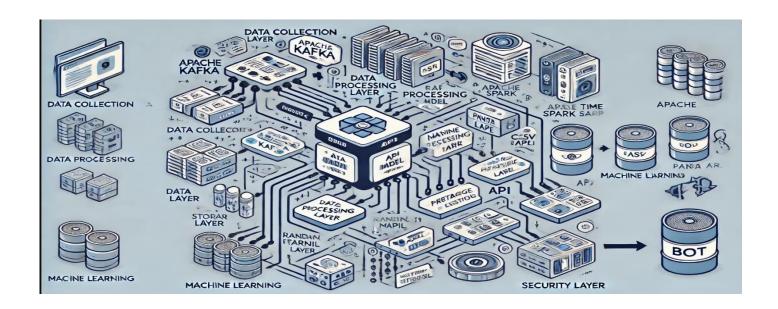
# **Technical Documentation**

# • System Architecture:

The Bot Detection System is designed with a modular architecture to ensure scalability, efficiency, and ease of deployment. The architecture consists of the following key components:

- Data Collection Layer: Utilizes Apache Kafka for real-time data ingestion from social media streams. This layer ensures the continuous flow of data to the processing units.
- 2. Data Processing Layer: Employs Pandas and Apache Spark for batch and real-time data preprocessing. This layer handles text cleaning, feature extraction, and data transformation.
- 3. Machine Learning Model Layer: Implements a Random Forest Classifier trained on a comprehensive dataset with TF-IDF features, sentiment scores, posting patterns, and engagement metrics.
- 4. **API Layer:** Developed with **FastAPI** to provide a RESTful interface for real-time predictions. It handles incoming requests, processes data, and returns bot detection results.
- 5. **Storage Layer:** Uses CSV files for report generation and can be extended to databases like PostgreSQL for persistent storage.
- 6. **Security Layer:** Incorporates **encryption** with the cryptography library to anonymize sensitive user data.



## Model Used:

#### 1. Random Forest Classifier:

- A robust ensemble learning method used for classification tasks.
- Trained with 100 estimators to balance performance and computational efficiency.
- Features include TF-IDF vectors, sentiment polarity, followers-to-following ratio, posting frequency, hashtag usage, and engagement metrics.

#### 2. TextBlob:

- Utilized for sentiment analysis to add contextual sentiment features to the model.
- TextBlob is a Python library for processing textual data. It provides a simple API for diving into common natural language processing (NLP) tasks such as part-of-speech tagging, noun phrase extraction, sentiment analysis, classification, and more.

#### 3. TF-IDF Vectorizer:

 Converts textual data into numerical features based on the importance of words in the dataset.

# Tools and Technologies:

- 1. **Python:** Core programming language for data processing, model development, and API implementation.
- Pandas & NumPy: For efficient data manipulation and numerical computations.
- 3. **Scikit-learn:** Provides machine learning algorithms and evaluation metrics.
- 4. **Apache Kafka:** Enables real-time data streaming and processing.
- 5. **Apache Spark:** Facilitates large-scale data processing for scalability.
- 6. **FastAPI:** Lightweight web framework for building APIs with high performance.
- 7. **Cryptography: Fernet** Ensures data security through encryption.

# Setup Guide:

### **Local Deployment:**

### 1. Clone the Repository

git clone https://github.com/bot-detection-system.git cd bot-detection-system

#### 2. Create a Virtual Environment

python -m venv venv source venv/bin/activate

### 3. Install dependencies

Pip install -r requirements.txt

#### 4. Start Kafka Server

bin/zookeeper-server-start.sh config/zookeeper.properties bin/kafka-server-start.sh config/server.properties

#### 5. Run the API

uvicorn bot\_detection\_system:app --reload

# Privacy Measures:

### Data Encryption:

- Utilizes the cryptography library to encrypt sensitive information like usernames before storage or transmission.
- AES (Advanced Encryption Standard) algorithm ensures strong encryption.

### Data Anonymization:

 Anonymizes user data during processing to prevent unauthorized identification.

### • Secure API Endpoints:

- o Implements HTTPS for secure communication.
- Validates and sanitizes API inputs to prevent injection attacks.

#### Access Control:

 Restricts access to sensitive data through authentication and authorization mechanisms.

The Bot Detection System integrates machine learning, real-time data processing, and robust security measures to deliver accurate bot detection. Its flexible architecture supports both local and cloud deployments, making it suitable for diverse operational environments.