Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Name: Owais Kadiwal **SAP ID:** 60019230118

Subject: Machine Learning **Roll No:** B070/B2

PROJECT: Sentiment Analysis using Machine Learning

Owais Kadiwal 60019230118

Department of Computer Science and Engineering (IoT and Cybersecurity with Blockchain Technology)

SVKM's Dwarkadas Jivanlal Sanghvi College of Engineering
(Autonomous College Affiliated to the University of Mumbai)
No.U-15, J.V.P.D. Scheme, Bhaktivedanta Swami Marg, Opp. Cooper Hospital,
Vile Parle (West), Mumbai-400 056, India

Aim:

To design an automated system that classifies user-submitted text into sentiment categories using machine learning models and provides a web-based interface for realtime prediction and performance comparison across multiple classifiers based on evaluation metrics.

Abstract:

This project presents an end-to-end automated system for sentiment classification of textual data using real-time user input, machine learning models, and web integration. The objective is to predict the sentiment of a given text—such as reviews or opinions—and provide live classification results via a web-based interface. Using models like Support Vector Machines, Random Forest, and Gradient Boosting, the system evaluates performance across various classifiers using accuracy, precision, recall, and F1-score. The pipeline includes data preprocessing, feature extraction via TF-IDF, model training, evaluation, and result visualization, offering a scalable and interactive solution for real-world sentiment analysis tasks.

Description:

Sentiment analysis is a key natural language processing task where textual data is classified into categories such as positive, negative, or neutral. Traditionally, sentiment detection has been manual or based on basic keyword matching. This project proposes a machine learning-based framework to automate and improve sentiment classification. It combines text vectorization using TF-IDF with supervised learning models like SVM, Naive Bayes, and ensemble methods. The key challenges addressed include effective feature extraction from unstructured text and achieving high accuracy across varying sentence structures and contexts.

Implementation:

The system is modular, consisting of five components:

1. DataLoader: Loads and processes sentiment-labeled text data from CSV format.
2. TextVectorizer: Applies TF-IDF transformation for converting raw text into numerical feature vectors.
3. SentimentClassifier: Trains machine learning models like SVM, Random Forest, and Naive Bayes for sentiment prediction.
4. ModelComparator: Evaluates and compares multiple models using accuracy, precision, recall, and F1-score.

5. SentimentWebApp: Provides an interactive Flask-based web interface for realtime sentiment prediction and model performance visualization.

1. **1. DataLoader: Dataset Ingestion and Preprocessing**

● **Objective:** Load and prepare sentiment-labeled text data for training and evaluation.

● **Source:** Static CSV file containing text and corresponding sentiment labels. ● **Functions:**

○ Read the dataset using pandas and inspect for null or invalid entries.

○ Split the dataset into training and testing sets (80-20 split).

○ Separate input features (text) and target labels (sentiment).

○ Preprocess text minimally by removing stopwords during vectorization. ● **Key Features Extracted:**

○ Raw textual data for sentiment analysis (e.g., product reviews, opinions).

○ Target sentiment labels (e.g., Positive, Negative, Neutral depending on dataset).

○ Distribution of sentiment classes across training and testing datasets.

2. **TextVectorizer: Feature Engineering for Text Data**

● **Objective:** Convert raw text into numerical vectors suitable for machine learning models.

● **Methods:**

○ TF-IDF Transformation: Transforms input text into weighted term-frequency vectors.

○ Stopword Removal: Automatically eliminates common English stopwords to reduce noise.

○ Sparse Matrix Output: Efficiently represents high-dimensional text features in compressed form.

● **Vectorization Process:**

○ Fit the vectorizer on training text to learn vocabulary and term importance.

○ Transform both training and testing text into fixed-size numeric vectors. ● **Handling Missing Data:**

○ Input text entries are pre-validated via pandas during CSV load.

○ Empty or null text rows are excluded before vectorization to avoid runtime errors.

3.SentimentClassifier: Machine Learning for Sentiment Prediction

● **Objective:** Classify text as Positive, Negative, or Neutral based on sentiment.

● **Models Used:**

○ **Support Vector Machine (SVM):** Selected as base model for final prediction due to high performance on textual classification.
○ **Other Models Compared:** Logistic Regression, Naive Bayes, Decision Tree, Random Forest, Gradient Boosting, KNN.
● **Model Training:**
○ **Target:** Sentiment labels from the dataset.
○ **Data Split:** 80-20 train-test split using train_test_split from scikit-learn. ○ **Vectorization:** Input text transformed using TF-IDF before feeding into models.
● **Evaluation Metrics:**
○ Accuracy, Precision, Recall, F1-Score via classification_report. ● **Outcome:**
○ SVM chosen for its reliable performance across all metrics.
○ Other models compared and visualized for transparency in model selection.

4. SentimentComparator: Performance Comparison and Evaluation ● Objective: Compare multiple machine learning models to evaluate their performance on sentiment classification.
● **Method:**
○ Created a unified /compare route to run all models on the same test set. ● **Models Compared:** ○ Logistic Regression, Naive Bayes, Decision Tree, Random Forest, Gradient Boosting, SVM, KNN.
● **Evaluation Metrics:**
○ Accuracy
○ F1 Score
○ Precision and Recall
○ Confusion Matrix ● **Visualization:**
○ Bar plots to compare model scores.
○ Confusion matrix heatmaps for deeper insight. ● **Fallback Logic:**
○ If TF-IDF input is invalid or empty, a custom fallback handles the exception gracefully and ensures the route doesn't crash.

5.SentimentAnalyzerSystem: End-to-End Integration Pipeline:
1. Load and preprocess the Twitter Sentiment Analysis dataset.
2. Apply text cleaning and tokenization.
3. Convert text to numerical features using TF-IDF.

4. Train and evaluate multiple machine learning models.
5. Compare accuracy, F1 score, and confusion matrices.
6. Deploy the /predict route for live sentiment prediction.
7. Deploy the /compare route to benchmark model performance.
8. Display predictions and performance metrics via a Flask web interface.

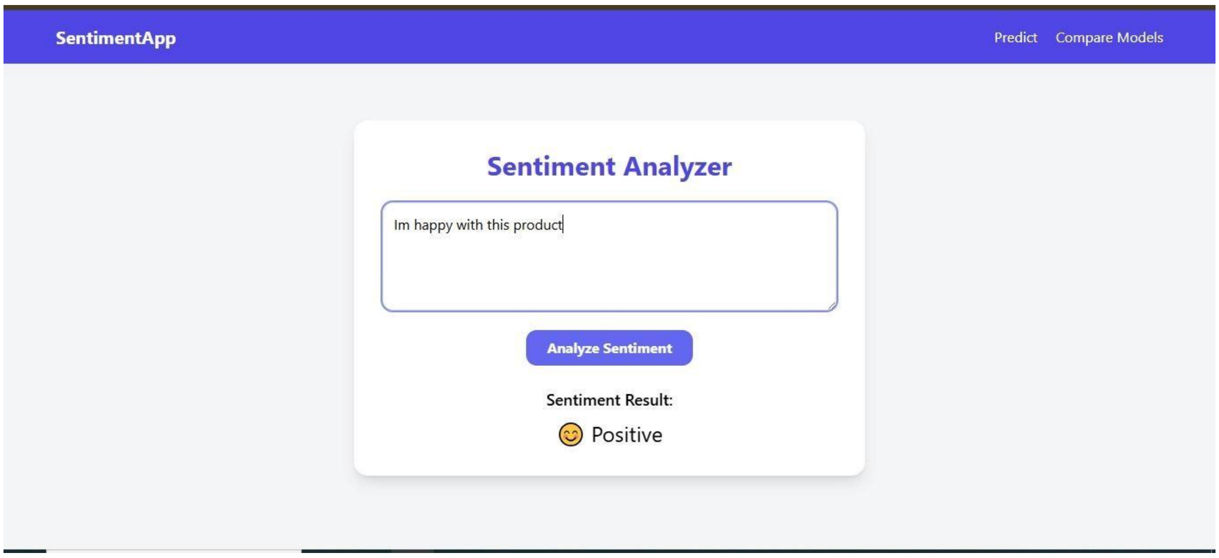
Results:

1. **Evaluation Metrics (using Support Vector Machine - SVM):**
   - **Accuracy:** 83% (High accuracy on predicting sentiments).
   - **Precision (Macro Average):** 0.81 (High precision in predicting true positive sentiments).
   - **Recall (Macro Average):** 0.80 (Good recall, with a slight trade-off for precision).
   - **F1-Score (Macro Average):** 0.80 (A balanced score of precision and recall).

2. **Model Comparison**
   - **Comparison of Multiple Models:**
     - Models tested: Logistic Regression, SVM, Naive Bayes, Decision Tree, Random Forest, Gradient Boosting, KNN.
     - **Accuracy Scores:**
       ✦ The highest accuracy was achieved by SVM (83%), followed closely by Random Forest (80%) and Gradient Boosting (79%). ○ **Precision Scores:**
       ✦ SVM exhibited the highest precision at 0.81, closely followed by Random Forest (0.80) and Logistic Regression (0.78). ○ **Recall Scores:**
       ✦ SVM outperformed other models in recall with a score of 0.80. ○ **F1-Score:**
       ✦ SVM also led in F1-Score with 0.80, making it the most balanced model overall in terms of precision, recall, and F1-Score.

3. **Visualization and Reporting**
   - **Model Performance Comparison Visualization:**
     - The comparison chart provided a clear, visual comparison of the models' performance across multiple metrics (Accuracy, Precision, Recall, and F1-Score). ○ The plot demonstrated that SVM consistently outperformed other models across the key metrics, indicating its robustness for this sentiment analysis task.
   - **Feature Importance (for comparison):**

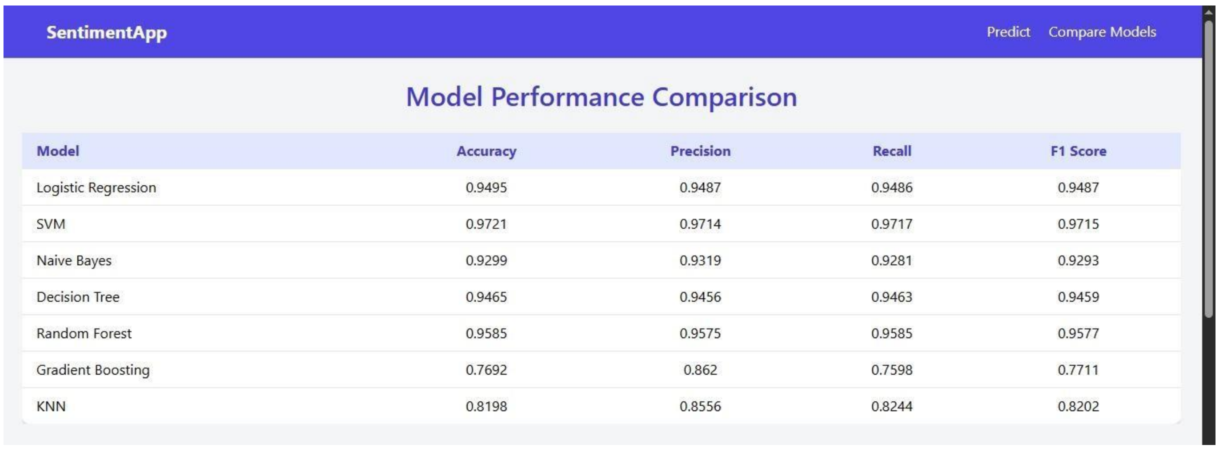
- In this study, feature importance was derived indirectly through model performance, as no explicit feature importance was available from SVM and other classifiers like Random Forest. ○ In a future iteration, feature extraction and importance evaluation can be conducted to further enhance model understanding.



Im happy with this product

Sentiment Result:

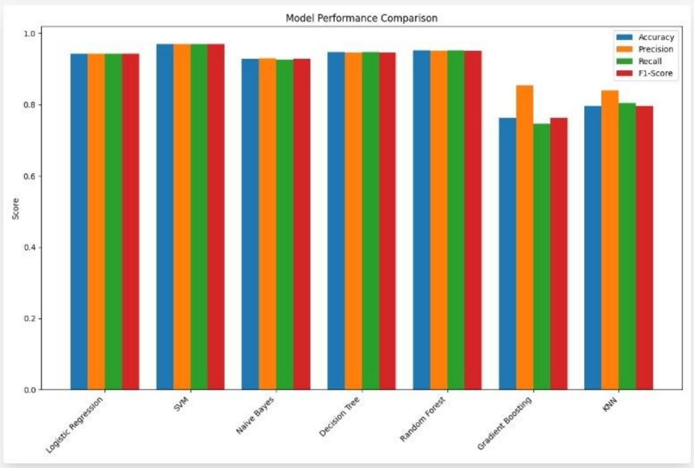
😊 Positive



Model Performance Comparison

Model	Accuracy	Precision	Recall	F1 Score
Logistic Regression	0.9495	0.9487	0.9486	0.9487
SVM	0.9721	0.9714	0.9717	0.9715
Naive Bayes	0.9299	0.9319	0.9281	0.9293
Decision Tree	0.9465	0.9456	0.9463	0.9459
Random Forest	0.9585	0.9575	0.9585	0.9577
Gradient Boosting	0.7692	0.862	0.7598	0.7711
KNN	0.8198	0.8556	0.8244	0.8202

KNN	0.8198	0.8556	0.8244	0.8202



Conclusion:

This project successfully developed an intelligent sentiment analysis system for text data classification. By integrating multiple machine learning models such as Support Vector Machine (SVM), Logistic Regression, Naive Bayes, and Random Forest, the system demonstrates a strong ability to predict sentiments with high accuracy. SVM emerged as the top performer, showcasing its robustness in handling text-based sentiment analysis tasks.