



## Final Project

**Student Name:** Shivam Agarwal  
**Branch:** CSE  
**Semester:** 5th  
**Subject Name:** DAA

**UID:** 23BCS13100  
**Section/Group:** KRG-3B  
**Subject Code:** 23CSH-301

### 1. Aim:

To develop a basic web application using **Flask** and **Machine Learning** (specifically **Logistic Regression** trained on **VADER**-labeled data) that can classify the sentiment (Positive or Negative) of a given news article text.

### 2. Objective:

- Implement a full-stack application using Python's Flask framework.
- Utilize the **VADER (Valence Aware Dictionary and Entiment Reasoner)** lexicon for initial dataset labelling.
- Train a text classification model (Logistic Regression) using **TF-IDF vectorization**.
- Provide a user interface (UI) where a user can input news text and receive a real-time sentiment prediction.

### 3. Technology Stack

Component	Technology	Role
<b>Backend Framework</b>	Python (Flask)	Handles web requests, processes data, serves the frontend.
<b>Data Processing</b>	Pandas, NLTK	Used for dataset loading, text cleaning, and sentiment analysis setup.
<b>Sentiment Labeling</b>	NLTK (VADER)	Provides pre-trained sentiment scores to create training labels.
<b>Machine Learning</b>	Scikit-learn (Logistic Regression, TfidfVectorizer)	Used for feature extraction and training the core classification model.
<b>Frontend</b>	HTML	Provides the structure for the input form and displays the result.

### 4. Methodology

#### 1. Data Preparation and Pre-labeling

The project uses the provided News\_Sentiment\_dataset.csv. Since this is a supervised learning task, the text data needed sentiment labels.

- **Feature Combination:** The title and text columns were combined into a single `full_text` feature.
- **VADER Labeling:** The `SentimentIntensityAnalyzer` (VADER) was used to generate a compound score for each news item. A custom threshold was applied to assign labels:
  - Compound score  $> 0.2 \rightarrow$  **Positive**
  - Compound score  $< -0.2 \rightarrow$  **Negative**
  - Scores between  $-0.2$  and  $0.2$  were labeled **Neutral** and subsequently filtered out to focus on a binary (Positive/Negative) classification problem.
- **Text Preprocessing:** A preprocess function converted text to lowercase and removed punctuation.

## 2. Model Training

The pre-labeled, cleaned data was used to train a `LogisticRegression` classifier.

- **Split Data:** The dataset was split into training and testing sets (`train_test_split`, 80/20 ratio).
- **Feature Extraction (TF-IDF):** The raw text data was converted into numerical features using `TfidfVectorizer`, limiting the vocabulary size to 5000 features. This step transforms text into a matrix representing the importance of words in the document corpus.
- **Training:** The Logistic Regression model was trained on the TF-IDF vectors of the training data.

## 3. Flask Application Integration

The prediction functionality was integrated into a Flask web application:

- **Route (/):** A single route handles both GET (displaying the form) and POST (receiving the user input).
- **Prediction Function:** The `predict_sentiment` function takes new text, applies the same preprocess steps, uses the fitted `TfidfVectorizer` to transform the text, and passes it to the trained `LogisticRegression` model for classification.
- **Frontend Rendering:** The result is passed to the `index.html` template for dynamic display.

## 3. IMPLEMENTATION AND CODES

### App.py:

```
from flask import Flask, request, render_template
import pandas as pd
import string
from nltk.sentiment import SentimentIntensityAnalyzer
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
import nltk
```

```
nltk.download('vader_lexicon')
```

```
# Setup
```

```
app = Flask(__name__)
```

```
sia = SentimentIntensityAnalyzer()
```



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
# Load and preprocess dataset
df = pd.read_csv('News_Sentiment_dataset.csv')
df['full_text'] = df['title'].fillna("") + ' ' + df['text'].fillna("")

# Label sentiment using VADER
def get_sentiment(text):
    score = sia.polarity_scores(text)['compound']
    if score > 0.2:
        return 'positive'
    elif score < -0.2:
        return 'negative'
    else:
        return 'neutral'

df['sentiment'] = df['full_text'].apply(get_sentiment)
df = df[df['sentiment'] != 'neutral'] # binary classification

# Preprocess text
def preprocess(text):
    text = text.lower()
    text = text.translate(str.maketrans("", "", string.punctuation))
    return text

df['clean_text'] = df['full_text'].apply(preprocess)

# Train model
X_train, X_test, y_train, y_test = train_test_split(df['clean_text'], df['sentiment'], test_size=0.2,
                                                    random_state=42)
vectorizer = TfidfVectorizer(max_features=5000)
X_train_vec = vectorizer.fit_transform(X_train)
model = LogisticRegression()
model.fit(X_train_vec, y_train)

# Prediction function
def predict_sentiment(text):
    cleaned = preprocess(text)
    vec = vectorizer.transform([cleaned])
    return model.predict(vec)[0]

# Flask routes
@app.route('/', methods=['GET', 'POST'])
def index():
    prediction = ""
    if request.method == 'POST':
        input_text = request.form['news_text']
        if input_text.strip() != "":
            prediction = predict_sentiment(input_text)
        else:
            prediction = "No input text provided"
```



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
sentiment = predict_sentiment(input_text)
prediction = f'Predicted Sentiment: {sentiment.upper()}'
return render_template('index.html', prediction=prediction)

if __name__ == '__main__':
    app.run(debug=True)
```

## index.html:

```
<!DOCTYPE html>
<html>
<head>
    <title>News Sentiment Analyzer</title>
</head>
<body>
    <h2>Enter News Text</h2>
    <form method="post">
        <textarea name="news_text" rows="10" cols="80" placeholder="Paste news article
here..."></textarea><br><br>
        <input type="submit" value="Analyze">
    </form>
    {% if prediction %}
        <h3>{{ prediction }}</h3>
    {% endif %}
</body>
</html>
```

## 6. Output:

### Output 1:

#### Enter News Text

Tesla boss Elon Musk has had a record-breaking pay package that could be worth nearly \$1tn (£760bn) approved by shareholders.

The unprecedented deal was approved by 75% of votes and drew huge applause from the audience at the firm's annual general meeting on Thursday.

Musk, who is already the world's richest man, must drastically raise the electric car firm's market value over 10 years. If he does this and meets various targets, he will be rewarded with hundreds of millions of new shares.

Analyze

**Predicted Sentiment: POSITIVE**

## Output 2:

### Enter News Text

What is GPS spoofing, which disrupted flights at Delhi airport?  
Delhi flights: This week, Delhi's Indira Gandhi International (IGI) airport experienced rare cases of GPS spoofing, where fake satellite signals are transmitted to mislead aircraft about their locations.

Analyze

**Predicted Sentiment: NEGATIVE**

## 7. Conclusion and Future Scope

The project successfully demonstrated an end-to-end machine learning application deployed via Flask, capable of classifying news sentiment. The use of VADER for rapid initial dataset labeling followed by TF-IDF and Logistic Regression proved effective for this binary classification task.

### Future Scope:

1. **Refinement:** Implement cross-validation and hyperparameter tuning on the Logistic Regression model to improve predictive accuracy.
2. **Advanced Models:** Explore more complex models like Support Vector Machines (SVM) or Deep Learning (LSTM/BERT) for better feature understanding.
3. **Visualization:** Add a feature to visualize the word importance (top TF-IDF features) that led to the positive or negative classification.
4. **Database Integration:** Integrate a database to store past analysis results or to manage and update the training dataset dynamically.