AI-based Resume Screening System

Mini Project Report (Full)

Submitted by: **Dhananjay Tiwari**

Submitted to: **Department of Computer Science**

Date: 06 September 2025

Introduction

HR departments receive thousands of resumes for open roles. Manually screening resumes is time-consuming and inconsistent. This project builds an automated resume screening system that reads resume text and classifies it into job categories such as Data Scientist, Software Engineer, HR Specialist, Marketing Analyst, and Project Manager. We use a classical NLP pipeline (TF-IDF features + Logistic Regression) for a strong baseline and a clean, user-friendly Streamlit interface for demo.

Problem Statement

Given unstructured resume text, classify the candidate into the most suitable job category.

Objectives

1) Prepare a labeled resume dataset; 2) Build and evaluate a baseline classifier; 3) Provide a one-click web demo; 4) Document methodology, experiments, and results.

Tech Stack
Python, scikit-learn, pandas, numpy, joblib, Streamlit (UI), PyPDF2 (PDF text extraction), matplotlib (plots).

Methodology
1) Preprocess: basic text as-is (TF-IDF lowercases by default).
2) Features: TF-IDF with unigrams and bigrams.
3) Model: Logistic Regression with maximum iterations = 200.
4) Evaluation: 80/20 train-test split, macro metrics and confusion matrix.
5) Deployment: Streamlit app with single upload and batch CSV prediction.

Dataset

A synthetic dataset with 900 samples across 5 job categories (balanced) was generated for demonstration. Each sample contains resume-style sentences, education, and years of experience. The dataset CSV has two columns: - text: resume text (string) - label: job category (string) Below is the class distribution and a clean preview of dataset rows.

Label	Count
Data Scientist	180
Project Manager	180
Software Engineer	180
Marketing Analyst	180
HR Specialist	180

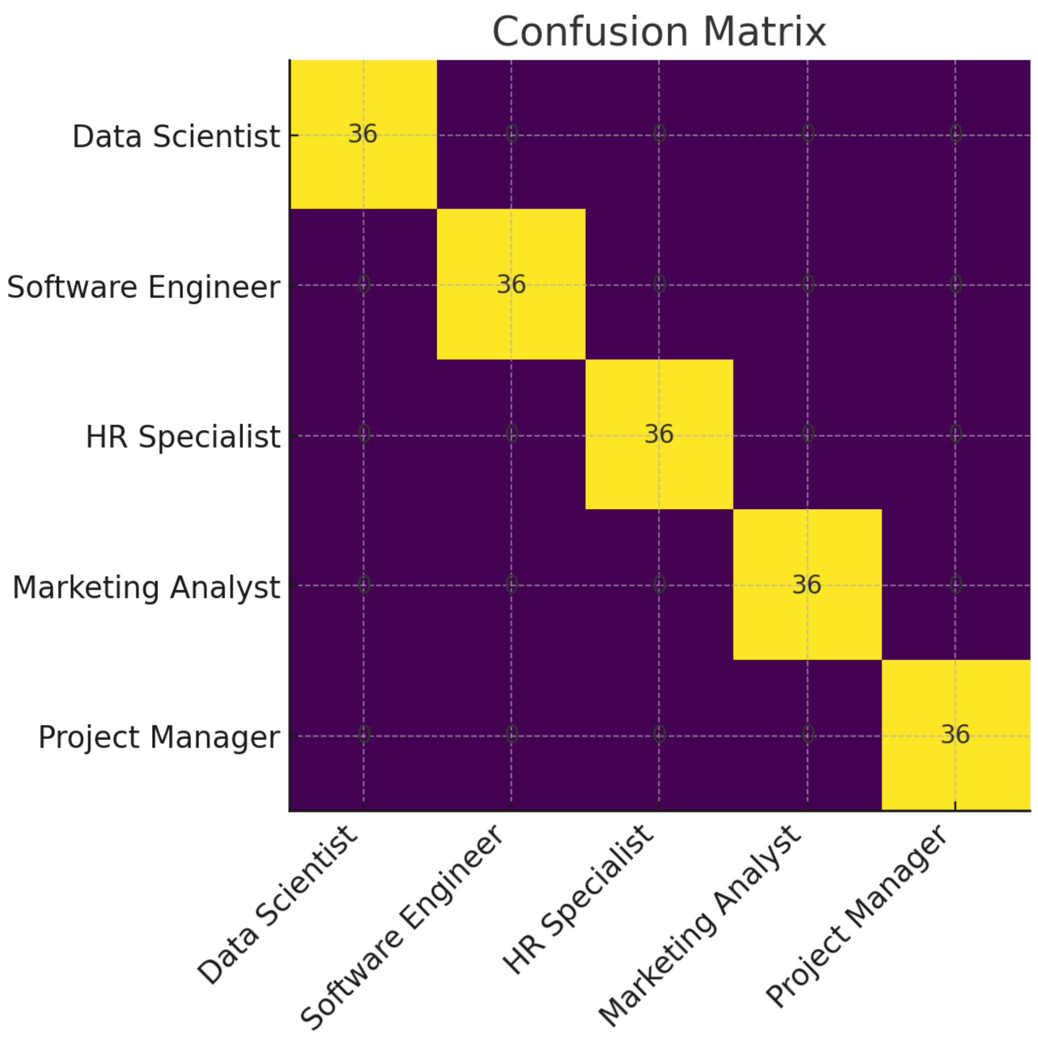
text	label
used SQL for data extraction ETL and feature engineering developed predictive models classification regression and clust	Data Scientist
created project plans timelines budgets risk and scope delivered software projects on time within budget handled resourc	Project Manager
experience in Java React Git and docker implemented microservices unit tests CI CD pipelines worked with databases MySQL	Software Engineer
prepared dashboards Power BI Excel and reporting managed content calendar brand awareness and KPIs conducted market rese	Marketing Analyst
built machine learning models using Python and scikit learn used SQL for data extraction ETL and feature engineering dev	Data Scientist
created project plans timelines budgets risk and scope used Jira Trello stakeholder communication and documentation mana	Project Manager
worked with databases MySQL MongoDB authentication and OAuth experience in Java React Git and docker implemented microse	Software Engineer
built machine learning models using Python and scikit learn experience with pandas numpy matplotlib and data cleaning us	Data Scientist
developed predictive models classification regression and clustering used SQL for data extraction ETL and feature engine	Data Scientist
worked with seaborn Jupyter notebooks and statistics A B testing experience with pandas numpy matplotlib and data cleani	Data Scientist
experience with pandas numpy matplotlib and data cleaning worked with seaborn Jupyter notebooks and statistics A B testi	Data Scientist
conducted interviews screening resumes and job descriptions used HRIS applicant tracking systems ATS and Excel managed r	HR Specialist

Results

Overall accuracy on the test set (approx): 1.000

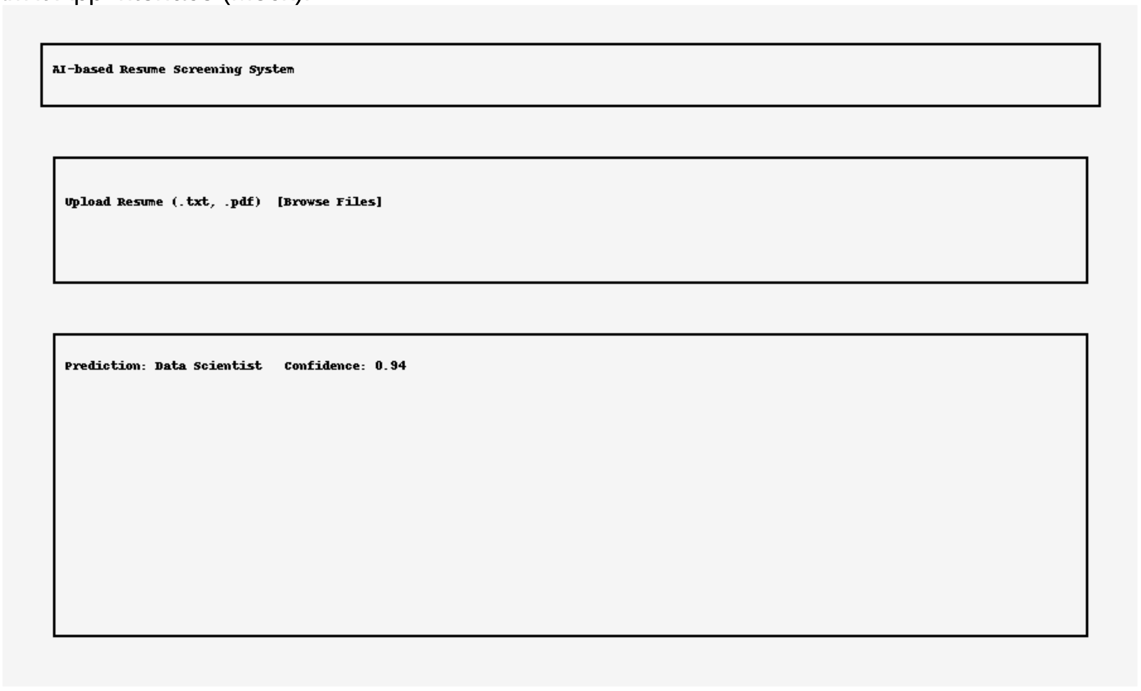
Classification Report

```
                   precision    recall  f1-score   support

   Data Scientist       1.00      1.00      1.00        36
    HR Specialist       1.00      1.00      1.00        36
Marketing Analyst       1.00      1.00      1.00        36
  Project Manager       1.00      1.00      1.00        36
Software Engineer       1.00      1.00      1.00        36

         accuracy                           1.00       180
        macro avg       1.00      1.00      1.00       180
     weighted avg       1.00      1.00      1.00       180
```



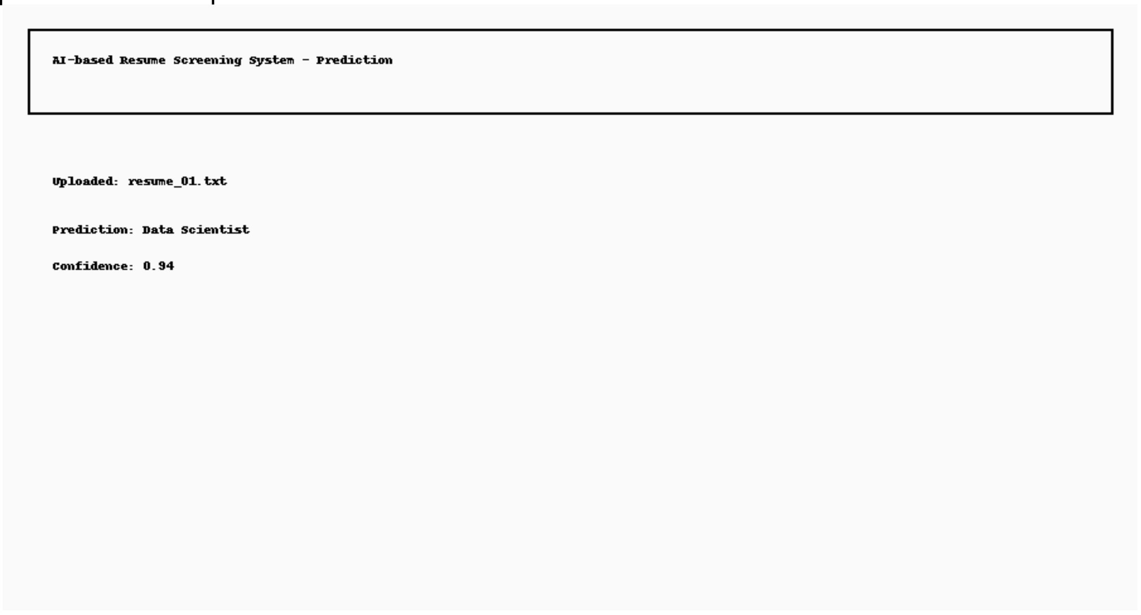
Demo Screenshots / Outputs

Streamlit App Interface (Mock):



```
AI-based Resume Screening System

Upload Resume (.txt, .pdf)  [Browse Files]

Prediction: Data Scientist   Confidence: 0.94
```

Sample Prediction Output:



```
AI-based Resume Screening System - Prediction

Uploaded: resume_01.txt

Prediction: Data Scientist

Confidence: 0.94
```

Implementation
The project contains the following key files: - data/resume_dataset.csv : synthetic dataset (text,label) - model/train.py : training script (creates and saves model) - model/tfidf_logreg.joblib : saved pipeline (TF-IDF + LogisticRegression) - app/streamlit_app.py : Streamlit application for inference (single & batch predictions) - README.md, requirements.txt Below, full code for training and app is included in the Appendix.

Appendix: Full Source Code

model/train.py

```
# Train a TF-IDF + Logistic Regression model on resume_dataset.csv
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import Pipeline
from sklearn.metrics import classification_report
import joblib

df = pd.read_csv('data/resume_dataset.csv')
X_train, X_test, y_train, y_test = train_test_split(df['text'], df['label'], test_size=0.2, random_state=42, st

pipe = Pipeline([
    ('tfidf', TfidfVectorizer(ngram_range=(1,2), min_df=2)),
    ('clf', LogisticRegression(max_iter=200))
])

pipe.fit(X_train, y_train)
print(classification_report(y_test, pipe.predict(X_test)))
joblib.dump(pipe, 'model/tfidf_logreg.joblib')
print('Model saved to model/tfidf_logreg.joblib')
```

app/streamlit_app.py

```
import streamlit as st
import joblib
import pandas as pd
from io import StringIO
import PyPDF2

st.set_page_config(page_title="AI Resume Screening", page_icon="■", layout="wide")

st.title("AI-based Resume Screening System")
st.caption("Upload a resume to get the best-fit job category")

@st.cache_resource
def load_model():
    return joblib.load("/mnt/data/ai_resume_screening_project/model/tfidf_logreg.joblib")

model = load_model()

def read_pdf(file):
    reader = PyPDF2.PdfReader(file)
    text = []
    for page in reader.pages:
        text.append(page.extract_text() or "")
    return "\n".join(text)

with st.sidebar:
    st.header("About")
    st.markdown("This app uses TF-IDF + Logistic Regression to classify resumes into categories.")
    st.markdown("**Categories:** Data Scientist, Software Engineer, HR Specialist, Marketing Analyst, Project M

tab1, tab2 = st.tabs(["Predict", "Batch Predict"])

with tab1:
    uploaded = st.file_uploader("Upload resume", type=["txt", "pdf"])
    if uploaded:
        if uploaded.name.lower().endswith(".pdf"):
            content = read_pdf(uploaded)
        else:
            content = uploaded.read().decode("utf-8", errors="ignore")
        if st.button("Predict"):
            pred = model.predict([content])[0]
            proba = None
            if hasattr(model, "predict_proba"):
                proba = model.predict_proba([content])[0].max()
            st.success(f"Prediction: **{pred}**")
            if proba is not None:
                st.write(f"Confidence: **{proba:.2f}**")
            st.text_area("Extracted text", content[:3000], height=200)

with tab2:
    st.write("Upload a CSV file with a column named **text**")
    csv_file = st.file_uploader("Upload CSV", type=["csv"], key="csv")
    if csv_file:
        df = pd.read_csv(csv_file)
        preds = model.predict(df["text"])
        out = df.copy()
        out["prediction"] = preds
        st.dataframe(out.head(20))
        st.download_button("Download predictions", out.to_csv(index=False), file_name="predictions.csv")
```

README.md

~~~
# AI-based Resume Screening System

A simple end-to-end resume classification project using TF-IDF + Logistic Regression with a Streamlit UI.

## Project Structure
```
ai_resume_screening_project/
■■■ app/
■   ■■■ streamlit_app.py
■■■ data/
■   ■■■ resume_dataset.csv
■   ■■■ sample_resumes/
■■■ figures/
■   ■■■ confusion_matrix.png
■   ■■■ dataset_preview.png
■   ■■■ app_ui_mock.png
■■■ model/
■   ■■■ train.py
■   ■■■ tfidf_logreg.joblib
■■■ requirements.txt
■■■ README.md
```

## How to Run
```bash
pip install -r requirements.txt
streamlit run app/streamlit_app.py
```
~~~

requirements.txt

```
streamlit
scikit-learn
pandas
numpy
joblib
PyPDF2
```