#### Project 1

#### **AI vs Human Text Detection**

# Machine Learning Classification Web Application with Streamlit

# **Project Overview**

In this project, students will build a live web application that can analyze MS word or PDF documents and determine whether the text was written by an AI or a human. The system will provide probability scores indicating the likelihood of AI vs human authorship.

#### What You'll Build

- Develop three classifiers (SVM, Decision Tree, AdaBoost) with hyperparameters fine tuning
- Train on AI vs Human text datasets
- Adapt the features for AI detection
- Customize the interface for your specific use case

# You will develop a web application where users can:

- 1. Upload text files or type or paste text directly
- Choose which model to use from your trained classifiers (SVM, Decision Tree, AdaBoost)
- 3. Get Instant Predictions: Real-time AI vs Human classification with confidence scores
- 4. See Visualizations: Feature importance, text statistics
- 5. Compare Models: Side-by-side performance comparison
- 6. Download Reports: Comprehensive analysis reports

#### **Technical Stack**

- Main Framework: Streamlit with Python
- scikit-learn (machine learning)
- pandas/numpy (data processing)

### **Document Processing**

- PyPDF2/pdfplumber (PDF text extraction)
- python-docx (Word document processing)
- NLTK/spaCy (text preprocessing)

### **Machine Learning**

- You will build 3 Classifiers: SVM, Decision Tree, AdaBoost
- Feature Engineering: TF-IDF, linguistic features, AI-detection specific features
- Model Evaluation: Cross-validation, performance metrics

#### Visualization

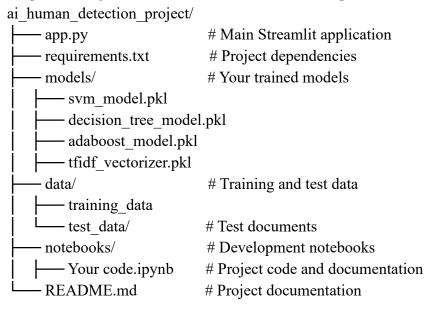
- matplotlib/seaborn (static plots)
- plotly (interactive visualizations)
- wordcloud (text visualization)

# **Learning Objectives**

By completing this project, you will:

- Learn document text extraction techniques (PDF, Word, plain text)
- Understand advanced text preprocessing for AI detection
- Implement and compare multiple classification algorithms
- Build a complete ML pipeline from data input to prediction
- Create a user-friendly interface for file uploads and text analysis
- Evaluate model performance on a challenging classification task
- Deploy a production-ready web application

# **Required Project Structure: Create the Following Folder Structure for Project 1:**



### **Required Deliverables & Grading = 100 Marks**

- 1. Jupyter Notebooks (50%): Your notebook should contain the following sections:
  - Data Analysis & Preprocessing: Data exploration and preprocessing sections where you analyze your AI vs Human text datasets, identify patterns, and prepare your data for model training. Implement different features: engineering and selection processes, extracting meaningful features.
  - Model Development & Optimization: Model training and hyperparameter tuning efforts for all three required algorithms (SVM, Decision Tree, and AdaBoost), optimized each model's performance through techniques like grid search or random search.
  - Performance Evaluation & Analysis: Performance evaluation and comparison sections that include accuracy metrics, confusion matrices, ROC curves, and detailed analysis comparing the strengths and weaknesses of each model on your specific dataset.

- 2. Streamlit Web Application (30%): You should deploy Streamlit web application that should make real-time predictions with explanations, and provide detailed explanations, and insights into why the model made its particular decision.
- 3. Documentation & Report (10%): Provide a comprehensive README file with detailed setup instructions that would allow another developer to clone your repository, install dependencies, and run your application successfully. Throughout your codebase, you should include thorough code documentation and comments that explain complex algorithms, design decisions, and any non-obvious implementation details.
- **4. Demo Video & Presentation (10%):** Create and submit a concise demo video (3-5 minutes) that showcases your application's key features, walks through the user interface, demonstrates predictions on sample texts, and highlights the unique aspects of your AI detection system.

# **Grading Rubric**

Component	Excellent (90-	Good (80-89%)	Satisfactory (70-	Needs
	100%)		79%)	Improvement
				(<70%)
Functionality	All features work	Minor bugs, core	Basic	Major issues or
	flawlessly, handles	features work	functionality	non-functional
	edge cases		present	
ML Models	All 3 models	All 3 models	All 3 models	Missing models or
	properly tuned,	working, 70-80%	implementation	poor performance
	>80% accuracy	accuracy		
Code Quality	Clean, documented,	Well-structured,	Functional but	Poor structure,
	efficient	mostly	messy	hard to follow
		documented		
User	Intuitive,	Good usability,	Functional but	Confusing or
Interface	professional design	clear layout	basic	poorly designed

#### **Technical Resources**

- Streamlit Documentation
- scikit-learn Model Selection
- Document Processing Libraries
- Scikit-learn User Guide
- NLTK Documentation
- Streamlit Gallery
- Deployment Guide: docs.streamlit.io/streamlit-community-cloud
- Cheat Sheet: docs.streamlit.io/library/cheatsheet