Twitter Sentiment Analysis

User Manual

Version 1.0

April 30, 2025

MLOPS AI PROJECT

Contents

1 Introduction **3**
- 1.1 About This Application . . . 3
- 1.2 System Requirements . . . 3
  - 1.2.1 Minimum Requirements . . . 3
  - 1.2.2 Recommended . . . 3

2 Getting Started **4**
- 2.1 Installation . . . 4
  - 2.1.1 Step 1: Install Docker . . . 4
  - 2.1.2 Step 2: Download the Application . . . 4
  - 2.1.3 Step 3: Launch the Application . . . 4
  - 2.1.4 Step 4: Verify Installation . . . 4

3 Using the Application **6**
- 3.1 The Interface . . . 6
- 3.2 Analyzing Text . . . 7
  - 3.2.1 Single Text Analysis . . . 7
  - 3.2.2 Batch Analysis(Future Tasks) . . . 7
- 3.3 Understanding Results . . . 7

4 Dashboard & Analytics **8**
- 4.1 Accessing the Dashboard . . . 8
- 4.2 Dashboard Panels . . . 8
- 4.3 Twitter Sentiment Dashboard . . . 8
  - 4.3.1 System Performance Metrics . . . 8
  - 4.3.2 Application Performance . . . 9
  - 4.3.3 Dashboard Configuration . . . 9
- 4.4 Grafana Interface Elements . . . 9
- 4.5 Customizing Your View . . . 9
- 4.6 Key Observations . . . 10

5 Maintenance **11**
- 5.1 Regular Maintenance Tasks . . . 11
  - 5.1.1 Daily Checks . . . 11
  - 5.1.2 Weekly Tasks . . . 11
  - 5.1.3 Monthly Activities . . . 11
- 5.2 Updating the System . . . 11

6 Evaluation & Testing **12**
6.1 Model Evaluation . . . 12
6.1.1 Performance Metrics . . . 12
6.1.2 Classification Report . . . 12
6.2 Model Monitoring . . . 13
6.2.1 Drift Detection . . . 13
6.2.2 Performance Over Time . . . 13
6.3 Unit Testing . . . 13
6.3.1 Test Types . . . 13
6.3.2 Running Tests . . . 13
6.4 Test Cases . . . 13
6.4.1 Data Pipeline Tests . . . 13
6.4.2 Model Tests . . . 14
6.4.3 Example Test Case . . . 14

7 Troubleshooting **15**
7.1 Common Issues . . . 15

8 Appendix **16**
8.1 Technical Details . . . 16
8.1.1 System Architecture . . . 16
8.1.2 File Structure . . . 16
8.2 Command Reference . . . 17
8.3 Glossary . . . 17

Chapter 1

Introduction

1.1 About This Application

The Twitter Sentiment Analysis application helps you understand the emotional tone behind tweets. Using advanced machine learning, it classifies text as positive or negative, allowing you to:

- Monitor brand perception
- Track campaign effectiveness
- Analyze customer feedback

1.2 System Requirements

1.2.1 Minimum Requirements

- Operating System: Windows 10/11, macOS 10.15+, or Linux Ubuntu 20.04+
- Docker Desktop (latest version)
- 4GB RAM
- 10GB free disk space
- Internet connection

1.2.2 Recommended

- 8GB+ RAM
- SSD storage
- Broadband internet connection

Chapter 2

Getting Started

2.1 Installation

Follow these simple steps to install the Twitter Sentiment Analysis application:

2.1.1 Step 1: Install Docker

If you don't have Docker installed:

1. Visit Docker's official website
2. Download and install the version for your operating system
3. Launch Docker Desktop and ensure it's running

2.1.2 Step 2: Download the Application

1. Download the application package from the provided link or clone from GitHub:

```
git clone https://github.com/Rajnishmaurya/twitter-sentiment-AI-project
```

2. Navigate to the application folder:

```
cd twitter-sentiment-AI-project
```

2.1.3 Step 3: Launch the Application

1. Open a terminal or command prompt
2. Run the following command:

```
docker-compose up --build -d
```

3. Wait for the setup to complete (approximately 5-7 minutes)

2.1.4 Step 4: Verify Installation

After installation, you can access:

- Main application: http://localhost:8000
- Monitoring dashboard: http://localhost:3000

- Default login: `admin`
- Default password: `admin` (please change after first login)

Chapter 3

Using the Application

3.1 The Interface

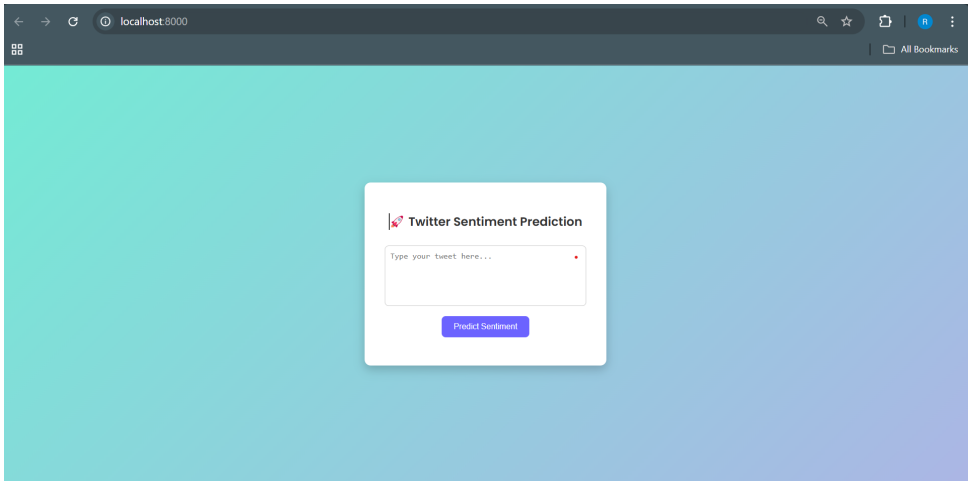

Figure 3.1: Twitter Sentiment Analysis Interface

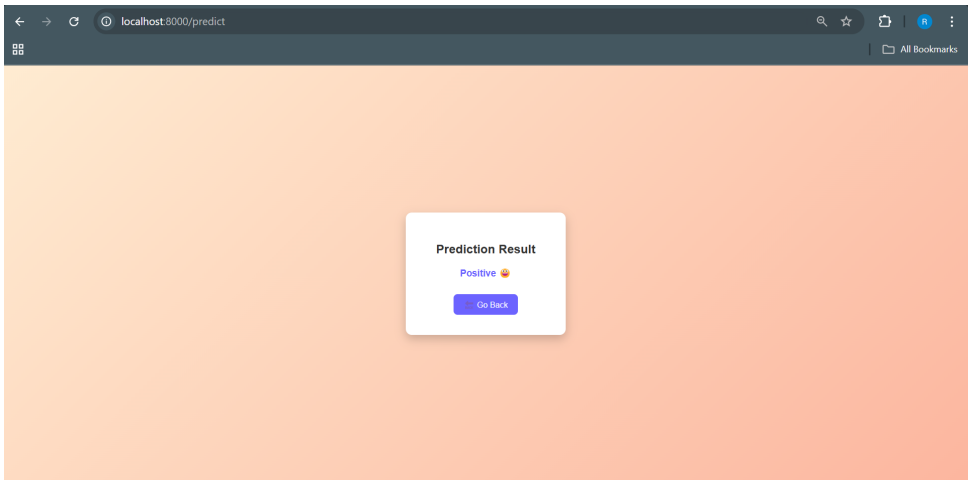

Figure 3.2: Twitter Sentiment Analysis Predict

1. **Input Area** - Enter tweets or text you want to analyze
2. **Predict Button** - Click to analyze the entered text

3. **Results Display** - Shows sentiment classification and confidence score
4. **History Panel** - Lists your recent analyses

3.2 Analyzing Text

3.2.1 Single Text Analysis

1. Enter text in the input field
2. Click "Predict Sentiment"
3. View the results:
   - **Positive** results appear in green with a ☺ emoji
   - **Negative** results appear in red with a ☹ emoji

3.2.2 Batch Analysis(Future Tasks)

For analyzing multiple tweets:

1. Prepare a CSV file with a column named "text" containing your tweets
2. Click the "Batch Analysis" tab
3. Upload your CSV file
4. Click "Analyze All"
5. Download the results when processing completes

3.3 Understanding Results

The application provides several insights:

- **Sentiment Classification** - Positive or Negative
- **Visualization** - Color-coded representation of sentiment

Chapter 4

Dashboard & Analytics

4.1 Accessing the Dashboard

1. Navigate to http://localhost:3000 in your browser
2. Log in with your credentials (default: admin/admin)
3. Select the "Twitter Sentiment Dashboard" from the home screen

4.2 Dashboard Panels

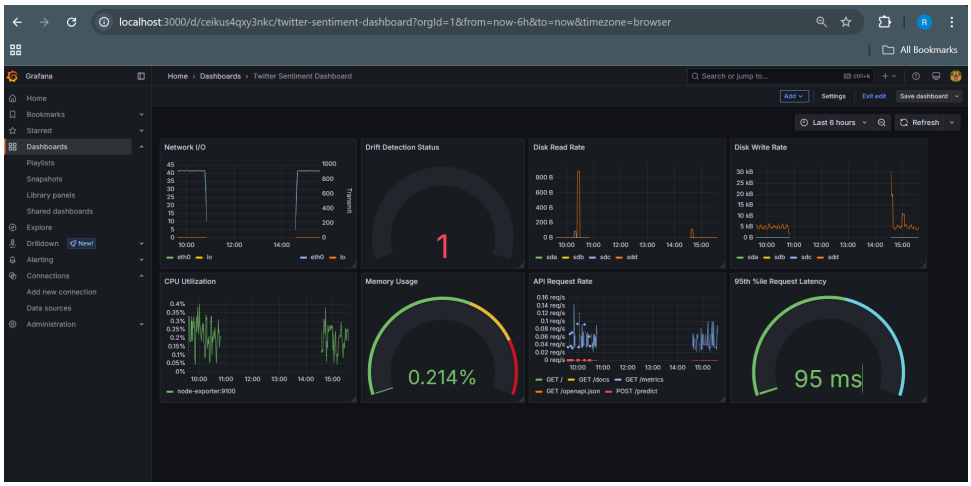

Figure 4.1: Dashboard Overview

The dashboard contains several informative panels:

4.3 Twitter Sentiment Dashboard

This dashboard displays monitoring metrics for a Twitter sentiment analysis system running in Grafana.

4.3.1 System Performance Metrics

- **Network I/O**: Shows network throughput with values up to 45 on left axis and 1000 on right axis, displaying eth0 and lo interfaces

- **CPU Utilization**: Low CPU usage (0.05% to 0.4%) for node-exporter:9100
- **Memory Usage**: Currently at 0.214% utilization, displayed as a gauge visualization with green-yellow-red gradient
- **Disk Performance**:
  - Disk Read Rate: Shows periodic spikes up to approximately 800 B
  - Disk Write Rate: Shows periodic activity with values up to 30 kB

4.3.2 Application Performance

- **API Request Rate**: Shows request rates for different endpoints (GET /, GET /docs, GET /metrics, GET /openapi.json, POST /predict)
- **95th %ile Request Latency**: Currently at 95 ms, displayed as a gauge visualization
- **Drift Detection Status**: Shows a value of 1 (likely indicating normal operation)

4.3.3 Dashboard Configuration

- **Time Range**: Last 6 hours selected in the dropdown
- **Refresh Rate**: Configurable via dropdown
- **Dashboard Controls**: Options to Add panels, Settings, Exit edit mode, and Save dashboard

4.4 Grafana Interface Elements

- **Navigation**: Left sidebar with Home, Bookmarks, Starred, Dashboards sections
- **Dashboards Section**: Expandable subsections for Playlists, Snapshots, Library panels, Shared dashboards
- **Additional Features**: Drilldown, Alerting, Connections, Administration sections available

4.5 Customizing Your View

To customize your dashboard:

1. Click the "Add" button in the top right to add new panels
2. Use the "Settings" button to modify dashboard properties
3. Click directly on a panel and select "Edit" to modify specific visualizations
4. Adjust the time range using the "Last 6 hours" dropdown
5. Use the refresh button to update data manually or set an auto-refresh interval

4.6 Key Observations

- The system appears to be running with minimal resource utilization (0.214% memory, low CPU)
- API endpoints show variable request rates with clear patterns
- The 95ms latency indicates good performance for the service
- Storage I/O shows periodic activity patterns rather than sustained load
- This appears to be a monitoring dashboard for a Twitter sentiment analysis API rather than displaying the sentiment analysis results themselves

Chapter 5

Maintenance

5.1 Regular Maintenance Tasks

5.1.1 Daily Checks

- Monitor dashboard for unusual activity
- Verify the application is accessible

5.1.2 Weekly Tasks

- Review log files for warnings or errors
- Check disk space usage

5.1.3 Monthly Activities

- Review model performance metrics
- Update Docker images if newer versions are available

5.2 Updating the System

When updates become available:

1. Download the latest version
2. Stop the current instance:

```
docker-compose down
```

3. Replace application files with the new version
4. Rebuild and restart:

```
docker-compose up --build -d
```

Chapter 6

Evaluation & Testing

6.1 Model Evaluation

6.1.1 Performance Metrics

The sentiment analysis model is evaluated using the following metrics:

- **Accuracy**: Overall correctness of predictions (currently 75.9%)
- **Precision**: Proportion of positive identifications that are correct
- **Recall**: Proportion of actual positives that are identified correctly
- **F1 Score**: Harmonic mean of precision and recall (currently 0.76)

6.1.2 Classification Report

```
Classification Report:
              precision    recall  f1-score   support
              precision    recall  f1-score   support

           0       0.78      0.73      0.75    159656
           1       0.74      0.79      0.77    159547

    accuracy                           0.76    319203
              precision    recall  f1-score   support

           0       0.78      0.73      0.75    159656
           1       0.74      0.79      0.77    159547
              precision    recall  f1-score   support

           0       0.78      0.73      0.75    159656
              precision    recall  f1-score   support

              precision    recall  f1-score   support
              precision    recall  f1-score   support
              precision    recall  f1-score   support
              precision    recall  f1-score   support

           0       0.78      0.73      0.75    159656
    accuracy                           0.76    319203
   macro avg       0.76      0.76      0.76    319203
weighted avg       0.76      0.76      0.76    319203
```

Figure 6.1: Classification Report for Sentiment Classification

6.2 Model Monitoring

6.2.1 Drift Detection

The system continuously monitors for concept drift:

- **Data Drift**: Changes in input data distribution
- **Concept Drift**: Changes in the relationship between input and output
- **Model Drift**: Degradation of model performance over time

6.2.2 Performance Over Time

The system tracks performance metrics over time to identify:

- Gradual degradation requiring retraining
- Sudden drops indicating potential issues
- Seasonal or cyclical performance patterns

6.3 Unit Testing

6.3.1 Test Types

- **Unit Tests**: Test individual functions and classes

6.3.2 Running Tests

To run the test suite:

```
# Run unit tests
pytest tests/
```

6.4 Test Cases

6.4.1 Data Pipeline Tests

Data Ingestion Tests

- Tests that data is loaded correctly from various sources
- Verifies handling of different file formats
- Checks error handling for corrupted data
- Validates schema enforcement

Data Validation Tests

- Tests for detection of invalid data
- Verifies data quality checks
- Validates handling of missing values
- Checks statistical profiling of datasets

Preprocessing Tests

- Tests text normalization
- Verifies tokenization and stemming
- Validates feature extraction
- Checks data transformation pipelines

6.4.2 Model Tests

Training Tests

- Tests model initialization
- Verifies training procedures
- Validates hyperparameter tuning
- Checks model serialization and loading

Inference Tests

- Tests prediction accuracy
- Verifies handling of edge cases
- Validates performance under load
- Checks response formatting

6.4.3 Example Test Case

Below is an example of a test case from the test suite:

```
def test_model_prediction():
    with open(MODEL_PATH, "rb") as f:
        model = pickle.load(f)
    with open(VECTORIZER_PATH, "rb") as f:
        vectorizer = pickle.load(f)

    text = "I am so happy!"
    transformed = vectorizer.transform([text])
    prediction = model.predict(transformed)

    assert isinstance(prediction, np.ndarray) or isinstance(prediction,
        list)
    assert len(prediction) == 1
    assert prediction[0] in [0, 1]  # assuming binary classification
```

Chapter 7

Troubleshooting

7.1 Common Issues

Application Not Accessible

Issue: Cannot access the application at http://localhost:8000
Solution:

1. Check if Docker is running
2. Verify all containers are up with `docker ps`
3. Check logs with `docker logs twitter-sentiment-fastapi`
4. Restart the application with `docker-compose restart`

Slow Performance

Issue: Analysis takes longer than expected
Solution:

1. Check system resources (CPU, memory)
2. Reduce batch size for large analyses
3. Clear application cache in Settings

Inaccurate Results

Issue: Sentiment predictions seem incorrect
Solution:

1. Ensure text is clear and in English
2. Check for sarcasm or ambiguity which may confuse the model
3. Report example to improve future versions

Chapter 8

Appendix

8.1 Technical Details

8.1.1 System Architecture

The application uses a microservices architecture:

- **Frontend**: FastAPI web interface
- **Backend**: Python-based ML inference service
- **Monitoring**: Prometheus and Grafana
- **ML Pipeline**: DVC and MLflow

8.1.2 File Structure

```
twitter-sentiment-AI-project/
          data/
                raw/
                processed/
          notebooks/
                eda.ipynb
          notebooks/
          src/
                data/
                      data_ingestion.py
                      data_validation.py
                      preprocessing.py
                models/
                      train.py
                evaluation/
                      evaluate.py
                deployment/
                    api.py
                    docker/
                        Dockerfile

          configs/
                config.yaml
          tests/
                test_data_pipeline.py
                test_model.py
          requirements.txt
```

```
28          .gitignore
29          dvc.yaml
30          docker-compose.yml
31          README.md
```

8.2 Command Reference

Command	Description
`docker-compose up -d`	Start all services
`docker-compose down`	Stop all services
`docker logs [container]`	View container logs
`dvc repro`	Rerun ML pipeline
`mlflow ui`	Launch experiment tracking UI

Table 8.1: Common Commands

8.3 Glossary

- **Sentiment Analysis**: Determining whether text expresses positive or negative
- **ML (Machine Learning)**: Computer systems that learn from data
- **SVM**: Support Vector Machine, the model used for classification
- **TF-IDF**: Term Frequency-Inverse Document Frequency, a text feature extraction method
- **Data Drift**: When new data differs significantly from training data
- **Docker**: Container platform for running applications consistently
- **FastAPI**: Web framework for building APIs
- **Prometheus**: Monitoring and alerting toolkit
- **Grafana**: Analytics and monitoring platform
- **MLflow**: Platform for ML lifecycle management
- **DVC**: Data Version Control for ML projects