Twitter Sentiment Analysis – Project Documentation

1. Objective

The goal of this project is to develop a machine learning model that can classify tweets as positive or negative based on their textual content.
This helps in understanding public opinion and sentiment trends, which can be valuable for businesses, politics, and social research.

---

2. Data Collection

- **Dataset**: Sentiment140 Dataset (downloaded via kagglehub)
- **Size**: 1.6 million tweets.
- **Features**:
    - target: Sentiment label (0 = negative, 4 = positive)
    - id: Tweet ID
    - date: Date of the tweet
    - flag: Query status (not relevant for analysis)
    - user: Username of the tweet author
    - text: Tweet content (used for sentiment analysis)

---

3. Data Preprocessing

Steps applied to clean and prepare text data for NLP processing:

1. **Remove URLs** using regex (http\S+).
2. **Remove hashtags, mentions, and special characters** (@username, #topic, punctuation).
3. **Convert text to lowercase** for uniformity.
4. **Remove stopwords** using NLTK stopwords list.
5. **Tokenization** to split tweets into words.

6. **Stemming** using PorterStemmer to reduce words to their root form.
7. **Vectorization** with TfidfVectorizer to convert text into numerical features.

---

4. Model Development

- **Train-Test Split**: 80% training, 20% testing.
- **Models Used**:
    1. **Logistic Regression** (baseline & primary model)
    2. **Naive Bayes** (BernoulliNB)
    3. **Support Vector Machine** (LinearSVC)
- **Implementation Tools**:
    - **Python Libraries**: pandas, numpy, nltk, scikit-learn.

---

5. Evaluation Metrics

Models were evaluated using:

- **Accuracy** – percentage of correct predictions.
- **Precision** – proportion of positive predictions that are correct.
- **Recall** – proportion of actual positives correctly identified.
- **F1-Score** – harmonic mean of precision and recall.

Results on Test Data:

Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**
Logistic Regression	**0.7767**	**0.7956**	0.7666	**0.7808**
Naive Bayes	0.7648	0.7384	**0.7796**	0.7585
SVM	0.7697	0.7812	0.7636	0.7723

---

6. Conclusion

- **Logistic Regression** achieved the highest **accuracy, precision, and F1-score**, making it the most balanced model.
- **Naive Bayes** had the highest recall but lower precision, meaning it identifies more positives but with more false positives.
- **SVM** performed moderately across all metrics but did not outperform Logistic Regression.

Chosen Model for Deployment: **Logistic Regression**, due to its balanced performance and robustness.

7. Deployment

- **Platform:** Flask Web Application
- **Steps:**
    1. Save the trained model using **pickle**.
    2. Create a **Flask API** that:
        - Accepts tweet text from the user.
        - Preprocesses it with the same pipeline used during training.
        - Predicts sentiment using the deployed model.
        - Returns the predicted sentiment label.
    3. Provide a **simple web interface** for users to input tweets and view sentiment results.