

# Chapter 1

## **INTRODUCTION**

Human language is so simple while we are talking to each other. We can understand what other people mean. Tone, semantics, expressions, etc. we can understand each other. While we are talking human beings' tone, semantics, expressions, etc. can mean other things or meanings, however we understand them all.

In writing also, one can write anything in his or her own words or sentences to express a meaning. Other people can understand them too. However, when we talk about machines (computer), machines don't understand the human language. Machines only understand machine language or binary codes. In order to make the machines understand the human language (words or sentences) we need to convert the human language (words or sentences) into numerical formats. We use these numerical formats to train and test the models, and then to predict the newly coming textual data.

In this project, the most crucial and first step is the preprocessing. Preprocessing is a must to reduce noise and unwanted things in our dataset. There are many steps or methods to preprocess a dataset. After preprocessing, the text data must convert into numerical values, so feature extraction is needed here. Applying feature extraction is a must thing, and we can use many feature extraction techniques. After converting the text into numerical forms or values, these values are then fed to a classifier to train and test the classifier. After training, a model is ready to test, then testing the model is there. Then the model is ready to deploy. This model will classify any newly coming text data base on the dataset.

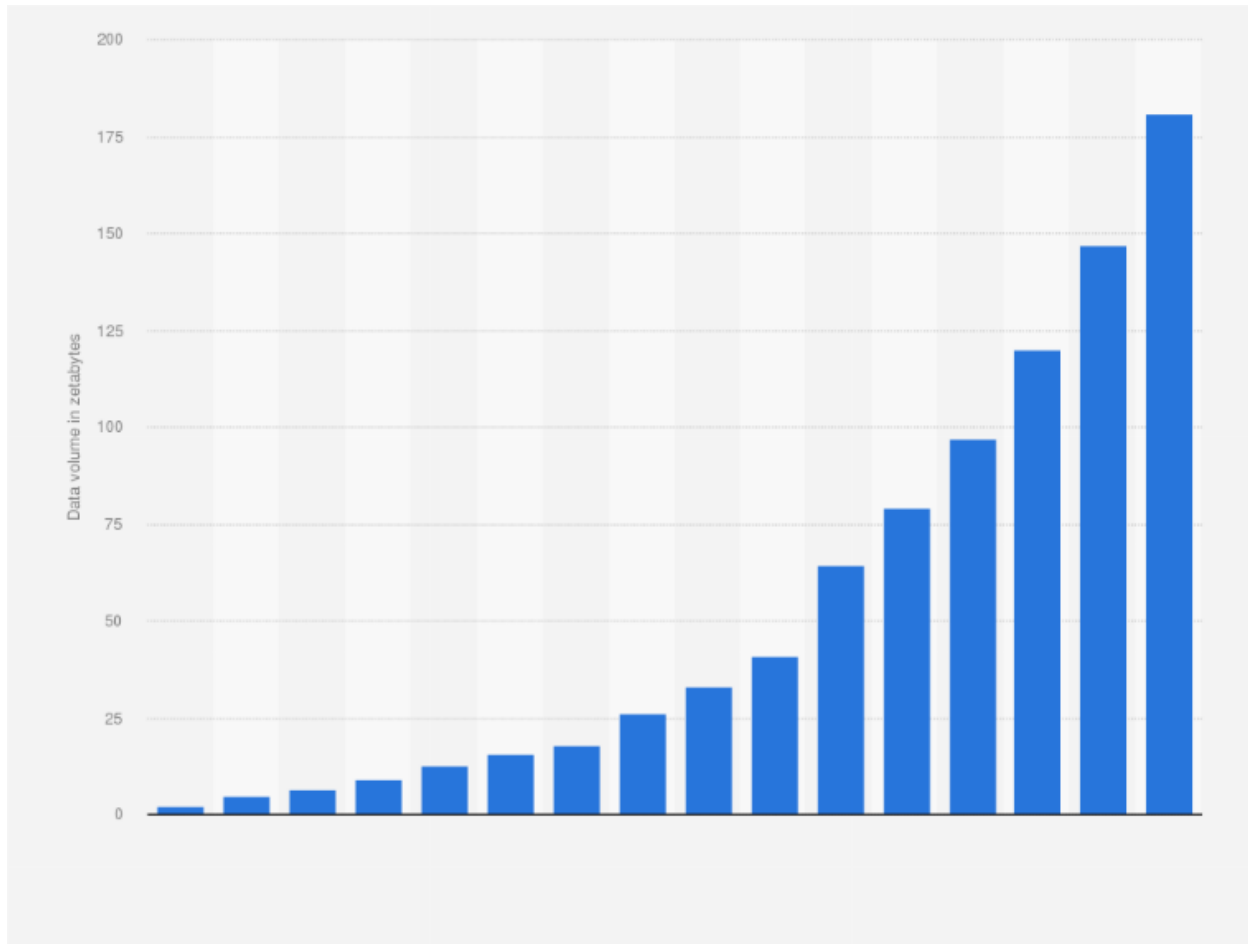
## **1.1 Data Growth:**

Data growth refers to the constant increase in the amount of data being produced by modern technological society as well as the increase in the amount of data an enterprise must store.

Data has been growing exponentially for the last few decades. According to the Data Services Market Report, the main drivers of growth are remote work, the digitization of existing processes, and the growing industrial sector using digital technologies, a rise in number of SMEs adopting digital technologies, growing used of Over-the-Top (OTT) services (a method of delivering film and television directly via the internet, without the need for traditional cable or satellite broadcasting), and the development of data-generating and data-hungry technologies, such as IoT or Machine Learning (ML).

The growth of data in recent years has been impressive, with the amount of data generated continuing to grow exponentially. The Internet and the rapid growth of Internet users over the years is one reason for this, as are the data created by large enterprises, tech companies, ecommerce platforms, health and scientific industries, governmental and non-governmental organizations. The tech giants, including Meta, Amazon, Google and Microsoft are major contributors to the growth of data.

The following chart is the growth of data from 2010 to 2025.



(Source: statista)

Fig-1: Growth of Data from 2010 to 2025

Since 2010 data has been growing (according to the chart above), and by 2030 the amount of data generated (grown) worldwide would increase significantly. According to current trends, experts estimate the data growth would reach 660 zettabytes or more.

## **1.2 Text Classification:**

Text or document classification or categorization is the process of assigning text documents into one or more classes or categories, assuming that we have a predefined set of classes. Documents here are textual documents, and each document can contain a sentence or even a paragraph of words. A text classification system would successfully be able to classify each document to its correct class(es) based on inherent properties of the document. The textual data involved can be anything ranging from a phrase, sentence, or a complete document with paragraphs of text, which can be obtained from corpora, blog or anywhere form the Web.

Text classification in NLP (Natural Language Processing) involves categorizing and assigning predefined labels or categories to text documents, sentences, or phrases based on their contents. Text classification aims to automatically determine the class or category to which a piece of text belongs. It's a fundamental task in NLP with numerous practical applications, including sentiment analysis, spam detection, topic labeling, language identification and more. Text classification algorithms analyze the features and patterns within the text to make accurate predictions about its category, enabling machines to organize, filter, and understand large volumes of textual data.

## **1.3 Problem Statement:**

Machines cannot understand our human languages. For example, English words can't be understood by machines, so how can we make machine understand those words? And how can we make machines those words to be classified or categorized?

The main problem is that when we are working with text data, first they are unstructured which is difficult to analyze, organize and extract meaningful insights. There will be information overload, lack of structure, inefficient search and limited automation.

Text (or document) classification or categorization comes into play. In text classification, the unstructured text data are converted into structured text data which makes easier to classify or organize the text data.

## **1.4 Motivation:**

First, I felt ‘how can we know this statement belongs to this class?’. In simple words, I got confused and rose a question like, for example in movie review, how can I know a review is positive or love review? And same goes to comments on various social media; Twitter, Facebook, Instagram, etc. like the question above. Now I know, we have NLP to do this by using many techniques like Machine Learning techniques, Deep Learning techniques, etc.

Using NLP, we can categorize or classify the text data. Let’s say, we have an incoming text data to a model or machine and we want to know to what this text data belongs. We do classification here. Even in the reviews, we can analyze what comments are negative comments and what comments are not.

## **1.5 Contribution:**

In order to classify some text data to the category or class it belongs, I used NLP with ML techniques. Here, in this project, I got twenty news group (20newsgroup) dataset (from UCI repository, uploaded by Tom Mitchell). This dataset has twenty news categories or classes. By using ML models, we can decide or know what a new coming text data does belong to which class.

To classify a new text data to which category it’s belonged is the main goal in this project. For any new text data, we are going to classify. Machines don’t understand the text data, so we convert the text data into numbers and feed them to the trained and test model (algorithm) to classify or predict to which category the text data belongs.

# Chapter 2

## **RELATED WORK**

Many beginners in NLP did works on text (or document) classification or categorization. This is done in Python (a general programming language). Some of the works are listed below:

- a. NLP Sentiment Analysis (COVID-19) by Tarik Emre on Kaggle. In this work, he used Machine Learning to analyze the sentiments.
- b. E-Commerce Text Classification by Mahendra on Kaggle. In this work, he used Machine Learning techniques.
- c. Exploring spam messages using NLP by Lamiaa Loukhmiri on Kaggle. In this work, she used ML techniques.
- d. Email Categorization by many developers (on many websites including medium) using ML techniques.

There are my related work in NLP which base on Machine Learning techniques. The works base on ML have some common things. The preprocessing step, feature extraction step, model training, model evaluation, model prediction, etc. In this, we clean, means preprocess, the raw dataset to remove noise, unusable words, special characters. Then, we assign numbers to the cleaned words to feed to the model, and to train and test the model because the model doesn't understand the text data. After training the model, we can use them to make some predictions.

# Chapter 3

## METHODOLOGY

The following is the conceptual figure of the project:

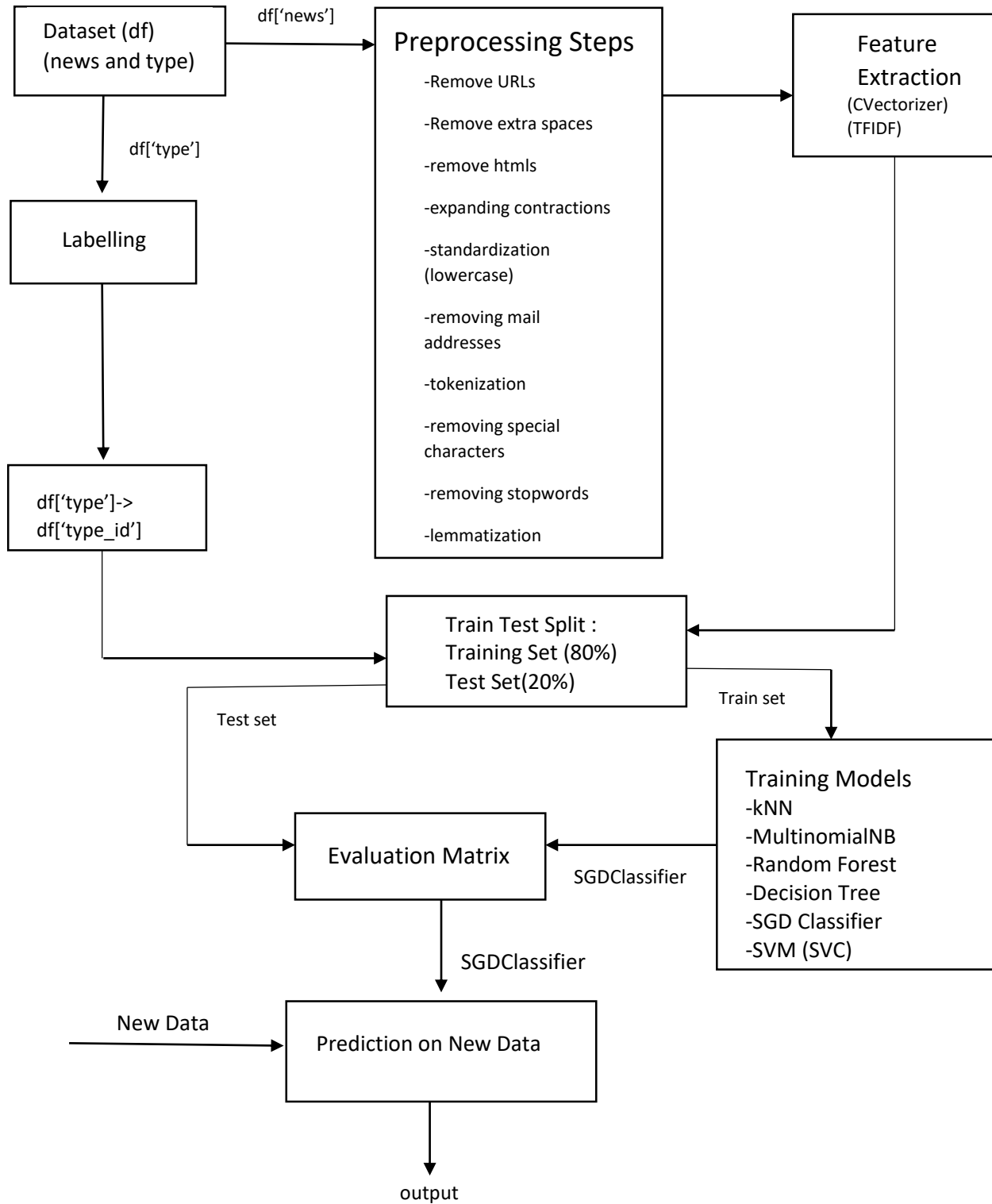


Fig-2: Conceptual figure

### **3.1 Dataset:**

The dataset 20newsgroups used in this project is from UCI Repository Dataset, created by Tom Mitchell.

The data type of the dataset is text. This dataset consists of 20000 messages taken from 20 newsgroups. There are 20 groups (folders) in the dataset. One thousand Usenet articles were taken from each of the 20 newsgroups (folder). Each newsgroup is stored in a subdirectory, with each article stored as a separate file.

(Source: <https://archive.ics.uci.edu/dataset/113/twenty+newsgroups>)

### **3.2 Preprocessing:**

Data preprocessing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model. Here in textual data, data preprocessing is also known as text preprocessing (NLP).

Text data can come from many sources like websites, social media, books, etc. The sources may have many unwanted things and noises. And unready or unreliable to train models. So, we do text preprocessing. Text preprocessing aims to remove or reduce the noise or unwanted things and making the dataset ready or reliable to use in training model.

In NLP, there are many steps involve in text preprocessing. Let's discuss the ones I included in this project.

1. Removing URLs:

When we have text dataset, there may be many URLs (Uniform Resource Locaters) pointed or linked to some websites. We don't need these in training the models, so we remove them.

2. Removing Extra Space:

There may be many extra white spaces, so in order to make our dataset look nice, we can remove those extra spaces.

3. Removing htmls:

The htmls tag are removed here, because they don't mean good to our models.



#### 4. Expanding Contractions:

Contractions are English words (I use English in my project). But in order to train models we don't need these contractions, so we expand common contractions.

#### 5. Standardization:

In here, I convert every word to lowercase to make training the model easier.

#### 6. Removing Mails addresses:

In the dataset I use, there are many mail addresses. And we don't need these in training the models, so I remove every one of them.

#### 7. Tokenization:

Tokenization is a technique that involves dividing a sentence or phrase into smaller units known as tokens. These tokens can encompass words, dates, punctuations marks, or even fragments of words.

#### 8. Removing Special Characters:

We don't special characters to train a model. Special characters include punctuations, ASCII code, etc. So, it's better to remove these characters.

#### 9. Removing Stopwords:

I used stopwords in NLTK library. We don't need these stop words to be included in training the model, so we need to remove them from the dataset.

#### 10. Lemmatization:

Lemmatization is a process of grouping together the different inflected forms of a word so they can be analyzed as a single item. Lemmatization is similar to stemming but it brings context to the words. Lemmatization is preferred over stemming because it does morphological analysis of the words.

### **3.3 Feature Extraction:**

In NLP, feature extraction is a fundamental task that involves converting raw data into a format that can be easily processed by Machine Learning (ML) algorithms.

Feature extraction is one of the most important steps to be followed for a better understanding of the context of what we are dealing with. After the initial text is cleaned (preprocessed), we need to transform it into its features to be used for modeling. Document data is not computable so it must be transformed into numerical data such as vectors. This transformation is generally called feature extraction in NLP. Feature extraction is also called Text Representation, Text Extraction, or Text Vectorization.

If we have textual data, that data we cannot feed to any ML algorithms because the ML algorithms don't understand text data. They only understand numerical data. So, we need Text Vectorization. There are some numbers of feature extraction (text vectorization) techniques. However, in my project I used only TF-IDF (Term Frequency and Inverse Document Frequency) which is one of the best for my project.

#### **3.3.1 TF-IDF (Term Frequency and Inverse Document Frequency):**

TF-IDF (Term Frequency-Inverse Document Frequency) is a statistical measure that evaluates how relevant a word is to a document in a collection of documents.

##### **Term Frequency (TF):**

The number of times a word appears in a document is divided by the total number of words in that document,  $0 < TF < 1$ .

$$TF_{ij} = \frac{\text{Number of times term } i \text{ appears in the document}}{\text{Total number of terms in the document } j}$$

### Inverse Document Frequency (IDF):

The logarithm of the number of documents in the corpus is divided by the number of documents where the specific term appears. In sklearn use  $\log(N/n_i)+1$  formula.

$$IDF_i = \log(\text{Total number of documents} / \text{Number of documents with term } i \text{ in it})$$

In TFIDF, we take a log to calculate IDF because if we have a very rare word, the IDF value without a log is very high, and then we have to calculate  $TF * IDF$ , at this time IDF value will dominate the TF value because TF lies in 0 and 1. That means we normalize the IDF value using a log.

```
(from sklearn.feature_extraction.text import TfidfVectorizer)
```

### **3.3.2 Count Vectorizer:**

CountVectorizer is a class in the scikit-learn library of Python that is used for converting a collection of text documents into a matrix of token counts. It creates a bag of words representation of the text corpus, where each document is represented as a vector of term frequencies. The countvectorizer class provides various options for preprocessing the text data, such as tokenization, removing stop words, and stemming. It also allows for the specification of the maximum vocabulary size and minimum document frequency required for a term to be included in the vocabulary. The resulting matrix can be used as input to various machine-learning models for tasks such as text classification and clustering.

```
(from sklearn.feature_extraction.text import CountVectorizer)
```

### **3.4 Classification:**

Classification is a supervised learning technique that is used to identify the category of new observations on the basis of training data. In classification, a program learns from the given dataset or observations and then classifies new observation into number of categories. The main objective of classification is to build a model that can accurately assign a label or category or class to a new observation based on its features.

Supervised Learning is a type of Machine Learning, where we have input variables (X) and an output variable (Y) and we use an algorithm to learn the mapping function from the input to the output  $Y=f(X)$ . The goal is to approximate the mapping function so well that when we have new input data (x), we can predict the output variables (Y) for that data.

In classification, we have two types: binary and multi-class classifier.

#### **Binary Classification:**

If the classification problem has only two possible outcomes (classes or labels or categories, etc.), then it is called as binary classification.

#### **Multi-Class Classification:**

If a classification problem has more than two classes or labels, etc., then it is called as multi-class classification.

The project I give is the multi-class classification having 20 (twenty) classes or labels or categories. The classifiers I used in this project are six in numbers. Let's define one by one:

### **3.4.1 Classifiers:**

A classifier is an algorithm that automatically categorizes data into one or more categories or classes. Targets, labels and categories are all terms used to describe classes.

#### **1. kNN:**

kNN is one of the simplest ML algorithms based on supervised learning technique. It assumes the similarity between the new case/date and available cases and put the new case into the category that is most similar to the available categories. When new data appears then it can be easily classified into a well suite category by using kNN algorithm. It is a non-parametric algorithm, which means it does not make any assumption on underlying data.

In kNN, k means the number of the nearest neighbors (we can select any number depending on the model). Calculate any distance producing method (Euclidean or Manhattan or any other) to find the similarity. The minimum distance between the new data point and some neighbors of a category is the solution, means the new data point will be fed or kept in that category.

I used kNN which is already available in sklearn library. And the value is k is 5.  
(from sklearn.neighbors import KNeighborsClassifier)

#### **2. Multinomial Naïve Bayes:**

Naïve Bayes algorithm or classifier is a supervised learning algorithm which is based on Bayes' Theorem of probability. It is probabilistic classifier, which means it predicts on the basis of the probability of an object.

In this project, I used Multinomial Naïve Bayes. It is a very popular and efficient ML algorithm that is based in Bayes' theorem. It is commonly used in text classification of multiple classes.

Multinomial Naïve Bayes is a probabilistic classifier to calculate the probability distribution of text data, which makes it well-suited for data with features that represent discrete frequencies or counts of events in various NLP tasks.

(from sklearn.naive\_bayes import MultinomialNB)

### 3. Random Forest:

Random Forest is a popular ML algorithm that belongs to the supervised learning technique. It can be used for both classification and regression problems. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.

As the name suggests, Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset. Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of prediction, and it predicts the final output.

I used sklearn Random Forest, which has default number of trees 100.

(from sklearn.ensemble import RandomForestClassifier)

### 4. Decision Tree:

Decision tree is a supervised learning technique that can be used for both classification and regression problems, but mostly it is preferred for solving classification problems. It is a tree-structured classifier, where internal nodes represent the outcome. In this, there are two nodes, which are the decision node and leaf node. Decision nodes are used to make any decision and have multiple branches, whereas leaf nodes are the output of those decisions and do not contain any further branches.

Decision tree is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions. It is called decision tree because it starts with the root node, which expands on further branches and constructs a tree-like structure. And they are non-parametric supervised learning method. And it can be used in both binary and multi-class classification.

(from sklearn.tree import DecisionTreeClassifier)

## 5. SGD Classifier:

SGD (Stochastic Gradient Descent) is one of the popular optimization methods in Deep Learning (DL) and ML. Large datasets and complicated models benefit greatly from its training. To minimize a loss function, SGD updates model parameters iteratively. It differentiates itself as stochastic by employing mini-batches, or random subsets, of the training data in each iteration, which introduces a degree of randomness while maximizing computational efficiency. By accelerating convergence, this randomness can aid in escaping local minima. Modern machine learning algorithms rely heavily on SGD because, despite its simplicity, it may be quite effective when combined with regularization strategies and suitable learning rate schedules.

SGD Classifier has been successfully applied to large-scale and sparse machine learning problems often encountered in text classification. It is merely an optimization technique and does not correspond to a specific family of machine learning models.

I used SGD Classifier given in sklearn library, which I can use in my project. It uses one-vs-rest approach. And default loss function is hinge loss.

(from sklearn.linear\_model import SGDClassifier)

## 6. SVM:

SVM (Support Vector Machine) is a supervised machine learning algorithm used for both classification and regression. The main objective of SVM is to find the optimal hyperplane in an N-dimensional space that can separate the data points in different classes in the feature space. The hyperplane tries that the margin between the closest points of different classes should be as maximum as possible. The dimension of the hyperplane depends upon the number of features.

In sklearn library, we have SVC and NuSVC. However, I used SVC in this project. In multi-class classification, SVC can perform one-vs-one and one-vs-all. Library sklearn gives decision\_function in SVC to specify it's ovo or ovr. Also, we can use LinearSVC for multiclass classification given by sklearn which implements ovr strategy. (from sklearn import svm)

### **3.5 Evaluation Metric:**

Evaluation metrics are tied to ML tasks. By looking at this matrix we can evaluate the models' performance. In ML, we have confusion matrix.

A confusion matrix gives a comparison between actual and predicted values. It is used for the optimization of ML models. The confusion matrix is an  $N \times N$  matrix, where  $N$  is the number of classes or outputs.

In this, we have; TP = True Positive, TN= True Negative, FP= False Positive, FN= False Negative.

TP = true positive value is where the actual value and predicted value are the same.

TN = true negative value for a class will be the sum of the values of all columns and rows except the values of that class we are calculating the values for.

FP = false positive value for a class will be the sum of values of the corresponding column except for the TP value.

FN = false negative value for a class will be the sum of values of corresponding rows except for the TP value.

The confusion matrix allows us to measure Recall and Precision, which, along with Accuracy and the AUC-ROC curve, are the metrics used to measure the performance of ML models.



# Chapter 4

## **TOOLS**

I used Anaconda navigator which has jupyter notebook in building this project, python programming language and also some libraries are used.

Choosing the right tools is one of the most important part in building any project or software. My project is based on ML and NLP so I chose Python as my base programming language, Anaconda navigator which is an open source distribution and some libraries of Python that are needed in NLP.

### **4.1 Python Programming Language:**

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective.

### **4.1.1 Python Over Other Programming Languages:**

Let's describe some advantages of Python over other programming language.

- Presence of third-party modules:  
Python has a rich ecosystem of third-party modules and libraries that extend its functionality for various tasks.
- Open source and large active community base:  
Python is open source, and it has a large and active community that contributes to its development and provides support.
- Versatile, easy to read and write:  
Python is known for its simplicity and readability, making it an excellent choice for both beginners and experienced programmers.
- User-friendly data structures:  
Python offers intuitive and easy-to-use data structures, simplifying data manipulation and management.
- Dynamically typed language:  
Python is dynamically typed, meaning we don't need to declare data types explicitly, making it flexible but still reliable.

### **4.1.2 Python in Machine Learning:**

Machine Learning (ML) is the field of study that gives computers the capability to learn without being explicitly programmed. It is one of the most exciting technologies that one has ever come across. As it is evident from the name, it gives the computer something that makes it more similar to humans: the ability to learn. ML is actively being used today, perhaps in many more places than one would expect.

Python has a crucial role in ML because Python provides libraries like NumPy, Pandas, Scikit-Learn, Tensorflow, and Keras. These libraries offer tools and functions essential for data manipulation, analysis, and building machine learning models. It is well-known for its readability and offers platform independence. These all things make it the perfect language choice for ML.

### **4.1.3 Python in Natural Language Processing:**

Natural Language Processing (NLP) is a subfield of Artificial Intelligence (AI). This is widely used technology for personal assistants that are used in various business fields/areas. This technology works on the speech provided by the user breaks it down for proper understanding and processes it accordingly. This is a very recent and effective approach due to which it has a really high demand in today's market. NLP is an upcoming field where already many transitions such as compatibility with smart devices, and interactive talks with a human have been made possible. Knowledge representation, logical reasoning, and constraint satisfaction were the emphasis of AI applications in NLP.

Python is a popular programming language for NLP tasks because it is easy to learn and use, and it has a large community of developers who contribute to its libraries and tools.

Python has a number of powerful libraries for NLP tasks, such as NLTK, spaCy, and TextBlob. These libraries provide a variety of functions and tools for tasks such as tokenization, stemming, lemmatization, POS tagging, NER (Named Entity Recognition), and sentiment analysis.

Python's string and file operations are straightforward, making tasks such as splitting a sentence at the white spaces a one-line command.

## **4.2 Libraries:**

A Python library is a collection of related modules. It contains bundles of code that can be used repeatedly in different programs. It makes Python programming simpler and convenient for the programmer. As we don't need to write the same code again and again for different programs. Python libraries play a very vital role in fields of ML, DS, DV, etc.

Below are libraries I used in this project:

### **NLTK (Natural Language Toolkit):**

NLTK is the main library for building Python projects to work with human language data. It gives simple to-utilize interfaces to more than 50 corpora and lexical assets like WordNet, alongside a set-up of text preprocessing libraries for tagging, parsing, classification, stemming, tokenization, and semantic conversation discussion. It is accessible for Windows, Mac OS and Linux. The best part is that NLTK is a free, open-source, local area-driven venture. It has some disadvantages as well. It is slow and difficult to match the demands of production usage. The learning curve is somehow steep.

### **Scikit-learn:**

It is one of the greater Python libraries for NLP and is most used among data scientists for NLP tasks. It provides a large number of algorithms to build machine learning models. It has excellent documentation that helps data scientists and makes learning easier. The main advantage of scikit learn is it has great intuitive class methods. It offers many functions for bag-of-words to convert text into numerical vectors. It has some disadvantages as well. It doesn't provide us with neural networks for text preprocessing. It is better to use other NLP libraries if we want to carry out more complex preprocessing, such as POS tagging for text corpora.

### **Other:**

In this project, I used pandas, string, re. Pandas for creating data frame, string for some preprocessing step, and re (Regular Expression) to remove or replace some symbols or words.

Then I used pickle and streamlit to make prediction interface for the new text data.

# Chapter 5

## RESULT, DISCUSSION AND IMPLEMENTATION

### 5.1 Result:

Result(s) of a project is the output(s) of the project.

First, let's put the result of classifiers using CountVectorizer.

MultinomialNB Classifier:

	precision	recall	f1-score	support
0	0.76	0.82	0.79	200
1	0.74	0.85	0.79	200
2	0.89	0.56	0.69	200
3	0.71	0.81	0.76	200
4	0.84	0.90	0.87	200
5	0.82	0.84	0.83	200
6	0.90	0.79	0.84	200
7	0.92	0.92	0.92	200
8	0.96	0.96	0.96	200
9	0.98	0.95	0.97	200
10	0.97	0.98	0.97	200
11	0.88	0.93	0.90	200
12	0.88	0.83	0.85	200
13	0.95	0.93	0.94	200
14	0.92	0.94	0.93	200
15	0.94	0.98	0.96	200
16	0.78	0.87	0.82	200
17	0.91	0.95	0.93	200
18	0.67	0.72	0.70	200
19	0.68	0.51	0.58	200
accuracy			0.85	4000
macro avg	0.85	0.85	0.85	4000
weighted avg	0.85	0.85	0.85	4000

---

0.9391760955179096

Training Accuracy

For MultinomialNB, we got the accuracy 85% in test and 93% in training.

kNN classifier:

	precision	recall	f1-score	support
0	0.37	0.62	0.46	200
1	0.29	0.47	0.36	200
2	0.35	0.59	0.44	200
3	0.57	0.38	0.45	200
4	0.30	0.59	0.40	200
5	0.29	0.55	0.38	200
6	0.40	0.63	0.49	200
7	0.73	0.49	0.59	200
8	0.67	0.47	0.55	200
9	0.84	0.43	0.57	200
10	0.89	0.58	0.70	200
11	0.95	0.54	0.69	200
12	0.68	0.36	0.47	200
13	0.76	0.45	0.56	200
14	0.87	0.61	0.72	200
15	0.94	0.89	0.91	200
16	0.69	0.56	0.61	200
17	0.72	0.69	0.70	200
18	0.77	0.47	0.59	200
19	0.58	0.39	0.46	200
accuracy			0.54	4000
macro avg	0.63	0.54	0.56	4000
weighted avg	0.63	0.54	0.56	4000

0.7361380258798524

Training Accuracy

For kNN classifier, we got accuracy 54% in testing and 73% in training.

For RandomForestClassifier:

	precision	recall	f1-score	support
0	0.77	0.73	0.75	200
1	0.65	0.76	0.70	200
2	0.72	0.80	0.75	200
3	0.75	0.73	0.74	200
4	0.83	0.80	0.82	200
5	0.83	0.80	0.82	200
6	0.77	0.85	0.81	200
7	0.86	0.91	0.88	200
8	0.96	0.92	0.94	200
9	0.91	0.92	0.92	200
10	0.95	0.96	0.96	200
11	0.94	0.94	0.94	200
12	0.85	0.72	0.78	200
13	0.87	0.84	0.85	200
14	0.92	0.92	0.92	200
15	0.95	1.00	0.97	200
16	0.77	0.84	0.80	200
17	0.91	0.94	0.93	200
18	0.70	0.56	0.63	200
19	0.54	0.49	0.51	200
accuracy			0.82	4000
macro avg	0.82	0.82	0.82	4000
weighted avg	0.82	0.82	0.82	4000

0.979246108645371

Training Accuracy

We got the accuracy of 82% in testing and 97% in training.

For DecisionTreeClassifier:

	precision	recall	f1-score	support
0	0.62	0.65	0.63	200
1	0.49	0.49	0.49	200
2	0.59	0.60	0.60	200
3	0.51	0.56	0.53	200
4	0.63	0.59	0.61	200
5	0.58	0.51	0.54	200
6	0.68	0.68	0.68	200
7	0.67	0.77	0.71	200
8	0.83	0.81	0.82	200
9	0.72	0.74	0.73	200
10	0.88	0.79	0.83	200
11	0.91	0.79	0.84	200
12	0.49	0.51	0.50	200
13	0.66	0.69	0.68	200
14	0.71	0.73	0.72	200
15	1.00	1.00	1.00	200
16	0.63	0.63	0.63	200
17	0.84	0.78	0.81	200
18	0.43	0.53	0.48	200
19	0.42	0.36	0.39	200
accuracy			0.66	4000
macro avg	0.66	0.66	0.66	4000
weighted avg	0.66	0.66	0.66	4000

---

0.979246108645371

### Training Accuracy

We got the accuracy of 66% in testing and 97% in training.

For SVM (SVC) classifier:

	precision	recall	f1-score	support
0	0.83	0.68	0.75	200
1	0.53	0.79	0.63	200
2	0.78	0.65	0.70	200
3	0.74	0.73	0.73	200
4	0.88	0.81	0.84	200
5	0.78	0.73	0.76	200
6	0.73	0.86	0.79	200
7	0.88	0.85	0.87	200
8	0.93	0.83	0.88	200
9	0.89	0.83	0.86	200
10	0.98	0.89	0.93	200
11	0.98	0.83	0.90	200
12	0.62	0.82	0.71	200
13	0.88	0.84	0.86	200
14	0.95	0.82	0.88	200
15	0.99	0.95	0.97	200
16	0.80	0.75	0.78	200
17	0.64	0.85	0.73	200
18	0.74	0.58	0.65	200
19	0.63	0.69	0.66	200
accuracy			0.79	4000
macro avg	0.81	0.79	0.79	4000
weighted avg	0.81	0.79	0.79	4000

0.9040445083453148

### Training Accuracy

We got accuracy of 79% in testing and 90% in training..



For SGD Classifier:

	precision	recall	f1-score	support
0	0.72	0.77	0.74	200
1	0.70	0.77	0.73	200
2	0.72	0.73	0.73	200
3	0.81	0.68	0.74	200
4	0.77	0.80	0.78	200
5	0.79	0.81	0.80	200
6	0.88	0.81	0.84	200
7	0.95	0.89	0.92	200
8	0.96	0.94	0.95	200
9	0.95	0.94	0.95	200
10	0.93	0.97	0.95	200
11	0.94	0.82	0.88	200
12	0.73	0.83	0.78	200
13	0.91	0.87	0.89	200
14	0.88	0.93	0.91	200
15	0.99	0.97	0.98	200
16	0.83	0.81	0.82	200
17	0.90	0.90	0.90	200
18	0.64	0.68	0.66	200
19	0.55	0.55	0.55	200
accuracy			0.82	4000
macro avg	0.83	0.82	0.82	4000
weighted avg	0.83	0.82	0.82	4000

0.9780583859473652

Training Accuracy

We got the accuracy of 82% in testing and 97% in training.

Results of classifiers of using TFIDF:

kNN classifier:

0	0.70	0.78	0.73	200
1	0.70	0.74	0.72	200
2	0.65	0.73	0.69	200
3	0.67	0.71	0.69	200
4	0.75	0.77	0.76	200
5	0.80	0.75	0.77	200
6	0.76	0.58	0.66	200
7	0.85	0.89	0.87	200
8	0.95	0.89	0.92	200
9	0.92	0.93	0.92	200
10	0.89	0.96	0.93	200
11	0.88	0.94	0.91	200
12	0.84	0.82	0.83	200
13	0.88	0.84	0.86	200
14	0.91	0.90	0.90	200
15	0.92	0.88	0.90	200
16	0.85	0.80	0.82	200
17	0.90	0.94	0.92	200
18	0.76	0.68	0.72	200
19	0.56	0.56	0.56	200
accuracy				0.80
macro avg				0.81
weighted avg				0.81

0.8862911795961743

Training Accuracy

We got the accuracy of 80% in testing and 88% in training.

MultinomialNB:

	precision	recall	f1-score	support
0	0.74	0.81	0.78	200
1	0.85	0.80	0.82	200
2	0.81	0.81	0.81	200
3	0.75	0.81	0.77	200
4	0.85	0.90	0.87	200
5	0.89	0.85	0.87	200
6	0.92	0.78	0.85	200
7	0.91	0.91	0.91	200
8	0.96	0.95	0.96	200
9	0.96	0.97	0.97	200
10	0.97	0.98	0.98	200
11	0.88	0.94	0.91	200
12	0.90	0.81	0.85	200
13	0.98	0.89	0.93	200
14	0.91	0.95	0.93	200
15	0.89	0.99	0.94	200
16	0.69	0.91	0.79	200
17	0.89	0.95	0.92	200
18	0.73	0.65	0.69	200
19	0.66	0.47	0.55	200
accuracy				0.86
macro avg				0.86
weighted avg				0.86

---

0.9341126461211477

### Training Accuracy

We got the accuracy of 86% in testing and 93% in training.

RandomForestClassifier:

	precision	recall	f1-score	support
0	0.77	0.73	0.75	200
1	0.66	0.75	0.70	200
2	0.74	0.79	0.76	200
3	0.72	0.69	0.70	200
4	0.84	0.81	0.82	200
5	0.83	0.79	0.81	200
6	0.74	0.88	0.80	200
7	0.87	0.91	0.89	200
8	0.93	0.90	0.92	200
9	0.92	0.94	0.93	200
10	0.95	0.97	0.96	200
11	0.96	0.94	0.95	200
12	0.82	0.69	0.74	200
13	0.90	0.84	0.87	200
14	0.91	0.92	0.92	200
15	0.93	1.00	0.96	200
16	0.76	0.84	0.80	200
17	0.92	0.95	0.93	200
18	0.71	0.60	0.65	200
19	0.54	0.49	0.51	200
accuracy			0.82	4000
macro avg	0.82	0.82	0.82	4000
weighted avg	0.82	0.82	0.82	4000

0.979246108645371

### Training Accuracy

We got the accuracy of 82% in testing and 97% in training.

DecisionTreeClassifier:

	precision	recall	f1-score	support
0	0.59	0.64	0.61	200
1	0.48	0.50	0.49	200
2	0.61	0.58	0.60	200
3	0.50	0.53	0.51	200
4	0.60	0.62	0.61	200
5	0.58	0.56	0.56	200
6	0.64	0.68	0.66	200
7	0.65	0.75	0.70	200
8	0.84	0.78	0.81	200
9	0.69	0.72	0.70	200
10	0.85	0.78	0.81	200
11	0.85	0.76	0.80	200
12	0.49	0.49	0.49	200
13	0.69	0.67	0.68	200
14	0.73	0.72	0.73	200
15	0.98	0.99	0.99	200
16	0.64	0.62	0.63	200
17	0.82	0.81	0.81	200
18	0.47	0.49	0.48	200
19	0.39	0.34	0.36	200
accuracy			0.65	4000
macro avg	0.65	0.65	0.65	4000
weighted avg	0.65	0.65	0.65	4000

0.979246108645371

Training Accuracy

We got the accuracy of 65% in testing and 97% in training.

SGDClassifier:

	precision	recall	f1-score	support
0	0.80	0.82	0.81	200
1	0.84	0.82	0.83	200
2	0.81	0.79	0.80	200
3	0.82	0.78	0.80	200
4	0.90	0.89	0.89	200
5	0.87	0.87	0.87	200
6	0.87	0.89	0.88	200
7	0.93	0.96	0.95	200
8	0.96	0.98	0.97	200
9	0.94	0.98	0.96	200
10	0.97	0.99	0.98	200
11	0.96	0.96	0.96	200
12	0.88	0.88	0.88	200
13	0.95	0.96	0.95	200
14	0.96	0.97	0.96	200
15	0.98	0.99	0.99	200
16	0.84	0.89	0.86	200
17	0.90	0.97	0.94	200
18	0.78	0.64	0.70	200
19	0.64	0.59	0.61	200
accuracy			0.88	4000
macro avg	0.88	0.88	0.88	4000
weighted avg	0.88	0.88	0.88	4000

0.9650559479902482

Training Accuracy

We got the accuracy of 88% in testing and 96% in training.

SVM (SVC):

	precision	recall	f1-score	support
0	0.78	0.77	0.77	200
1	0.74	0.88	0.81	200
2	0.87	0.81	0.84	200
3	0.80	0.80	0.80	200
4	0.92	0.84	0.88	200
5	0.84	0.88	0.86	200
6	0.86	0.90	0.88	200
7	0.92	0.94	0.93	200
8	0.99	0.95	0.97	200
9	0.98	0.96	0.97	200
10	0.99	0.97	0.98	200
11	1.00	0.92	0.96	200
12	0.81	0.89	0.85	200
13	0.96	0.95	0.95	200
14	0.98	0.95	0.97	200
15	0.99	0.99	0.99	200
16	0.85	0.84	0.84	200
17	0.94	0.93	0.93	200
18	0.69	0.69	0.69	200
19	0.60	0.60	0.60	200
accuracy			0.87	4000
macro avg	0.88	0.87	0.87	4000
weighted avg	0.88	0.87	0.87	4000

0.9754954053885103

Training Accuracy

We got the accuracy of 87% in testing and 97% in training.

## **5.2 Discussion:**

We trained the models (six models) with the extracted features using CountVectorizer and TFIDF. Among them, let's choose which classifier has the highest accuracy.

Let's draw a table of accuracy:

<b>Classifiers</b>	<b>Count Vectorizer (Training Accuracy)</b>	<b>Count Vectorizer (Testing Accuracy)</b>	<b>TFIDF (Training Accuracy)</b>	<b>TFIDF (Testing Accuracy)</b>
kNN classifier	73%	54%	88%	80%
MultinomialNB	93%	85%	93%	86%
RandomForestClassifier	97%	82%	97%	82%
DecisionTreeClassifier	97%	66%	97%	65%
SGDClassifier	97%	82%	96%	88%
SVM (SVC)	90%	79%	97%	87%

After training the models, we got the different accuracy. From the table above, we can say, the accuracy of the model using those classifier are different. The model which is trained using feature extraction CountVectorizer produce somewhat less accuracy when compared to the model which is trained using TFIDF. However, as we can see, RandomForestClassifier, MultinomialNB classifier, and SGDClassifier, they have the high accuracy. Among them, if we look closely, we can say SGDClassifier has better accuracy in both cases.

In the training with CountVectorizer, we got 82pc for SGDClassifier and we got 88pc with TFIDF. And from the above table, we can conclude that the training with TFIDF somewhat produces better accuracy. So, TFIDF is better in my project.

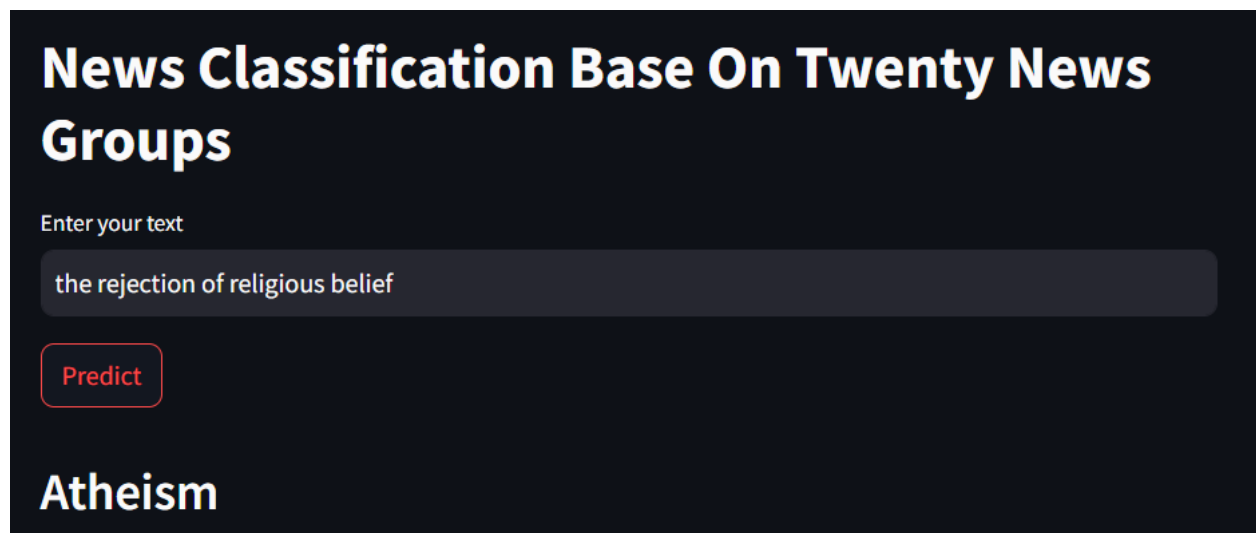
To implement or to predict in real time, I choose SGDClassifier, which follows 'ovr' strategy, with TFIDF.

### 5.3 Implementation:

In implementation, as mentioned above, I use the trained model with SGDClassifier and TFIDF feature extraction.

And, we can write code for prediction of new data very shortly by using model.predict (as an example), however I wanted my project to look nice so I used pickle and streamlit. In my project, pickle is to dump the trained model and the vectorizer files. And streamlit is for create a web based interface. After we create the web based interface, we can run the code as streamlit run appName.py.

Below are some outputs I got when I run the model to predict new text data.



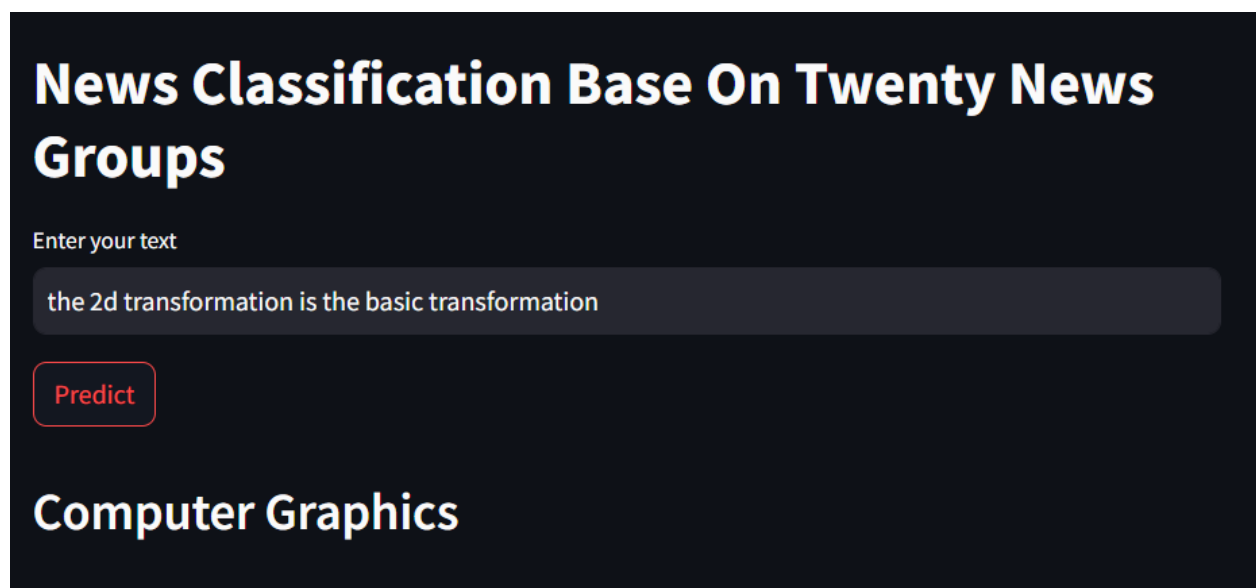
**News Classification Base On Twenty News Groups**

Enter your text

the rejection of religious belief

Predict

**Atheism**



**News Classification Base On Twenty News Groups**

Enter your text

the 2d transformation is the basic transformation

Predict

**Computer Graphics**



# News Classification Base On Twenty News Groups

Enter your text

the windows operating system is the best for beginners in computer world. it gives basic user friendly L

Predict

Computer os windows

# News Classification Base On Twenty News Groups

Enter your text

if you are looking for laptop to use in programming, video editing, apple laptops have the best OS

Predict

Computer system MAC hardware

# News Classification Base On Twenty News Groups

Enter your text

the new car by hyundai is available for sale

Predict

Forsale

# News Classification Base On Twenty News Groups

Enter your text

Kawasaki ninja h2 is one of the fastest bike in the world

Predict

**Motorcycles**

# News Classification Base On Twenty News Groups

Enter your text

recently NASA found new planet which is like out planet Earth

Predict

**Science Space**

# News Classification Base On Twenty News Groups

Enter your text

Stanley Cup is the oldest professional sports trophy in northern America

Predict

**Sports Hockey**

# News Classification Base On Twenty News Groups

Enter your text

we need an Earth wire to connect to the Earth surface, it is also known as neutral wire

Predict

**Science Electronics**

These are some outputs of my project.

# Chapter 6

## CONCLUSION

In my project, I used ML techniques to classify the news. When a user gives a new news to the model, it will classify to which news group it belongs.

I used ten steps in preprocessing, the most important steps are standardization, tokenization, removing stop words and lemmatization. The dataset we got is raw dataset, no preprocessed means the unwanted things or noises are in the dataset, so preprocessing is the way to remove these.

The text data cannot be understood by classifiers, classifiers only understand the numerical values or data. So, we need to convert the preprocessed text data into numerical data. In this, I used both CountVectorizer and TFIDF. These two are feature extraction techniques.

After converting into numerical data, I trained and tested six classifiers. Among them, SGDClassifier with TFIDF is the one which gives the highest accuracy in testing. This classifier works efficiently on large-scale and sparse data due to its stochastic nature, making it suitable for datasets with a high number of samples and features. It can also handles large datasets with ease.

MultinomialNB classifier gives the highest accuracy in testing which is trained with CountVectorizer. It is a classifier which base on Naïve Bayes theorem of probability. It is popular in NLP in text classification. It calculates the likelihood for a given sample and outputs the tag with the greatest chance.

Here, my project's number of feature is more than 100k. When I give the number of features as 20k (reducing number of features), classifiers give less accuracy and again I give 50k number of features, and however the classifiers give same accuracy as in no specified number of features.

And for prediction of newly coming news, means for prediction purpose, I use SGDClassifier with TFIDF. Because it gives the highest accuracy in testing. When we want to classify a piece of news or a sentence or simply a news, my project will classify or predict the class to what it belongs base on the twenty news groups I used.

I am grateful that, my guide, Dr. Ksh. Nareshkumar helped me a lot in achieving this.

## **6.1 Scope:**

Human languages are difficult to understand. I am human being, but even I cannot understand most of the languages in the world. However in this project, I made a model to understand twenty news groups which are written in English. Briefly, this model can understand English language (but not all human interpretation).

I did this project with ML techniques, but we can also do this using DL techniques. I didn't do the data augmentation here in my project, we can also do this. In NLP, we have easy data augmentation, backtranslation and generative models to perform data augmentation. Using these, may increase the accuracy.

I used only twenty news groups here. We can also try using more than this, and more features than this. This may give higher accuracy.

CountVectorizer and TFIDF are the feature extraction techniques I used in this project. We can also use Word2Vec, FastText, Glove in feature extraction. This may rise the accuracy and predict with higher accuracy too. I didn't use dimensional reduction techniques like PCA, LDA, etc. we can also use this to reduce the dimension. Dimensional reduction techniques are the ones that can reduce the high dimension to lower dimension ones. Means, a large number of features can be reduced to a lower number of features.

The classifiers I used are from ML. I also think, we can use CNN (Convolution Neural Networks), RNN (Recurrent Neural Networks), Transformer, LSTM, etc. These are DL models.

# Chapter 7

## **REFERENCES**

1. Sarkar D. 2016. “Text Analytics with Python- A practical Real-World Approach to Gaining Actionable Insights from Your Data”. Apress.
2. [linkedin.com/pulse/data-growth-what-means-your-business-101-data-solutions](https://www.linkedin.com/pulse/data-growth-what-means-your-business-101-data-solutions)
3. [idera.com/glossary/data-growth/](https://idera.com/glossary/data-growth/)
4. Phiri M. 2022. “Exponential Growth of Data”. [medium.com/@mwlimu/exponential-growth-of-data-2f53df89124](https://medium.com/@mwlimu/exponential-growth-of-data-2f53df89124)
5. Afzal A. 2022. “Step-by-step Explanation of Text Classification”. [analyticsvidhya.com/blog/2022/08/step-by-step-explantation-of-text-classification/](https://analyticsvidhya.com/blog/2022/08/step-by-step-explantation-of-text-classification/)
6. Kaggle.com
7. Medium.com
8. Analyticsvidhya.com
9. Towardsdatascience.com
10. Scikit-learn.org