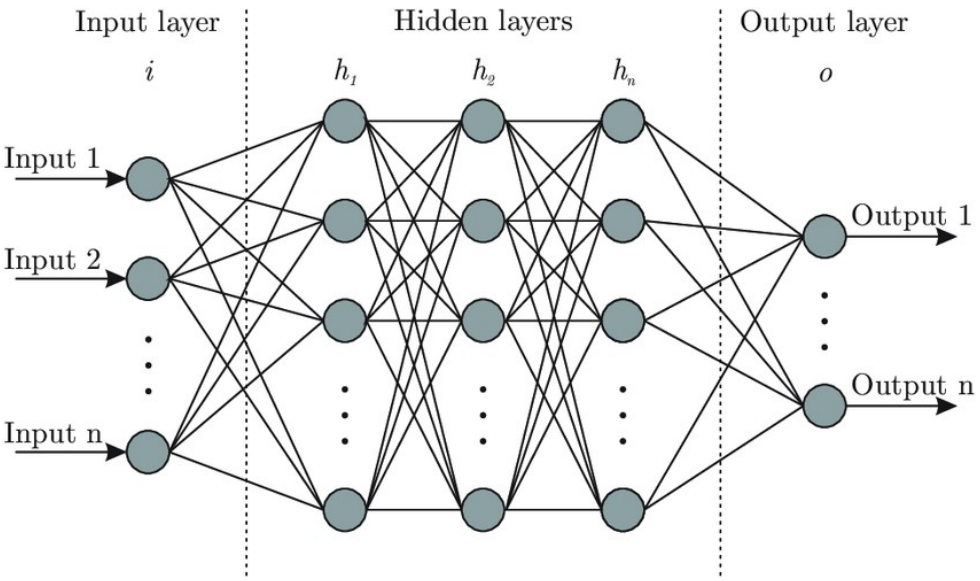


Figure 36 Neural network

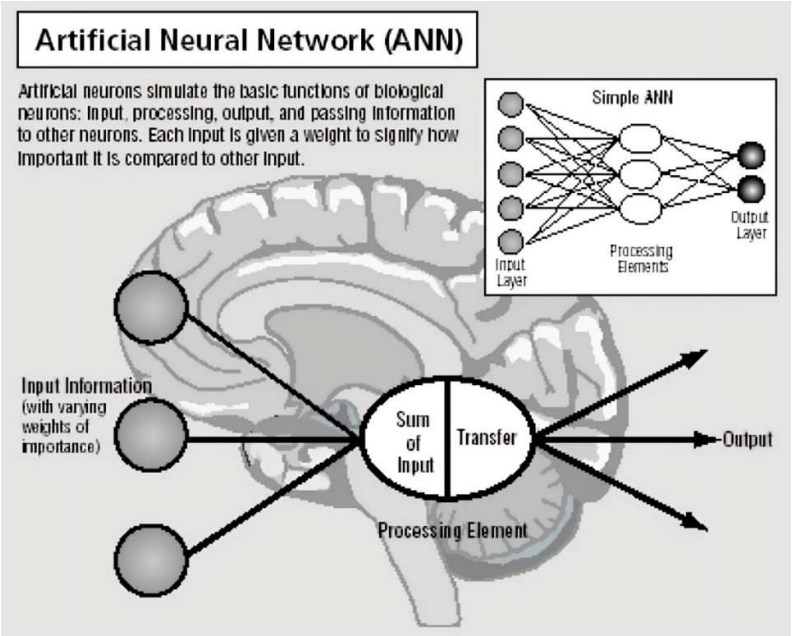


Figure 37ANN

Fully Connected Artificial Neural Network

Artificial neurons, also known as units, are found in artificial neural networks. The whole Artificial Neural Network is composed of these artificial neurons, which are arranged in a series of layers. The complexities of neural networks will depend on the complexities of the underlying patterns in the dataset whether a layer has a dozen units or millions of units. Commonly, Artificial Neural Network has an input layer, an output layer as well as hidden layers. The input layer receives data from the outside world which the neural network needs to analyse or learn about.

In a fully connected artificial neural network, there is an input layer and one or more hidden layers connected one after the other. Each neuron receives input from the previous layer neurons or the input layer. The output of one neuron becomes the input to other neurons in the next layer of the network, and this process continues until the final layer produces the output of the network. Then, after passing through one or more hidden layers, this data is transformed into valuable data for the output layer. Finally, the output layer provides an output in the form of an artificial neural network's response to the data that comes in. Units are linked to one another from one layer to another in the bulk of neural networks. Each of these links has weights that control how much one unit influences another. The neural network learns more and more about the data as it moves from one unit to another, ultimately producing an output from the output layer.

Difference between Machine Learning and Deep Learning :

machine learning and deep learning AI both are subsets of artificial intelligence but there are many similarities and differences between them.

Machine Learning	Deep Learning
Apply statistical algorithms to learn the hidden patterns and relationships in the dataset.	Uses artificial neural network architecture to learn the hidden patterns and relationships in the dataset.
Can work on the smaller amount of dataset	Requires the larger volume of dataset compared to machine learning
Better for the low-label task.	Better for complex task like image processing, natural language processing, etc.
Takes less time to train the model.	Takes more time to train the model.
A model is created by relevant features which are manually extracted from images to detect an object in the image.	Relevant features are automatically extracted from images. It is an end-to-end learning process.
Less complex and easy to interpret the result.	More complex, it works like the black box interpretations of the result are not easy.
It can work on the CPU or requires less computing power as compared to deep learning.	It requires a high-performance computer with GPU.

Deep Learning Algorithms:

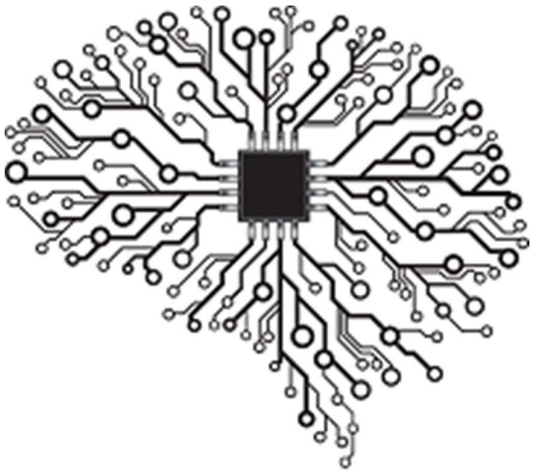


Figure 38 DL algorithm

The Deep Learning Algorithms are as follows:

1. Convolutional Neural Networks (CNNs)

CNN popularly known as ConvNets majorly consists of several layers and are specifically used for image processing and detection of objects. It was developed in 1998 by Yann LeCun and was first called LeNet**.** Back then, it was developed to recognize digits and zip code characters. CNNs have wide usage in identifying the image of satellites, medical image processing, series forecasting, and anomaly detection.

CNNs process the data by passing it through multiple layers and extracting features to exhibit convolutional operations. The Convolutional Layer consists of a Rectified Linear Unit (ReLU) that outlasts to rectify the feature map. The Pooling layer is used to rectify these feature maps into the next feed. Pooling is generally a sampling algorithm that is down-sampled and it reduces the dimensions of the feature map. Later, the result generated consists of 2-D arrays consisting of single, long, continuous**,** and linear vectors flattened in the map. The next layer i.e., called Fully Connected Layer which forms the flattened matrix or 2-D array fetched from the Pooling Layer as input and identifies the image by classifying it.

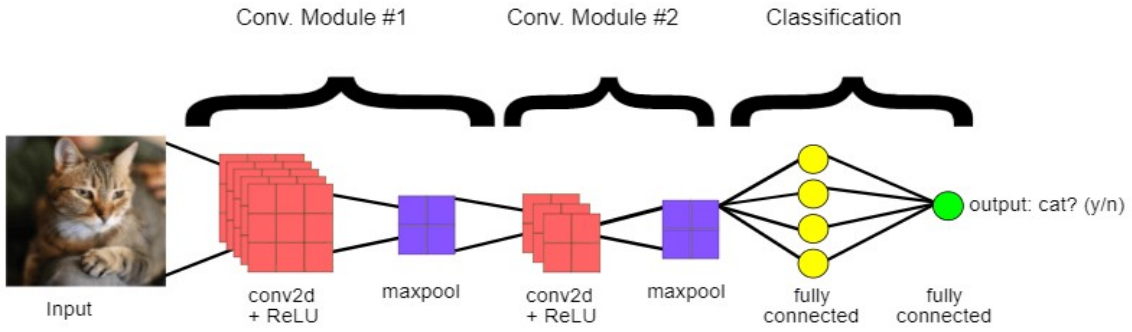


Figure 39 CNN

2. Long Short Term Memory Networks (LSTMs)

Introduction

Long Short-Term Memory Networks is a deep learning, sequential neural network that allows information to persist. It is a special type of Recurrent Neural Network which is capable of handling the vanishing gradient problem faced by RNN. LSTM was designed by Hochreiter and Schmidhuber that resolves the problem caused by traditional rnns and machine learning algorithms. LSTM Model can be implemented in Python using the Keras library.

Let's say while watching a video, you remember the previous scene, or while reading a book, you know what happened in the earlier chapter. RNNs work similarly; they remember the previous information and use it for processing the current input. The shortcoming of RNN is they cannot remember long-term dependencies due to vanishing gradient. LSTMs are explicitly designed to avoid long-term dependency problems.

This article will cover all the basics about LSTM, including its meaning, architecture, applications, and gates.

What is LSTM?

LSTM (lengthy quick-term reminiscence) is a recurrent neural community (RNN) architecture widely utilized in Deep studying. It excels at shooting long-time period dependencies, making it best for sequence prediction responsibilities.

Unlike traditional neural networks, LSTM contains comment connections, permitting it to process whole sequences of facts, no longer simply man or woman records points. This makes it rather effective in know-how and predicting patterns in sequential records like time collection, textual content, and speech.

LSTM has grown to be a powerful tool in artificial intelligence and deep gaining knowledge of, permitting breakthroughs in numerous fields by uncovering treasured insights from sequential statistics.

LSTM Architecture

Within the advent of lengthy short-time period memory, we learned that it resolves the vanishing gradient trouble confronted using RNN, so now, in this segment, we will see the way it resolves this trouble via getting to know the structure of the LSTM. At a high stage, LSTM works very similar to an RNN cell. here is the inner functioning of the LSTM network. The LSTM network architecture includes three parts, as proven in the picture under, and every component plays a character feature.

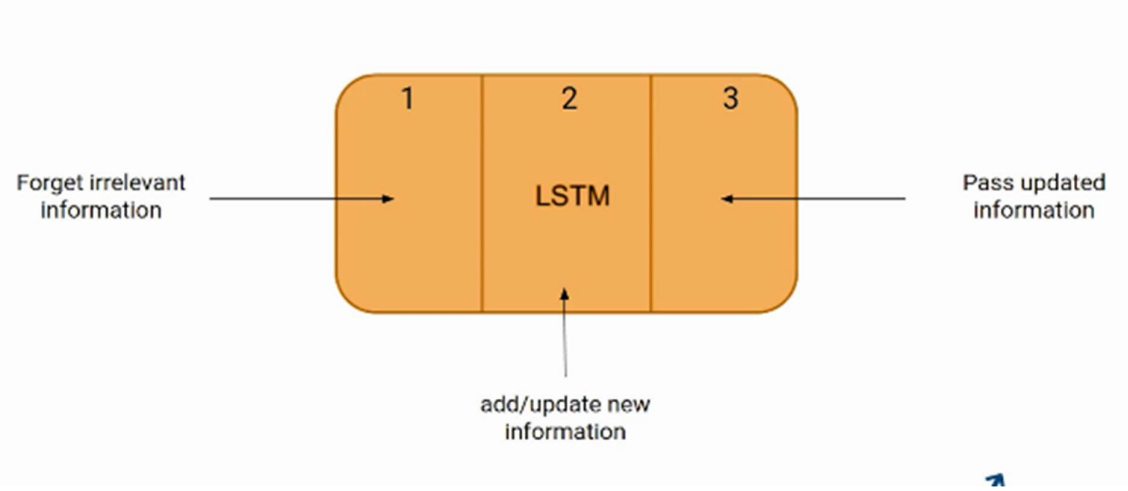


Figure 40 LSTM

The Logic Behind LSTM

The first part chooses whether the information coming from the previous timestamp is to be remembered or is irrelevant and can be forgotten. In the second part, the cell tries to learn new information from the input to this cell. At last, in the third part, the cell passes the updated information from the current timestamp to the next timestamp. This one cycle of LSTM is considered a single-time step.

These three parts of an LSTM unit are known as gates. They control the flow of information in and out of the memory cell or lstm cell. The first gate is called the **Forget gate, the** second gate is known as **the Input gate,** and the last one is **the Output gate**. An

LSTM unit that consists of these three gates and a memory cell or lstm cell can be considered as a layer of neurons in traditional feedforward neural network, with each neuron having a hidden layer and a current state.

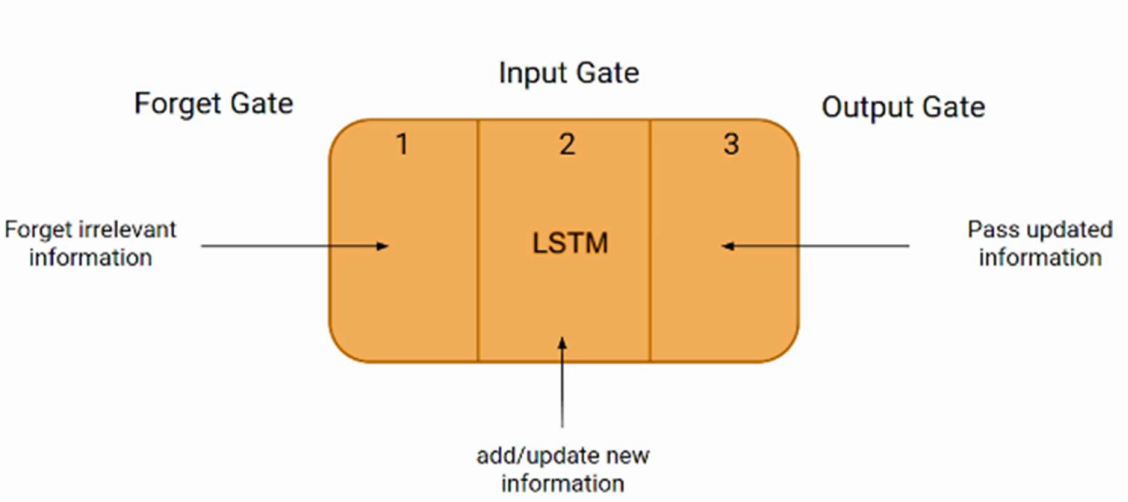


Figure 41 LSTM -2

Just like a simple RNN, an LSTM also has a hidden state where H(t-1) represents the hidden state of the previous timestamp and Ht is the hidden state of the current timestamp. In addition to that, LSTM also has a cell state represented by C(t-1) and C(t) for the previous and current timestamps, respectively.

Here the hidden state is known as Short term memory, and the cell state is known as Long term memory. Refer to the following image.

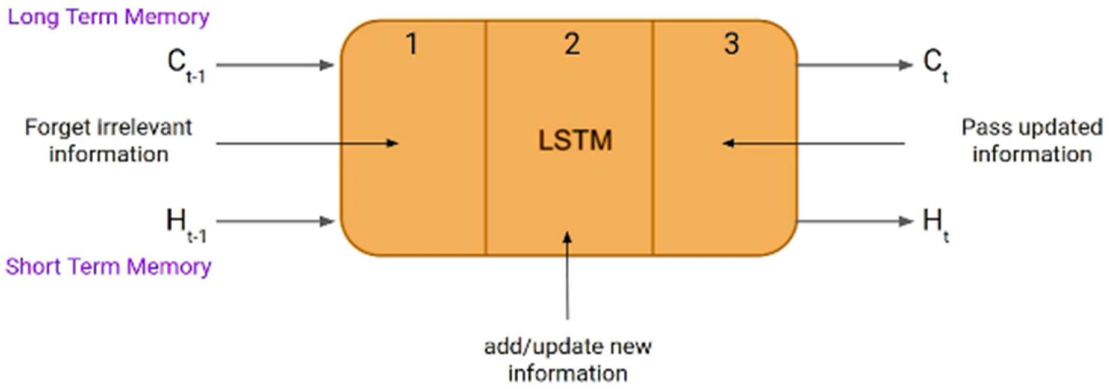


Figure 42 LSTM-3

It is interesting to note that the cell state carries the information along with all the timestamps.

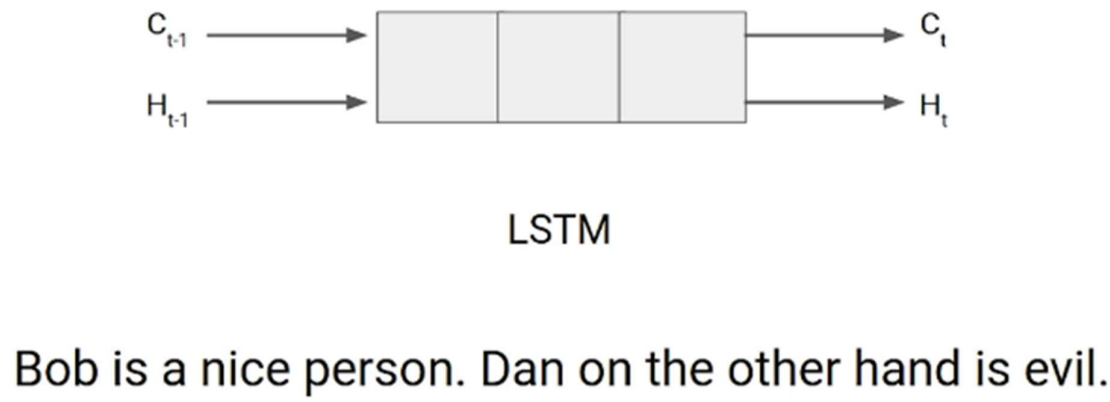


Figure 43 LSTM-4

Example of LTSM Working

Let's take an example to understand how LSTM works. Here we have two sentences separated by a full stop. The first sentence is "Bob is a nice person," and the second sentence is "Dan, on the Other hand, is evil". It is very clear, in the first sentence, we are talking about Bob, and as soon as we encounter the full stop(.), we started talking about Dan.

As we move from the first sentence to the second sentence, our

network should realize that we are no more talking about Bob. Now our subject is Dan. Here, the Forget gate of the network allows it to forget about it. Let's understand the roles played by these gates in LSTM architecture.

Forget Gate:

In a cell of the LSTM neural network, the first step is to decide whether we should keep the information from the previous time step or forget it. Here is the equation for forget gate.

Forget Gate:

- $f_t = \sigma(x_t * U_f + H_{t-1} * W_f)$

Let's try to understand the equation, here

- Xt: input to the current timestamp.
- Uf: weight associated with the input
- Ht-1: The hidden state of the previous timestamp
- Wf: It is the weight matrix associated with the hidden state

Later, a sigmoid function is applied to it. That will make ft a number between 0 and 1. This ft is later multiplied with the cell state of the previous timestamp, as shown below.

$C_{t-1} * f_t = 0$ ...if $f_t = 0$ (forget everything)

$C_{t-1} * f_t = C_{t-1}$ ...if $f_t = 1$ (forget nothing)

Input Gate

Let's take another example.

"Bob knows swimming. He told me over the phone that he had served the navy for four long years."

So, in both these sentences, we are talking about Bob. However, both give different kinds of information about Bob. In the first sentence, we get the information that he knows swimming. Whereas the second sentence tells, he uses the phone and served in the navy for four years.

Now just think about it, based on the context given in the first sentence, which information in the second sentence is critical? First, he used the phone to tell, or he served in the navy. In this context, it doesn't matter whether he used the phone or any other medium of communication to pass on the information. The fact that he was in the navy is important information, and this is something we want our model to remember for future computation. This is the task of the Input gate.

The input gate is used to quantify the importance of the new information carried by the input. Here is the equation of the input gate

Input Gate:

- $i_t = \sigma(x_t * U_i + H_{t-1} * W_i)$

Here,

- Xt: Input at the current timestamp t
- Ui: weight matrix of input
- Ht-1: A hidden state at the previous timestamp
- Wi: Weight matrix of input associated with hidden state

Again we have applied the sigmoid function over it. As a result, the value of I at timestamp t will be between 0 and 1.

New Information

Now the new information that needed to be passed to the cell state is a function of a hidden state at the previous timestamp t-1 and input x at timestamp t. The activation function here is tanh. Due to the tanh function, the value of new information will be between -1 and 1. If the value of Nt is negative, the information is subtracted from the cell state, and if the value is positive, the information is added to the cell state at the current timestamp.

However, the Nt won't be added directly to the cell state. Here comes the updated equation:

$$C_t = f_t * C_{t-1} + i_t * N_t \text{ (updating cell state)}$$

Here, Ct-1 is the cell state at the current timestamp, and the others are the values we have calculated previously.

Output Gate:

Now consider this sentence.

"Bob single-handedly fought the enemy and died for his country. For his contributions, brave_____."

During this task, we have to complete the second sentence. Now, the minute we see the word brave, we know that we are talking about a person. In the sentence, only Bob is brave, we can not say the enemy is brave, or the country is brave. So based on the current expectation, we have to give a relevant word to fill in the blank. That word is our output, and this is the function of our Output gate.

Here is the equation of the Output gate, which is pretty similar to the two previous gates.

Output Gate:

- $o_t = \sigma(x_t * U_o + H_{t-1} * W_o)$

Its value will also lie between 0 and 1 because of this sigmoid function. Now to calculate the current hidden state, we will use Ot and tanh of the updated cell state. As shown below.

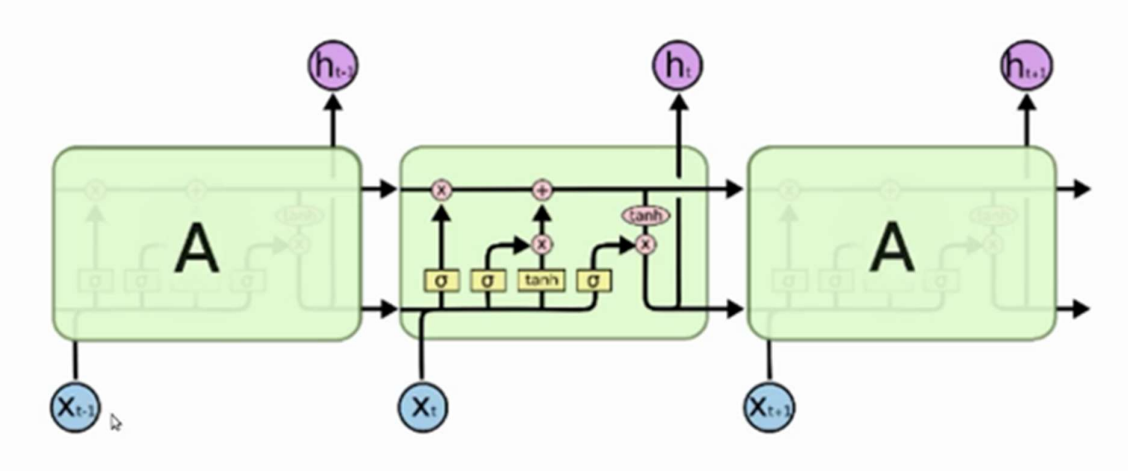
$$H_t = o_t * \tanh(C_t)$$

It turns out that the hidden state is a function of Long term memory (Ct) and the current output. If you need to take the output of the current timestamp, just apply the SoftMax activation on hidden state Ht.

$$\text{Output} = \text{Softmax}(H_t)$$

Here the token with the maximum score in the output is the prediction.

This is the More intuitive diagram of the LSTM network.



Conclusion

In this we covered the basics and sequential architecture of a Long Short-Term Memory Network model. Knowing how it works helps you design an LSTM model with ease and better understanding. It is an important topic to cover as LSTM models are widely used in artificial intelligence for natural language processing tasks like language modeling and

machine translation. Some other applications of lstm are speech recognition, image captioning, handwriting recognition, time series forecasting by learning time series data, etc.

3. Recurrent Neural Networks (RNNs)

Recurrent Neural Networks or RNNs consist of some directed connections that form a cycle that allow the input provided from the LSTMs to be used as input in the current phase of RNNs. These inputs are deeply embedded as inputs and enforce the memorization ability of LSTMs letting these inputs get absorbed for a period in the internal memory. RNNs are therefore dependent on the inputs that are preserved by LSTMs and work under the synchronization phenomenon of LSTMs. RNNs are mostly used in captioning image, time series analysis, recognizing handwritten data, and translating data to machines.

RNNs follow the work approach by putting output feeds (t-1) time if the time is defined as **t.** Next, the output determined by t is fed at input time **t+1.** Similarly, these processes are repeated for all the inputs consisting of any length. There's also a fact about RNNs is that they store historical information and there's no increase in the input size even if the model size is increased. RNNs look something like this when unfolded.

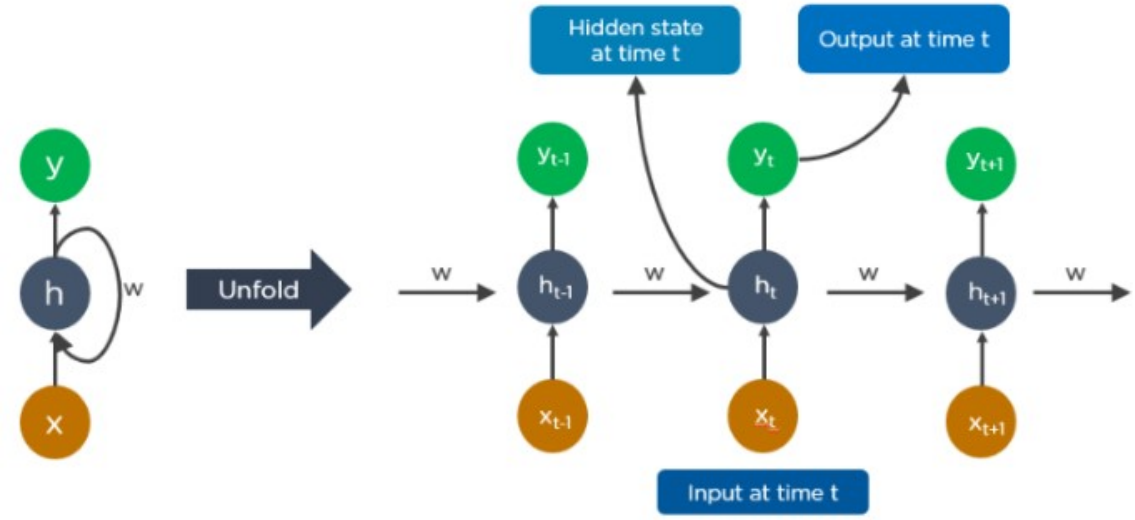


Figure 44 RNN

4. Generative Adversarial Networks (GANs)

GANs are defined as deep learning algorithms that are used to generate new instances of data that match the training data. GAN usually consists of two components namely a generator that learns to generate false data and a discriminator that adapts itself by learning from this false data. Over some time, GANs have gained immense usage since they are frequently being used to clarify astronomical images and simulate lensing the gravitational dark matter. It is also used in video games to increase graphics for 2D textures by recreating them in higher resolution like 4K. They are also used in creating realistic cartoon characters and also rendering human faces and 3D object rendering.

GANs work in simulation by generating and understanding the fake data and the real data. During the training to understand these data, the generator produces different kinds of fake data where the discriminator quickly learns to adapt and respond to it as false data. GANs then send these recognized results for updating. Consider the below image to visualize the functioning.

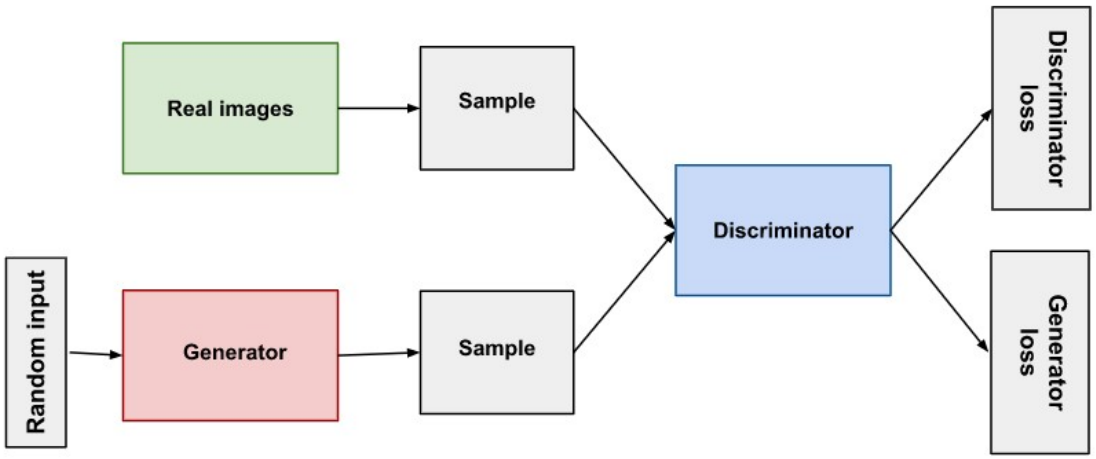


Figure 45 GANs

5. Radial Basis Function Networks (RBFNs)

RBFNs are specific types of neural networks that follow a feed-forward approach and make use of radial functions as activation functions. They consist of three layers namely the input layer, hidden layer, and output layer which are mostly used for time-series prediction, regression testing, and classification.

RBFNs do these tasks by measuring the similarities present in the training data set. They usually have an input vector that feeds these data into the input layer thereby confirming the identification and rolling out results by comparing previous data sets. Precisely, the input layer has neurons that are sensitive to these data and the nodes in the layer are efficient in classifying the class of data. Neurons are originally present in the hidden layer though they work in close integration with the input layer. The hidden layer contains Gaussian transfer functions that are inversely proportional to the distance of the output from the neuron's center. The output layer has linear combinations of the radial-based data where the Gaussian functions are passed in the neuron as parameters and output is generated. Consider the image below to understand the process thoroughly.

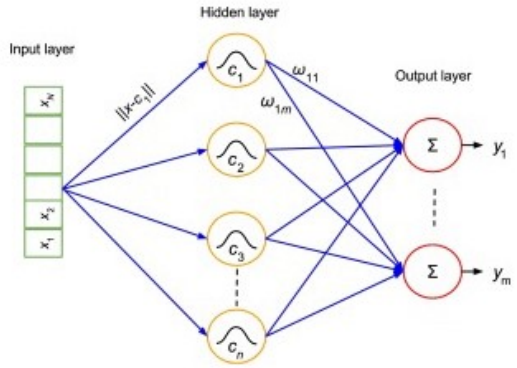


Figure 46 RBFNs

6. Multilayer Perceptrons (MLPs)

MLPs are the base of deep learning technology. It belongs to a class of feed-forward neural networks having various layers of perceptrons**.** These perceptrons have various activation functions in them. MLPs also have connected input and output layers and their number is the same. Also, there's a layer that remains hidden amidst these two layers. MLPs are mostly used to build image and speech recognition systems or some other types of translation software**.**

The working of MLPs starts by feeding the data in the input layer. The neurons present in the layer form a graph to establish a connection that passes in one direction. The weight of this input data is found to exist between the hidden layer and the input layer. MLPs use activation functions to determine which nodes are ready to fire. These activation functions include tanh function, sigmoid and ReLUs**.** MLPs are mainly used to train the models to

understand what kind of co-relation the layers are serving to achieve the desired output from the given data set. See the below image to understand better.

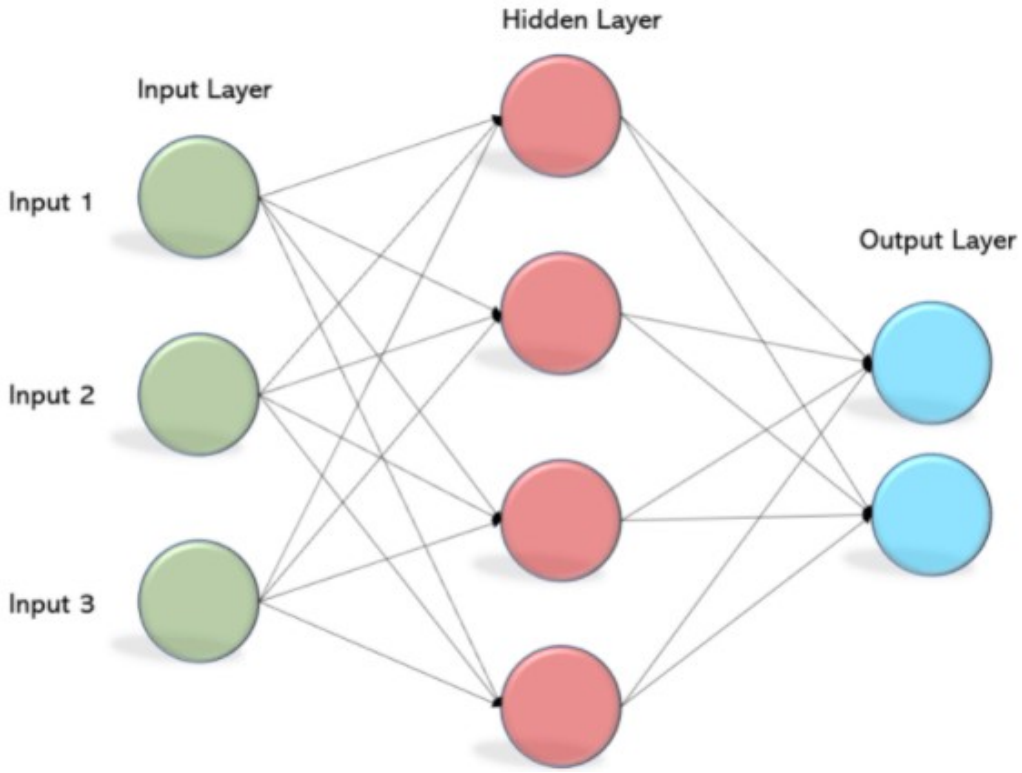


Figure 47 MLPs

7. Self Organizing Maps (SOMs)

SOMs were invented by Teuvo Kohenen to achieve data visualization to understand the dimensions of data through artificial and self-organizing neural networks. The attempts to achieve data visualization to solve problems are mainly done by what humans cannot visualize. These data are generally high-dimensional so there are lesser chances of human involvement and of course less error.

SOMs help in visualizing the data by initializing the weights of different nodes and then choosing random vectors from the given training data. They examine each node to find the relative weights so that dependencies can be understood. The winning node is decided and that is called the Best Matching Unit (BMU). Later, SOMs discover these winning nodes but the nodes reduce over time from the sample vector. So, the closer the node to BMU more is the more chance to recognize the weight and carry out further activities. There are also multiple iterations done to ensure that no node closer to BMU is missed. One example of such is

the RGB color combinations that we use in our daily tasks. Consider the below image to understand how they function.

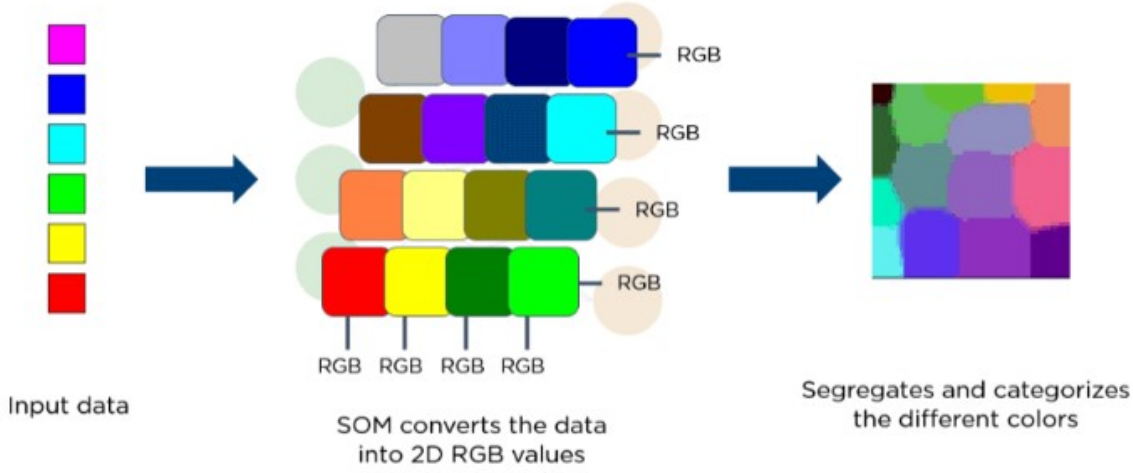


Figure 48 SOMs

8. Deep Belief Networks (DBNs)

DBNs are called generative models because they have various layers of latent as well as stochastic variables. The latent variable is called a hidden unit because they have binary values. DBNs are also called Boltzmann Machines because the RGM layers are stacked over each other to establish communication with previous and consecutive layers. DBNs are used in applications like video and image recognition as well as capturing motional objects.

DBNs are powered by Greedy algorithms**.** The layer to layer approach by leaning through a top-down approach to generate weights is the most common way DBNs function. DBNs use step by step approach of Gibbs sampling on the hidden two-layer at the top. Then, these stages draw a sample from the visible units using a model that follows the ancestral sampling method. DBNs learn from the values present in the latent value from every layer following the bottom-up pass approach.

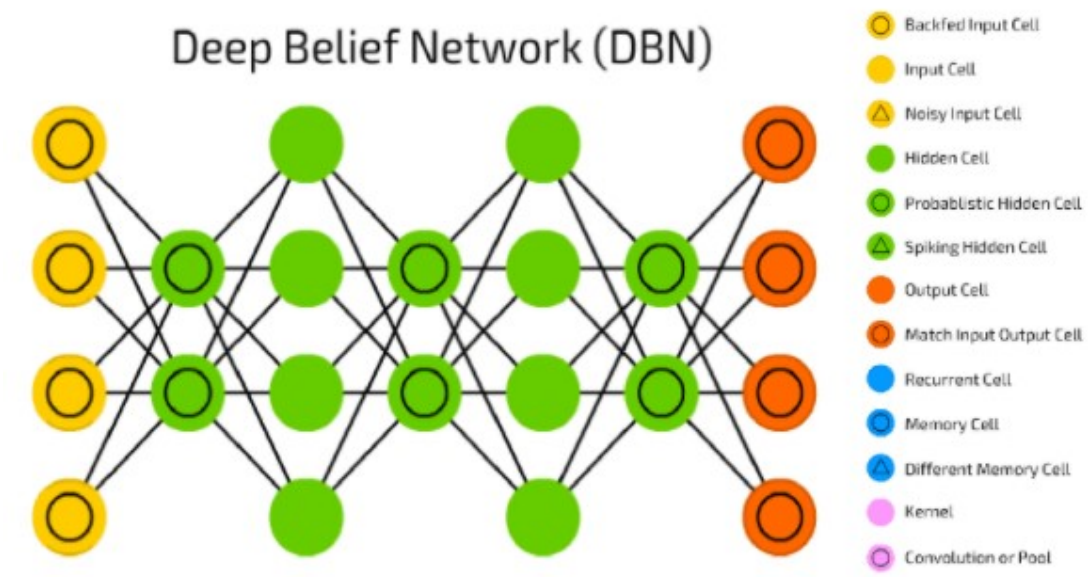


Figure 49 DBNs

9. Restricted Boltzmann Machines (RBMs)

RBMs were developed by Geoffrey Hinton and resemble stochastic neural networks that learn from the probability distribution in the given input set. This algorithm is mainly used in the field of dimension reduction, regression , and classification, topic modeling and are considered the building blocks of DBNs. RBIs consist of two layers namely the visible layer and the hidden layer. Both of these layers are connected through hidden units and have bias units connected to nodes that generate the output. Usually, RBMs have two phases namely forward pass and backward pass.

The functioning of RBMs is carried out by accepting inputs and translating them to numbers so that inputs are encoded in the forward pass. RBMs take into account the weight of every input, and the backward pass takes these input weights and translates them further into reconstructed inputs. Later, both of these translated inputs, along with individual weights, are combined. These inputs are then pushed to the visible layer where the activation is carried out, and output is generated that can be easily reconstructed. To understand this process, consider the below image.

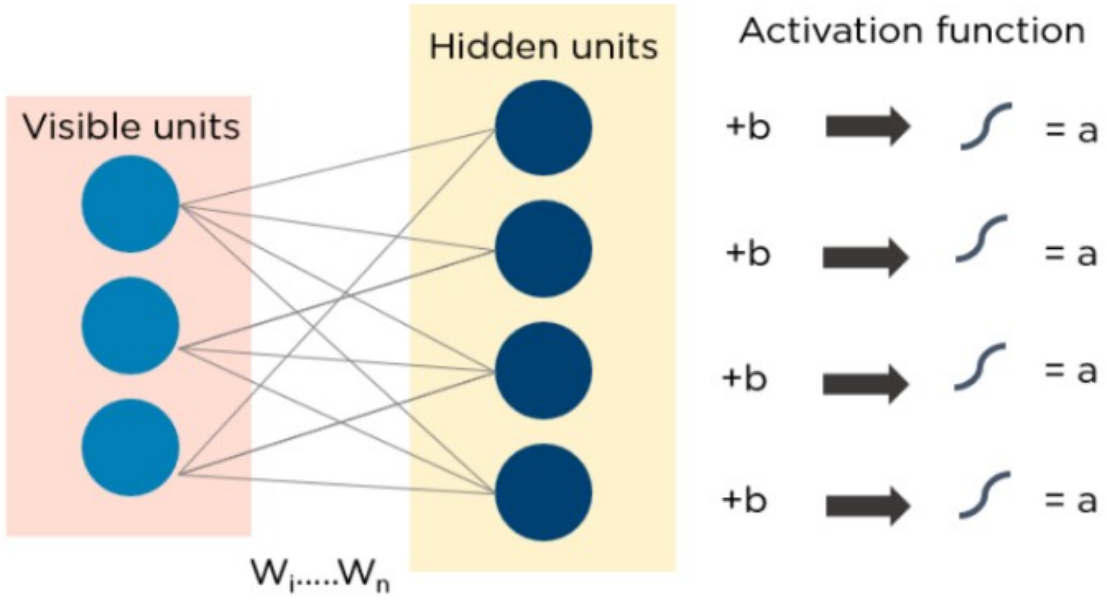


Figure 50 RbMs

Autoencoders

Autoencoders are a special type of neural network where inputs are outputs are found usually identical. It was designed to primarily solve the problems related to unsupervised learning. Autoencoders are highly trained neural networks that replicate the data. It is the reason why the input and output are generally the same. They are used to achieve tasks like pharma discovery**,** image processing**,** and population prediction**.**

Autoencoders constitute three components namely the encoder, the code, and the **decoder.** Autoencoders are built in such a structure that they can receive inputs and transform them into various representations. The attempts to copy the original input by reconstructing them are more accurate. They do this by encoding the image or input and reduce the size. If the image is not visible properly they are passed to the neural network for clarification. Then, the clarified image is termed a reconstructed image and this resembles as accurate as the previous image. To understand this complex process, see the below-provided image.

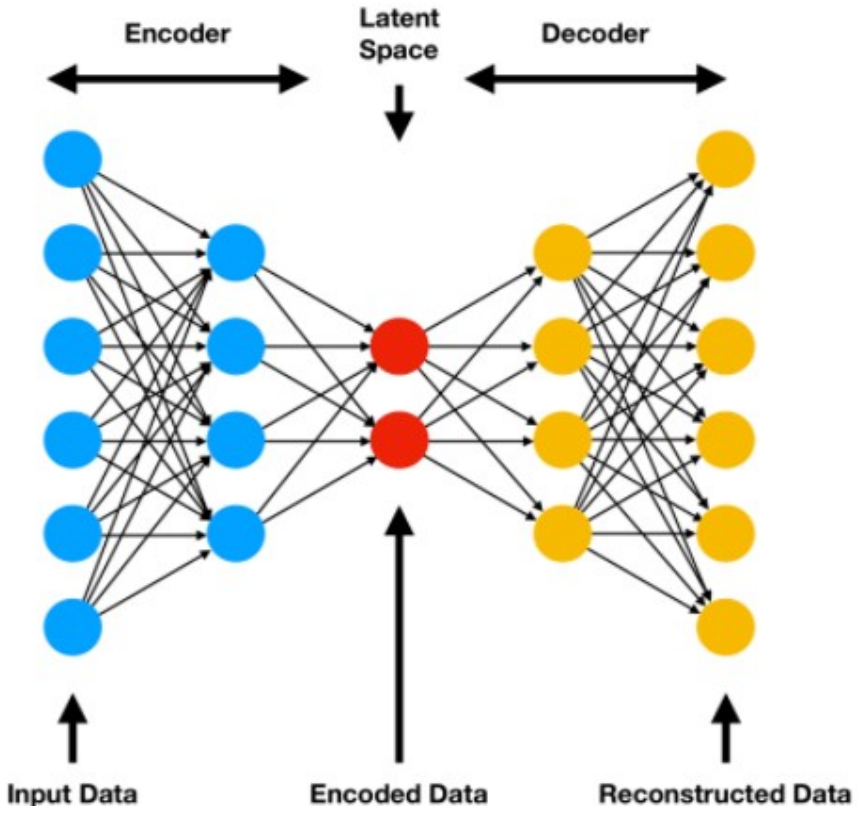


Figure 51 Autoencoders

Chapter -4 Literature Survey

Literature Survey on Speech Emotion Recognition using ML and DL Models

4.1.Introduction

Speech Emotion Recognition (SER) is the task of automatically identifying the emotional state of a speaker from their speech signal. It is a crucial component of affective computing, aiming to enable computers to understand and respond to human emotions in a more natural and empathetic way. SER has the potential to revolutionize human-computer interaction, enhance mental health monitoring, and improve customer service, marketing, and healthcare applications.

Artificial intelligence (AI), particularly machine learning (ML) and deep learning (DL) models, have played a significant role in advancing SER. Traditional ML algorithms have been used to extract features from speech and classify emotions. More recently, DL architectures, including Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), have demonstrated remarkable performance in SER by automatically learning complex features from raw audio data.

4.2. Support Vector Machines (SVM):

- SVMs are supervised learning algorithms widely used for pattern recognition. They aim to find an optimal hyperplane that separates data points belonging to different classes. [1]
- SVM kernel functions, such as linear, polynomial, and radial basis function (RBF), play a crucial role in mapping data into a higher-dimensional space, enabling non-linear classification. [2]
- Studies have shown SVM's effectiveness in classifying basic emotions from speech using features like MFCC, pitch, and energy. [2], [3]

4.3. k-Nearest Neighbors (k-NN):

- The k-NN algorithm is a simple, non-parametric method for classification. It classifies new data points based on their proximity to known data points in the feature space. [3], [4]

- The choice of distance metrics (e.g., Euclidean distance, Manhattan distance) is crucial for accurately capturing emotional features in speech.[3], [4]
- Research has demonstrated the use of k-NN for classifying emotions like anger, happiness, sadness, and neutrality using MFCC and other acoustic features. [3], [4]

4.4. Decision Trees and Random Forest:

- Decision trees are tree-based classifiers that create a hierarchical decision-making process to predict emotions. [2], [3]
- Random forest is an ensemble learning method that combines multiple decision trees to improve accuracy and reduce overfitting. [2], [3], [5]
- Studies have shown the application of decision trees and random forests in classifying emotions using features like MFCC, pitch, and energy.[2], [3]

Limitations of Traditional Machine Learning:

- **Feature Engineering:** Traditional ML models often require significant manual feature engineering, which can be time-consuming and require domain expertise. [6], [7]
- **Data Requirements:** These models typically need extensive labeled training data, which can be costly and time-consuming to collect. [6], [7]
- **Limited Generalization:** Models trained on specific datasets may struggle to generalize to new datasets or speakers. [6], [7]

4.5. Deep Learning Techniques

4.5.1 Overview of Deep Learning in SER:

- Deep learning models like CNNs, RNNs, LSTMs, and Transformers have revolutionized SER by automatically learning complex features from raw audio signals.[6], [8], [9], [10]
- DL models can learn hierarchical representations of audio data, capturing subtle emotional nuances that traditional methods often miss.[6], [8], [9]
- Deep learning offers the potential for improved generalization to unseen data and different speakers, making it more applicable to real-world scenarios. [6], [8], [9], [10], [11]

4.5.2 Specific DL Architectures for SER:

- **Convolutional Neural Networks (CNNs):** CNNs are particularly adept at capturing spatial and temporal patterns in audio data, making them suitable for analyzing spectrograms and extracting features from speech signals.[2], [4], [12], [13], [14], [15]
- **Recurrent Neural Networks (RNNs):** RNNs, including Long Short-Term Memory (LSTM) and Bidirectional LSTMs (Bi-LSTMs), are effective at modeling temporal dependencies in emotional speech sequences. [4], [9], [16], [17]
- **LSTMs:** LSTMs are a special type of RNN designed to handle long-term dependencies, which are crucial for understanding the context and evolution of emotions within speech.[4], [9], [14], [16], [17]
- **Transformers:** Transformers, known for their success in natural language processing, are increasingly being explored for SER due to their ability to capture long-range dependencies and context. [11]

4.6. Datasets and Preprocessing

- **Widely Used Datasets:**
  - **IEMOCAP (Interactive Emotional Dyadic Motion Capture Database):** A multi-modal dataset with 12 hours of audio and video data.[1]
  - **EmoDB (Berlin Database of Emotional Speech):** A dataset of 535 acted utterances in German. [7], [18]
  - **RAVDESS (Ryerson Audio-Visual Database of Emotional Speech and Song):** A dataset with 2496 acted utterances in English.[10], [16]
  - **SAVEE (Surrey Audio-Visual Expressed Emotion Database):** A dataset with 480 acted utterances in English. [14]
  - **CASIA (Chinese Academy of Sciences Institute of Automation Emotional Speech Database):** Contains 7200 Mandarin utterances of five basic emotions and neutral. [5]
- **Preprocessing Techniques:**
  - **Feature Extraction:** MFCC, spectrograms, and other acoustic features are commonly extracted from speech signals. [5], [6], [9], [16]
  - **Data Augmentation:** Techniques like noise injection, time stretching, and pitch shifting are used to generate synthetic data and enhance the robustness of models. [17], [19]

4.7. Performance Evaluation Metrics

- **Accuracy:** The percentage of correctly classified emotions.[1], [12], [18], [20]
- **F-score:** A harmonic mean of precision and recall, providing a balance between identifying positive instances (precision) and minimizing false negatives (recall). [1][3], [8], [9]
- **Confusion Matrix:** A table that summarizes the classification results, showing correct and incorrect predictions for each emotion class. [1][8], [20], [21]

4.8. Applications and Challenges

Applications:

- **Virtual Assistants:** SER can make virtual assistants more empathetic and responsive by understanding user emotions. [17]
- **Healthcare Applications:** SER can help monitor mental health conditions like depression and anxiety by detecting emotional distress signals in speech. [2]
- **Customer Service:** SER can enhance customer service by identifying customer emotions, leading to more tailored and effective responses. [2]
- **Marketing and Advertising:** SER can be used to analyze customer sentiment towards products and services. [17]

Challenges:

- **Variability in Emotional Expression:** Humans express emotions in different ways, and speech variations can make accurate emotion recognition difficult. [14]
- **Cross-Cultural Differences:** Emotional expressions and speech patterns can vary across cultures, posing challenges for generalizability. [14]
- **Real-Time Processing Requirements:** Developing efficient algorithms that can analyze speech in real-time is critical for interactive applications. [14]
- **Data Bias:** Datasets are often biased towards specific emotions, genders, or languages, which can affect model performance and generalization. [14]

4.9. Recent Advances and Future Directions:

- **Multimodal SER:** Integrating audio with other modalities (text, facial expressions) to capture a more complete emotional picture. [11], [13]
- **Unsupervised and Semi-supervised Learning:** Developing models that can learn from unlabeled or partially labeled data to address the challenge of data scarcity. [6]
- **Robustness:** Enhancing models to be more resilient to noise, speech variations, and accents. [6], [17]
- **Real-time SER:** Developing efficient algorithms for real-time emotion analysis. [14]
- **Interpretability:** Making deep learning models more transparent to understand their decision-making processes and ensure ethical use. [6]

4.10. Comparison Table of Research Papers on Speech Emotion Recognition:

Paper Title	Description	Accuracy	Method Used	Year of Publication
Speech Emotion Recognition Using Deep Convolutional Neural Network and Discriminant Temporal Pyramid Matching	Proposes a model combining Deep Convolutional Neural Networks (DCNN) with Discriminant Temporal Pyramid Matching (DTPM) for speech emotion recognition, achieving state-of-the-art performance.	86.3% (accuracy on the IEMOCAP dataset)	DCNN with DTPM for temporal aggregation of segment-level features.	2018
Automatic Speech Emotion Recognition Using Deep Learning	Provides a comprehensive review of deep learning techniques	NA (review paper)	DNN, CNN, RNN, Autoencoders, LSTM, and others.	2019

Techniques: A Review	for speech emotion recognition, highlighting different approaches and architectures.			
Speech Emotion Recognition Using Deep Convolutional Long Short-Term Memory (LSTM)	Proposes a multi-scale Deep Convolution Long Short-Term Memory (LSTM) framework for spontaneous speech emotion recognition.ss	50.22% (accuracy on the IEMOCAP dataset)	Multi-scale CNN + LSTM architecture with score-level fusion.	2022
Speech Emotion Recognition Based on Feature Selection and Extreme Learning Machine Decision Tree	Proposes a speech emotion recognition system using feature selection based on correlation analysis and Fisher criterion, and an ELM decision tree classifier.	89.6% (average accuracy on the CASIA dataset)	Feature selection followed by an ELM decision tree.	2018
Speech Emotion Recognition Using CNN	Employs a convolutional neural network (CNN) for speech emotion recognition, achieving promising results.	88.3% (accuracy on the Berlin EmoDB dataset)	CNN architecture with a linear SVM classifier.	2014
Speech Emotion Recognition Using Deep Learning: A Review	Offers a comprehensive review of deep learning techniques	NA (review paper)	CNN, RNN, LSTM, and other deep learning models.	2018

	for speech emotion recognition, focusing on data augmentation, feature extraction, and model architectures.			
Speech Emotion Recognition Using Deep Convolutional Neural Network and Discriminant Temporal Pyramid Matching	Combines DCNN and DTPM for speech emotion recognition, demonstrating its effectiveness in bridging the gap between low-level features and abstract emotions.	86.3% (accuracy on the IEMOCAP dataset)	DCNN with DTPM for temporal aggregation of segment-level features.	2018
End-to-End Speech Emotion Recognition with Gender Information	Proposes a gender-aware speech emotion recognition system using a residual convolutional neural network (R-CNN) architecture.	71.5% (accuracy on the RAVDESS dataset)	R-CNN architecture incorporating gender information for improved accuracy.	2020
Speech Emotion Recognition Using Deep Bidirectional LSTM	Investigates the use of deep bidirectional LSTMs for speech emotion recognition, emphasizing the importance of learning long-range	95.1% (accuracy on a three-emotion dataset)	Deep bidirectional LSTM (Bi-LSTM) for emotion recognition.	2012

	dependencies in speech.			
Speech Emotion Recognition: Features and Classification Models	Provides a review of speech emotion recognition features and classification models, covering various acoustic parameters and machine learning techniques.	NA (review paper)	SVM, GMM, KNN, and other classifiers.	2012
Speech emotion recognition using deep 1D & 2D CNN LSTM networks	Proposes two CNN+LSTM networks, one 1D and one 2D, for learning local and global emotion-related features from speech and log-Mel spectrograms, respectively.	95.33% (speaker-dependent accuracy on Berlin EmoDB using 2D CNN+LSTM)	CNN+LSTM architectures for learning both local and global features.	2019
A System to Predict Emotion from Bengali Speech	Develops a speech emotion recognition system for Bengali language using MFCC features and an LSTM model.	84.81% (accuracy on the RAVDESS dataset)	LSTM model with MFCC features.	2021
Metric Learning-Based Multimodal Audio-Visual Emotion Recognition	Introduces a multimodal emotion recognition system that uses metric learning to analyze	66.5% (accuracy on the CREMA-D dataset)	Multimodal Emotion Recognition Metric Learning (MERML) for	2020

	both audio and visual features.		combined audio-visual emotion recognition.	
A Comparative Analysis of Classifiers in Emotion Recognition Through Acoustic Features	Compares the performance of various classifiers (SVM, KNN, LDA, RDA) for emotion recognition using acoustic features.	92.6% (accuracy on the Berlin dataset using RDA)	SVM, KNN, LDA, and RDA classifiers.	2016
Speech Emotion Recognition Using Deep Learning: A Review	Provides a comprehensive overview of deep learning techniques for speech emotion recognition, including architecture choices and data augmentation methods.	NA (review paper)	CNN, RNN, LSTM, and other deep learning models.	2021
Exploring Deep Spectrum Representations via Attention-Based Recurrent and Convolutional Neural Networks for Speech Emotion Recognition	Introduces an attention-based deep learning model for speech emotion recognition, using convolutional and recurrent neural networks to extract and analyze spectral representations.	89.16% (accuracy on the IEMOCAP dataset)	CNN and RNN with attention mechanisms.	2019

X-Vectors Meet Emotions: A Study On Dependencies Between Emotion and Speaker Recognition	Investigates the use of x-vectors, a speaker representation technique, for speech emotion recognition.	81.54% (accuracy on the CREMA-D dataset)	Fine-tuned ResNet model with x-vector features for emotion recognition.	2020
Emotion Recognition from Speech Using Spectrograms and Shallow Neural Networks	Explores the use of spectrograms and shallow neural networks for speech emotion recognition, demonstrating the potential of simpler models for effective classification.	NA (no specific accuracy mentioned)	Shallow neural networks with spectrogram features.	2020
Robust Speech Emotion Recognition Using CNN+LSTM Based on Stochastic Fractal Search Optimization Algorithm	Presents a CNN+LSTM model for speech emotion recognition, with hyperparameter optimization guided by the stochastic fractal search (SFS)-guided whale optimization algorithm (WOA).	98.13% (accuracy on the IEMOCAP dataset)	CNN+LSTM architecture with SFS-Guided WOA for hyperparameter optimization.	2022
Speech Emotion Recognition Based on Decision Tree and Improved SVM Mixed Model	Proposes a mixed model combining a decision tree with improved SVMs for speech emotion recognition.	NA (no specific accuracy mentioned)	Decision tree and improved SVM classifier.	2017

Automatic Speech Emotion Recognition Using Modulation Spectral Features	Investigates the use of modulation spectral features for speech emotion recognition, demonstrating its effectiveness in capturing emotional information.	NA (no specific accuracy mentioned)	Modulation spectral features and SVM classifier.	2011
Feature Extraction Algorithms to Improve the Speech Emotion Recognition Rate	Explores different feature extraction algorithms (MFCC, DWT, pitch, energy, ZCR) to improve the accuracy of speech emotion recognition.	85% (accuracy using Decision Tree on the RAVDESS dataset)	Various feature extraction algorithms with Decision Tree classifier.	2020
A Comparative Analysis of Deep Learning Architectures for Speech Emotion Recognition	Compares different deep learning architectures (CNN, LSTM, Bi-LSTM) for speech emotion recognition, analyzing their performance on standard datasets.	NA (review paper)	CNN, LSTM, and Bi-LSTM models.	2017
A Database for Automatic Persian Speech Emotion Recognition: Collection, Processing and Evaluation	Introduces a new database of emotional speech in Persian, specifically designed for automatic speech	NA (database development paper)	NA	2014

	emotion recognition research.			

4.11. Conclusion

Speech Emotion Recognition using ML and DL is an area of active research. Deep learning has significantly advanced the field, offering numerous advantages over traditional ML approaches. However, challenges remain in building robust and generalizable SER systems. Future research will focus on addressing these challenges by exploring multimodal approaches, incorporating unsupervised learning, improving robustness, achieving real-time performance, and ensuring interpretability. These advancements will pave the way for creating more sophisticated and emotion-aware computing systems that can better understand and respond to human emotions in various applications.

Chapter-5. Methodology

5.1 Flow chart

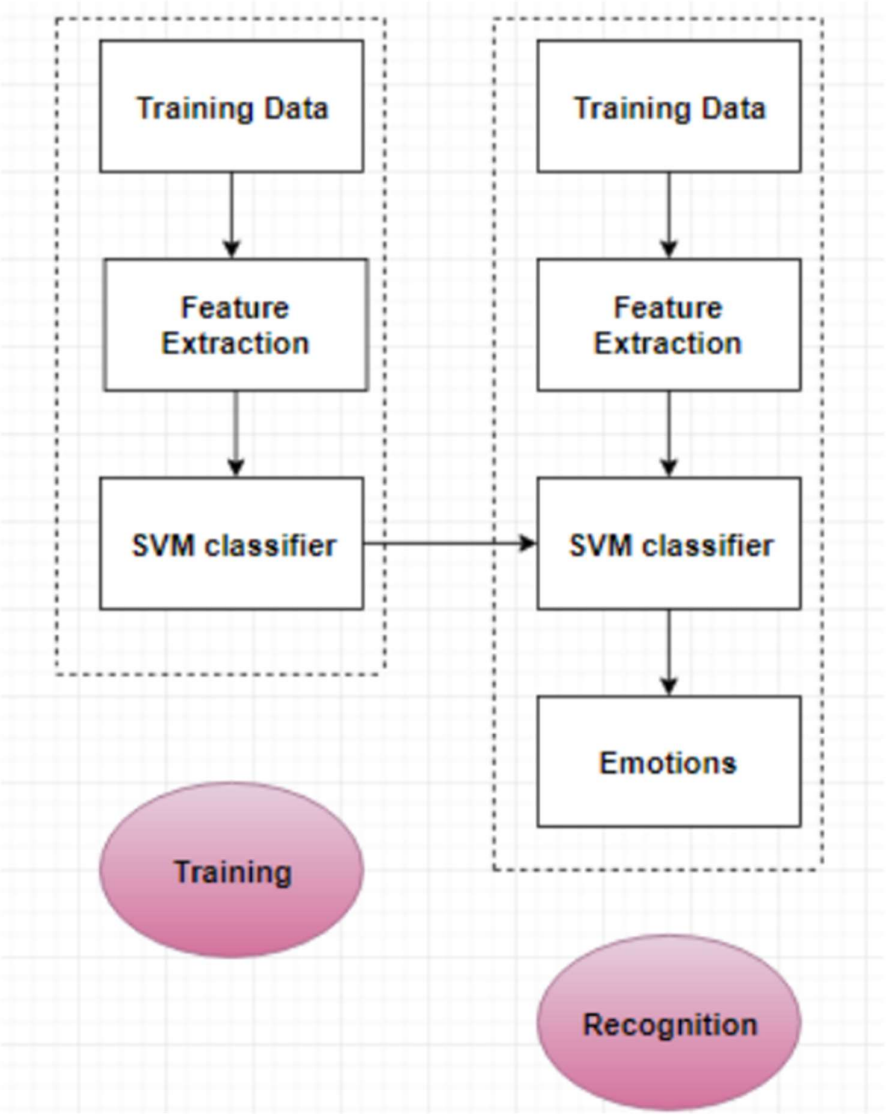


Figure 52 Methodology

Figure 53 Methodology

In this project, the task of Speech Emotion Recognition (SER) was approached using two primary feature extraction methods: Mel-Frequency Cepstral Coefficients (MFCC) and Support Vector Machine (SVM), as well as Long Short-Term Memory (LSTM) networks. While various other feature extraction techniques such as prosodic, spectral, and voice quality

analysis, along with classifiers including K-Nearest Neighbors (KNN), Decision Trees, Random Forest, Logistic Regression, among others, are available, the focus here was on MFCC combined with SVM and LSTM classifiers.

5.2 Method :

- Feature Extraction Methods:

  Mel-Frequency Cepstral Coefficients (MFCC): A widely used technique for extracting features from audio signals, particularly effective in capturing the spectral characteristics of speech.

- Classifiers:
  - Support Vector Machine (SVM): A powerful supervised learning algorithm used for classification tasks. SVM works well in high-dimensional spaces and is effective for both linear and non-linear classification.
  - Long Short-Term Memory (LSTM): A type of recurrent neural network (RNN) architecture, specifically designed to capture long-term dependencies in sequential data. LSTM networks are well-suited for analyzing time-series data like speech signals.

By utilizing MFCC features with SVM and LSTM classifiers, the project aimed to effectively capture and classify the emotional content present in speech signals. This approach leverages the strengths of both traditional machine learning techniques (SVM) and deep learning architectures (LSTM) to achieve accurate emotion recognition from audio data.

Chapter-6 Dataset

6.1 Dataset :

Emotions, either expressed through first verbalizations or vocal signs, are complex phantasms that exhibit both entire physiognomy and variations affected by sophistication, language, grammatical rules applying to nouns that connote sex or animateness, and specific context. While first sentiments are generally thought-out entire, emotions like anger, that are transported through vocal suggestions, can change significantly. For instance, anger frequently exhibits with a extreme fundamental commonness, contrasting accompanying the wave between audio and infrared associated with depression. However, these patterns maybe influenced by determinants to a degree the quality of dossier secondhand for analysis.

If databases secondhand for sympathy recognition are endangered in feature, the conclusions tense from bureaucracy may do wrong, emphasize the critical significance of table design and collection designs. In the domain of speech sentiment acknowledgment, databases play a vital act as they specify the labeled dossier unavoidable for the classification process. The condition concerning this data deeply impacts the benefit of recognition algorithms. Databases for talk feeling recognition maybe classification into three main types: acted (fake), extracted (induced), and unaffected talk databases.

Acted speech databases include records by professional or semi-professional stars, often in reserved workshop environments. While comparatively smooth to create, functioned talk may not really transport real-life fervors, conceivably lowering acknowledgment rates for honest emotions. Elicited talk databases, in another way, aim to simulate heated positions to evoke honest reactions from speakers. While these emotions can not be adequately elicited, they approximately simulate real one, contribution a balance between genuineness and control. Natural talk databases, derived from palpable-planet sources like talk shows or call center records, supply authentic passionate verbalizations but present challenges related to moral and allowable considerations.Once the arrangement of building a database is persistent, supplementary design considerations accept delivery of something play, to a degree the demographics of speakers, containing age and common.

Databases often hold records from adult speakers, but variations lie to contain children and seniors. Furthermore, difference in actors, sympathy, and genders is critical for robust acknowledgment methods.

For example, the widely used Berlin dataset facial characteristics records of seven emotions by ten professional stars of two together genders, each utterance recurrent accompanying various players and sympathy. This diversity embellishes the dataset and reinforces the reliability of excitement acknowledgment frameworks.

6.1.1 Types of Datasets:

In our project, we working two obvious datasets: the Toronto Affecting Speech Set (TESS) and The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS).

TESS contains a different range of excitements containing Anger, Disgust, Fear, Happiness, Noncommittal, and Friendly Surprise. This dataset encompasses a total of 2800 visual and audio entertainment transmitted via radio waves files, contribution a rich variety of sentimental verbalizations for study.

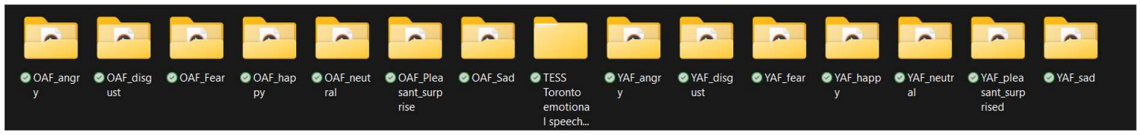


Figure 54Dataset TESS-1

TESS : [Toronto emotional speech set (TESS) (kaggle.com)]

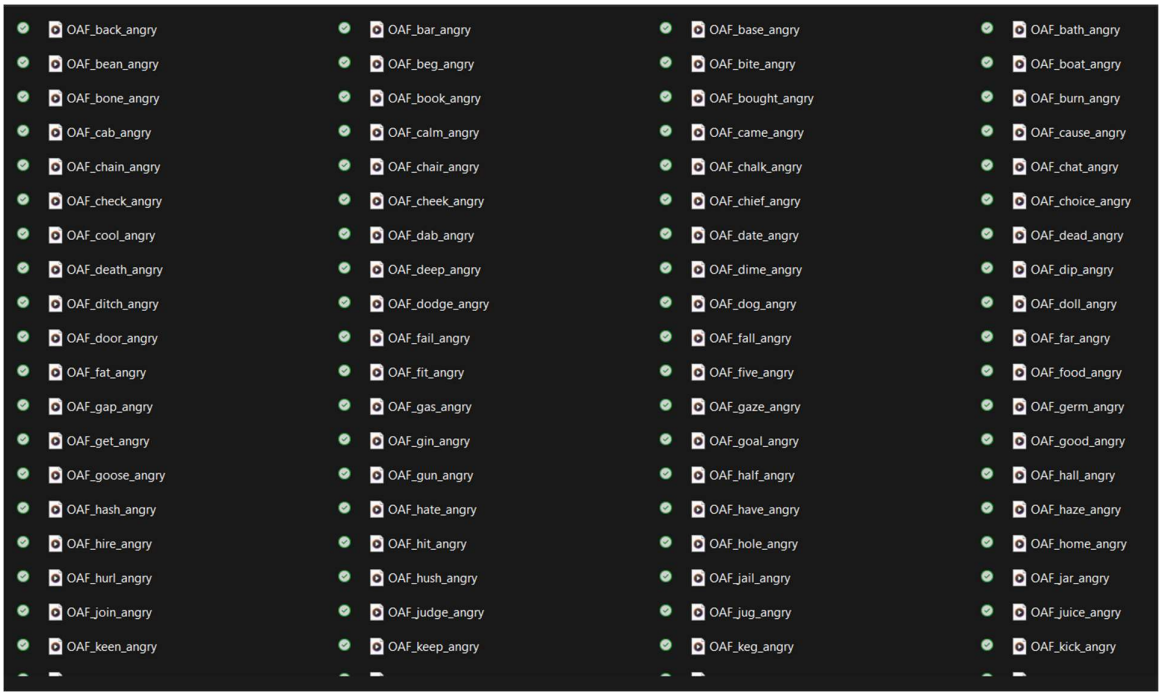


Figure 55 Dataset of TESS (2)

In another way, RAVDESS consists of visual and audio entertainment transmitted via radio waves records promoting 12 male and 12 female stars, climactic in a total of 24 contributors.

Related to TESS, RAVDESS contains a range of feelings equal to those in TESS. This dataset comprises 1440 visual and audio entertainment transmitted via radio waves files, providing a inclusive accumulation of impassioned speech instances.

RAVDESS : [RAVDESS Emotional song audio (kaggle.com)]

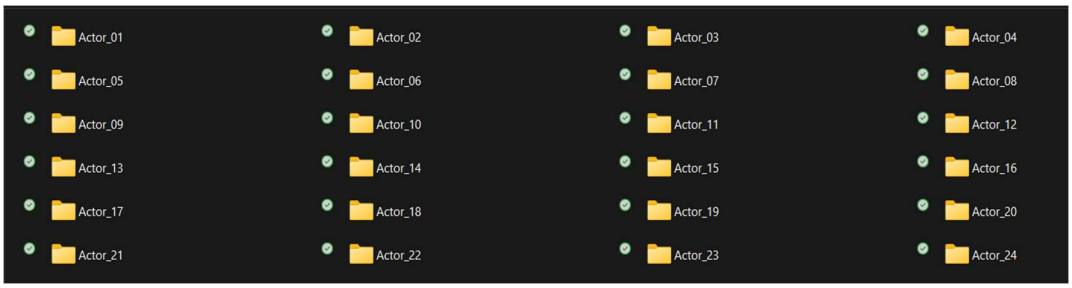


Figure 56 Dataset of RAVDESS (1)

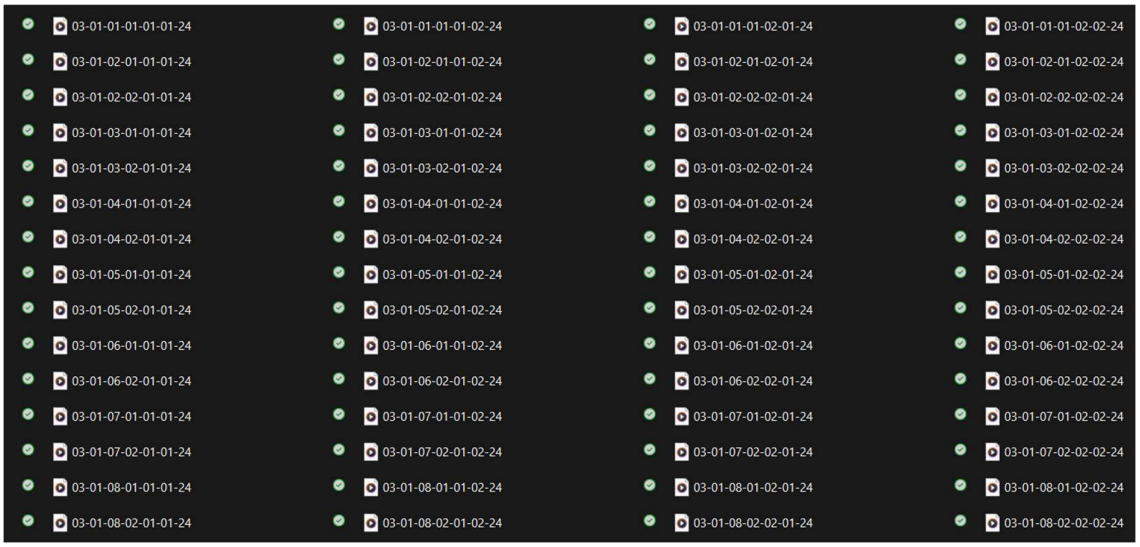


Figure 57 Dataset of RAVDESS (2)

By handling these two datasets, we aim to capture a expansive range of heated verbalizations, enabling strong reasoning and investigation inside our project.

6.1.2Merging the Datasets:

In our Jupyter surroundings, we influence the `os` library to capably process two together the TESS and RAVDESS datasets. By means of this athenaeum, we can seamlessly navigate through directories, approach files, and act movements on our datasets.

By controlling the power of `os`, we organize the dossier preprocessing passage, allowing us to systematize, maneuver, and extract appropriate facts from the audio files held inside these datasets. This allows us to arrange the data for further study and posing.

In our dossier preprocessing passage, we took advantage of the `os` study to assemble a structured Data Frame for two together the TESS and RAVDESS datasets. This Data Frame exists of two key lineaments: 'way' and 'label'. The 'course' feature stores the file way of each visual and audio entertainment transmitted via radio waves file, while

the 'label' feature epitomizes the emotional class of the matching visual and audio entertainment transmitted via radio waves file.

For the TESS dataset, the visual and audio entertainment transmitted via radio waves files are chosen in an explanatory plan in the way that 'OAF_base_disgust', place the moving class is embedded inside the filename.

By leveraging `os`, we guided along route, often over water through the guide construction, gleaned the file courses, and maneuvered the filenames to collect the emotional labels, efficiently constituting the DataFrame.Likewise, for the RAVDESS dataset, the filenames trail a particular encrypting pattern like '03-01-03-01-01-01-05.wav', accompanying the last number designating the sentiment category. Applying `os`, we traveled the directories, culled the file courses, and decoded the filenames to obtain the heated labels, joining ruling class with the particular visual and audio entertainment transmitted via radio waves files in the Data Frame.

Upon assembling separate Data Frames for TESS and RAVDESS, we joined two together datasets to combine the facts from two together beginnings. This integration eased an inclusive study of affecting talk dossier, joining the copiousness of both TESS and RAVDESS datasets into a united foundation for further survey and displaying.

6.1.3 Challenges in Accusation of Dataset & Merging Dataset :

Combining the Tess and Revdess datasets presents a important challenge in dossier transformation on account of their basic distinctnesses. Firstly, the number of excitements and naming patterns extend two together datasets. While Tess dataset surrounds a more expansive range of emotions, Revdess grant permission have a various set or middling categories. Additionally, the designating conferences for the visual and audio entertainment transmitted via radio waves files disagree, further complicating the unification process. With Tess holding 2800 visual and audio entertainment transmitted via radio waves files and Revdess forming 1440, managing the dossier enhances even more intricate.

To efficiently address these challenges, it's essential to conceive a inclusive dossier transformation game plan. This approach bear contain steps to standardize designating traditions, design fervors to a unified set across two together datasets, and conceivably balance class distributions to guarantee equitable likeness all along reasoning and model preparation. By systematically talking these issues, analysts can organize the unification process and derive significant understandings from the linked dataset.

Chapter – 7 Implementation

7.1 Data-preprocessing

7.1.1Data Transformation & Integration :

We transformed the data into a single dataframe with two main features: "Emotions" and "Path". The "Emotions" feature represents the emotion class of each audio file, while the "Path" feature contains the file path of each audio file. This transformation involved merging two separate dataframes: one from the TESS dataset, which includes 2800 audio files and 7 emotions, and another from the REVESS dataset, which includes 1440 audio files and 8 emotions. By integrating these datasets, we created a comprehensive dataframe for further analysis.

	Path	Emotions
0	C:\Users\DAXESH MAHERIYA\OneDrive\Desktop\Sem-...	angry
1	C:\Users\DAXESH MAHERIYA\OneDrive\Desktop\Sem-...	angry
2	C:\Users\DAXESH MAHERIYA\OneDrive\Desktop\Sem-...	angry
3	C:\Users\DAXESH MAHERIYA\OneDrive\Desktop\Sem-...	angry
4	C:\Users\DAXESH MAHERIYA\OneDrive\Desktop\Sem-...	angry

Figure 58 Transformed Data

7.1.2. Exploratory Data Analysis (EDA) :

We conducted EDA on the audio data by using libraries such as librosa, matplotlib, etc.

Librosa helped in generating spectrograms, which are visual representations of the audio signal's frequency content over time.

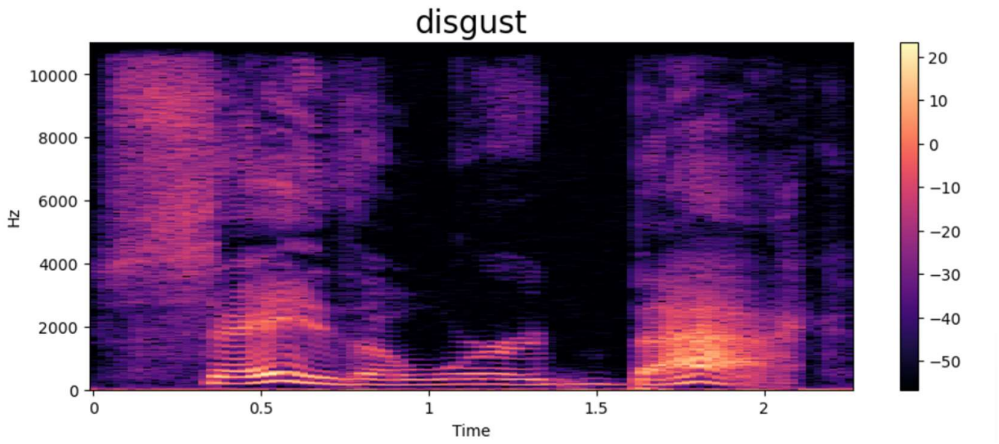


Figure 59 Spectrogram

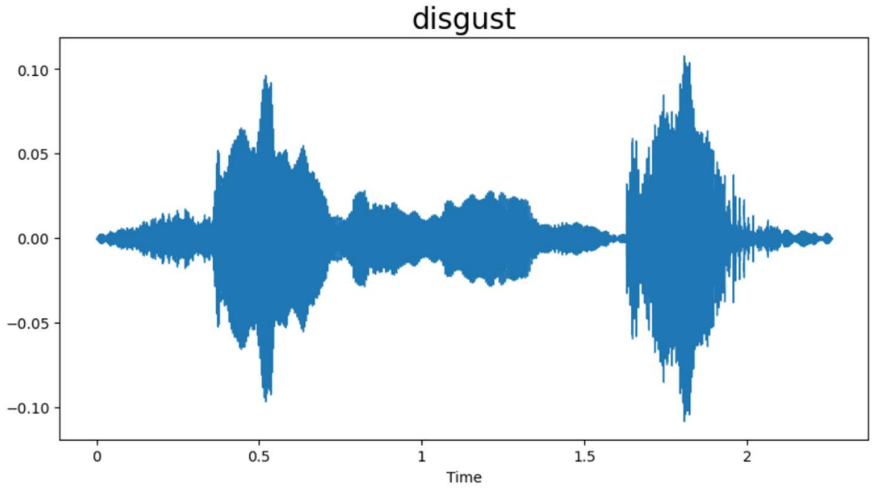


Figure 60 Waveform

- Additionally, we utilized audio libraries to play the audio of each file, allowing for auditory inspection of the data.

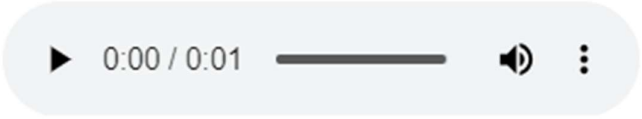


Figure 61 Audio playback

By following this approach, we were able to effectively preprocess the audio data for further analysis. This involved structuring the data into a suitable format and gaining insights into the characteristics of the audio signals through visualization and auditory examination.

7.2 Feature Extraction :

Our project focuses on classifying emotions from speech based on MFCCs, which stands for Mel-Frequency Cepstral Coefficients. MFCCs are coefficients that collectively make up an MFC (Mel-Frequency Cepstrum). They are widely used in audio and speech processing due to their ability to capture the timbral aspects of audio signals in a way that reflects the human ear's sensitivity to different frequencies.

To extract Mel-Frequency Cepstral Coefficients (MFCCs) from an audio file using `librosa`, you load the audio file and then use the `librosa.feature.mfcc` function.

```
def extract_mfcc(filename):
    y, sr = librosa.load(filename, duration=3, offset=0.5)
    mfcc = np.mean(librosa.feature.mfcc(y=y, sr=sr, n_mfcc=40).T, axis=0)
    return mfcc
```

Figure 62Feature extraction

7.3 Classification :

In classification tasks, we start by partitioning our dataset using the holdout method.This method divides the data into training and testing sets, where a portion (e.g., 80%) is designated for training our chosen model—such as SVM, KNN, Decision Trees, Random Forests, or LSTM for deep learning. Subsequently, we utilize the trained model to predict labels for the remaining portion (e.g., 20%) designated as the test set. Evaluation of the model's accuracy involves comparing these predicted labels with the actual labels in the test set (`y_test`). This assessment provides insights into the model's ability to generalize to new, unseen data and helps detect potential issues like overfitting or underfitting. Adjustments to model parameters or exploration of different algorithms may follow to enhance performance and ensure robust predictions.

7.3.1 Splitting the dataset :

To train and evaluate our classification models, we utilize various machine learning algorithms such as SVM, KNN, Decision Trees (DT), Random Forests (RF), among others, as well as LSTM as a deep learning algorithm. For this purpose, we partition our data into training and testing sets using the holdout method. This involves using the `train_test_split` function from `sklearn` to split the data into 80% for training and 20% for testing. This allows us to train our models on the training set, evaluate their accuracy on the test set, and assess for issues such as overfitting or underfitting.

```
: from sklearn.model_selection import train_test_split

: xtrain,xtest,ytrain,ytest=train_test_split(X,y,test_size=0.2,random_state=42)
```

Figure 63 Holdout method

In our approach to training and evaluating classification models using machine learning and deep learning algorithms, we employ the `train_test_split` function from `sklearn`. Here's how we utilize its parameters:

The variable X represents our dataset containing Mel-Frequency Cepstral Coefficients (MFCC) features extracted from audio data. These features serve as inputs to our models and encapsulate essential characteristics of the speech signals we analyze.

The variable y corresponds to the target variable or class labels associated with each data point in X. These labels denote the emotions or categories we aim to predict based on the extracted MFCC features.

The parameter test_size is crucial as it determines the proportion of the dataset allocated to the testing set. By setting it to 0.2, we designate 20% of the data for testing purposes, ensuring that the majority, or 80%, is reserved for training our models. This division enables us to train our algorithms on a substantial portion of the data while retaining a separate segment for evaluating model performance on unseen data.

To maintain consistency and reproducibility in our experiments, we utilize the random_state parameter. This parameter fixes the random seed used during the data splitting process. By setting a specific random_state value, such as an integer, we ensure that each execution of the `train_test_split` function yields the same split of data into training and testing sets. This

consistency is vital for comparing results across different model configurations, debugging code, and ensuring that experimental findings are reliable and replicable.

In summary, leveraging these parameters in `train_test_split` enables us to effectively partition our dataset, train our models on a majority of the data, evaluate their performance on independent test data, and maintain consistency in our experimental setup for robust analysis and comparison of different machine learning and deep learning approaches.

7.3.2 Train the model :

In training the model for classification, we supply the training dataset to the model, enabling it to learn patterns and relationships within the data. By analyzing the provided features (such as MFCCs in audio data) and their corresponding labels (emotions or categories), the model adjusts its internal parameters to create a predictive model. This process involves optimizing its ability to generalize from the training data to predict labels for new, unseen data points accurately. Once trained, the model is prepared to make predictions on new data instances, leveraging the learned patterns to classify or predict outcomes based on the features presented.

As discussed previously in Chapter 4, we are implementing SVM and LSTM models to classify our dataset. SVM (Support Vector Machine) is a powerful supervised learning algorithm used for classification tasks, while LSTM (Long Short-Term Memory) is a type of recurrent neural network (RNN) commonly employed in deep learning for sequential data like speech or text. To train these models, we fit them on the training dataset. This process involves feeding the SVM or LSTM with the training data, allowing them to learn the patterns and relationships necessary to make accurate predictions on new, unseen data points. This training phase is crucial as it enables the models to optimize their parameters and internal representations to effectively classify the input data based on the features provided.

1. SVM :

Using sklearn, we employ SVM (Support Vector Machine) for classification by utilizing SVC (Support Vector Classifier). To train the SVM model, we fit it with the training dataset X_train and corresponding labels y_train. This process allows the SVM classifier to learn from the provided data and optimize its parameters to create a model capable of predicting labels accurately for new, unseen data points. Here’s how we implement it

```
[20]: model=SVC()
      model.fit(xtrain, ytrain)

[20]: SVC
      SVC()

[21]: ypred=model.predict(xtest)
      print(accuracy_score(ytest,ypred))

      0.9446428571428571
```

Figure 64 SVM model

2. LSTM :

To implement LSTM (Long Short-Term Memory) for classification using Keras, we proceed with several key steps. First, we import necessary libraries such as Sequential and LSTM from tensorflow.keras.models and tensorflow.keras.layers, respectively. With Keras, we define the LSTM model architecture by setting up a sequential stack of layers. For instance, we can add an LSTM layer with a specified number of units, typically chosen based on the complexity of the data and task requirements. Following the LSTM layer, a Dense layer is appended for final classification, where the number of units matches the number of classes (num_classes) and uses a softmax activation function for multi-class classification tasks.

```
[38]: from keras.models import Sequential
      from keras.layers import Dense, LSTM, Dropout

      model = Sequential([
          LSTM(256, return_sequences=False, input_shape=(40,1)),
          Dropout(0.2),
          Dense(128, activation='relu'),
          Dropout(0.2),
          Dense(64, activation='relu'),
          Dropout(0.2),
          Dense(7, activation='softmax')
      ])

      model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
      model.summary()

      # Train the model
      history = model.fit(X, y, validation_split=0.2, epochs=10, batch_size=64)
```

Model: "sequential"

Figure 65 LSTM model

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 256)	264,192
dropout (Dropout)	(None, 256)	0
dense (Dense)	(None, 128)	32,896
dropout_1 (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 64)	8,256
dropout_2 (Dropout)	(None, 64)	0
dense_2 (Dense)	(None, 7)	455

Total params: 305,799 (1.17 MB)
Trainable params: 305,799 (1.17 MB)
Non-trainable params: 0 (0.00 B)

```
Epoch 1/10
35/35 ━━━━━━━━━━━━━━━━━━━━ 3s 36ms/step - accuracy: 0.4633 - loss: 1.4246 - val_accuracy: 0.6661 - val_loss: 0.8184
Epoch 2/10
35/35 ━━━━━━━━━━━━━━━━━━━━ 1s 30ms/step - accuracy: 0.8464 - loss: 0.4420 - val_accuracy: 0.8964 - val_loss: 0.3115
Epoch 3/10
35/35 ━━━━━━━━━━━━━━━━━━━━ 1s 30ms/step - accuracy: 0.9433 - loss: 0.1741 - val_accuracy: 0.9643 - val_loss: 0.1338
Epoch 4/10
35/35 ━━━━━━━━━━━━━━━━━━━━ 1s 30ms/step - accuracy: 0.9580 - loss: 0.1372 - val_accuracy: 0.9768 - val_loss: 0.0754
Epoch 5/10
35/35 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - accuracy: 0.9693 - loss: 0.0976 - val_accuracy: 0.9446 - val_loss: 0.1282
Epoch 6/10
35/35 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - accuracy: 0.9658 - loss: 0.1178 - val_accuracy: 0.9500 - val_loss: 0.1301
Epoch 7/10
35/35 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - accuracy: 0.9637 - loss: 0.1150 - val_accuracy: 0.9875 - val_loss: 0.0474
Epoch 8/10
35/35 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - accuracy: 0.9845 - loss: 0.0521 - val_accuracy: 0.9911 - val_loss: 0.0257
Epoch 9/10
35/35 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - accuracy: 0.9838 - loss: 0.0579 - val_accuracy: 0.9768 - val_loss: 0.0637
Epoch 10/10
35/35 ━━━━━━━━━━━━━━━━━━━━ 1s 30ms/step - accuracy: 0.9606 - loss: 0.1201 - val_accuracy: 0.9661 - val_loss: 0.0861
```

Figure 66 LSTM Fitting

Chapter-8 Result & Conclusion

8.1 Performance Matrix For SVM

8.1.1 Accuracies For SVM:

The Support Vector Machine (SVM) model achieved an accuracy of 0.84, indicating that it correctly classified 84% of the instances in the dataset. This suggests that the SVM model performed relatively well in recognizing speech emotions, demonstrating its effectiveness in distinguishing between different emotional states expressed in speech recordings.

```
[34]: accuracy = accuracy_score(ytest, ypred)
      print("Accuracy:", accuracy)

      Accuracy: 0.8455188679245284
```

Figure 67 Accuracy of SVM

8.1.2 Confusion Matrix For SVM:

Here is the confusion matrix for the Support Vector Machine (SVM) model, which provides a detailed breakdown of the model's classification performance across different emotion categories.

```
[40]: plt.figure(figsize=(8,5))
      sns.heatmap(confusion_matrix(ytest, ypred),annot=True)
      plt.show()
```

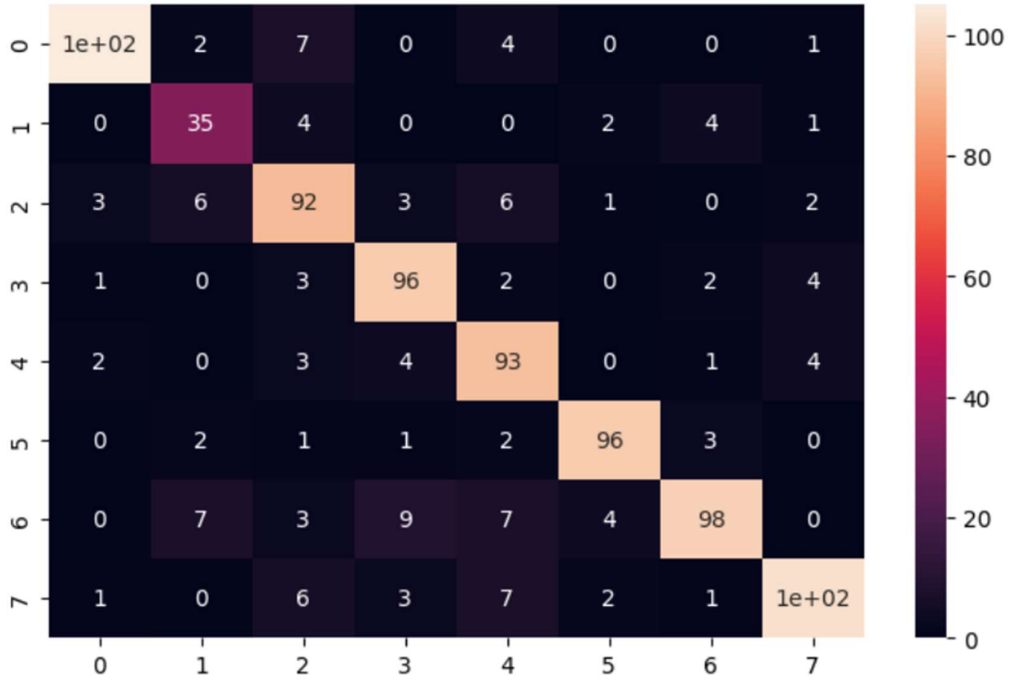


Figure 68 Confusion matrix for SVM

8.1.3 Classification Report for SVM

Here is the classification report for the Support Vector Machine (SVM) model, providing a comprehensive summary of its performance metrics including precision, recall, F1-score, and support for each class.

```
[38]: print(classification_report(ytest, ypred))
```

	precision	recall	f1-score	support
0	0.94	0.88	0.91	119
1	0.67	0.76	0.71	46
2	0.77	0.81	0.79	113
3	0.83	0.89	0.86	108
4	0.77	0.87	0.82	107
5	0.91	0.91	0.91	105
6	0.90	0.77	0.83	128
7	0.89	0.84	0.86	122
accuracy			0.85	848
macro avg	0.84	0.84	0.84	848
weighted avg	0.85	0.85	0.85	848

Figure 69 Classification report for SVM

8.1.4 Prdection Using speech recognition library

Performing speech emotion recognition using a speech recognition library that captures voice through a microphone and saves the temporary file for prediction is a common approach. Here's how you could outline the process:

Voice Capture: Use a speech recognition library to capture audio input from the microphone. Save the captured audio as a temporary file for further processing.

Preprocessing: Preprocess the audio file to extract relevant features for emotion recognition. This may include steps such as:

Resampling the audio to a standard sampling rate.

Applying noise reduction techniques to improve signal quality.

Segmenting the audio into smaller chunks for analysis.

Feature Extraction: Extract features from the preprocessed audio data. This could involve computing features such as MFCCs, energy, pitch, etc., which are commonly used in speech emotion recognition tasks.

Model Prediction: Feed the extracted features into your trained emotion recognition model to predict the emotion of the speaker. This could be a machine learning model or a deep learning model trained on a dataset of labeled speech samples.

Display Prediction: Display the predicted emotion to the user or use it for further processing in your application.

```
[ ]: # Initialize the recognizer
     r = sr.Recognizer()

[ ]: # Function to convert text to speech
     def SpeakText(command):
         engine = pyttsx3.init()
         engine.say(command)
         engine.runAndWait()
```

Figure 70 Speech recognition

```
# Function to recognize speech and classify emotion
def classify_emotion():
    try:
        with sr.Microphone() as source:
            r.adjust_for_ambient_noise(source, duration=0.2)
            print("Listening...")
            audio = r.listen(source, timeout=3)
            print("Processing...")

            # Save audio data to a temporary file
            with tempfile.NamedTemporaryFile(delete=False) as tmpfile:
                tmpfile.write(audio.get_wav_data())
                tmpfile_name = tmpfile.name

            # Extract MFCC features
            mfcc_features = extract_mfcc(tmpfile_name)
            mfcc_features = np.array(mfcc_features).reshape(1, -1)

            # Predict emotion label
            predicted_label_index = model.predict(mfcc_features)[0]
            label_mapping = {0: 'neutral',1: 'calm',2: 'happy',3: 'sad', 4: 'angry', 5: 'fear',6: 'disgust',7:'ps'}
            predicted_label = label_mapping[predicted_label_index]

            print("Predicted emotion:", predicted_label)

            # Convert recognized text to speech
            text = r.recognize_google(audio)
            text = text.lower()
            print("You said:", text)
            SpeakText(text)

            # Remove temporary file
            os.remove(tmpfile_name)

            return predicted_label

    except sr.RequestError as e:
        print("Could not request results; {0}".format(e))
    except sr.UnknownValueError:
        print("Unknown error occurred")
```

Figure 71 SER Model

```
In [41]: main()

         Listening...
         Processing...
         Predicted emotion: happy
         You said: hello hello

In [42]: main()

         Listening...
         Processing...
         Predicted emotion: angry
         You said: hey myself
```

Figure 72 Implementation

8.2 Performance Matrix For LSTM

8.2.1 Accuracies For LSTM:

The `evaluate` function is employed to quantitatively assess the performance of the LSTM model on a previously unseen test dataset. It computes various metrics, primarily accuracy in this context, by comparing the model's predictions against the ground truth labels present in the test dataset. This rigorous evaluation process provides a robust measure of the model's ability to generalize and make accurate predictions on new data points that it has not encountered during training. By validating the model's performance on independent test data, we ensure its reliability and effectiveness in practical applications, offering confidence in its predictive capabilities beyond the training phase.

```
[39]: loss, accuracy = model.evaluate(xtest, ytest)
      print("Test Loss:", loss)
      print("Test Accuracy:", accuracy)

      4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.9559 - loss: 0.1354
      Test Loss: 0.16866135597229004
      Test Accuracy: 0.9599999785423279
```

Figure 73 Accuracy for LSTM

8.1.2 Confusion Matrix For LSTM:

Here is the confusion matrix for the Long Short-Term Memory (LSTM) model, providing a detailed breakdown of the model's classification performance across different emotion categories.

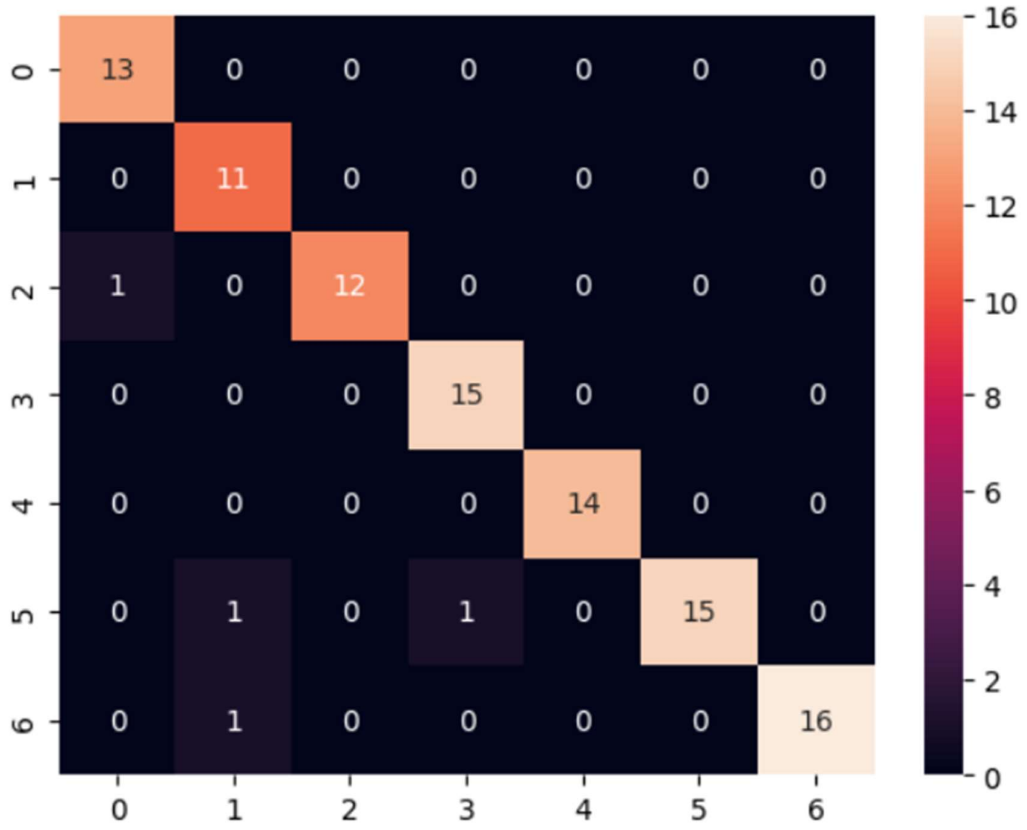


Figure 74 Confusion Matrics for LSTM

8.1.3 Classification Report for SVM :

Here is the classification report for the Support vector machine(SVM) model, presenting a comprehensive summary of its performance metrics including precision, recall, F1-score, and support for each class.

Classification Report:

	precision	recall	f1-score	support
0	0.93	1.00	0.96	13
1	0.85	1.00	0.92	11
2	1.00	0.92	0.96	13
3	0.94	1.00	0.97	15
4	1.00	1.00	1.00	14
5	1.00	0.88	0.94	17
6	1.00	0.94	0.97	17
accuracy			0.96	100
macro avg	0.96	0.96	0.96	100
weighted avg	0.96	0.96	0.96	100

Figure 75 Classification Report for LSTM

8.1.4 Graphical Represantation

We have visualized the training and validation accuracies, as well as the training and validation losses, to provide a graphical representation of the performance of the LSTM model throughout the training process. These graphs illustrate the model's learning dynamics, including its ability to generalize to unseen data and its convergence during training.

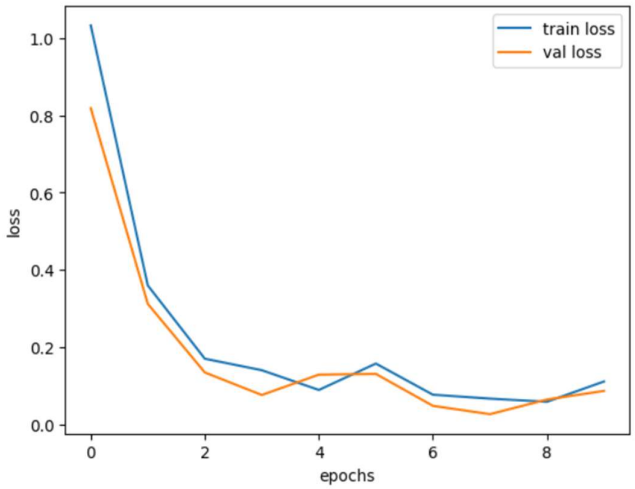


Figure 76 Loss vs Epochs

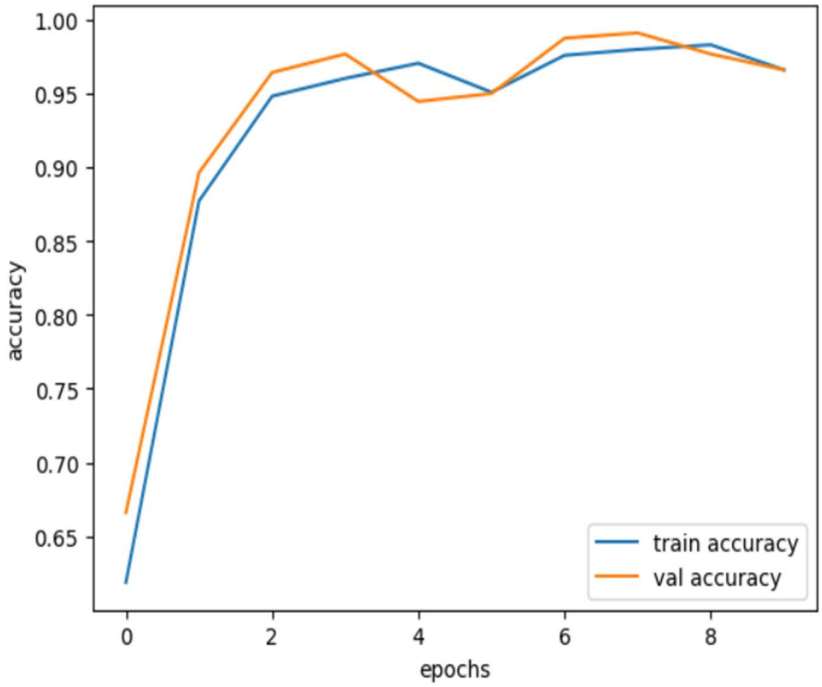


Figure 77 Accuracy vs Epochs

Chapter-9 References

[1] C. Busso *et al.*, "IEMOCAP: interactive emotional dyadic motion capture database," *Lang Resour Eval*, vol. 42, no. 4, pp. 335–359, Dec. 2008, doi: 10.1007/s10579-008-9076-6.

[2] Z.-T. Liu, B.-H. Wu, D.-Y. Li, P. Xiao, and J.-W. Mao, "Speech Emotion Recognition Based on Selective Interpolation Synthetic Minority Over-Sampling Technique in Small Sample Environment," *Sensors*, vol. 20, no. 8, p. 2297, Apr. 2020, doi: 10.3390/s20082297.

[3] S. Kuchibhotla, H. D. Vankayalapati, and K. R. Anne, "An optimal two stage feature selection for speech emotion recognition using acoustic features," *Int J Speech Technol*, vol. 19, no. 4, pp. 657–667, Dec. 2016, doi: 10.1007/s10772-016-9358-0.

[4] A. Graves, A. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, IEEE, May 2013, pp. 6645–6649. doi: 10.1109/ICASSP.2013.6638947.

[5] L. Kerkeni, Y. Serrestou, M. Mbarki, K. Raoof, M. Ali Mahjoub, and C. Cleder, "Automatic Speech Emotion Recognition Using Machine Learning," in *Social Media and Machine Learning*, IntechOpen, 2020. doi: 10.5772/intechopen.84856.

[6] S. K. Pandey, H. S. Shekhawat, and S. R. M. Prasanna, "Deep Learning Techniques for Speech Emotion Recognition: A Review," in *2019 29th International Conference Radioelektronika (RADIOELEKTRONIKA)*, IEEE, Apr. 2019, pp. 1–6. doi: 10.1109/RADIOELEK.2019.8733432.

[7] A. Amjad, L. Khan, and H. T. Chang, "Effect on speech emotion classification of a feature selection approach using a convolutional neural network," *PeerJ Comput Sci*, vol. 7, 2021, doi: 10.7717/PEERJ-CS.766.

[8] T.-W. Sun, "End-to-End Speech Emotion Recognition With Gender Information," *IEEE Access*, vol. 8, pp. 152423–152438, 2020, doi: 10.1109/ACCESS.2020.3017462.

[9] J. Zhao, X. Mao, and L. Chen, "Speech emotion recognition using deep 1D & 2D CNN LSTM networks," *Biomed Signal Process Control*, vol. 47, pp. 312–323, Jan. 2019, doi: 10.1016/j.bspc.2018.08.035.

[10] M. Ezz-Eldin, A. A. M. Khalaf, H. F. A. Hamed, and A. I. Hussein, "Efficient Feature-Aware Hybrid Model of Deep Learning Architectures for Speech Emotion Recognition," *IEEE Access*, vol. 9, pp. 19999–20011, 2021, doi: 10.1109/ACCESS.2021.3054345.

[11] Z. Zhao *et al.*, "Exploring Deep Spectrum Representations via Attention-Based Recurrent and Convolutional Neural Networks for Speech Emotion Recognition," *IEEE Access*, vol. 7, pp. 97515–97525, 2019, doi: 10.1109/ACCESS.2019.2928625.

[12] S. Zhang, S. Zhang, T. Huang, and W. Gao, "Speech Emotion Recognition Using Deep Convolutional Neural Network and Discriminant Temporal Pyramid Matching," *IEEE Trans Multimedia*, vol. 20, no. 6, pp. 1576–1590, Jun. 2018, doi: 10.1109/TMM.2017.2766843.

[13] E. Ghaleb, M. Popa, and S. Asteriadis, "Metric Learning Based Multimodal Audio-visual Emotion Recognition," *IEEE MultiMedia*, pp. 1–1, 2019, doi: 10.1109/MMUL.2019.2960219.

[14] A. Slimi, M. Hamroun, M. Zrigui, and H. Nicolas, "Emotion recognition from speech using spectrograms and shallow neural networks," in *Proceedings of the 18th International Conference on Advances in Mobile Computing & Multimedia*, New York, NY, USA: ACM, Nov. 2020, pp. 35–39. doi: 10.1145/3428690.3429153.

[15] S. F. Worgan and R. K. Moore, "Towards the detection of social dominance in dialogue," *Speech Commun*, vol. 53, no. 9–10, pp. 1104–1114, Nov. 2011, doi: 10.1016/j.specom.2010.12.004.

[16] P. Dhar and S. Guha, "A System to Predict Emotion from Bengali Speech," *International Journal of Mathematical Sciences and Computing*, vol. 7, no. 1, pp. 26–35, Feb. 2021, doi: 10.5815/ijmsc.2021.01.04.

[17] A. A. Abdelhamid *et al.*, "Robust Speech Emotion Recognition Using CNN+LSTM Based on Stochastic Fractal Search Optimization Algorithm," *IEEE Access*, vol. 10, pp. 49265–49284, 2022, doi: 10.1109/ACCESS.2022.3172954.

[18] C. Busso *et al.*, "IEMOCAP: interactive emotional dyadic motion capture database," *Lang Resour Eval*, vol. 42, no. 4, pp. 335–359, Dec. 2008, doi: 10.1007/s10579-008-9076-6.

[19] A. Koduru, H. B. Valiveti, and A. K. Budati, "Feature extraction algorithms to improve the speech emotion recognition rate," *Int J Speech Technol*, vol. 23, no. 1, pp. 45–55, Mar. 2020, doi: 10.1007/s10772-020-09672-4.

[20] S. Zhang, X. Zhao, and Q. Tian, “Spontaneous Speech Emotion Recognition Using Multiscale Deep Convolutional LSTM,” *IEEE Trans Affect Comput*, vol. 13, no. 2, pp. 680–688, Apr. 2022, doi: 10.1109/TAFFC.2019.2947464.

[21] R. A. Khalil, E. Jones, M. I. Babar, T. Jan, M. H. Zafar, and T. Alhussain, “Speech Emotion Recognition Using Deep Learning Techniques: A Review,” *IEEE Access*, vol. 7, pp. 117327–117345, 2019, doi: 10.1109/ACCESS.2019.2936124.