

# **Lucid: A User-Adaptive Real/Fake Information Detection System**

## Technical Documentation

November 23, 2025

This document provides comprehensive technical documentation for Lucid, a web application designed to detect manipulated or AI-generated content in texts and images. Lucid offers real-time predictions and allows users to enhance the model's accuracy by uploading their own datasets. This documentation details the system architecture, methodology, innovative user-driven dataset integration, installation, deployment, and evaluation.

# Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	Problem Statement . . . . .	3
1.2	Project Objectives . . . . .	3
<b>2</b>	<b>System Overview</b>	<b>3</b>
2.1	Architecture . . . . .	3
2.2	Technology Stack . . . . .	3
<b>3</b>	<b>Methodology</b>	<b>4</b>
3.1	Overview . . . . .	4
3.2	Data Collection and Preprocessing . . . . .	4
3.3	Feature Extraction: TF-IDF . . . . .	4
3.4	Machine Learning Model . . . . .	4
3.5	Training Pipeline . . . . .	4
3.6	Inference Pipeline . . . . .	5
<b>4</b>	<b>Frontend System</b>	<b>5</b>
<b>5</b>	<b>Backend System (Flask)</b>	<b>5</b>
<b>6</b>	<b>User-Driven Dataset Integration</b>	<b>5</b>
6.1	Dataset Format and Validation . . . . .	6
6.2	Retraining Workflow . . . . .	6
6.3	Impact and Advantages . . . . .	6
<b>7</b>	<b>Installation and Deployment</b>	<b>6</b>
7.1	Prerequisites . . . . .	6
7.1.1	Backend Installation . . . . .	6
7.1.2	Starting the Backend . . . . .	6
7.2	Frontend Hosting . . . . .	7
7.3	Deployment Options . . . . .	7
<b>8</b>	<b>Limitations and Future Work</b>	<b>7</b>
8.1	Current Limitations . . . . .	7
8.2	Future Enhancements . . . . .	7
<b>9</b>	<b>Conclusion</b>	<b>8</b>

# 1 Introduction

## 1.1 Problem Statement

The proliferation of fabricated and AI-generated content has created an urgent need for reliable detection mechanisms. Misinformation can impact academic, social, and political domains, making automated verification critical. Lucid addresses this challenge by offering a user-adaptive system capable of classifying content as authentic or fake.

## 1.2 Project Objectives

- Develop a robust classification system for detecting fake and real content.
- Implement an intuitive web interface for text and image submission.
- Achieve high accuracy using classical machine learning techniques.
- Introduce an innovative user dataset upload feature for adaptive learning.
- Ensure transparency through confidence scores and interpretable outputs.

# 2 System Overview

## 2.1 Architecture

Lucid follows a three-tier client-server architecture for scalability and maintainability. The workflow consists of:

- **Frontend:** HTML/CSS/JavaScript interface for user interaction.
- **Backend:** Flask server handling requests, processing inputs, and communicating with the ML layer.
- **Machine Learning Layer:** TF-IDF vectorization and LinearSVC model for prediction.

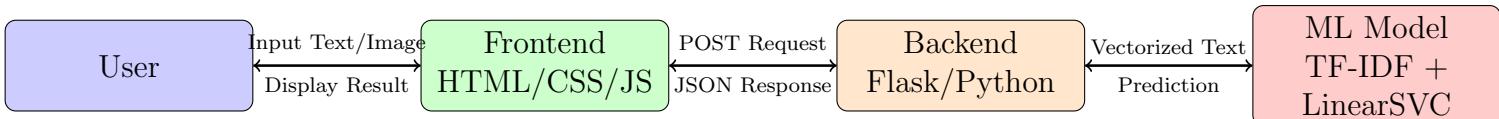


Figure 1: System Architecture of Lucid

## 2.2 Technology Stack

Component	Technology
Frontend	HTML, CSS, JavaScript
Backend	Flask (Python)
ML Framework	Scikit-learn (TF-IDF + LinearSVC)
Data Processing	Pandas, NumPy
Version Control	Git

## 3 Methodology

### 3.1 Overview

Lucid uses a classical ML approach to classify input text as **REAL** or **FAKE**, augmented by the innovative capability for user dataset integration. The methodology includes dataset preprocessing, TF-IDF feature extraction, model training, saving/loading of models, and integration with a Flask-based backend.

### 3.2 Data Collection and Preprocessing

The initial dataset `fake_or_real_news.csv` contains labeled textual news articles. Preprocessing steps include:

- Extracting the `text` and `label` columns.
- Mapping labels: FAKE → 0, REAL → 1.
- Splitting into training (80%) and testing (20%) sets.
- Cleaning text implicitly during TF-IDF vectorization (stopword removal, normalization).

### 3.3 Feature Extraction: TF-IDF

- Converts raw text into numerical vectors.
- `TF-IDF Vectorizer(stop_words='english', max_df=0.7)`.
- Captures term importance while reducing noise from frequent words.
- Persisted using Joblib for later inference.

### 3.4 Machine Learning Model

Linear Support Vector Classifier (`LinearSVC`):

- Performs well on sparse high-dimensional data.
- Fast training and binary classification.
- Stored as `linear_svc_model.joblib`.

### 3.5 Training Pipeline

1. Load dataset.
2. Split into training/testing.
3. Fit TF-IDF vectorizer.
4. Transform text into vectors.
5. Train `LinearSVC` classifier.
6. Save model/vectorizer using Joblib.
7. Load automatically in Flask backend.

### **3.6 Inference Pipeline**

1. User submits text/image via frontend.
2. Flask retrieves input from POST request.
3. Text is vectorized using TF-IDF model.
4. LinearSVC predicts REAL/FAKE.
5. Backend returns result.
6. Frontend displays prediction.

## **4 Frontend System**

- Text input form for news classification.
- Image upload section.
- User dataset upload interface.
- Responsive layout with modern blur effects.

## **5 Backend System (Flask)**

- Loads ML model/vectorizer.
- Handles text submission.
- Processes optional dataset uploads.
- Returns classification results.

## **6 User-Driven Dataset Integration**

Lucid's unique innovation is allowing users to upload their own datasets, enabling:

- Adaptive learning to new misinformation patterns.
- Culturally aware content verification.
- Personalized domain-specific training (e.g. historical, medical, political).

## 6.1 Dataset Format and Validation

Uploaded CSV must include:

```
text,label  
"Some news text",REAL  
"Fake news content",FAKE
```

Validation checks:

- Required columns exist.
- Labels follow binary classification.
- UTF-8 encoding.

## 6.2 Retraining Workflow

1. Validate and load CSV.
2. Append to existing dataset.
3. Refit TF-IDF vectorizer.
4. Retrain LinearSVC classifier.
5. Persist updated model/vectorizer.

## 6.3 Impact and Advantages

- User-adaptive, evolving detection system.
- Domain-specialized content detection.
- Collaborative improvement without advanced deep learning.

# 7 Installation and Deployment

## 7.1 Prerequisites

- Python 3.8+
- Pip
- Flask, scikit-learn, pandas, joblib

### 7.1.1 Backend Installation

```
1 pip install flask scikit-learn pandas joblib numpy
```

### 7.1.2 Starting the Backend

```
1 python app.py
```

## 7.2 Frontend Hosting

- Static HTML/CSS/JS served via Flask templates.
- Optionally host on Netlify, Vercel, or any static server.

## 7.3 Deployment Options

- Docker with exposed port 5000.
- Cloud hosting: AWS EC2, Render, Railway, DigitalOcean.
- Hybrid: Frontend on Netlify/Vercel, backend on Render/Railway.

# 8 Limitations and Future Work

## 8.1 Current Limitations

- **Model Complexity:** The TF-IDF + LinearSVC approach, while interpretable, may lack the sophistication of transformer-based models for complex linguistic patterns.
- **Domain Specificity:** Performance may vary across different content domains (historical, political, scientific) without domain-specific fine-tuning.
- **Language Restriction:** Currently optimized for English documents, limiting global applicability.
- **Computational Constraints:** Retraining with large user-uploaded datasets may require significant computational resources.
- **Real-time Performance:** Batch processing of large datasets may impact response times during retraining phases.
- **Feature Engineering Limitations:** TF-IDF may not capture complex semantic relationships as effectively as deep learning approaches.

## 8.2 Future Enhancements

- **Multimodal Integration:**
  - Image analysis using CNN-based feature extraction
  - Cross-modal consistency checking between text and images
- **Advanced Model Architectures:**
  - Integration of transformer models (BERT, RoBERTa) for improved semantic understanding
  - Ensemble methods combining multiple classifiers
  - Deep learning approaches for image forensics and deepfake detection

- **User Experience Improvements:**

- Explanatory AI features showing decision rationale
- Progressive web application (PWA) for mobile compatibility
- Batch processing interface for multiple content verification

- **Technical Enhancements:**

- Video and document analysis capabilities
- Multi-language support using multilingual models
- Incremental learning for continuous model improvement
- Advanced dataset validation and preprocessing pipelines
- Caching mechanisms for improved performance

- **Deployment and Scalability:**

- Microservices architecture for better scalability
- Containerization with Docker and Kubernetes
- Cloud-native deployment with auto-scaling capabilities
- API rate limiting and usage analytics

- **Community and Collaboration:**

- Public API for developer integration
- Community dataset sharing platform
- Collaborative model improvement programs
- Integration with fact-checking organizations

## 9 Conclusion

Lucid provides a user-adaptive framework for real/fake historical content detection. Its modular design integrates TF-IDF vectorization, LinearSVC classification, and a Flask backend for seamless web interaction. The innovative user dataset upload feature empowers individuals to contribute new domain-specific data, enhancing cultural awareness and continuous learning. The system balances interpretability, efficiency, and flexibility, forming a strong foundation for combating misinformation in diverse contexts. Future extensions include transformer-based models, multimodal analysis, and broader deployment strategies to further strengthen detection capabilities.