PROJECT REPORT

Project Title: AI-Powered Smart File Organizer

Course: INF-340- Artificial Intelligence

Team Members: Serhat Yiğit Aydın (210715034) , Alper Murtezaoğlu (220715007)

Problem Definition

Project Description:

In this project, we developed the "Smart File Organizer," an intelligent automation tool designed to resolve file clutter on computers. The system categorizes files by analyzing the semantic content of filenames rather than relying solely on file extensions.

The Problem:

Traditional operating systems sort files strictly by type (e.g., .pdf). However, in modern workflows, a .pdf file could be a "Lecture Note," an "Invoice," or a "Contract." Our goal was to build a system capable of understanding these contexts without compromising user privacy or system performance.

Project Type:

We defined this project as a Supervised Learning task utilizing Zero-Shot Classification. The fundamental challenge was to predict file categories based on short text strings (filenames) without a pre-trained dataset.

Data Description & Preprocessing

Data Source:

To test our system under realistic conditions, we created a synthetic dataset consisting of Turkish and English filenames frequently found on a student's desktop:

- **Academic:** "fizik_odevi_final.pdf", "matematik_notlari.docx"
- **Finance:** "elektrik_faturasi.pdf", "banka_dekontu.pdf"
- **Multimedia:** "tarkan_yolla.mp3", "tatil_fotografi.jpg"

Model Architecture & Methodology

Design Philosophy: Privacy-First & Hybrid Architecture

We designed the system with a **Privacy-First** approach. Instead of the slow and potentially privacy-invasive method of "reading file content," we focused solely on filename analysis. To optimize performance, we engineered a **Hybrid Filtering System**:

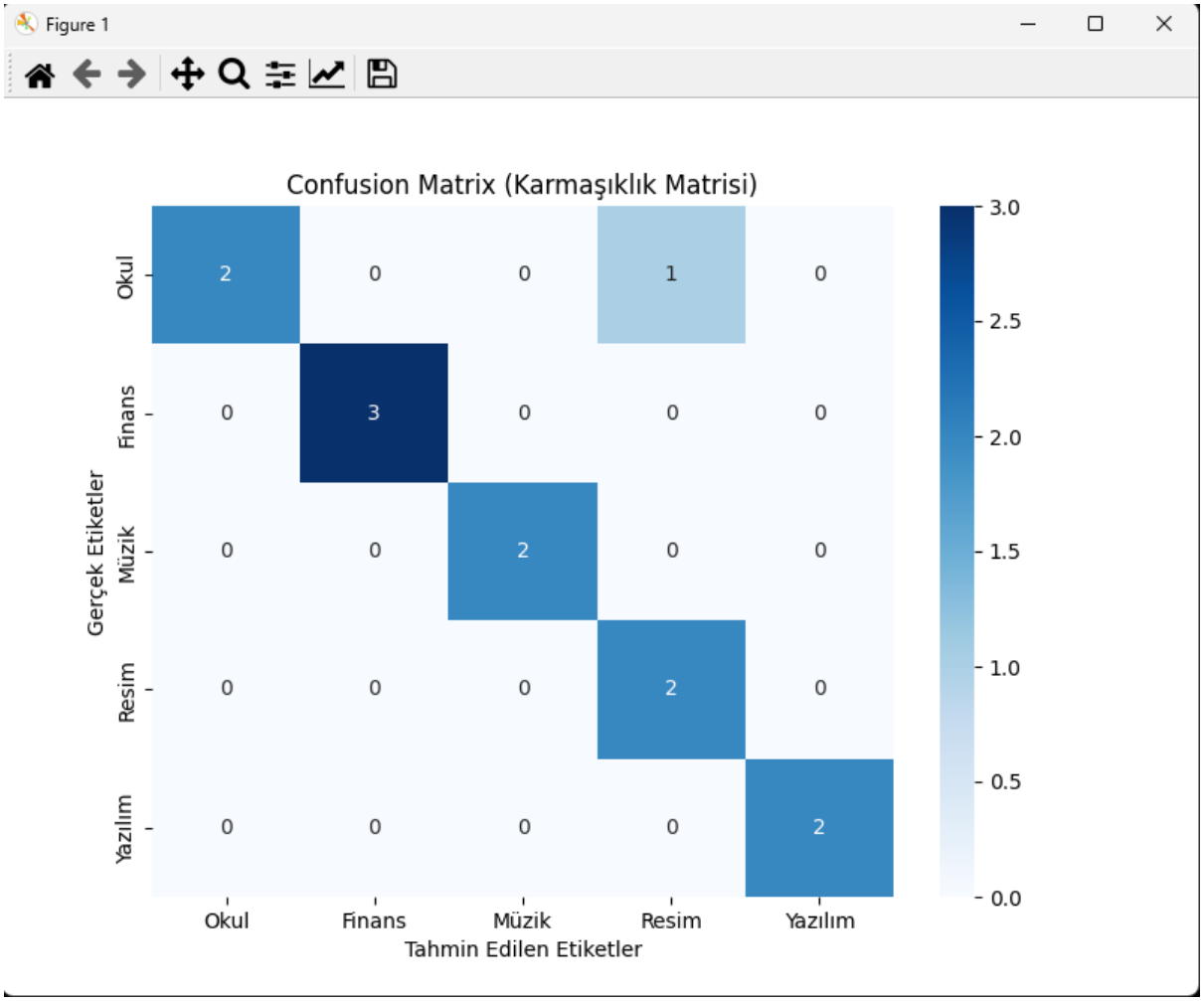
1. **Layer 1 (Rule-Based Filter):** We implemented a fast-path for files with clear content types (e.g., .mp3, .exe, .jpg). These files are processed directly without entering the AI model to reduce latency.
2. **Layer 2 (Semantic AI Analysis):** Files with ambiguous content (.pdf, .docx, .txt) are processed by the NLP model.

Results & Evaluation

Metrics:

We evaluated the system on a test set consisting of 12 different files.

Confusion Matrix:



AI & Prompt Engineering Log

During the development process, Generative AI (Gemini) was used as a coding assistant, particularly for **creating basic code skeletons** and **debugging**. Critical logic decisions were made by us, while the AI served as an accelerator during the implementation phase.

Task	**My Prompt**	**AI Output Summary**	**My Refinement / Action**
GUI Coding	"Write me a basic interface code in Python using a modern library instead of tkinter."	The AI suggested the customtkinter library and provided **boilerplate code** for a window.	The interface was freezing while the AI code was processing; therefore, I revised the code by adding a **Threading** structure.
Model Integration	"I need an NLP model code that can understand Turkish filenames."	The AI recommended the mDeBERTa model and provided a **ready-to-use code block** on	I integrated the provided code into my project's file reading loop and limited it to

		how to use it with the transformers library.	run only on .pdf/.docx files.
Debugging	"Keras 3 and Transformers libraries are conflicting, getting 'ModuleNotFoundError'. What is the solution?"	The AI identified the version mismatch and provided the necessary terminal commands (pip install tf-keras) for the solution.	The solution was applied, and the necessary **environment variable** was added to the beginning of the code.
Hybrid Logic Setup	"Asking AI for .mp3 files is too slow, how can I speed this up?"	The AI suggested an if-else logic, stating that using AI for files with obvious extensions is unnecessary.	Taking this suggestion into account, I implemented the **Hybrid Architecture** into the code, which moves media files directly and routes documents to the AI.