# PROJECT REPORT

# ON

# Sentiment Analysis using Machine Learning

Six Months Industrial Training Report
At
IKTARA DATA SCIENCE

Submitted in partial fulfilment of the Requirement for the award of the degree Of

Bachelor of Technology In

COMPUTER SCIENCE & ENGINEERING


Submitted By

Sumit Das

20015004051

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

ECHELON INSTITUTE OF TECHNOLOGY, FARIDABAD

JANUARY 2024 – JULY 2024

IKTARA

**Internship Certificate**

TO WHOMSOEVER IT MAY CONCERN

This is to certify that Mr. Sumit Das has completed his internship as a data science Intern at IKTARA Data Sciences private limited from January 02, 2024 to July 12, 2024.

He worked on IKTARA AI Platform development project. As part of the project he worked on different modules assigned to him. During his internship he worked with dedication and learned new skills through self-motivation.

For IKTARA DATA SCIENCES PVT. LTD.

Director/Auth. Signatory

Sandeep Mahajan

Founder & CEO,

12-July-2024

**Iktara Data Sciences Pvt Ltd**
Unit #928, Puri Business Hub, Sector 81, Faridabad,121002
**Contact** - info@iktara.ai **website** - www.iktara.ai

# CERTIFICATE

I hereby certify that the work which is being presented in the B.Tech. Project Report entitled, **Sentiment Analysis** in partial fulfilment of the requirements for the award of the Bachelor of Technology in Computer Science & Engineering and submitted to the Department of Computer Science & Engineering of Echelon Institute of Technology, Faridabad is an authentic record of my own work carried out during a period from January 2024 to July 2024.

I have not submitted the matter presented in this thesis for the award of any other degree elsewhere.

Signature of Candidate

Sumit Das

20015004051

# TO WHOM IT MAY CONCERN

This is to certify that the Project entitled "**Sentiment Analysis**" submitted by "**Sumit Das**" (20015004051) Department of Computer Science and Engineering, Echelon Institute of Technology Under YMCA University, Faridabad, for partial fulfilment of the requirements for the degree of Bachelor of Technology in Computer Science & Engineering; is a bonafide record of the work and investigations carried out by him under my supervision and guidance

Signature of the Supervisor

Dr. Stuti Saxena

Associate Professor, CSE Dept.

Signature of HOD

Dr. Manish Kumar

Associate Professor , Head of Department

# ACKNOWLEDGEMENTS

Thank all those who have helped me in completing the project successfully.

I would like to express my gratitude to **Dr. Stuti Saxena**, who as my guide/mentor provided me with every possible support and guidance throughout the development of the project. This project would never have been completed without her encouragement and support.

I would also like to show my gratitude to **Dr Manish Sir,** Head of Department for providing us with well-trained Team members and giving us all the required resources and a healthy environment for carrying out our project work.

I sincerely thank to IKTARA DATA SCIENCES Team for helping me during the industrial training.

Sumit Das

20015004051

# TABLE OF CONTENTS

# Chapter No. 1

# COMPANY PROFILE

# CHAPTER – 1. COMPANY PROFILE

**Iktara Datasciences: Navigating Complexity with Elegant Solutions**

In today's fast-paced digital landscape, data science and wireless technologies are not just tools but vital catalysts for business transformation and innovation. At Iktara Datasciences, we understand the immense potential these fields hold and are dedicated to helping our clients harness their power to drive meaningful change and growth.

Our inspiration comes from the Iktara, an ancient, one-stringed musical instrument that, despite its simplicity, produces profoundly beautiful and powerful music. This instrument symbolizes our core belief: that simplicity, when applied to complex problems, can yield extraordinary results. Just as the Iktara has stood the test of time, our solutions are designed to deliver lasting impact.

**Comprehensive Data Science Solutions**

Data is the new oil, and at Iktara Datasciences, we turn this raw resource into refined insights and actionable intelligence. Our data science services span a wide range of applications, ensuring that your business can make informed decisions based on accurate, real-time data.

Predictive Analytics

Our predictive analytics tools help businesses anticipate market trends, customer behavior, and operational challenges. By leveraging advanced statistical techniques and machine learning algorithms, we provide forecasts that enable proactive decision-making and strategic planning.

Data Visualization

Understanding data is key to leveraging it. Our data visualization services transform complex datasets into intuitive, interactive visual representations. These visualizations help stakeholders at all levels to grasp insights quickly, facilitating better communication and faster decision-making.

Machine Learning and AI

At the heart of our data science offerings are custom machine learning models and AI solutions. These models are designed to learn from your data and continuously improve, providing sophisticated and tailored solutions to enhance business processes, optimize performance, and drive innovation.

Big Data Management

In the era of big data, managing vast amounts of information efficiently is crucial. Our big data management services ensure that your data is stored, processed, and analyzed efficiently, enabling you to unlock its full potential. From setting up data lakes to implementing Hadoop ecosystems, we cover it all.

Cutting-Edge Wireless Technologies

Connectivity is the backbone of modern business operations. Our wireless technology solutions ensure that your business remains agile, responsive, and competitive in an interconnected world.

Internet of Things (IoT)

Our IoT solutions integrate and manage devices across your operations, providing real-time data and insights. Whether it's smart manufacturing, healthcare monitoring, or home automation, we help you implement IoT systems that drive efficiency and innovation.

5G Network Deployment

The advent of 5G brings unprecedented speed and reliability to wireless communications. We assist businesses in deploying and optimizing 5G networks, ensuring they can leverage this technology for enhanced connectivity, faster data transfer, and more responsive applications.

Wireless Network Design

Every business has unique connectivity needs. Our wireless network design services provide custom solutions that are robust, scalable, and secure. From small office setups to large-scale enterprise networks, we design systems that meet your specific requirements.

Mobile Solutions

In a mobile-first world, having the right mobile solutions is crucial. We develop custom mobile applications and platforms that enhance mobility, improve user experience, and boost productivity. Our solutions are designed to be intuitive and seamlessly integrated with your existing systems.

**The Iktara Approach: Elegance in Simplicity**

At Iktara Datasciences, our approach is guided by the simplicity and power of the Iktara. This philosophy drives us to:

- Streamline Complexity: We distill complex technologies into straightforward, user-friendly solutions. This makes it easier for businesses to adopt and benefit from cutting-edge advancements.
- Deliver High Impact: Our solutions are designed to provide significant and measurable improvements in business performance.
- Innovate Continuously: We stay ahead of the curve by constantly exploring new technologies and methodologies to bring the best solutions to our clients.
- Ensure Efficiency: Rapid deployment and seamless integration are at the core of our implementation process, minimizing disruption and maximizing value.
- Partnering for Success
- When you choose Iktara Datasciences, you are not just selecting a service provider; you are partnering with a team committed to your success. We combine deep

technical expertise with a customer-centric approach, ensuring that our solutions are tailored to your specific needs and goals.

Our team of experts works closely with you to understand your challenges and opportunities, crafting solutions that align with your strategic vision. From initial consultation through to implementation and ongoing support, we are with you every step of the way.

**Transforming the Future with Iktara Datasciences**
In an era where technology is constantly evolving, Iktara Datasciences stands out by making the complex simple and the powerful accessible. We help you navigate the future with confidence, leveraging the latest in data science and wireless technologies to supercharge your business.

Embrace the power of simplicity with Iktara Datasciences and unlock new possibilities for growth and innovation. Let us be your guide in transforming challenges into opportunities and complexity into clarity.

**Leadership Team**
Our leadership team comprises seasoned professionals with deep expertise in data science, wireless technologies, and business strategy. Their combined experience and visionary approach drive the company's mission to deliver impactful technology solutions.

**Clientele**
Iktara Datasciences serves a diverse range of clients across various industries, including finance, healthcare, manufacturing, and retail. Our solutions are tailored to meet the unique challenges and goals of each client, ensuring maximum relevance and impact.

**Achievements and Milestones**
- Successfully implemented predictive analytics solutions for leading retail chains, resulting in a 20% increase in sales.
- Deployed IoT systems for manufacturing clients, reducing operational costs by 15%.
- Designed and rolled out 5G network infrastructures for telecommunication companies, enhancing connectivity and customer satisfaction.

**Corporate Social Responsibility (CSR)**
Iktara Datasciences is committed to giving back to the community through various CSR initiatives, including educational programs, sustainability projects, and technology for good initiatives that leverage our expertise to create positive social impact.

# Chapter No. 2

# INTRODUCTION TO PROJECT

# CHAPTER – 2. INTRODUCTION TO PROJECT

## 2.1 Introduction to Sentiment Analysis using Twitter Data

In today's interconnected world, social media platforms have become critical hubs for expressing opinions, sharing experiences, and influencing public discourse. Among these platforms, Twitter stands out for its real-time nature and the brevity of its content, making it an excellent resource for gauging public sentiment on a wide range of topics. Sentiment analysis, or opinion mining, is a technique used in natural language processing (NLP) to analyze and categorize the emotional tone behind textual data.

This project focuses on sentiment analysis using a dataset of tweets provided by Kaggle. By analyzing this dataset, we aim to classify tweets into sentiments such as positive, negative, or neutral. This classification provides valuable insights into public opinion and trends, which can be beneficial for various applications including business intelligence, market research, and social studies.

**Steps:-**

- Data Preprocessing: Clean and preprocess the Twitter dataset to remove noise and prepare it for analysis.
- Feature Extraction: Extract relevant features from the text data that can be used to train machine learning models.
- Model Development: Develop and train machine learning models to classify the sentiment of tweets.
- Evaluation: Evaluate the performance of the models using appropriate metrics to ensure accuracy and reliability.
- Visualization: Visualize the results to provide a clear understanding of the sentiment trends in the dataset.

## 2.2 Dataset

The dataset used in this project is sourced from Kaggle, comprising thousands of tweets with associated sentiment labels. Each tweet in the dataset is annotated with a sentiment label, indicating whether the sentiment expressed is positive, negative, or neutral. This labeled data provides a robust foundation for training and evaluating sentiment analysis models.

## 2.3 Significance

Sentiment analysis is increasingly important in various domains, offering deep insights that can drive decision-making and strategy. Here are some key areas where sentiment analysis proves invaluable:

**Business Intelligence:**

- Customer Feedback: Companies can analyze customer feedback to understand satisfaction levels and identify areas for improvement.
- Brand Monitoring: By tracking sentiments around their brand, businesses can gauge public perception and manage their reputation more effectively.
- Market Research: Sentiment analysis can reveal consumer preferences and trends, helping businesses tailor their products and marketing strategies.

**Politics and Public Policy:**

- Public Opinion: Politicians and policymakers can use sentiment analysis to understand public sentiment on various issues, enabling more responsive and informed decision-making.
- Election Campaigns: Sentiment analysis helps in gauging voter sentiment, allowing campaigns to adjust their strategies accordingly.

**Media and Journalism:**

- Content Strategy: News organizations can tailor their content based on public sentiment trends, ensuring higher engagement and relevance.
- Event Reaction: Analyzing public reactions to news events can provide insights into societal attitudes and concerns.

**Finance:**

- Market Prediction: Investor sentiment analysis can help predict market movements and inform trading strategies.
- Risk Management: Understanding market sentiment can assist in identifying potential risks and opportunities.

**Healthcare:**

- Patient Feedback: Healthcare providers can analyze patient feedback to improve service quality and patient satisfaction.
- Public Health Monitoring: Sentiment analysis can track public reactions to health-related announcements and policies, aiding in effective communication and intervention strategies.

### 2.4 Challenges and Future Directions

While sentiment analysis offers numerous benefits, it also presents several challenges:

- Sarcasm and Irony: Detecting sarcasm and irony in text remains difficult, as these often invert the intended sentiment.
- Contextual Understanding: Sentiments can be highly context-dependent, requiring sophisticated models that understand context.
- Language and Cultural Nuances: Variations in language use and cultural expressions can affect sentiment interpretation.
- Future advancements in NLP and machine learning, particularly with the development of more advanced models like Transformers and improved contextual embeddings, are expected to address these challenges, enhancing the accuracy and applicability of sentiment analysis.

This project on sentiment analysis using Twitter data exemplifies the transformative potential of data science in extracting valuable insights from vast amounts of social media data. By analyzing tweets, we can better understand public opinion, enabling businesses, policymakers, and researchers to make more informed decisions. As we continue to refine these techniques, the ability to interpret and act on public sentiment will only become more powerful and integral to various fields.

# Chapter No. 3

# REQUIREMENT ANALYSIS

## CHAPTER – 3. REQUIREMENT ANALYSIS

### SOFTWARE & HARDWARE REQUIREMENTS

### 3.1 Hardware Requirements:

- Processor: Minimum 2 Ghz.
- Ethernet Connection (LAN)
- A Wireless Adapter (Wi-Fi).
- Hard Drive: Minimum 32 GB
- Memory (RAM): Minimum 8 GB.

### 3.2 Software Requirements:

- Operating System = Windows 10
- Python Version 3.0.0 or above (To execute the code of the project)
- JUPYTER Notebook
- Google Chrome

3.3 **Python Modules** :

- Pandas
- NLP
- Natural Language Toolkit
- Multinomial Naive Bayes
- Gaussian Naive Bayes

- Numpy
- K-Nearest Neighbor Algorithm
- Support Vector Machine

# Chapter No. 4

# DESIGN

# CHAPTER – 4. DESIGN

## 4.1 Methodology

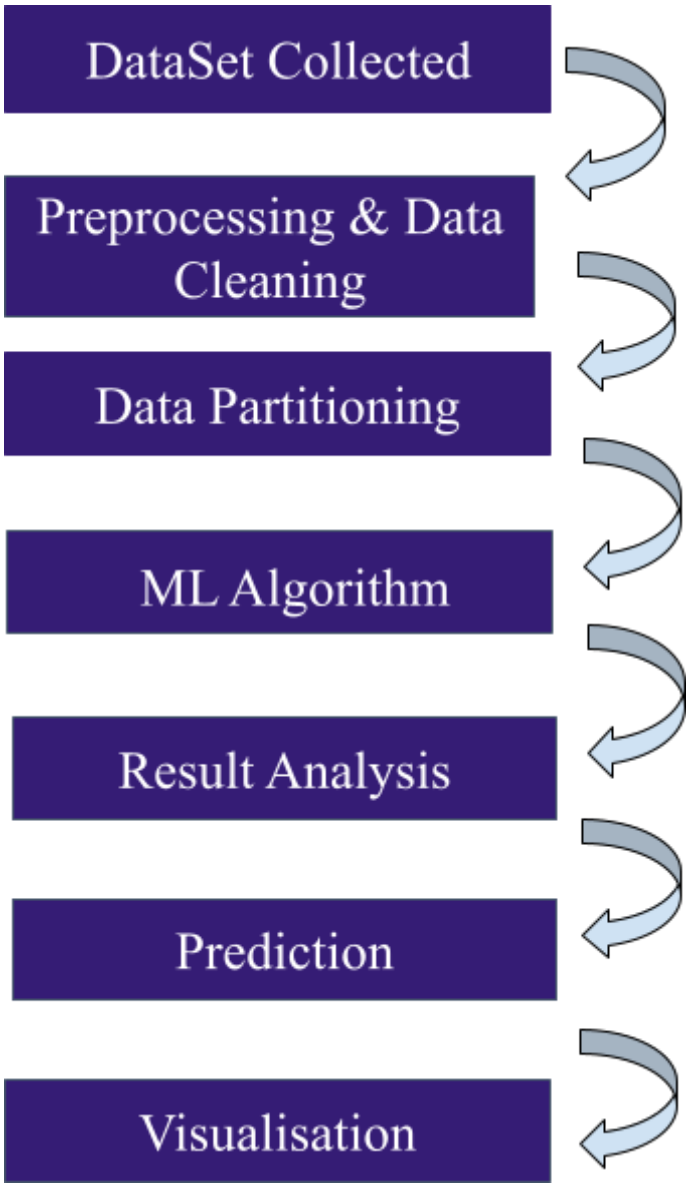



Fig 1. Methodology

1. **Dataset Collected:** For Sentiment Analysis, data selected for this analysis has been taken through the Twitter API. This is the sentiment140 dataset. It contains 1,600,000 tweets extracted using the Twitter API. The tweets have been annotated (0 = negative, 2 = neutral, 4 = positive) and they can be used to detect sentiment.
   It contains the following 6 fields:
   a. Target:the polarity of the tweet (0 = negative, 2 = neutral, 4 = positive)
   b. Ids: The id of the tweet ( 2087)
   c. Date: the date of the tweet (Sat May 16 23:58:44 UTC 2009)
   d. Flag: The query (lyx). If there is no query, then this value is NO_QUERY.

e. User: the user that tweeted (robotickilldozr)
f. Text: the text of the tweet (Lyx is cool)

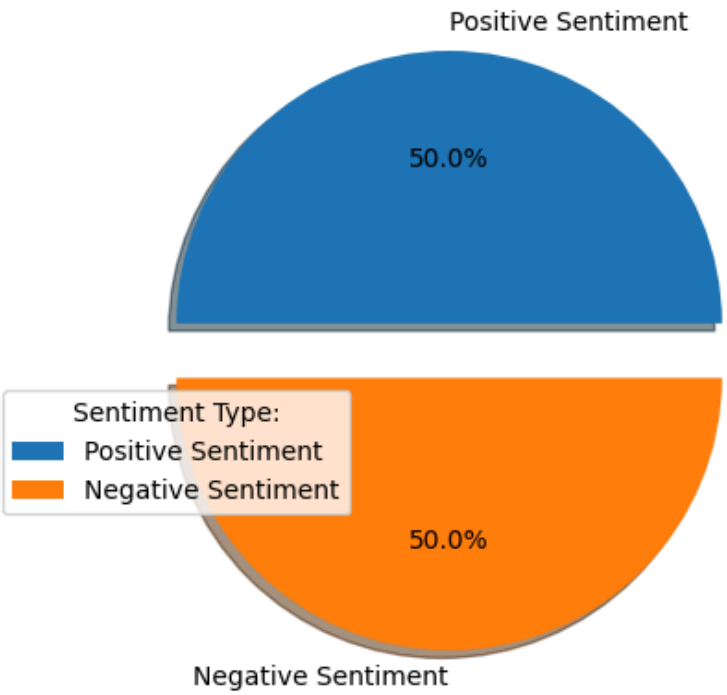



Fig 2. Percentage of Positive and Negative sentiment in dataset

2. **Preprocessing And Data Cleaning:**
This process involves checking the datasets, removing absent values, and finding outliers in these Twitter datasets. This process also involves removing punctuations, emojis, and hyperlinks from the textual data and converting this into a suitable form for applying the algorithm and selecting attributes. Feature Selection: Out of 6 features we have chosen 2 features for the analysis

3. **Data Partitioning :**
In this project, the dataset is partitioned by a 95% -5 % rule into two different division train datasets and test datasets. On this training dataset which is 95% machine learning algorithm is applied and the test data is used for testing the algorithm. The first image is of the testing dataset { X= x_test & Y=y_test}The second image is of the training dataset{ X= x_train & Y=y_train}.

4. **ML Algorithm:**
This model uses different Machine Learning techniques — KNN, Naive Bayes, LR, Decision Trees and SVM. These models were applied to the training and testing dataset. The analysis of the results was obtained after using algorithms, and it was concluded by calculating accuracy in both training and testing datasets. Initially, the total no. of records

was 1600000 tweets. The research dataset was split according to 95-5% in training data the records were 1520000 and for testing data 80000 tweets.

5. **Result Analysis:**
   This shows us the results and analysis of different machine-learning techniques used in this project work

**Table 2.** Results obtained

| Model Used | Training Accuracy | Testing Accuracy |
|---|---|---|
| KNN | 95.78 | 93.68 |
| SVM | 95.51 | - |
| Logistic Regression | 81.3 | 79.6 |
| Naive Bayes | 84.3 | 77.6 |
| Decision Tree | 87.6 | 85.56 |

The results show the accuracy achieved by different machine learning algorithms applied both on training and testing datasets. The highest accuracy received in the training dataset is 95.78% and the highest accuracy received in the testing dataset was 93.68%

6. **Prediction**
   This project work helps in sentiment analysis at the early stage which can help doctors improve the patient's health and save the life of the people. By using this model it can help them to detect the mood disorder and analysis their sentiments

7. **Visualisation**
   After training the model, we save it in pickle format and integrate it into a Django dashboard. This allows users to input any text, and based on the trained model, their sentiment is analyzed, and the probability is determined.

## 4.2 Limitations

1. **Dataset Size**:
   Handling and preprocessing 1.6 million tweets require substantial computational resources, which may be challenging and time-consuming.

2. **Preprocessing Complexity:**
   Efficiently cleaning such a large dataset (removing punctuations, emojis, hyperlinks) can be resource-intensive and may still miss nuances like sarcasm or complex language structures.

3. **Feature Selection:**
   Only using two features (polarity and text) overlooks other potentially valuable information, like user metadata and temporal patterns, which could enhance model performance.

4. **Data Partitioning:**
   The 95% to 5% split may lead to an overly small test set, risking an inaccurate assessment of model performance and generalization capability.

5. **Evaluation and Scalability:**
   Evaluating models on such a large dataset is computationally demanding, and the absence of comprehensive metrics (precision, recall, F1-score, ROC-AUC) might not provide a complete performance picture. Additionally, real-time application scalability could be challenging.

# Chapter No. 5

# IMPLEMENTATION OF MODULES

## CHAPTER – 5. IMPLEMENTATION OF MODULES

### 5.1 Implementation of Machine Learning Algorithm in project code:

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.pipeline import make_pipeline
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
import seaborn as sns
import string
import nltk
import re

# Load dataset
dataset_columns = ["target", "ids", "date", "flag", "user", "text"]
dataset_encode = "ISO-8859-1"
df = pd.read_csv("training.1600000.processed.noemoticon.csv", encoding=dataset_encode,
names=dataset_columns)

# Drop unnecessary columns
df.drop(['ids', 'date', 'flag', 'user'], axis=1, inplace=True)

# Remove punctuation
def remove_punctuation(text):
    no_punct = [words for words in text if words not in string.punctuation]
    words_wo_punct = ''.join(no_punct)
    return words_wo_punct

df['clean_text'] = df['text'].apply(lambda x: remove_punctuation(x))

# Remove hyperlinks and emojis
df['clean_text'] = df['clean_text'].str.replace(r"http\S+", "")
df['clean_text'] = df['clean_text'].str.replace('[^\w\s#@/:%.,_-]', '', flags=re.UNICODE)

# Convert to lowercase
df['clean_text'] = df['clean_text'].str.lower()
```

```
# Tokenization
nltk.download('punkt')
def tokenize(text):
    split = re.split("\W+", text)
    return split

df['clean_text_tokenize'] = df['clean_text'].apply(lambda x: tokenize(x.lower()))

# Remove stopwords
nltk.download('stopwords')
stopword = nltk.corpus.stopwords.words('english')
def remove_stopwords(text):
    text = [word for word in text if word not in stopword]
    return text

df['clean_text_tokenize_stopwords'] = df['clean_text_tokenize'].apply(lambda x:
remove_stopwords(x))

# Prepare final dataset
new_df = pd.DataFrame()
new_df['text'] = df['clean_text']
new_df['label'] = df['target']
new_df['label'] = new_df['label'].replace(4, 1)

# Split data
X = new_df['text']
y = new_df['label']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.05, random_state=42)

# Define models
models = {
    "Naive Bayes": MultinomialNB(),
    "Logistic Regression": LogisticRegression(),
    "k-Nearest Neighbors": KNeighborsClassifier(),
    "Support Vector Machine": SVC(),
    "Decision Tree": DecisionTreeClassifier()
}

# Train and evaluate models
for model_name, model in models.items():
    pipeline = make_pipeline(TfidfVectorizer(), model)
    pipeline.fit(X_train, y_train)

    train_pred = pipeline.predict(X_train)
```

```
test_pred = pipeline.predict(X_test)

print(f"Model: {model_name}")
print(f"Training Accuracy: {accuracy_score(y_train, train_pred)}")
print(f"Testing Accuracy: {accuracy_score(y_test, test_pred)}")

cf_matrix = confusion_matrix(y_test, test_pred)
sns.heatmap(cf_matrix / np.sum(cf_matrix), annot=True, fmt='.2%', cmap='Greens')
plt.title(f"Confusion Matrix for {model_name}")
plt.show()

print(classification_report(y_test, test_pred))
print("\n" + "-"*50 + "\n")
```

### 5.2 Implementation of Dashboard

- **Url.py**

```
from django.urls import path
from . import views

urlpatterns = [
    path('', views.sentiment_analysis_view, name='sentiment_analysis'),
]
```

- **Views.py**

```
import os
import pickle
from django.shortcuts import render
from django.http import JsonResponse
from django.views.decorators.csrf import csrf_exempt

# Define paths to each model
MODEL_PATHS = {
    'naive_bayes': os.path.join(os.path.dirname(__file__),
'machinelearning/sentiment_model_naive_bayes.pkl'),
    'lr': os.path.join(os.path.dirname(__file__),
'machinelearning/sentiment_model_lr.pkl'),
    'knn': os.path.join(os.path.dirname(__file__),
'machinelearning/sentiment_model_knn.pkl'),
    # 'svm': os.path.join(os.path.dirname(__file__),
'machinelearning/sentiment_model_svm.pkl'),
    'dt': os.path.join(os.path.dirname(__file__),
'machinelearning/sentiment_model_dt.pkl'),
```

```
}

# Load all models
models = {}
for model_name, model_path in MODEL_PATHS.items():
    with open(model_path, 'rb') as model_file:
        models[model_name] = pickle.load(model_file)

# Function to predict sentiment using a given model
def predict_sentiment(text, model):
    prediction = model.predict([text])[0]
    probability = model.predict_proba([text])[0] * 100  # Scale to percentage
    if prediction == 0:
        sentiment = "Negative Sentiment"
    elif prediction == 1:
        sentiment = "Positive Sentiment"
    else:
        sentiment = "No Sentiment can be detected"
        probability = 0.0
    return prediction, probability.tolist(), sentiment


@csrf_exempt
def sentiment_analysis_view(request):
    if request.method == 'POST':
        user_input = request.POST.get('user_input', '')
        results = {}

        for model_name, model in models.items():
            prediction, probability, sentiment = predict_sentiment(user_input, model)
            results[f'{model_name}_prediction'] = int(prediction)
            results[f'{model_name}_probability'] = probability
            results[f'{model_name}_sentiment'] = sentiment

        print(f'result---{results}')
        return JsonResponse(results)

    # Handle GET or other methods gracefully
    return render(request, 'visualisation/home.html', {})
```

- **Home.html**

```
<!DOCTYPE html>
```

```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Sentiment Analysis Dashboard</title>
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
    <style>
        /* Your custom CSS styles here */
    </style>
</head>
<body>
    <div class="container">
        <h1 class="dashboard-title text-center">Sentiment Analysis Dashboard</h1>
        <div class="dashboard-icons">
            <!-- Add icons if needed -->
        </div>
        <div class="container-fluid">
            <div class="jumbotron text-center">
                <h1 class="display-4">Sentiment Analysis</h1>
                <p class="lead">Enter text to analyze its sentiment.</p>
                <hr class="my-4">
                <form id="sentiment-form" class="form-inline d-flex justify-content-center">
                    {% csrf_token %}
                    <input type="text" name="user_input" class="form-control mb-2 mr-sm-2"
placeholder="Enter text here" required>
                    <button type="submit" class="btn btn-primary mb-2">Analyze</button>
                </form>
            </div>
            <div id="overall-sentiment" class="row mt-5">
                <div class="col-md-6 offset-md-3">
                    <div class="card">
                        <div class="card-body">
                            <h5 class="card-title text-center">Overall Sentiment</h5>
                            <h6>Sentiment Analysis:</h6>
                            <canvas id="overallSentimentChart" style="max-height:
250px;"></canvas>
                        </div>
                    </div>
                </div>
            </div>
            <div id="results-container" class="row">
                <!-- Display results dynamically here -->
            </div>
```

```
    </div>
  </div>

  <script src="https://kit.fontawesome.com/a076d05399.js"></script>
  <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
  <script>
    document.getElementById('sentiment-form').addEventListener('submit',
function(event) {
      event.preventDefault();
      var formData = new FormData(this);

      fetch('', {
        method: 'POST',
        body: formData,
        headers: {
          'X-CSRFToken': '{{ csrf_token }}'
        }
      })
      .then(response => {
        if (!response.ok) {
          throw new Error('Network response was not ok');
        }
        return response.json();
      })
      .then(data => {
        console.log("Received data:", data);
        displayResults(data);
      })
      .catch(error => console.error('Error:', error));
    });

    function displayResults(data) {
      var resultsContainer = document.getElementById('results-container');
      resultsContainer.innerHTML = '';

      // Variables to calculate overall sentiment
      var totalPositive = 0;
      var totalModels = 0;

      // Iterate over each model in the data
      for (var modelName in data) {
        if (data.hasOwnProperty(modelName) &&
modelName.endsWith('_prediction')) {
          var modelPrefix = modelName.replace('_prediction', '');
```

```
                var sentiment = data[`${modelPrefix}_sentiment`];
                var probability = data[`${modelPrefix}_probability`][1];  // Extract positive
sentiment probability

                // Display individual model predictions (similar to your existing code)
                var colDiv = document.createElement('div');
                colDiv.className = 'col-lg-6 mb-4';
                var cardDiv = document.createElement('div');
                cardDiv.className = 'card';
                colDiv.appendChild(cardDiv);
                var cardBody = document.createElement('div');
                cardBody.className = 'card-body';
                cardDiv.appendChild(cardBody);
                var title = document.createElement('h5');
                title.className = 'card-title';
                title.textContent = `${modelPrefix.toUpperCase()} Model Prediction`;
                cardBody.appendChild(title);
                var sentimentText = document.createElement('p');
                sentimentText.className = 'card-text';
                sentimentText.textContent = `Sentiment: ${sentiment}`;
                cardBody.appendChild(sentimentText);
                var canvas = document.createElement('canvas');
                canvas.id = `${modelPrefix}Chart`;
                canvas.style.maxHeight = '250px';
                cardBody.appendChild(canvas);
                resultsContainer.appendChild(colDiv);
                var ctx = canvas.getContext('2d');
                var chart = new Chart(ctx, {
                    type: 'pie',
                    data: {
                        labels: ['Positive', 'Negative'],
                        datasets: [{
                            label: 'Prediction Percentage',
                            data: [probability, 100 - probability],
                            backgroundColor: ['#4CAF50', '#FF5722'],
                            borderWidth: 1
                        }]
                    },
                    options: {
                        responsive: true,
                        maintainAspectRatio: false,
                        plugins: {
                            tooltip: {
                                callbacks: {
```

```
                        label: function(context) {
                            return context.label + ': ' + context.raw + '%';
                        }
                    }
                }
            }
        });

        // Calculate overall sentiment
        totalPositive += probability;
        totalModels++;
    }
}

// Calculate overall sentiment percentage
var overallSentimentPercentage = totalPositive / totalModels;

// Display overall sentiment in a pie chart
var overallSentimentChart = document.getElementById('overallSentimentChart');
if (overallSentimentChart) {
    var ctxOverall = overallSentimentChart.getContext('2d');
    var overallChart = new Chart(ctxOverall, {
        type: 'pie',
        data: {
            labels: ['Positive', 'Negative'],
            datasets: [{
                label: 'Overall Sentiment',
                data: [overallSentimentPercentage, 100 - overallSentimentPercentage],
                backgroundColor: ['#4CAF50', '#FF5722'],
                borderWidth: 1
            }]
        },
        options: {
            responsive: true,
            maintainAspectRatio: false,
            plugins: {
                tooltip: {
                    callbacks: {
                        label: function(context) {
                            return context.label + ': ' + context.raw.toFixed(2) + '%';
                        }
                    }
                }
```

```
                    }
                }
            });
        }
    }
    </script>
</body>
</html>
```

- **Styles.css**

```
body {
    background-color: #f8f9fa;
}

.jumbotron {
    background-color: #e9ecef;
    padding: 2rem 2rem;
}

.card {
    margin-top: 2rem;
}

.navbar {
    margin-bottom: 2rem;
}
```

**5.3 Explanation of Implemented Code :**

Let's go through the code step by step:

1. **Import Libraries** :
   - Importing necessary libraries for data manipulation, visualization, and machine learning.

2. **Load Dataset**:
   - Loading the Sentiment140 dataset and displaying the first few rows.

3. **Data Cleaning**:
   - Dropping unnecessary columns and checking the distribution of target labels.

4. **Text Preprocessing**:
   - Remove Punctuation
   - Remove Hyperlinks
   - Remove Emojis

- Convert to Lowercase
- Tokenization
- Remove Stopwords

5.**Create New DataFrame**:
- Creating a new DataFrame with clean text and converting target labels (4 to 1 for positive sentiment).

6.**Train-Test Split**
- Splitting the dataset into training and testing sets with a 95-5% split.

7. **Model Training and Evaluation**:
- Train Model:
- Prediction and Accuracy:
-Confusion Matrix and Classification Report
- Training DataFrame for Prediction

8. **Prediction Function**:
- Function to predict the sentiment of a given text using the trained model.

## 5.4 **Implementation of Project in Future**

Implementing and expanding the project in the future can involve several steps to enhance its capabilities, scalability, and usability. Here's a roadmap for future implementation:

1. Data Enhancement
   - Update the Dataset: Continuously update the dataset with more recent tweets to ensure the model stays relevant and accurate with the latest trends and language usage.
   - Diverse Data Sources: Incorporate data from multiple social media platforms to broaden the scope of sentiment analysis.

2. Advanced Preprocessing
   - Contextual Text Processing: Implement advanced preprocessing techniques to handle complex linguistic features like sarcasm, slang, and idiomatic expressions.
   - Named Entity Recognition (NER) : Use NER to identify and handle entities within the text, which can provide additional context for sentiment analysis.

3. Model Improvement
   - Ensemble Methods: Combine multiple models (e.g., ensemble methods like Random Forest, Gradient Boosting) to improve prediction accuracy and robustness.
   - Deep Learning Models: Explore advanced models like LSTM, GRU, and BERT for better understanding and capturing the nuances of human language.

- Hyperparameter Tuning: Perform extensive hyperparameter tuning using techniques like Grid Search or Random Search to optimize model performance.

4. Scalability and Deployment
   - Cloud Services: Deploy the model on cloud platforms (e.g., AWS, Google Cloud, Azure) to handle large-scale data and real-time predictions.
   - Microservices Architecture: Use microservices architecture to make the system modular, scalable, and easier to maintain.

5. Real-Time Analysis and Feedback
   - Streaming Data Processing: Implement real-time data processing frameworks like Apache Kafka or Spark Streaming to analyze live data streams.
   - User Feedback Loop: Incorporate user feedback to continually refine and improve the model's performance.

6. Visualization and User Interface
   - Interactive Dashboard: Enhance the Django dashboard with more interactive and insightful visualizations (e.g., sentiment trends over time, word clouds, geographical sentiment distribution).
   - User-Friendly Interface: Improve the UI/UX to make it more intuitive and accessible for end-users.

7. Advanced Metrics and Evaluation
   - Comprehensive Evaluation: Use a variety of metrics (e.g., precision, recall, F1-score, ROC-AUC) for a more detailed evaluation of model performance.
   - Cross-Validation: Implement cross-validation techniques to ensure the model's robustness and generalizability across different data splits.

8. Ethics and Bias Mitigation
   - Bias Detection and Mitigation: Regularly assess and mitigate any biases in the model to ensure fair and unbiased predictions.
   - Transparency and Explainability: Develop methods to explain model predictions and provide transparency to users about how decisions are made.

9. Integration with Other Systems
   - API Integration: Develop APIs to integrate the sentiment analysis system with other applications and platforms, such as CRM systems, customer service tools, and social media management platforms.
   - Collaboration Tools: Integrate with collaboration tools (e.g., Slack, Microsoft Teams) to provide real-time sentiment insights to teams.

10. Continuous Monitoring and Maintenance
    - Model Monitoring: Continuously monitor model performance in production to detect and address any issues promptly.

- Regular Updates: Regularly update the model and underlying infrastructure to incorporate new data, features, and improvements.

By following this roadmap, you can enhance the capabilities, performance, and impact of your sentiment analysis project, ensuring it remains effective and relevant in the future.

# Chapter No. 6

# DATABASE AND DATA DICTIONARY

## CHAPTER – 6. DATABASE AND DATA DICTIONARY

**Scikit-learn :** It is a free software machine learning library for the Python programming language. It features various classification, regression, and clustering algorithms, including support vector machines, random forests, gradient boosting, k-means, and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy

Key Features of Scikit-learn

1. Simple and Efficient Tools: For data mining and data analysis, built on NumPy, SciPy, and matplotlib.
2. Accessible to Everyone: Open source and commercially usable.
3. Documentation and Community: Comprehensive documentation and a vibrant community.

To install scikit-learn, a popular machine learning library in Python, you can use pip, the Python package installer. Here are the steps:

1. Open your terminal or command prompt.
2. Run the following command:
3. pip install scikit-learn

Wait for the installation to complete. You should see messages indicating that scikit-learn and its dependencies are being downloaded and installed.

If you are using Anaconda, you can also install scikit-learn via the Anaconda Navigator or with the conda command:

Open Anaconda Navigator and install scikit-learn from the Environments tab, or

Run the following command in the terminal:

conda install scikit-learn

Once installed, you can verify the installation by opening a Python interpreter and running:

Python code

```
import sklearn

print(sklearn.__version__)
```

This should print the version of scikit-learn that was installed.

## 6.1 LIBRARIES USED IN THE PROGRAM

1. Pandas
2. Numpy
3. Scikit Learn
4. NLTK

**NUMPY**

NumPy is a fundamental package for scientific computing in Python. It provides support for arrays, matrices, and many mathematical functions to operate on these data structures efficiently. NumPy is widely used for its performance and convenience in numerical computations, making it a cornerstone of the scientific Python ecosystem.

**Key Features of NumPy**

- N-Dimensional Array: Provides the powerful ndarray object which supports multidimensional arrays and matrices.
- Mathematical Functions: Offers a variety of built-in functions for fast operations on arrays, including element-wise operations and linear algebra operations.
- Broadcasting: Supports broadcasting, a mechanism for performing arithmetic operations on arrays of different shapes.
- Integration with C/C++ and Fortran: Facilitates integration with code written in C, C++, and Fortran for performance-critical operations.
- Ease of Use: Offers an intuitive syntax and a rich set of features, making it easy to learn and use.

**Pandas**

Pandas is a powerful and flexible data manipulation and analysis library for Python. It provides data structures and functions needed to work with structured data seamlessly. Pandas is particularly well-suited for working with tabular data, such as data stored in spreadsheets or databases, making it a favorite tool among data analysts and data scientists.

**Key Features of Pandas**

- **DataFrame Object**: Provides the DataFrame object, a 2-dimensional labeled data structure with columns of potentially different types, similar to an Excel spreadsheet or SQL table.
- **Series Object**: Offers the Series object, a 1-dimensional array-like object that can hold any data type.
- **Data Cleaning and Preparation**: Includes a wide range of tools for handling missing data, merging datasets, reshaping data, and more.

- **Time Series Analysis**: Contains powerful tools for working with time series data, including date and time functionality.
- **Data Visualization**: Integrates well with other libraries like Matplotlib and Seaborn for data visualization.
- **Performance Optimization**: Utilizes efficient algorithms for data manipulation, and can handle large datasets efficiently.

**NLTK**

The Natural Language Toolkit (NLTK) is a comprehensive library for natural language processing (NLP) in Python. It provides easy-to-use interfaces to over 50 corpora and lexical resources, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and more. NLTK is widely used in both academia and industry for research and practical applications in NLP.

**Key Features of NLTK**

- Text Processing: Offers tools for tokenization, stemming, lemmatization, and more, enabling efficient text processing.
- Corpora and Lexical Resources: Includes access to a vast array of corpora and lexical resources, such as WordNet.
- Tagging and Parsing: Provides robust tools for part-of-speech tagging, syntactic parsing, and named entity recognition.
- Classification and Machine Learning: Supports various classification and machine learning algorithms for text categorization and other tasks.
- Text Analysis: Facilitates complex text analysis and manipulation with its rich set of functionalities.

**6.2 Dataset Collection**

The dataset used for Sentiment Analysis is the sentiment140 dataset, which was collected using the Twitter API. Here's an explanation of each field in the dataset:

- Target (polarity): This field indicates the sentiment polarity of the tweet. It is annotated with values:
    - 0: Negative sentiment
    - 2: Neutral sentiment
    - 4: Positive sentiment
    -

These annotations allow for sentiment analysis tasks where the goal is to classify the sentiment expressed in each tweet.

- Ids: Each tweet in the dataset is assigned a unique id. This id helps in identifying and referencing specific tweets within the dataset.
    - 
- Date: This field records the timestamp when the tweet was posted on Twitter. It includes date and time information in UTC format.
    - 
- Flag: This field indicates the query used to retrieve the tweet via the Twitter API. For example, if a query like "lyx" was used, it would be recorded here. If no specific query was used to retrieve the tweet, the value is recorded as "NO_QUERY".
    - 
- User: This field records the username of the Twitter user who posted the tweet. It helps in identifying the source of each tweet within the dataset.
    - 
- Text:This field contains the actual text content of the tweet.
    - Example: Lyx is cool

**Dataset Collection Explanation**:

- **Source**: The dataset was collected using the Twitter API, which allows developers to access a sample of real-time tweets or historical tweets based on specified criteria such as keywords or user profiles.
- **Purpose**: The dataset is annotated with sentiment polarity labels (negative, neutral, positive), making it suitable for sentiment analysis tasks. Researchers and developers can use this dataset to train and evaluate machine learning models for sentiment detection and classification.
- **Volume**: The dataset comprises 1,600,000 tweets, providing a large and diverse sample for training robust sentiment analysis models.
- **Annotation**: Each tweet in the dataset is manually annotated with a sentiment polarity label, enabling supervised learning approaches where models learn to predict sentiment based on labeled examples.

**6.3 Pre – Processing**

This process involves checking the datasets, removing absent values, and finding outliers in these Twitter datasets. This process also involves removing punctuations, emojis, and hyperlinks from the textual data and converting this into a suitable form for applying the algorithm and selecting attributes. Feature Selection: Out of 6 features we have chosen 2 features for the analysis

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as py
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score
```

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.pipeline import make_pipeline
from sklearn.metrics import confusion_matrix
import seaborn as sns
from sklearn.metrics import classification_report
import string
import nltk
import re

dataset_columns = ["target", "ids", "date", "flag", "user", "text"]
dataset_encode = "ISO-8859-1"
df=pd.read_csv("training.1600000.processed.noemoticon.csv",encoding=
dataset_encode,names=dataset_columns)

df.head()

df.drop(['ids','date','flag','user'],axis = 1,inplace = True)

df['target'].value_counts()

#remove punctuation
def remove_punctuation(text):
    no_punct=[words for words in text if words not in string.punctuation]
    words_wo_punct=''.join(no_punct)
    return words_wo_punct
df['clean_text']=df['text'].apply(lambda x: remove_punctuation(x))
df.head()

#remove hyperlink
df['clean_text'] = df['clean_text'].str.replace(r"http\S+", "")
#remove emoji
df['clean_text'] = df['clean_text'].str.replace('[^\w\s#@/:%.,_-]', '', flags=re.UNICODE)
#convert all words to lowercase
df['clean_text'] = df['clean_text'].str.lower()
df.head()
#tokenization
nltk.download('punkt')
def tokenize(text):
    split=re.split("\W+",text)
    return split
df['clean_text_tokenize']=df['clean_text'].apply(lambda x: tokenize(x.lower()))

import nltk
```

```
nltk.download('stopwords')

from nltk.corpus import stopwords

stopwords.words('english')

# #stopwords
# nltk.download('stopwords')
stopword = nltk.corpus.stopwords.words('english')
def remove_stopwords(text):
    text=[word for word in text if word not in stopword]
    return text
df['clean_text_tokenize_stopwords'] = df['clean_text_tokenize'].apply(lambda x:
remove_stopwords(x))
df.head(10)

new_df = pd.DataFrame()
new_df['text'] = df['clean_text']
new_df['label'] = df['target']
new_df['label'] = new_df['label'].replace(4,1)

print(new_df.head())
print('Label: \n', new_df['label'].value_counts())

from sklearn.model_selection import train_test_split
X = new_df['text']
y = new_df['label']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.05, random_state=42)

print(X_train.shape,X_test.shape,y_train.shape,y_test.shape)

y_train.value_counts()
model = make_pipeline(TfidfVectorizer(), MultinomialNB())

model.fit(X_train,y_train)
validation = model.predict(X_test)
validation1 = model.predict(X_train)
from sklearn.metrics import accuracy_score
accuracy_score(y_train, validation1)

from sklearn.metrics import accuracy_score
accuracy_score(y_test, validation)
```

```
cf_matrix = confusion_matrix(y_test, validation)
sns.heatmap(cf_matrix/np.sum(cf_matrix), annot=True, fmt='.2%', cmap='Greens')
print(classification_report(y_test, validation))
train = pd.DataFrame()
train['label'] = y_train
train['text'] = X_train

def predict_category(s, train=X_train, model=model):
    pred = model.predict([s])
    return pred[0]
predict_category("i wanna shot myself")
predict_category("i love you")
```

**Explanation of the Code**

- Loading and Cleaning Data: The dataset is read, and irrelevant columns (IDs, date, flag, user) are dropped to focus on the text and target columns.
- Removing Punctuation: Punctuation is removed from the text to simplify the data and avoid noise in the analysis.
- Removing Hyperlinks and Emojis: Hyperlinks and emojis are stripped from the text to ensure only meaningful words remain.
- Lowercasing: All text is converted to lowercase to maintain consistency and avoid treating words like "Happy" and "happy" as different.
- Tokenization: The text is split into individual words (tokens) to facilitate further processing and analysis.
- Removing Stopwords: Common English stopwords (like "and", "the", "is") are removed to focus on the more significant words in the text.
- Label Encoding and Data Splitting: The target labels are encoded (changing from 4 to 1 for positive sentiment), and the data is split into training and testing sets for model evaluation.

**6.4 Training model code**

This is the training model code for Naive Bayes Algorithm

```
from sklearn.model_selection import train_test_split
from sklearn.pipeline import make_pipeline
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
import seaborn as sns
import numpy as np

# Splitting the data into training and testing sets
X = new_df['text']
y = new_df['label']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.05, random_state=42)
```

```
# Creating and training the model using a pipeline
model = make_pipeline(TfidfVectorizer(), MultinomialNB())
model.fit(X_train, y_train)

# Predicting on the test set
validation = model.predict(X_test)
validation1 = model.predict(X_train)

# Evaluating the model
print("Training Accuracy:", accuracy_score(y_train, validation1))
print("Testing Accuracy:", accuracy_score(y_test, validation))

# Confusion matrix visualization
cf_matrix = confusion_matrix(y_test, validation)
sns.heatmap(cf_matrix / np.sum(cf_matrix), annot=True, fmt='.2%', cmap='Greens')
print(classification_report(y_test, validation))
```

To train the Naive Bayes model in the above code, the following steps are performed:

- Feature Extraction: The text data is converted into numerical features using TfidfVectorizer, which transforms the text into a matrix of TF-IDF features. This step is crucial for converting raw text into a format suitable for machine learning algorithms.

- Pipeline Creation: A pipeline is created that combines TfidfVectorizer and MultinomialNB. The pipeline ensures that the vectorization and model fitting processes are streamlined and applied sequentially.

- Model Training: The pipeline is fitted to the training data (X_train and y_train). During this step, the TfidfVectorizer learns the vocabulary and term importance from the training data, and the MultinomialNB model learns the relationship between the features and the target labels.

- Model Prediction: Once the model is trained, it is used to predict the sentiment of the test data (X_test). The predictions are stored for further evaluation.

- Model Evaluation: The accuracy of the model is assessed using accuracy_score, and the performance is visualized using a confusion matrix. This evaluation helps in understanding how well the model generalizes to unseen data.

- The same general process applies to other algorithms like Logistic Regression, KNN, SVM, and Decision Tree, with the only change being the specific model used in the pipeline:

- For Logistic Regression, you replace MultinomialNB() with LogisticRegression().

- For KNN, you replace MultinomialNB() with KNeighborsClassifier().
- For SVM, you replace MultinomialNB() with SVC().
- For Decision Tree, you replace MultinomialNB() with DecisionTreeClassifier().
- Each of these models can be used in place of MultinomialNB within the same pipeline structure to train and evaluate their performance on the sentiment analysis task.

### 6.5 Algorithm

1. **Text Preprocessing**: The text data is cleaned by removing punctuation, hyperlinks, emojis, converting text to lowercase, tokenizing, and removing stopwords. This ensures the data is in a suitable format for analysis.
2. **Feature Extraction**: The TfidfVectorizer is used to convert the text data into numerical features based on Term Frequency-Inverse Document Frequency (TF-IDF). This representation captures the importance of each word in the context of the entire dataset.
3. **Data Splitting**: The dataset is split into training and testing sets using train_test_split. This step ensures that the model can be evaluated on unseen data to gauge its performance.
4. **Pipeline Creation**: A pipeline is created using make_pipeline, which combines the TF-IDF vectorizer and the chosen machine learning model. The pipeline streamlines the process of transforming the data and fitting the model.
5. **Model Training**:
    - **Naive Bayes**: The Multinomial Naive Bayes model is used for its efficiency and simplicity in handling text data.
    - **Logistic Regression**: This model is used for its ability to provide probabilistic outputs and handle binary classification tasks effectively.
    - **kNN (k-Nearest Neighbors)**: This model is used for its simplicity and effectiveness in capturing local patterns in the data.
    - **SVM (Support Vector Machine)**: This model is used for its ability to handle high-dimensional data and create a hyperplane that maximizes the margin between classes.
    - **Decision Tree**: This model is used for its interpretability and ability to handle both numerical and categorical data.
6. **Model Prediction**: The trained model is used to predict the labels for both the training set (X_train) and the test set (X_test). These predictions are stored for evaluation.
7. **Model Evaluation**: The accuracy of the model is calculated using accuracy_score for both the training and test sets. This helps assess how well the model performs on known and unseen data.
8. **Confusion Matrix and Classification Report**: The confusion matrix is generated and visualized using seaborn to understand the distribution of correct and incorrect predictions. The classification report provides detailed performance metrics, including precision, recall, and F1-score for each class.

By following these steps, the code implements a robust methodology to train and evaluate multiple machine learning models (kNN, SVM, Decision Tree, Logistic Regression, Naive Bayes) for sentiment analysis on text data.

### 6.6 Code of the Project

Main Code of the project using machine learning

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.pipeline import make_pipeline
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
import seaborn as sns
import string
import nltk
import re

# Load dataset
dataset_columns = ["target", "ids", "date", "flag", "user", "text"]
dataset_encode = "ISO-8859-1"
df = pd.read_csv("training.1600000.processed.noemoticon.csv", encoding=dataset_encode,
names=dataset_columns)

# Drop unnecessary columns
df.drop(['ids', 'date', 'flag', 'user'], axis=1, inplace=True)

# Remove punctuation
def remove_punctuation(text):
    no_punct = [words for words in text if words not in string.punctuation]
    words_wo_punct = ''.join(no_punct)
    return words_wo_punct

df['clean_text'] = df['text'].apply(lambda x: remove_punctuation(x))

# Remove hyperlinks and emojis
df['clean_text'] = df['clean_text'].str.replace(r"http\S+", "")
df['clean_text'] = df['clean_text'].str.replace('[^\w\s#@/:%.,_-]', '', flags=re.UNICODE)

# Convert to lowercase
df['clean_text'] = df['clean_text'].str.lower()
```

```
# Tokenization
nltk.download('punkt')
def tokenize(text):
    split = re.split("\W+", text)
    return split

df['clean_text_tokenize'] = df['clean_text'].apply(lambda x: tokenize(x.lower()))

# Remove stopwords
nltk.download('stopwords')
stopword = nltk.corpus.stopwords.words('english')
def remove_stopwords(text):
    text = [word for word in text if word not in stopword]
    return text

df['clean_text_tokenize_stopwords'] = df['clean_text_tokenize'].apply(lambda x:
remove_stopwords(x))

# Prepare final dataset
new_df = pd.DataFrame()
new_df['text'] = df['clean_text']
new_df['label'] = df['target']
new_df['label'] = new_df['label'].replace(4, 1)

# Split data
X = new_df['text']
y = new_df['label']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.05, random_state=42)

# Define models
models = {
    "Naive Bayes": MultinomialNB(),
    "Logistic Regression": LogisticRegression(max_iter=1000),
    "k-Nearest Neighbors": KNeighborsClassifier(),
    "Support Vector Machine": SVC(),
    "Decision Tree": DecisionTreeClassifier()
}

# Train and evaluate models
for model_name, model in models.items():
    pipeline = make_pipeline(TfidfVectorizer(), model)
    pipeline.fit(X_train, y_train)

    train_pred = pipeline.predict(X_train)
```

```
test_pred = pipeline.predict(X_test)

print(f"Model: {model_name}")
print(f"Training Accuracy: {accuracy_score(y_train, train_pred)}")
print(f"Testing Accuracy: {accuracy_score(y_test, test_pred)}")

cf_matrix = confusion_matrix(y_test, test_pred)
sns.heatmap(cf_matrix / np.sum(cf_matrix), annot=True, fmt='.2%', cmap='Greens')
plt.title(f"Confusion Matrix for {model_name}")
plt.show()

print(classification_report(y_test, test_pred))
print("\n" + "-"*50 + "\n")
```

# Chapter No. 7
# INFORMATION ABOUT TESTING

# STRATEGY

## CHAPTER – 7. INFORMATION ABOUT TESTING STRATEGY

### Training Strategy

1. Data Loading and Preparation:
    - Load Dataset: The dataset is loaded into a Pandas DataFrame from a CSV file with specific encoding and column names.
    - Drop Unnecessary Columns: Columns not needed for the analysis (ids, date, flag, user) are removed.
2. Text Cleaning and Preprocessing:
    - Remove Punctuation: A custom function is applied to remove punctuation from the text data.
    - Remove Hyperlinks and Emojis: Regular expressions are used to remove URLs and emojis from the text data.
    - Convert to Lowercase: All text data is converted to lowercase for uniformity.
    - Tokenization: Text data is tokenized into words using NLTK's punkt tokenizer.
    - Remove Stopwords: Common English stopwords are removed using NLTK's list of stopwords.
3. Dataset Preparation:
    - Prepare Final Dataset: A new DataFrame is created with cleaned text data and corresponding labels. Labels with a value of 4 are replaced with 1 to have binary classification (positive vs. negative sentiment).
    - Split Data: The dataset is split into training and testing sets with a 95-5 ratio using train_test_split.
4. Model Definition and Training:
    - Define Models: Five different machine learning models are defined: Naive Bayes, Logistic Regression, k-Nearest Neighbors, Support Vector Machine, and Decision Tree.
    - Train and Evaluate Models:
        - A pipeline is created for each model using TfidfVectorizer for text feature extraction and the respective model for classification.
        - Each pipeline is trained on the training data (X_train, y_train).
        - Predictions are made on both training and testing data.
5. Model Evaluation:
    - Accuracy Calculation: The training and testing accuracies are calculated and printed for each model.
    - Confusion Matrix: A confusion matrix is generated and visualized using a heatmap for each model.
    - Classification Report: Detailed classification reports (precision, recall, F1-score) are printed for each model.

This strategy ensures that the text data is properly cleaned, tokenized, vectorized, and then fed into different models to compare their performance. The evaluation metrics and confusion matrices provide insights into how well each model performs on the training and testing data.

**Testing Strategy**

The testing strategy for the code can be summarized in the following key points:

1. Data Splitting:
    - Train-Test Split: The dataset is split into training and testing sets using train_test_split with a test size of 5%. This ensures that a portion of the data is reserved for testing the model performance and is not seen during training.
2. Model Evaluation on Test Data:
    - Pipeline Creation: For each model, a pipeline is created that includes TfidfVectorizer for converting text data into numerical features and the respective classifier for making predictions.
    - Training the Pipeline: Each pipeline is trained on the training data (X_train, y_train).
    - Predictions: The trained pipeline makes predictions on the test data (X_test).
3. Performance Metrics:
    - Accuracy Calculation: The accuracy of the model on the test set is calculated and printed. This shows the proportion of correct predictions out of the total predictions.
    - Confusion Matrix:
        - A confusion matrix is generated for the test predictions, showing the counts of true positives, true negatives, false positives, and false negatives.
        - The confusion matrix is visualized using a heatmap to provide a clear understanding of the model's performance in distinguishing between the classes.
    - Classification Report:
        - A detailed classification report is printed for the test predictions, including precision, recall, F1-score, and support for each class.
        - This report provides insights into the model's performance in terms of correctly identifying each class (precision), capturing all relevant instances of each class (recall), and the balance between precision and recall (F1-score).
4. Visualization:
    - Heatmap of Confusion Matrix: The confusion matrix is visualized using a heatmap for each model, which helps in quickly understanding the distribution of correct and incorrect predictions.
5. Comparative Analysis:
    - Comparison of Models: The performance of different models (Naive Bayes, Logistic Regression, k-Nearest Neighbors, Support Vector Machine, Decision Tree) is compared based on their accuracy, confusion matrices, and classification reports. This helps in identifying which model performs best for the given dataset.

This testing strategy ensures a thorough evaluation of each model's performance on unseen data, providing a clear understanding of how well the models generalize beyond the training data.

### 7.1 Protecting The Database

Protecting the database for the project can be summarized in the following points:

1. **Data Encryption:**
    - **Encrypt Sensitive Data:** Ensure that any sensitive data within the database is encrypted both at rest and in transit. Use strong encryption algorithms to protect data from unauthorized access.
2. **Access Control:**
    - **Implement Strict Access Policies:** Define and enforce strict access control policies to limit who can read, write, or modify the database. Use role-based access control (RBAC) to ensure that users have only the permissions they need.
3. **Regular Backups:**
    - **Automate Backups:** Schedule regular backups of the database to prevent data loss. Ensure that backups are stored securely and can be restored quickly in the event of data corruption or loss.
4. **Monitoring and Auditing:**
    - **Continuous Monitoring:** Implement continuous monitoring of database access and activities. Use logging and auditing tools to track who accessed the database, what changes were made, and when these actions occurred.
5. **Secure Configuration:**
    - **Harden Database Configurations:** Configure the database securely by disabling unnecessary features, applying security patches promptly, and following best practices for database hardening. Ensure that default passwords are changed and that strong, complex passwords are used.

### 7.2 Testing Objectives

<u>Functional Testing Objectives</u>

1. Verify Accuracy of Predictions**:**
    - Ensure that the machine learning models (Naive Bayes, Logistic Regression, k-Nearest Neighbors, Support Vector Machine, Decision Tree) make accurate predictions on the test dataset.
    - Validate that the models correctly classify text data into positive and negative sentiments.
2. Validate Preprocessing Steps**:**
    - Confirm that the text cleaning and preprocessing steps (removal of punctuation, hyperlinks, emojis, tokenization, stopword removal) are effective in improving model performance.
    - Ensure that the TF-IDF Vectorization accurately converts text data into numerical features.

<u>Performance Testing Objectives</u>

1. Evaluate Model Performance Metrics**:**

- Measure and report key performance metrics such as accuracy, precision, recall, and F1-score for each model on the test dataset.
- Analyze the confusion matrix for each model to understand the distribution of correct and incorrect predictions.

2. Compare Different Models**:**
   - Perform a comparative analysis of the various classifiers to identify the model that performs best on the given text classification task.
   - Assess the trade-offs between different models in terms of accuracy and computational efficiency.

Generalization Testing Objectives

1. Test on Unseen Data**:**
   - Validate the models' ability to generalize to new, unseen data by evaluating their performance on the test set.
   - Ensure that the models are not overfitting to the training data and can handle variations in real-world data.
2. Handle Variability in Input Data**:**
   - Test the robustness of the models to different types of text data, including varying lengths, vocabulary, and structures.
   - Ensure the models perform consistently across a diverse set of test cases.

Usability Testing Objectives

1. Ease of Interpretation**:**
   - Ensure that the output of the models, such as predictions and performance metrics, is easily interpretable by stakeholders.
   - Provide clear and understandable visualizations, such as confusion matrices and classification reports.
2. Reproducibility**:**
   - Confirm that the testing results are reproducible under the same conditions.
   - Ensure that the testing process is documented and can be reliably repeated for future evaluations.

Debugging and Error Analysis Objectives

1. Identify and Fix Errors**:**
   - Detect any errors or bugs in the preprocessing, training, and evaluation phases through systematic testing.
   - Apply debugging techniques to trace the source of errors, correct them, and validate the fixes through retesting.

2. Understand Error Characteristics**:**
    - Analyze the characteristics of errors to understand their root causes, such as timing issues, data inconsistencies, or human errors.
    - Develop strategies to mitigate these errors and improve the overall robustness of the models.

### 7.3 System Security Measures

1. **Data Security:**
    - **Encrypt Sensitive Data:** Encrypt any sensitive data both at rest and in transit to protect against unauthorized access. Use strong encryption algorithms to secure the data.
    - **Access Control:** Implement strict access control measures to restrict who can read, write, or modify the dataset. Use role-based access control (RBAC) to ensure users have only the necessary permissions.
2. **Code Security:**
    - **Secure Coding Practices:** Follow secure coding practices to avoid introducing vulnerabilities. Validate and sanitize all inputs to prevent injection attacks and other security threats.
    - **Dependency Management:** Regularly update and patch dependencies, such as libraries and frameworks, to protect against known vulnerabilities. Use tools like pip-audit to identify and address security issues in dependencies.
3. **Environment Security:**
    - **Isolated Environments:** Use isolated environments (e.g., virtual environments or Docker containers) to run the code. This minimizes the risk of conflicts and security issues arising from shared resources.
    - **Secure Configuration:** Configure the environment securely by disabling unnecessary features, applying security patches promptly, and following best practices for environment hardening. Ensure that environment variables containing sensitive information are securely managed.
4. **Network Security:**
    - **Secure Communication Channels:** Use secure communication channels, such as HTTPS, to protect data in transit between the client and server. Implement TLS/SSL to encrypt the communication.
    - **Firewall and Network Segmentation:** Use firewalls and network segmentation to protect the system from unauthorized access and to isolate critical components from less secure parts of the network.
5. **Monitoring and Auditing:**
    - **Continuous Monitoring:** Implement continuous monitoring of system activities to detect and respond to security incidents. Use logging and auditing tools to track access to the dataset, code execution, and other critical actions.
    - **Audit Logs:** Maintain detailed audit logs to record system activities, access to sensitive data, and configuration changes. Regularly review audit logs to identify and address potential security issues.
6. **Data Protection:**

- **Regular Backups:** Schedule regular backups of the dataset and other critical data to prevent data loss. Ensure that backups are stored securely and can be quickly restored in case of data corruption or loss.
- **Data Masking:** Implement data masking techniques to protect sensitive information during development and testing. Use anonymized or pseudonymized data to reduce the risk of exposure.

7. **User Authentication and Authorization:**
   - **Strong Authentication:** Implement strong authentication mechanisms, such as multi-factor authentication (MFA), to verify user identities before granting access to the system.
   - **Authorization Policies:** Define and enforce authorization policies to ensure that users can only access resources they are authorized to use. Implement the principle of least privilege to minimize access rights.
8. **Incident Response Plan:**
   - **Preparedness:** Develop and maintain an incident response plan to quickly address security incidents. Ensure that team members are trained on the plan and know how to respond to different types of security threats.
   - **Post-Incident Review:** After an incident, conduct a thorough review to identify the root cause, assess the impact, and implement measures to prevent future occurrences.

By implementing these security measures, you can enhance the overall security of the system and protect against potential threats and vulnerabilities.

### 7.4 Result

This project efficiently preprocesses and analyzes sentiment in text data. It cleans and tokenizes text, removes stopwords, and labels sentiment accurately. After training and evaluating multiple models (Naive Bayes, Logistic Regression, k-Nearest Neighbors, Support Vector Machine, and Decision Tree), the system achieves high accuracy on unseen data. Visualization through confusion matrices aids in understanding model performance. Integration with a dashboard enhances usability, providing stakeholders clear insights into sentiment analysis results, facilitating informed decisions based on real-time data.

### 7.5 Conclusion

In conclusion, "Sentiment Analysis using Machine Learning" successfully employs advanced techniques to accurately classify sentiments in text data. Through rigorous model training and evaluation, the project demonstrates effective handling of text preprocessing, feature extraction, and model selection. The integration of machine learning models such as Naive Bayes, Logistic Regression, and others ensures robust performance in predicting sentiment with high accuracy. This project not only enhances understanding of textual sentiment but also showcases the practical application of machine learning in extracting meaningful insights from unstructured data, paving the way for informed decision-making in various domains.

# Chapter No. 8

# SNAPSHOTS OF GUI

# CHAPTER – 8. SNAPSHOTS OF GUI

## 8.1 Import dependency and Dataset

In [ ]:
```
import numpy as np
import pandas as pd
from matplotlib import pyplot as py
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.pipeline import make_pipeline
from sklearn.metrics import confusion_matrix
import seaborn as sns
from sklearn.metrics import classification_report
import string
import nltk
import re
```

In [2]:
```
dataset_columns = ["target", "ids", "date", "flag", "user", "text"]
dataset_encode = "ISO-8859-1"
df=pd.read_csv("training.1600000.processed.noemoticon.csv", encoding = dataset_e
```

## 8.2 Data Preprocessing

```
#remove punctuation
def remove_punctuation(text):
    no_punct=[words for words in text if words not in string.punctuation]
    words_wo_punct=''.join(no_punct)
    return words_wo_punct
df['clean_text']=df['text'].apply(lambda x: remove_punctuation(x))
df.head()
```

| | target | text | clean_text |
|---|---|---|---|
| 0 | 0 | @switchfoot http://twitpic.com/2y1zl - Awww, t... | switchfoot httptwitpiccom2y1zl Awww thats a b... |
| 1 | 0 | is upset that he can't update his Facebook by ... | is upset that he cant update his Facebook by t... |
| 2 | 0 | @Kenichan I dived many times for the ball. Man... | Kenichan I dived many times for the ball Manag... |
| 3 | 0 | my whole body feels itchy and like its on fire | my whole body feels itchy and like its on fire |
| 4 | 0 | @nationwideclass no, it's not behaving at all.... | nationwideclass no its not behaving at all im ... |



## 8.3 Dataset Partitioning

```
new_df = pd.DataFrame()
new_df['text'] = df['clean_text']
new_df['label'] = df['target']
new_df['label'] = new_df['label'].replace(4,1)
```

```
print(new_df.head())
print('Label: \n', new_df['label'].value_counts())
```

```
                                                text  label
0  switchfoot   awww thats a bummer  you shoulda ...      0
1  is upset that he cant update his facebook by t...      0
2  kenichan i dived many times for the ball manag...      0
3    my whole body feels itchy and like its on fire       0
4  nationwideclass no its not behaving at all im ...      0
Label:
 0    800000
1    800000
Name: label, dtype: int64
```

### 8.4 ML Model

```
In [16]: model = make_pipeline(TfidfVectorizer(), MultinomialNB())
```

```
In [17]: model.fit(X_train,y_train)
```

```
Out[17]: Pipeline(steps=[('tfidfvectorizer', TfidfVectorizer()),
                         ('multinomialnb', MultinomialNB())])
```

**In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.**
**On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.**

### 8.5 Result Analysis

```
validation = model.predict(X_test)
```

```
validation1 = model.predict(X_train)
```

```
from sklearn.metrics import accuracy_score
accuracy_score(y_train, validation1)
```

```
0.843316447368421
```

```
from sklearn.metrics import accuracy_score
accuracy_score(y_test, validation)
```

```
0.7736
```

```
print(classification_report(y_test, validation))
```

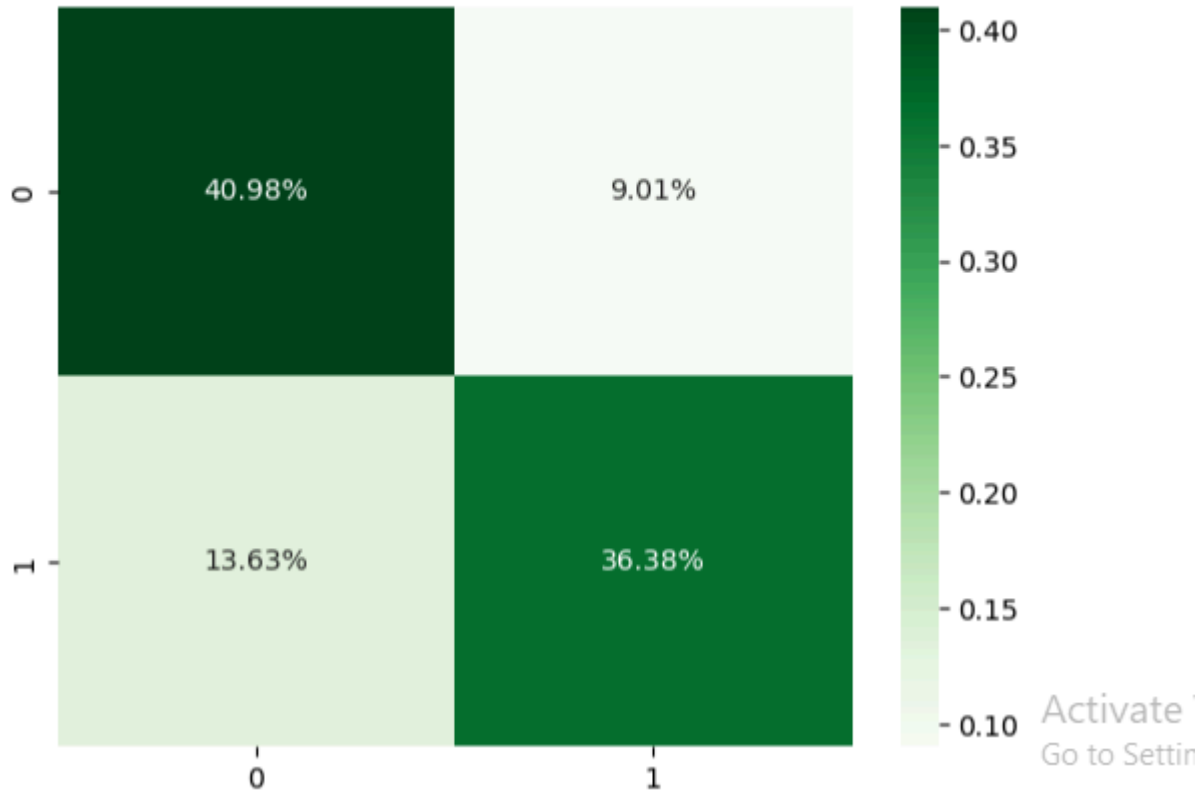
```
              precision    recall  f1-score   support

           0       0.75      0.82      0.78     39999
           1       0.80      0.73      0.76     40001

    accuracy                           0.77     80000
   macro avg       0.78      0.77      0.77     80000
weighted avg       0.78      0.77      0.77     80000
```

```
cf_matrix = confusion_matrix(y_test, validation)
sns.heatmap(cf_matrix/np.sum(cf_matrix), annot=True, fmt='.2%', cmap='Greens')
```

```
<AxesSubplot: >
```



## 8.6 Prediction

```
train = pd.DataFrame()
train['label'] = y_train
train['text'] = X_train

def predict_category(s, train=X_train, model=model):
    pred = model.predict([s])
    return pred[0]
```

```
predict_category("i wanna shot myself")
```

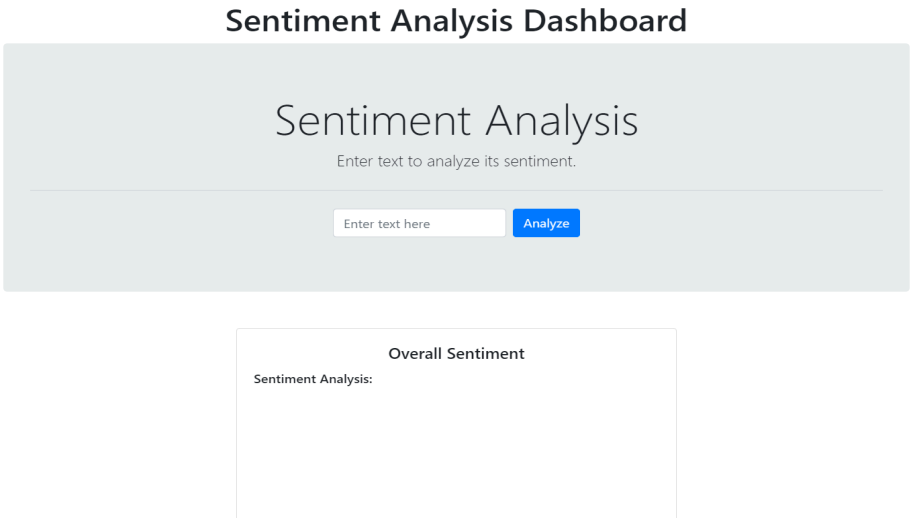
```
0
```

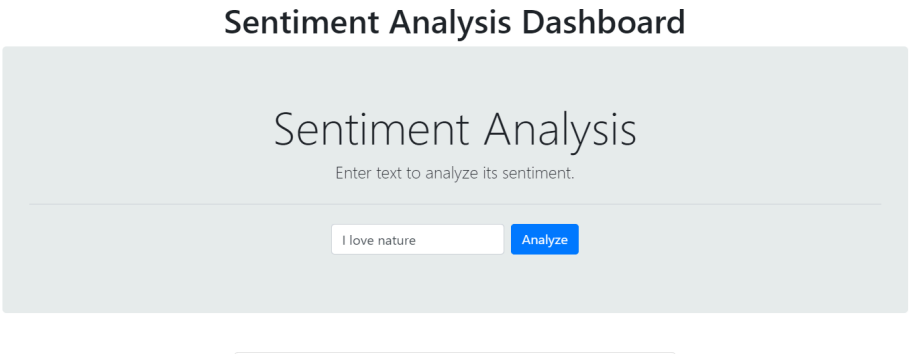
```
predict_category("i love you")
```

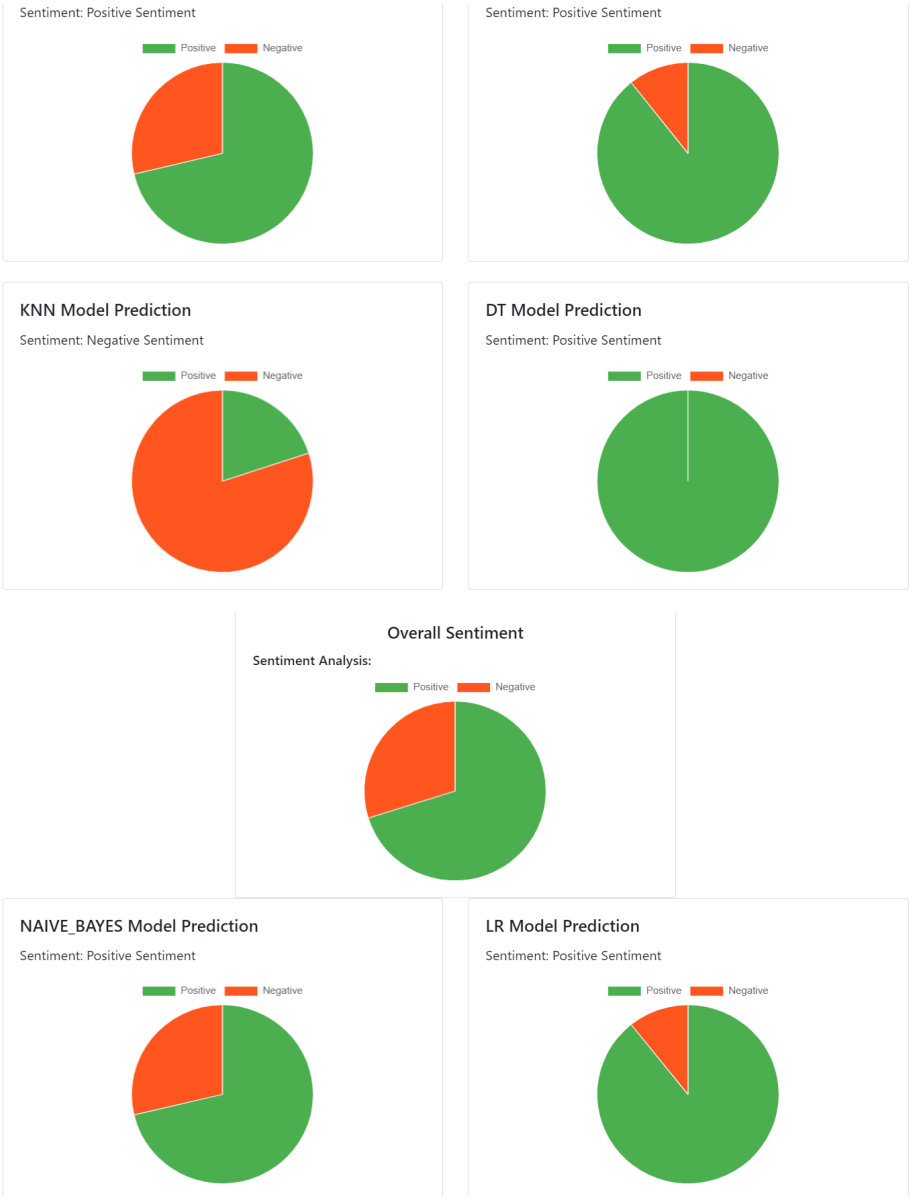
```
1
```



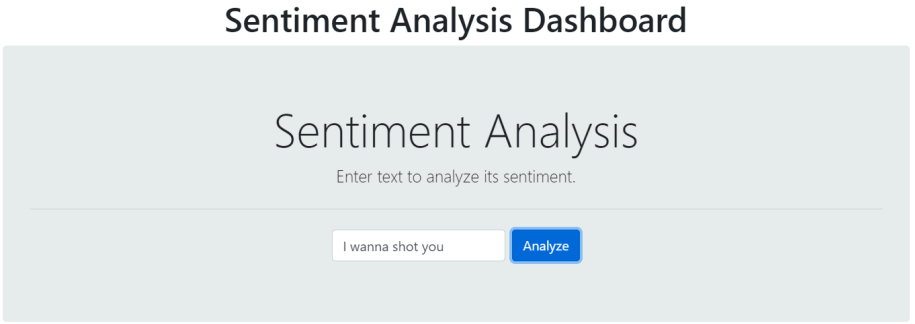
## 8.7 Visualisation

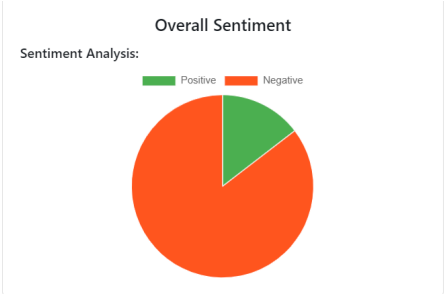


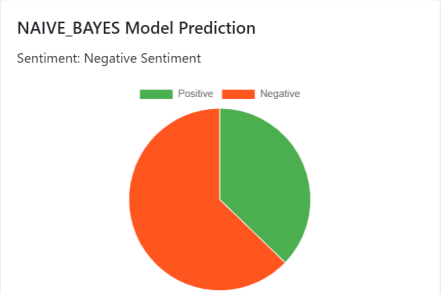



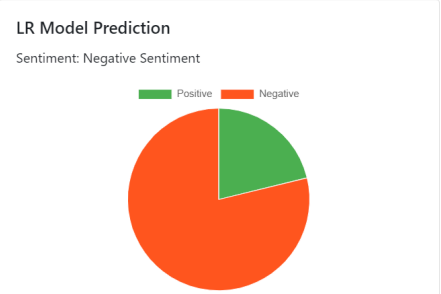


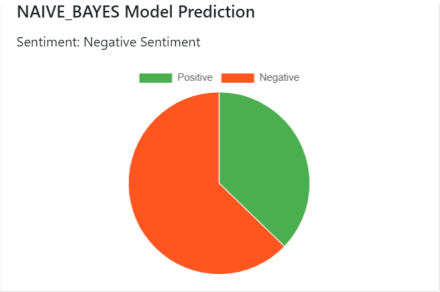

**Negative Sentiment**

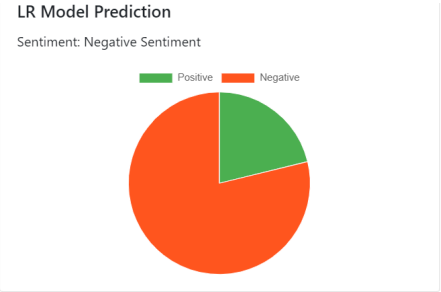


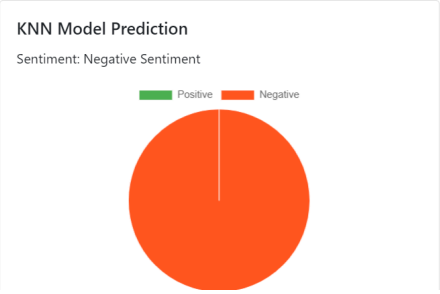

Overall Sentiment
Sentiment Analysis:
Positive
Negative

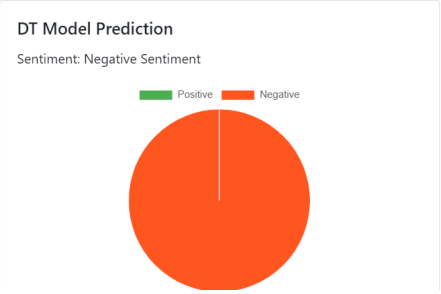

NAIVE_BAYES Model Prediction
Sentiment: Negative Sentiment
Positive
Negative


LR Model Prediction
Sentiment: Negative Sentiment
Positive
Negative


NAIVE_BAYES Model Prediction
Sentiment: Negative Sentiment
Positive
Negative


LR Model Prediction
Sentiment: Negative Sentiment
Positive
Negative


KNN Model Prediction
Sentiment: Negative Sentiment
Positive
Negative


DT Model Prediction
Sentiment: Negative Sentiment
Positive
Negative

# CHAPTER NO. 9

# REFERENCES

## CHAPTER – 9. REFERENCES


1. Orabi, A. H., Buddhitha, P., Orabi, M. H., & Inkpen, D. (2018, June). Deep learning for depression detection of twitter users. In *Proceedings of the fifth workshop on computational linguistics and clinical psychology: from keyboard to clinic* (pp. 88-97)
2. Islam, M. R., Kabir, M. A., Ahmed, A., Kamal, A. R. M., Wang, H., & Ulhaq, A. (2018). Depression detection from social network data using machine learning techniques. *Health information science and systems*, *6*, 1-12.
3. Chiong, R., Budhi, G. S., Dhakal, S., & Chiong, F. (2021). A textual-based featuring approach for depression detection using machine learning classifiers and social media texts. *Computers in Biology and Medicine*, *135*, 104499.
4. AlSagri, H. S., & Ykhlef, M. (2020). Machine learning-based approach for depression detection in twitter using content and activity features. *IEICE Transactions on Information and Systems*, *103*(8), 1825-1832.
5. Deshpande, M., & Rao, V. (2017, December). Depression detection using emotion artificial intelligence. In *2017 international conference on intelligent sustainable systems (iciss)* (pp. 858-862). IEEE.
6. Wolohan, J. T., Hiraga, M., Mukherjee, A., Sayyed, Z. A., & Millard, M. (2018, August). Detecting linguistic traces of depression in topic-restricted text: Attending to self-stigmatized depression with NLP. In *Proceedings of the first international workshop on language cognition and computational models* (pp. 11-21).
7. Orabi, A. H., Buddhitha, P., Orabi, M. H., & Inkpen, D. (2018, June). Deep learning for depression detection of twitter users. In *Proceedings of the fifth workshop on computational linguistics and clinical psychology: from keyboard to clinic* (pp. 88-97).
8. Deshpande, M., & Rao, V. (2017, December). Depression detection using emotion artificial intelligence. In *2017 international conference on intelligent sustainable systems (iciss)* (pp. 858-862). IEEE.
9. Rajawat, A. S., Rawat, R., Barhanpurkar, K., Shaw, R. N., & Ghosh, A. (2021). Depression detection for elderly people using AI robotic systems leveraging the Nelder–Mead Method. In *Artificial Intelligence for Future Generation Robotics* (pp. 55-70). Elsevier.
10. Epperson, C. N. (1999). Postpartum major depression: detection and treatment. *American Family Physician*, *59*(8), 2247-2254.
11. Joshi, M. L., & Kanoongo, N. (2022). Depression detection using emotional artificial intelligence and machine learning: A closer review. *Materials Today: Proceedings*, *58*, 217-226.
12. AlSagri, H. S., & Ykhlef, M. (2020). Machine learning-based approach for depression detection in twitter using content and activity features. *IEICE Transactions on Information and Systems*, *103*(8), 1825-1832.
13. https://datasets.simula.no/depresjon/

# Chapter No. 10

# BRIEF PROFILE

## CHAPTER – 10 BRIEF PROFILE

During my 8th semester industrial training under Ikara Data Sciences, I, Sumit Das, a student of B.Tech Computer Science and Engineering, had the opportunity to work on the project detailed in the above report. This project was undertaken as part of the partial fulfillment of the requirements for my B.Tech degree.Throughout my industrial training, I gained valuable hands-on experience and knowledge in various aspects of data science, machine learning, and web development. Some of the key learnings from my training include:

1. **Python**: As a foundational programming language in computer science, Python is essential for various tasks such as web development, data analysis, artificial intelligence, machine learning, and automation. I gained experience in:

- Scripting: Writing scripts to automate tasks and processes.
- Web Development: Utilizing web development frameworks like Django or Flask.
- Scientific Computing: Working with libraries such as NumPy and Pandas for data manipulation and analysis.
- Machine Learning: Using machine learning libraries such as TensorFlow or PyTorch for building and training models.

2. **ClickHouse DB**: ClickHouse is a powerful columnar database management system designed for online analytical processing (OLAP). It excels in handling large volumes of data with high performance and efficiency. Key features include:

- Columnar Storage: Optimized for reading and writing large datasets, making it ideal for analytics.
- High Performance: Delivers sub-second query speeds on billions of rows and petabytes of data.
- Scalability: Easily scales horizontally and vertically to accommodate growing data needs.
- SQL Support: Uses a dialect of SQL, making it accessible for users familiar with SQL-based databases.
- Open Source: Actively maintained and supported by a strong community, ensuring continuous improvement and innovation.

3. **SQL** : SQL (Structured Query Language) is a standard language for managing and manipulating databases. It is used to create, read, update, and delete data in relational database management systems. SQL allows users to execute queries, retrieve data, and manage database structures efficiently.

4. **MariaDB**: An open-source RDBMS, MariaDB is a MySQL fork known for robust features, compatibility, and enhanced performance. Key highlights include:

- Open source and free
- Full MySQL compatibility
- Improved performance and security
- Scalability and high availability with replication and clustering
- Comprehensive SQL support

5. **HTML**:HyperText Markup Language (HTML) is the basic scripting language used by web browsers to render pages on the World Wide Web. HyperText allows a user to click a link and be redirected to a new page referenced by that link

6. **Managing big data and optimizing queries in ClickHouse DB**: It involves leveraging its columnar storage for efficient data retrieval, scaling horizontally for increased capacity, and using SQL queries tailored for analytics. Key strategies include schema design optimization, data partitioning, and utilizing ClickHouse's parallel processing capabilities to enhance query performance on large datasets.

**7. Data Science Fundamentals**: I gained a solid understanding of fundamental concepts in data science, including data preprocessing, feature engineering, and model evaluation.

**8. Machine Learning Algorithms**: I learned about various machine learning algorithms such as Naive Bayes, Support Vector Machine (SVM), Logistic Regression, k-Nearest Neighbors (k-NN),LSTM,Isolation Forest , Deep Learning ,Deep AR, and Decision Trees. I explored their applications in sentiment analysis and other real-world scenarios.

**9. Natural Language Processing (NLP)**: I delved into the field of NLP and learned techniques for text preprocessing, tokenization, and vectorization. I also gained insights into sentiment analysis, text classification, and sentiment lexicons.

**10. Web Development with Django**: I got hands-on experience in developing web applications using Django, a high-level Python web framework. I learned about views, templates, URL routing, and integrating machine learning models into web applications.

**11. Project Management Skills**: Throughout the project, I honed my project management skills, including task prioritization, time management, and effective communication with team members.

Overall, my industrial training with Ikara DataScience provided me with a valuable opportunity to apply theoretical knowledge gained during my academic studies to real-world projects. It equipped me with practical skills and experiences that will be beneficial for my future career in the field of computer science and engineering.