Bot Detection System - Technical Documentation

1. Introduction

The Bot Detection System is designed to identify automated accounts on social media platforms using machine learning and anomaly detection techniques. It leverages **BERT-based text classification** for tweet analysis and **Isolation Forest** for anomaly detection in user activity.

---

2. Architecture Overview

2.1 Components

- **BERT Model**: Pre-trained `bert-base-uncased` for text classification.
- **Isolation Forest**: Used for anomaly detection based on user activity.
- **Dask**: For scalable, parallelized data processing.
- **Cryptography**: Fernet encryption for data privacy.
- **Cloud Deployment**: Supports deployment on AWS/GCP/Azure.

2.2 Workflow

1. Load the pre-trained BERT model and tokenizer.
2. Preprocess input tweets using tokenization.
3. Predict whether a tweet is from a bot or a human.
4. Extract user activity features (e.g., retweets, mentions, followers).
5. Apply Isolation Forest for anomaly detection.
6. Encrypt sensitive data before storing results.
7. Generate reports summarizing findings.

---

3. Models Used

3.1 BERT for Text Classification

- **Tokenizer**: `BertTokenizer.from_pretrained('bert-base-uncased')`

- **Model**: `BertForSequenceClassification.from_pretrained('bert-base-uncased', num_labels=2)`
- **Training Data**: Social media datasets labeled as bot/human.
- **Evaluation Metrics**: Precision, Recall, F1-score, AUC-ROC.

3.2 Isolation Forest for Anomaly Detection

- **Features**: Retweet Count, Mention Count, Follower Count.
- **Scaler**: StandardScaler for normalization.
- **Implementation**: `IsolationForest(contamination=0.1, random_state=42)`

---

4. Setup Guide

4.1 Local Deployment

Prerequisites

- Install dependencies:
  pip install torch transformers scikit-learn pandas dask cryptography
- Run the script:
  python bot_detection_api.py

4.2 Cloud Deployment

AWS (EC2 + S3)

1. Launch an **EC2 instance** (GPU recommended for BERT).
2. Install dependencies inside the instance.
3. Upload CSV data to **Amazon S3**.
4. Modify the script to read from S3 instead of local storage.
5. Deploy as an API using Flask/FastAPI with **AWS Lambda**.

GCP (Vertex AI + Cloud Storage)

1. Store datasets in **Cloud Storage**.
2. Deploy BERT model on **Vertex AI**.
3. Run Isolation Forest on **Cloud Functions**.
4. Store results in **BigQuery** or Cloud Storage.

Azure (VM + Blob Storage)

1. Launch an **Azure VM** with GPU.
2. Use **Azure Blob Storage** for CSV files.
3. Deploy BERT model on **Azure ML**.

4. Process data and store results in **Cosmos DB**.

---

5. Privacy & Security Measures

- **Data Encryption**:
  - Uses **Fernet encryption** before storing user IDs and usernames.
  - Example: `cipher.encrypt(data.encode()).decode()`
- **Anonymization**:
  - Direct identifiers (e.g., usernames) are replaced with encrypted values.
- **Secure Storage**:
  - Cloud-based deployments use **IAM roles** and **VPC security groups**.
- **Compliance**:
  - Designed to comply with **GDPR** & **CCPA** privacy regulations.

---

6. Conclusion

This documentation provides an overview of the **Bot Detection System**, detailing its architecture, models, setup, and privacy measures. For further improvements, integration with real-time API services and enhanced NLP models can be explored.