Università di Pisa

Artificial Intelligence and Data Engineering
Data Mining and Machine Learning Project

RESUME CLASSIFICATION USING MACHINE LEARNING

by: Filippo Gambelli and Andrea Bochicchio

INTRODUCTION

PROBLEMS

- Recruiters often receive hundreds of resumes for each job offer
- Manual review is time-consuming and subject to bias

SOLUTIONS

- A Machine Learning-based Classifier that automatically categorizes resumes into professional categories
- A Matching System between Resumes and Job descriptions

BENEFITS

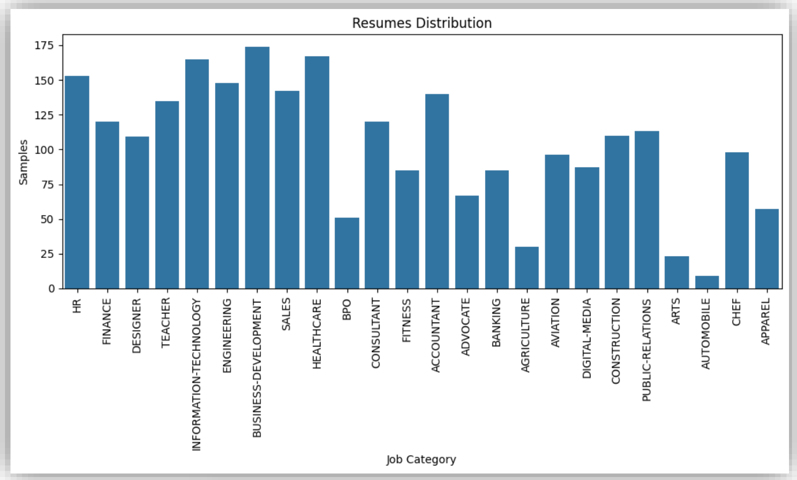
- Reduces processing time, minimises bias
- Improves hiring decisions by automating early screening stages

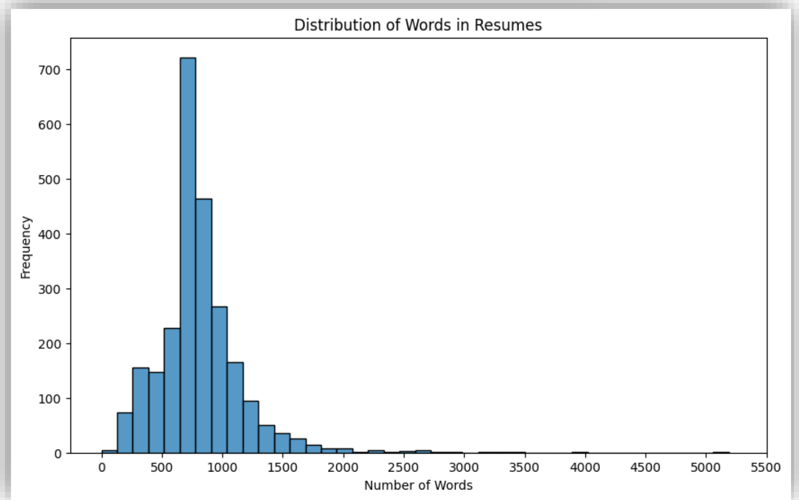
DATASET RESUME

Collection of **RESUMES** from LiveCareer.com, including both CVs in PDF format and the text extracted from them

2400+
Resumes

24
Categories





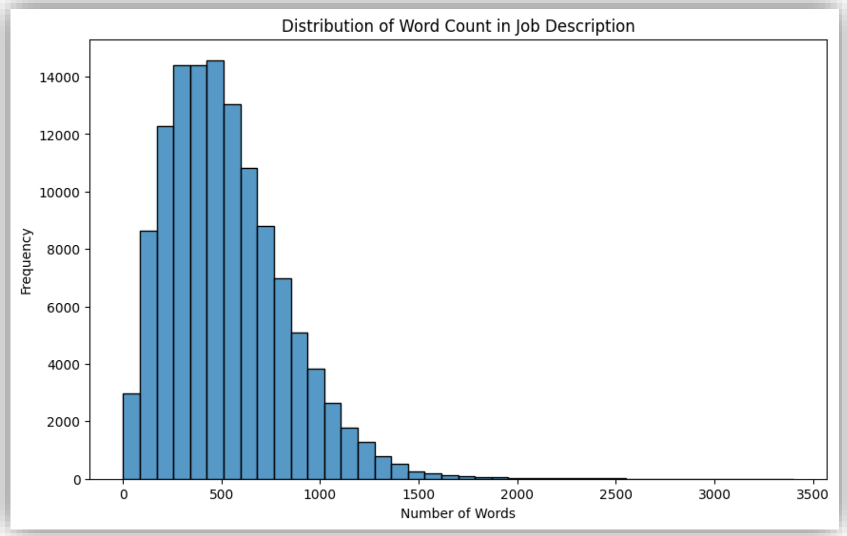
DATASET COLUMNS

Column	Description
ID	Unique identifier
Resume_str	Resume content in plain text format, extracted from the PDF files
Category	Job role, target label

DATASET JOB DESCRIPTION

Includes all **JOB POSTINGS** published on LinkedIn in 2023 and 2024

120.000+
Job Descriptions



DATASET COLUMNS

Column	Description
Job_id	Unique identifier
title	Job title as listed in the posting
description	Description of the published job posting

DATA PREPROCESSING

1. **DATA CLEANING**
   Removal of irrelevant or invalid records and fields

2. **TEXT NORMALIZATION**
   Convert to lowercase, remove punctuation and underscore, collapse multiple whitespace

3. **TOKENIZATION AND FILTERING**
   Splitting text into tokens. Remove tokens containing numerical digits and stopwords

4. **MORPHOLOGICAL PROCESSING**
   Applying lemmatization or stemming to reduce words to their base forms

5. **DATASET SPLITTING**
   Dividing data into training and test sets with stratified sampling

TEXT RAPPRESENTATION

TF-IDF

Term Frequency-Inverse Document Frequency is a method that turns text into numbers by giving more weight to words that appear often in one document but rarely across others, highlighting their importance and uniqueness

Word2Vec

Word2Vec maps words to dense vectors based on context, capturing semantic meaning. To represent documents, we computed a weighted average of word vectors using TF-IDF scores

Doc2Vec

Doc2Vec extends Word2Vec by generating vector representations for entire documents rather than individual words. It captures the semantic context of documents directly, without needing to aggregate words vector

SBERT

Sentence-BERT is a transformer-based model that provides semantically meaningful document embeddings. Unlike TF-IDF and Word2Vec, SBERT captures deep contextual and semantic information across entire documents

RESUME CLASSIFICATION

1

TRAIN-TEST SPLIT

Loaded a pre-split dataset (80% train, 20% hold-out test). The test set remained untouched for final evaluation

2

TEXT PREPROCESSING

Different cleaning steps applied depending on the feature extraction method

3

FEATURE EXTRACTION

TF-TDF, W2V, D2V, SBERT

4

IMBALANCE HANDLING

SMOTE oversampling applied to minority classes

5

CLASSIFICATION MODELS

LOGISTIC REGRESSION, **RANDOM FOREST** and **SUPPORT VECTOR MACHINE**

RESUME CLASSIFICATION

6

CROSS-VALIDATION AND HYPERPARAMETER TUNING

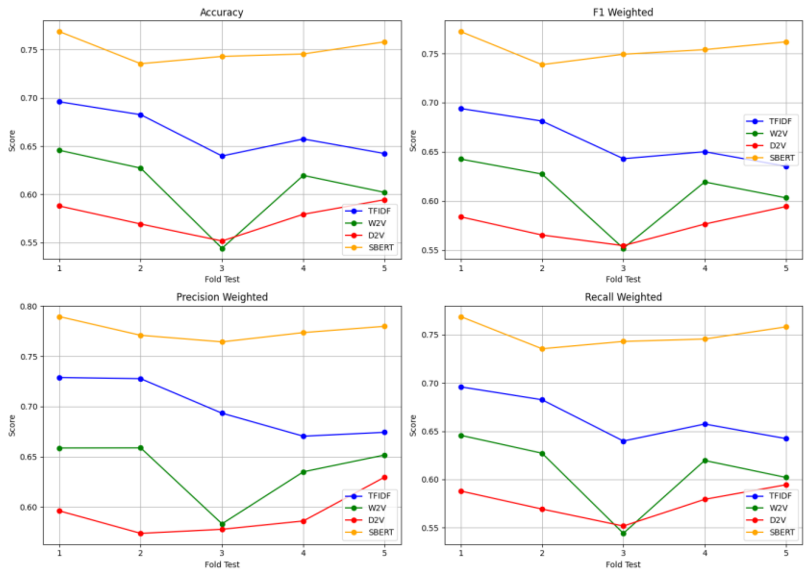
For each of the 3 classifier models and 4 feature extraction techniques (12 combinations total), we performed grid search with 5-fold stratified cross-validation to find the optimal hyperparameters

7

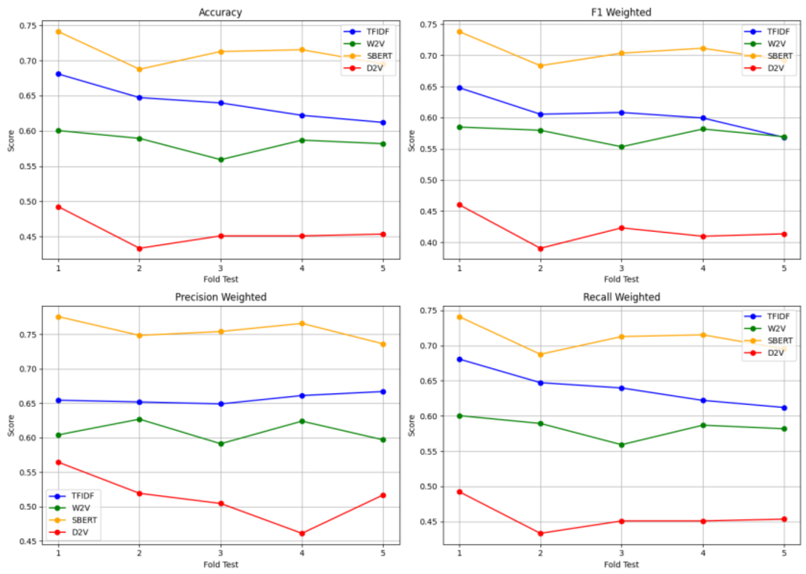
MODEL EVALUATION AND VISUALIZATION

Best Configuration per Classifier: for each classifier, we selected the best-performing feature extraction methods

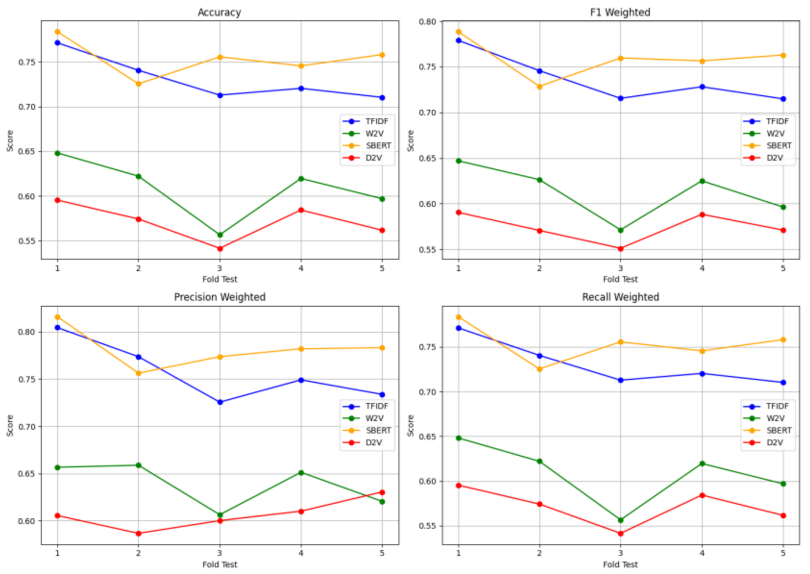
Final Model Comparison: the top configuration from each classifier was compared to identify the overall best-performing classifier model



Support Vector Machine



Random Forest



Logistc Regression

RESUME CLASSIFICATION

BEST-PERFORMING FEATURE EXTRACTION METHODS

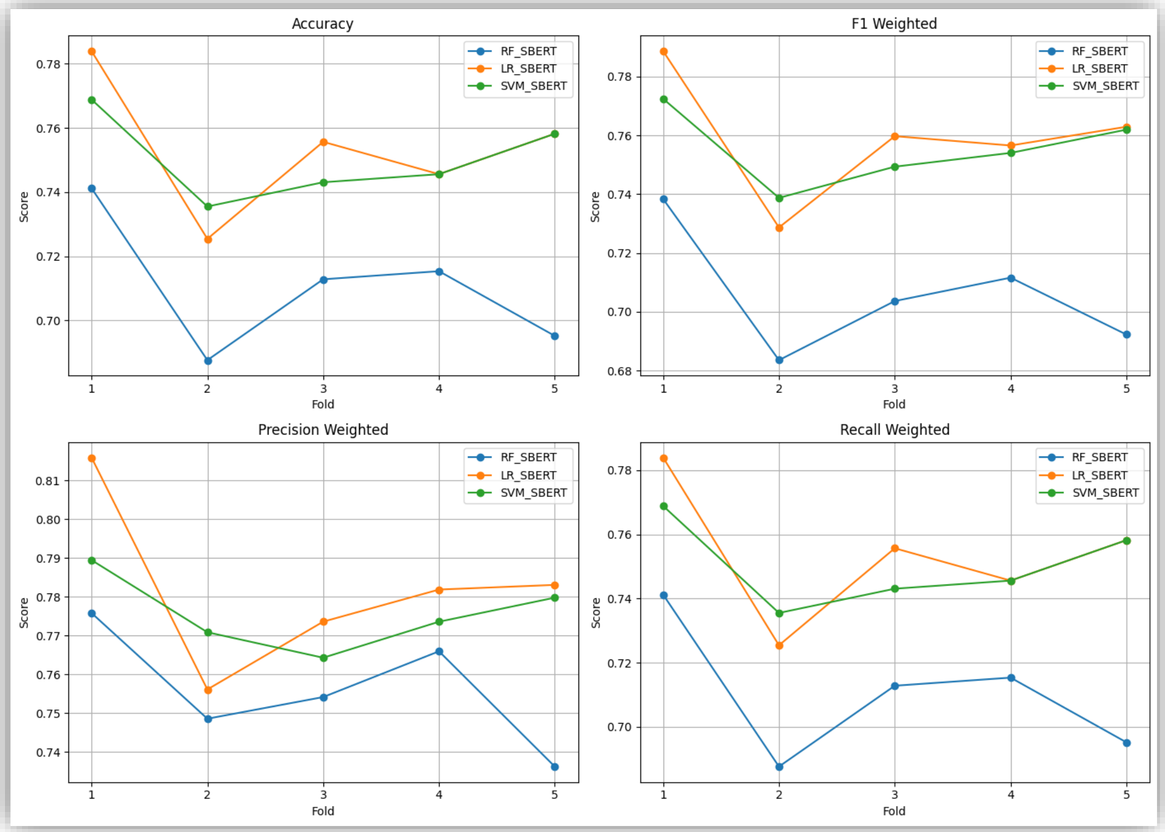
All three top-performing classifiers (Logistic Regression, Random Forest, SVM) used **Sentence-BERT (SBERT)** embeddings.
SBERT, a transformer-based model trained for **semantic similarity**, is well-suited for processing short to medium-length texts like resumes.

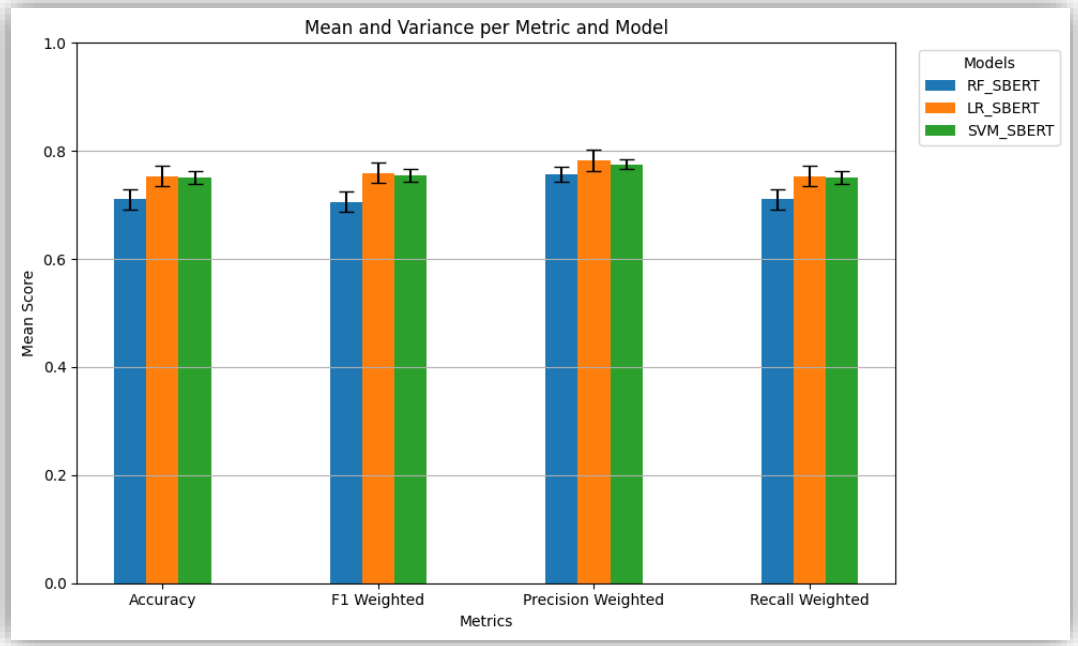
STATISTICAL SIGNIFICANCE TESTING

To evaluate whether performance differences between the top three models were meaningful, we used the **Wilcoxon signed-rank test**.
We tested differences in **F1-score** and **AUC** between each model pair.
Results: All p-values > 0.05 → **No statistically significant difference** between models.

RESUME CLASSIFICATION

FINAL MODEL CHOICE



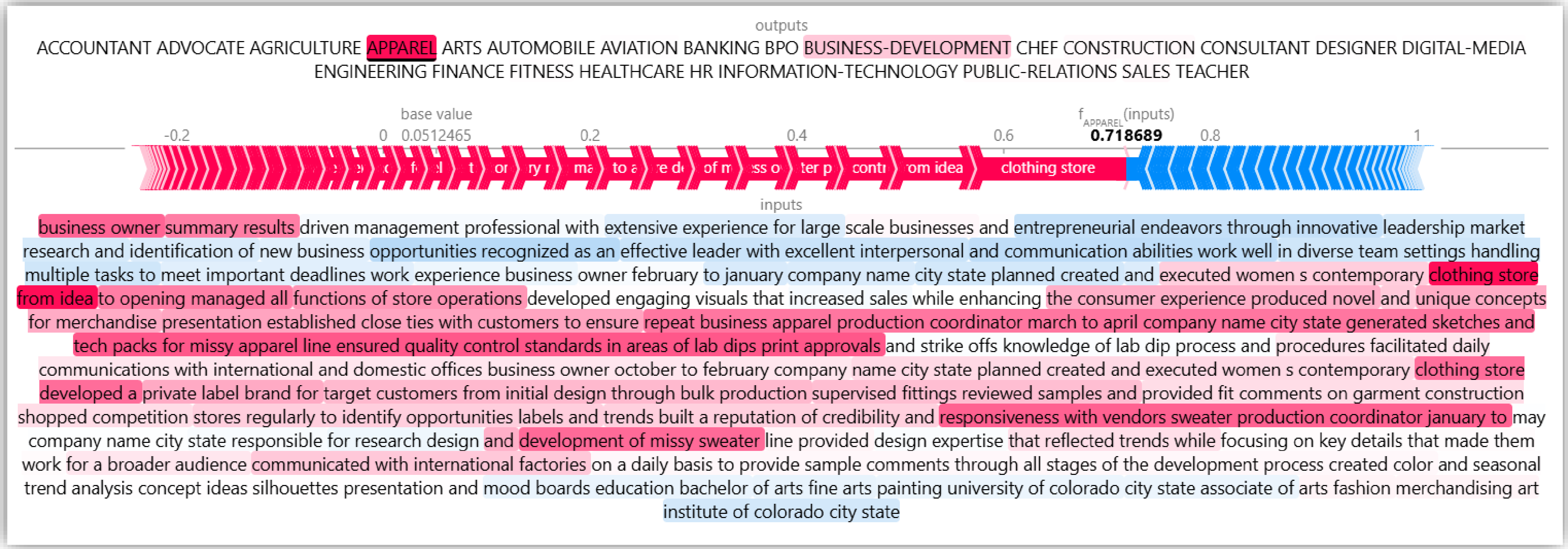


Based on this analysis, the **SUPPORT VECTOR MACHINE** with **SBERT EMBEDDINGS** was selected as the final model for resume classification, due to its strong overall performance and lower variance across evaluation folds

EXPLAINABILITY

GOAL: Explore **which parts of the input text** most influenced the model's classification for each job category

SHAP (SHapley Additive exPlanations) is model-agnostic interpretability method based on game theory



JOB & RESUME MATCHING

2nd APPLICATION FEATURE

Automate the selection of the most relevant resumes for a given job posting by computing the cosine similarity between the job description and all available resumes

1. **EMBEDDING WITH SBERT**
2. **SIMILARITY CALCULATION**
3. **TOP MATCHES**

$$\boxed{\text{cosine\_similarity}(A, B) = \frac{A \cdot B}{||A||\,||B||}}$$

JOB & RESUME MATCHING

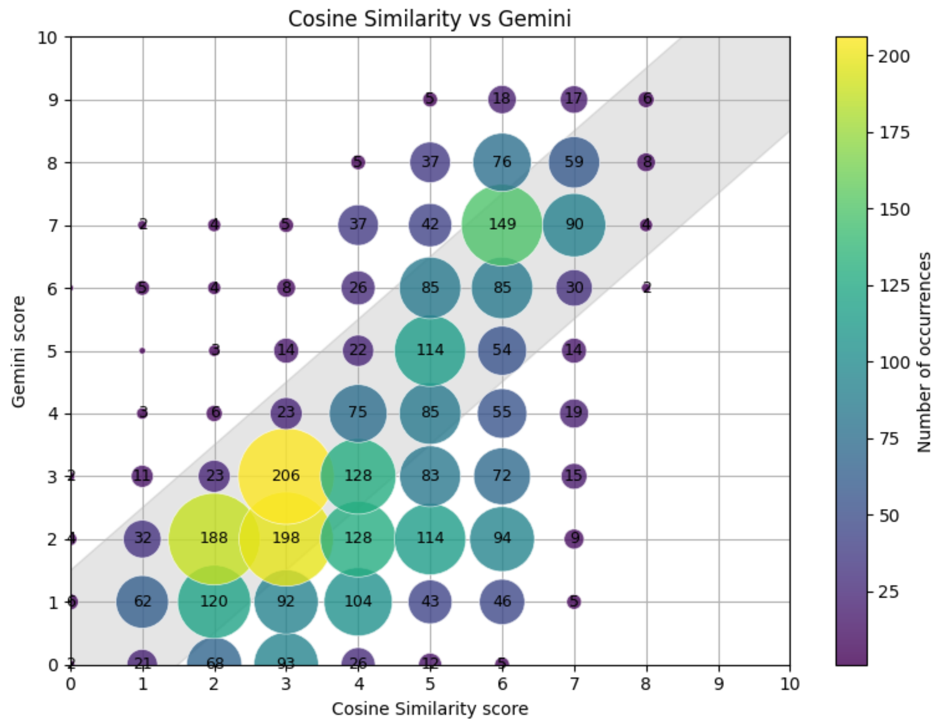
No labeled ground truth dataset available to benchmark resume-job matching accuracy

SOLUTION

Simulated human judgment with **LLM** (**Gemini 2.0 Flash)** to evaluate over 3,000 job-resume pairs and compare them with SBERT cosine similarity scores

EVALUATION METRICS

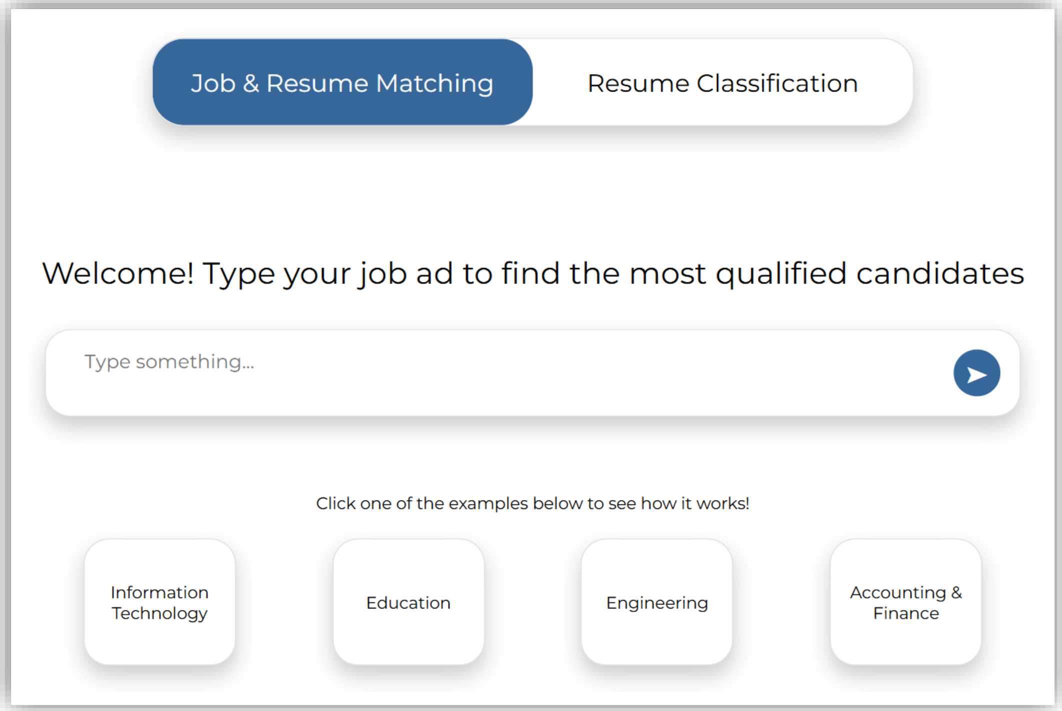
Metric	Value
Mean Absolute Error (MAE)	1.541
Mean Squared Error (MSE)	4.146
Pearson Correlation	0.623
% within ±2 Points of Gemini	75.9%

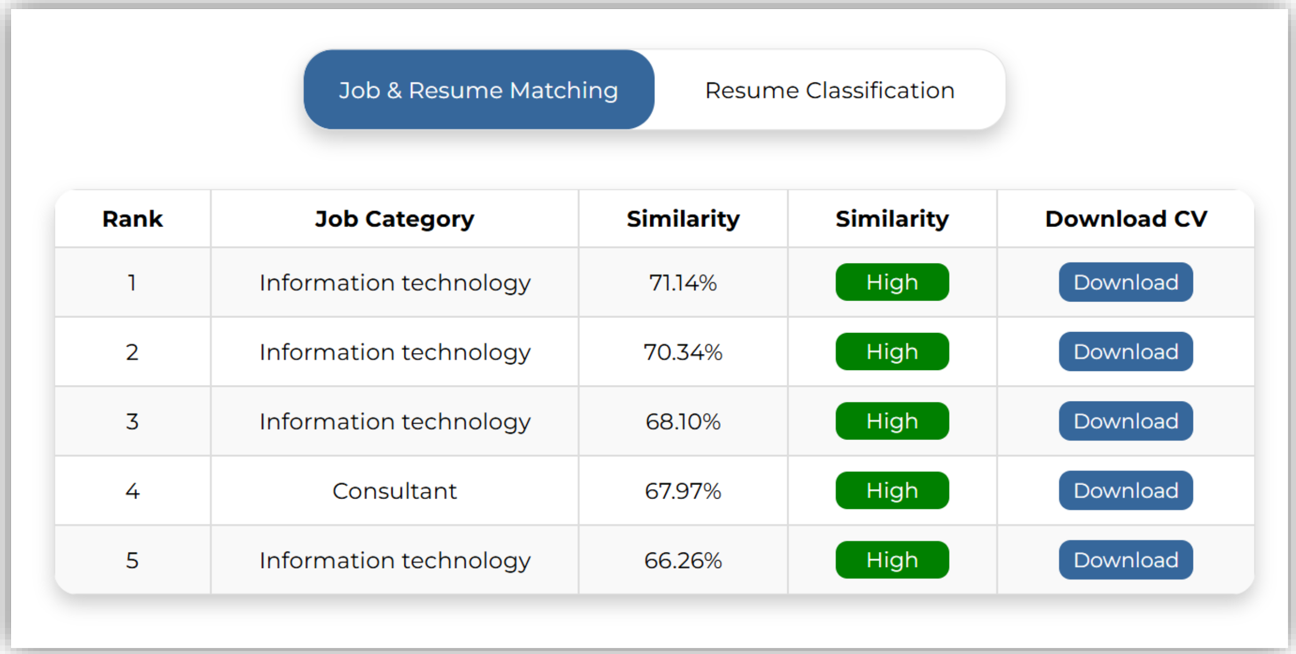


GRAPHICAL USER INTERFACE

The application features a simple, intuitive, and accessible user interface built with HTML, CSS, and JavaScript for the frontend, and Python with the Flask framework for the backend

JOB DESCRIPTION AND RESUME MATCHING PAGE



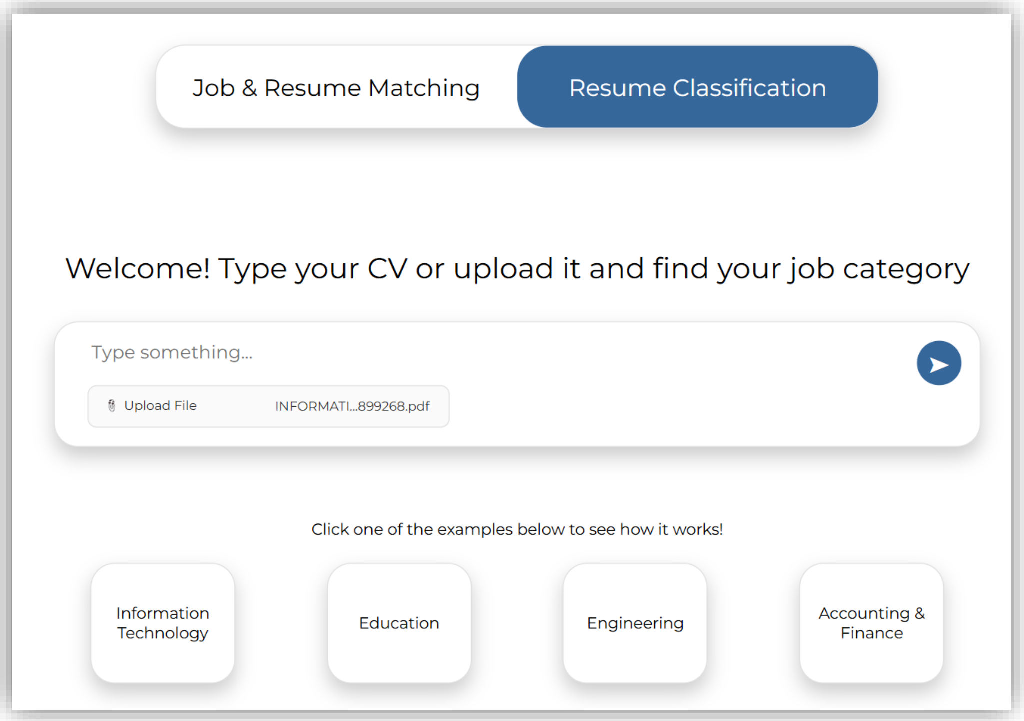


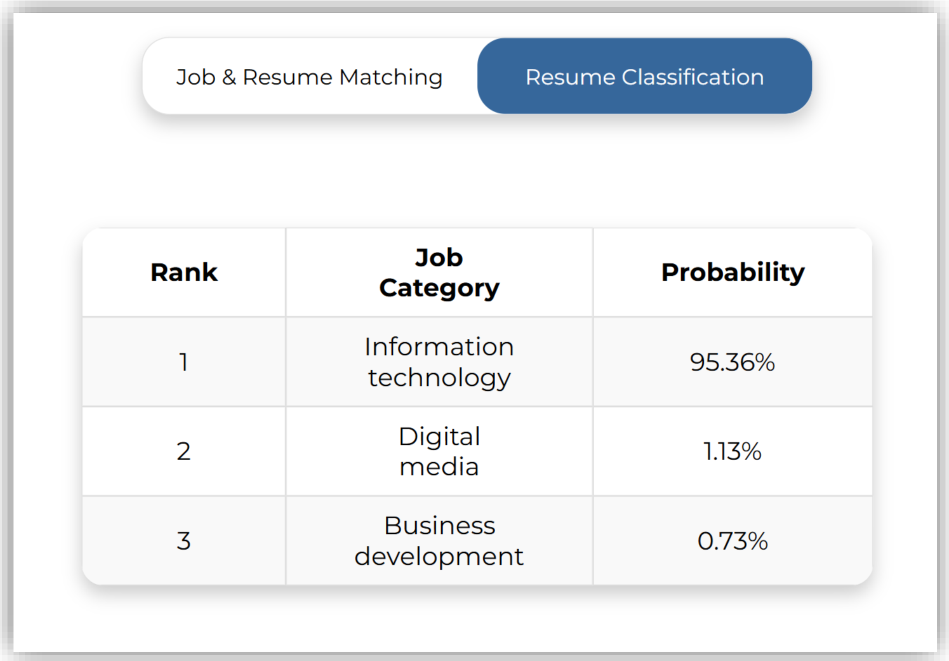
Rank	Job Category	Similarity	Similarity	Download CV
1	Information technology	71.14%	High	Download
2	Information technology	70.34%	High	Download
3	Information technology	68.10%	High	Download
4	Consultant	67.97%	High	Download
5	Information technology	66.26%	High	Download

GRAPHICAL USER INTERFACE

The application features a simple, intuitive, and accessible user interface built with HTML, CSS, and JavaScript for the frontend, and Python with the Flask framework for the backend

RESUME CLASSIFICATION PAGE





Rank	Job Category	Probability
1	Information technology	95.36%
2	Digital media	1.13%
3	Business development	0.73%