ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26may1118

Smart Resume Analysis and Ranking Using NLP and Machine Learning

Mrunali Wande[1]; Dr. Manisha Bharati[2]

[1]Department of Technology SPPU, Pune, India
[2]Department of Technology, SPPU, Pune, India

Publication Date: 2026/05/27

Abstract: However, modern recruitment process suffers from delays because of manual, inconsistent and biased analysis of numerous resumes. In this study, we propose to create a Smart Resume Analysis and Ranking System, which will be an advanced intelligent recruitment assistant designed as a four-tab Streamlit web application with multi-modal input/output interface. Our system includes Natural Language Processing (NLP) techniques and Machine Learning (ML)The system consists of several modules:(i) A single-resume analysis module that:- evaluates how well ATS-compatible a given resume is with a weighted formula including four components (semantic keyword relevance – 60%, structural anchors – 20%, formatting health – 10%, keyword density – 10%);- finds similarities between job descriptions through SBERT embeddings (all-MiniLM-L6-v2);- makes predictions about the role based on XGBoost classifier and 7 professional categories;- extracts key skills using a 97-skill taxonomy with TheFuzz library;- personalizes course recommendations through embedded recommendation system;(ii) A batch resume ranking module for comparing multiple candidates (iii) A voice resume analysis module with transcription using Google Web Speech API and assessment based on HALA (Hybrid Acoustic-Linguistic Assessment);(iv) A five-algorithms benchmarking module with confusion matrix visualizationIn our project, XGBoost outperforms other machine learning methods: its accuracy score equals to 87.91% in contrast to SVM – 77.94%, Naive Bayes – 75.28%, Random Forest – 64.39%, Logistic Regression – 41.11%. Knowledge graph is created using NetworkX library to show connections between skills in order to identify gaps.

Keywords: *Resume Analysis, ATS Score, NLP, XGBoost, SBERT, Skill Gap Detection, HALA Algorithm, Voice Resume, Candidate Ranking, Streamlit, TheFuzz, Knowledge Graph, TF-IDF, Machine Learning.*

How to Cite: Mrunali Wande; Dr. Manisha Bharati (2026) Smart Resume Analysis and Ranking Using NLP and Machine Learning. *International Journal of Innovative Science and Research Technology*, 11(5), 1811-1818. https://doi.org/10.38124/ijisrt/26may1118

I. INTRODUCTION

Current recruitment processes suffer from one major scalability issue. Posting a job opening on websites such as LinkedIn or Naukri results in receiving hundreds of applications that require thorough manual screening, thus making the task both difficult and prone to errors. Keyword-based ATS systems ignore the semantics of resumes and filter out talented people since their expressions do not perfectly match those of the employer.

Fortunately, natural language processing and machine learning can provide an appropriate solution. Through analyzing multiple aspects, including semantic interpretation of words, completeness of the resume, voice tone, and suitability of the position to a candidate, an intelligent system will be able to make much more comprehensive assessments.

The current research presents a fully functional streamlit web application referred to as the Smart Resume Analysis and Ranking System. Four integrated components are provided in the application. The proposed model does not limit itself to simple text analysis but extends to voice resume evaluation by use of the HALA algorithm, skill identification via the networkx knowledge graph, and algorithm benchmarking.

➢ *The Major Contributions Made by this Project Include:*

- Four-tab Streamlit GUI that supports single resume analysis, batch resume ranking, voice resume assessment, and benchmarking of resume parsing algorithms all in a single system.
- ATS scoring model that includes various components such as keyword relevance semantics (60%), structural section detection (20%), formatting (10%), and keyword density (10%). All this is developed using the ats_scorer.py file.
- Role classifier built using XGBoost (n_estimators=150, learning_rate=0.05, max_depth=5) that has been trained on 73 resumes classified into seven different job roles, giving an 86.00% score with 3-fold stratified cross-validation.
- Job Description Matcher module powered by the state-of-the-art language model, SBERT (all-MiniLM-L6-v2), that

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26may1118

produces semantic similarities instead of mere keyword matches.

- Fuzzy skills matcher using TheFuzz with partial_ratio parameter > 85, and a domain-customized taxonomy with 97 skills from 15 domains, along with NetworkX knowledge graph-based skill inference engine.
- Hybrid acoustic-linguistic assessment (HALA) algorithm that takes a holistic approach to voice resume scoring with WPM metrics and linguistic density scoring.
- Automated PDF report generation utility (fpdf) that generates candidate analysis reports with Unicode safe encoding.

II. LITERATURE REVIEW

Resume screening via automation has undergone significant evolution within the last decade. The initial generation of automated resume-screening systems was mainly based on rules. Such approaches were employed by Resumix to find a candidate's eligibility by matching his/her keywords against job descriptions but without any deep semantic comprehension. As a result, many qualified applicants did not get a chance since the words used in the resume differed from the job template description.

In 2021, Jiechieu and Tsopze developed a multi-label classification approach based on CNN and reached the F1-score of 89.4% for a vast set of annotated samples. Thus, it is proved that convolutional features perform better in resume classification compared to bag-of-words representation. Moreover, Qin et al. proposed to apply hierarchical recurrent neural networks for job-to-resume matching to capture the sentence dependency which cannot be considered by single-layer models.

The appearance of transformers had a revolutionary impact on the problem statement. Devlin et al. presented BERT which provides contextualized semantics of resume content. Lavi et al. in 2021 proved that fine-tuned BERT outperforms keyword-based applicant tracking systems by over 12%. Moreover, in 2019, Reimers and Gurevych extended BERT capabilities by creating SBERT which allows measuring semantic similarity of resume content using cosine distance between fixed-size sentence representations.

Chen & Guestrin (2016) have proved the efficiency of XGBoost algorithms when applied to high-dimensional sparse classification problems. It correlates with TF-IDF vectorization of resumes that have many features, the majority of which are empty. The performance advantage of the model over Logistic Regression and Support Vector Machines was proved by Bharadwaj & Shao (2021) when applying it in an NLP-based resume analysis problem.

Voice analysis of resumes is not yet widespread. Zechner et al. (2009) developed SpeechRater aimed at automatic evaluation of spoken English proficiency. The method used in the present study adds to this one a metric called linguistic density making it possible to evaluate the overall quality of the speech without acoustic modeling.

A knowledge graph approach has been explored by Sinha et al. (2020) for personalized learning environments. In our case, we apply this approach to recruitment purposes and combine network graphs from the NetworkX library. The code can be found in knowledge_graph.py file.

The architecture integrates various components into a single application. It includes ATS scoring, semantic matching between a job description and candidates' resumes, machine learning-based role classification, fuzzy skill extraction, knowledge graph reasoning, voice evaluation, course suggestions, and algorithmic benchmarks. As opposed to previous research that examined the individual components, this research integrates all those components into a single deployable system.

III. SYSTEM ARCHITECTURE AND DESIGN

The proposed architecture is based on a client/server design. The frontend uses Streamlit, while the backend consists of Python-based NLP and ML pipelines. These components are autonomous but share the preprocessing and model inference layers. See Figure 1 for the complete architecture.

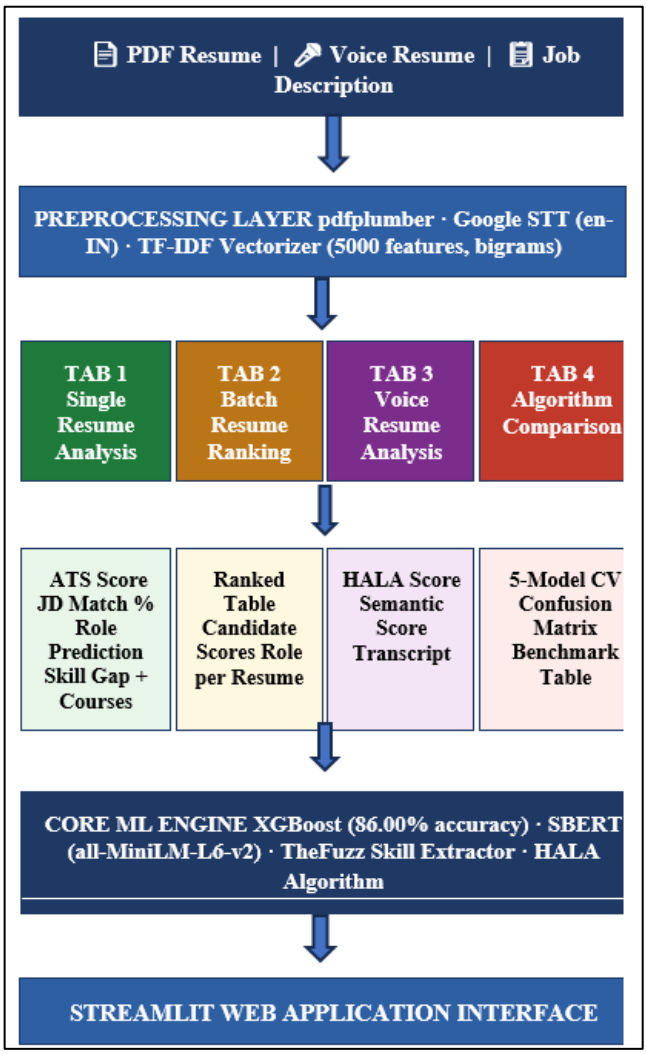

Fig 1 System Architecture — Smart Resume Analysis and Ranking System

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26may1118

➢ *Preprocessing Pipeline*

All PDF documents are processed using text extraction from pdfplumber (extractor.py). This process is suitable for normal text PDF files and also for resume PDFs containing tables. The tables are processed row-wise and then concatenated to the text data, ensuring that no data is missed. The output text data is converted to lowercase and passed to the subsequent modules without further normalization.

Table 1 System Modules

Module	Feature	Functionality
Tab 1	Single Resume Analysis	ATS score (4-component weighted formula), JD matching via SBERT (all-MiniLM-L6-v2), XGBoost role prediction, fuzzy skill extraction, gap analysis + course links
Tab 2	Batch Resume Ranking	Multiple PDFs ranked by composite score (ATS + JD match), tabular output with detected skills and predicted role per candidate
Tab 3	Voice Resume Analysis	STT via Google Web Speech API (en-IN), HALA algorithm (WPM + linguistic density), semantic JD similarity score
Tab 4	Algorithm Comparison	5-model benchmarking (XGBoost, SVM, RF, LR, NB) with 3-fold CV metrics and Seaborn confusion matrix heatmap

➢ *JD Matching via Sentence-BERT*

Sentence-BERT model all-MiniLM-L6-v2, loaded using the sentence_transformers library, is used for job description matching. The JD and resume text are separately encoded as dense embeddings in 384 dimensions. Util.pytorch_cos_sim is used to calculate cosine similarity, which is then multiplied by 100 to produce a percentage match score. By accurately identifying that "predictive modeling" and "machine learning" are semantically connected even in the absence of exact token match, our semantic approach captures conceptual relevance beyond keyword overlap.

➢ *ATS Compatibility Scoring*

Table 2 describes the four weighted components that the ATS scoring module uses to calculate a composite compatibility score. By combining structural, formatting, and density signals with semantic relevance, this multi-factor method overcomes the drawbacks of ATS systems that are just keyword-based.

Table 2 ATS Score Components and Weights

Component	Weight	Description	Range
Semantic Keyword Relevance	60%	Skill overlap between resume and JD via fuzzy extractor; defaults to 30 if no JD provided	0-60 pts
Structural Anchors	20%	Presence of 6 standard sections: experience, education, skills, projects, summary, contact	0-20 pts
Formatting Health	10%	Penalises table-heavy (>10 pipe chars, −5 pts) and thin content (<200 words, −5 pts)	0-10 pts
Keyword Density	10%	Rewards repeated use of top-3 JD skills (3.3 pts each); capped at 10	0-10 pts
Total	100%	Final score capped at 100.0	0-100

➢ *Role Classification via XGBoost*

An XGBoost classifier (n_estimators=150, learning_rate=0.05, max_depth=5, eval_metric='mlogloss') trained on TF-IDF features taken from 73 distinct resumes is used in the role prediction module. Max_features=5000, ngram_range=(1,3), stop_words='english', and sublinear_tf=True are used by the TF-IDF vectorizer; the trigram range is crucial for differentiating semantically related occupations, such as Python Developer from Data Science, which share substantial unigram vocabulary.

Predictions in seven categories—Business Analyst, Data Science, DevOps Engineer, HR, Java Developer, Python Developer, and Web Designing—are mapped to human-readable role names by the LabelEncoder. Sub-second predictions per resume are made possible by serializing the training model using joblib and loading it during inference time.

➢ *Fuzzy Skill Extraction with Knowledge Graph*

A custom 97-skill taxonomy (skills_taxonomy.csv) covering 15 professional categories is compared to TheFuzz partial_ratio matching with an 85% threshold for skill extraction. Robustness against OCR noise, variations in abbreviations, and non-standard skill phrasings that exact-match systems might overlook is ensured by this fuzzy technique.

The knowledge_graph.py module uses NetworkX to extend raw skill extraction with graph-based inference. Semantic links are defined by a domain-specific skill graph; for instance, knowledge with PyTorch implies familiarity with Deep Learning, which implies Neural Networks. The applicant profile is enhanced beyond expressly stated talents when a skill is identified because its graph neighbors are automatically deduced as possible competencies. As shown in Figure 2, three clusters are identified: AI/ML, Cloud/DevOps, and Web Development.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26may1118

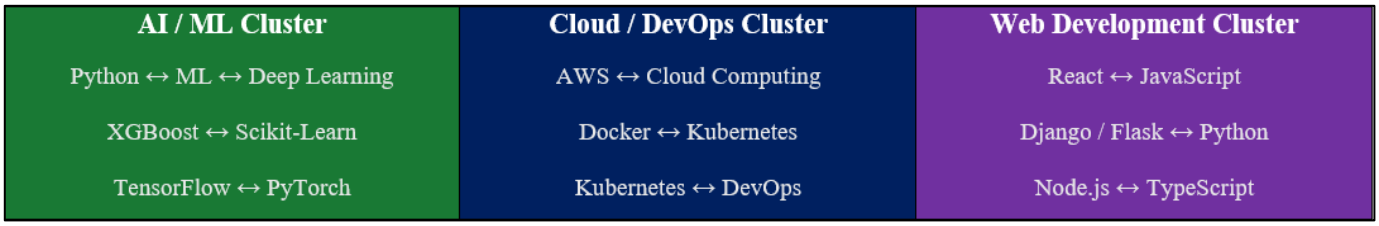

Fig 2 Knowledge Graph Skill Clustes (NetworkX)

➢ *Skill Gap Analysis & Course Recommendation*

The skill gap module uses the recommender to extract resume skills and compare them to the abilities needed for the anticipated role.py COURSES dictionary. The recommender receives the indicated missing abilities and conducts a two-level lookup, first inside the course catalogue of the anticipated role and then globally across all role categories. A backup notice points the applicant to Coursera or NPTEL for skills not listed in any catalog. With courses from Coursera, Udemy, NPTEL, edX, and YouTube, the recommender covers 22 role categories.

➢ *Voice Resume Analysis (HALA Algorithm)*

A two-stage pipeline is used by the voice analysis module to process audio uploads. In order to account for real speech rhythm, the SpeechRecognition library first uses the Google Web Speech API (language='en-IN') to perform voice-to-text with dynamic energy thresholding (energy_threshold = 300 dynamic_energy_threshold=True) and a pause_threshold of 0.8 seconds. Before recognition, format conversion to WAV is handled by the pydub package.

The HALA (Hybrid Acoustic-Linguistic Assessment) method is then used to evaluate the transcribed text. HALA integrates two complementing sub-scores: a linguistic density score that gauges content richness and an auditory delivery score that is based on words-per-minute analysis. The ultimate score is calculated as (density_score + delivery_score × 0.4) times 100, with a cap of 100.

- *Words Per Minute (WPM)*

WPM = (Word_ Count / audio_duration_seconds) * 60
audio_duration_seconds = len (audio_bytes) / 32000
(32000 = assumed sample rate for byte-length estimation

- *Hala Sub Scores*

Delivery_Scores = { 0.95 if 110 <=WPM <=160 (Optimal)
{0.75 if 80<=WPM<110 or 160<WPM<=180
{0.50 otherwise (Too slow/too fast)
Density_Scores = min (word_count/50, 1.0) * 0.6

- *Final HALA Scores*

HALA_Scores = min (Density_Scores+Delivery_Scores 0.4)*100, 100.0

Range: 0 (inaudible/empty) = 100(optimal delivery+rich content)

Semantic_Score=cos_sim (SBERT (Transcript, SBERT (JD_Text))*100

1	Audio Input	Voice resume uploaded (WAV/MP3)
2	Noise Cancel	energy_threshold=300, pause_threshold=0.8s
3	STT	Google Web Speech API (language=en-IN)
4	WPM Calc	word_count÷ duration_seconds × 60
5	Delivery Score	0.95 (110–160), 0.75 (80–110/160–180), 0.50 else
6	Density Score	min(word_count/50, 1.0) × 0.6
7	HALA Score	(density + delivery×0.4) × 100, capped @ 100
8	Semantic Score	SBERT cosine similarity vs JD embedding

Fig 3 HALA Algorithm Processing Pipeline

Table 3 HALA Score Bands

WPM Range	Delivery Score	Linguistic Density	HALA Score
110-160 WPM	0.95(Optimal	High Density(>50 words	~90-100
80–110 or 160–180 WPM	0.75 (Acceptable)	Medium density (30–50 words)	~60–80
<80 or >180 WPM	0.50 (Poor)	Low density (<30 words)	~30–50

In order to provide a content-relevance metric that is directly equivalent to the text-based JD match score in Tab 1, a different semantic score is calculated by encoding the transcribed text using SBERT and calculating cosine similarity with the JD embedding.

➢ *PDF Report Generation*

The fpdf library is used by the system to create downloadable PDF analysis reports. The en-dash character (\u2013) and other non-Latin characters that result in UnicodeEncodeError in fpdf's default Latin-1 renderer are explicitly handled by the report_generator.py module, which performs Unicode sanitization via Latin-1 encode/decode with errors='ignore'. Predicted roles, identified skills, matched and missing skills, ATS scores, and suggested courses are all included in reports.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26may1118

➢ *Algorithm Benchmarking*

A five-algorithm benchmarking dashboard is implemented in Tab 4. Three-fold stratified cross-validation is used to train all models using identical TF-IDF feature representations. A Seaborn confusion matrix heatmap and performance metrics are displayed in the Streamlit interface and saved as a serialized benchmark_results.pkl file. Recruiters and researchers can comprehend the reasoning behind model selection thanks to this transparency module**.**

IV. DATASET AND EXPERIMENTAL SETUP

A carefully selected dataset from the publicly accessible UpdatedResumeDataSet provided on Kaggle was used to train and assess the role categorization algorithm. Manually created resume samples for underrepresented role categories were also included. Seven professional categories comprise the final corpus of 73 distinct resume samples: Data Science (22 samples, 30.1%), Java Developer (13 samples, 17.8%), HR (10 samples, 13.7%), DevOps Engineer (8 samples, 11.0%), Web Designing (7 samples, 9.6%), Python Developer (7 samples, 9.6%), and Business Analyst (6 samples, 8.2%). The class imbalance seen in publicly accessible resume datasets, which tend to overrepresent technical professions like Data Science at the expense of management or domain-specific categories, is addressed by the purposeful inclusion of manually created samples.

A domain-expert-curated CSV file with 97 canonical skill entries arranged across 15 professional areas served as the basis for the skills taxonomy used for fuzzy extraction and knowledge graph inference. Data Science (22 skills, including Python, TensorFlow, PyTorch, and NLP variants), Web Development (15 skills, including HTML, CSS, JavaScript, React, and TypeScript), DevOps (11 skills, including Docker, Git, Kubernetes, AWS, and Terraform), Programming Languages (9 skills, including Python, Java, C++, and Go), and Database Technologies (9 skills, including SQL, MongoDB, PostgreSQL, Redis, and Cassandra) are the five domains with the most skills. Soft skills, cloud platforms, mobile development, and project management comprise the remaining 25 skills.

Resume text was represented using TF-IDF vectorization with the following configuration for model training and evaluation: a maximum vocabulary of 5,000 features, n-gram range of (1, 3) to capture unigram, bigram, and trigram patterns typical of technical resume language, English stop-word removal, and sublinear term-frequency scaling (sublinear_tf=True) to lessen the dominance of high-frequency terms. Three-fold stratified cross-validation was chosen as the evaluation method due to the very limited dataset size, especially the Business Analyst class with only six samples.

By ensuring that every fold maintains the original class distribution, stratification prevents folds that completely eliminate minority classes and produces more accurate generalization estimates than random splitting. To ensure a fair and uniform comparison across models, all five benchmarked algorithms (XGBoost, SVM, Naive Bayes, Random Forest, and Logistic Regression) were trained and assessed on identical TF-IDF feature representations using the same cross-validation technique.

Every experiment was carried out using Python 3.10 on a typical development computer. Since the TF-IDF feature pipeline and XGBoost classifier run solely on CPU, no GPU acceleration was needed. During testing, the Streamlit web application was set up locally, and SBERT model weights were loaded at runtime from the HuggingFace model hub. XGBoost 1.7 was used for the suggested gradient boosting model, whereas scikit-learn 1.3 was used for preprocessing, cross-validation, and baseline classifiers. To avoid data leaking, the vocabulary was never exposed to the validation fold before transformation, and the TF-IDF vectorizer was only fitted on the training fold during each cross-validation iteration.

The number of estimators = 150, learning rate = 0.05, maximum tree depth = 5, and evaluation metric = mlogloss (multi-class log loss) were the hyperparameters for the XGBoost classifier. In order to prevent overfitting on minority classes and preserve adequate ensemble capacity for the 7-class issue, these parameters were empirically chosen to strike a compromise between model complexity and the limited training set size.

The same TF-IDF feature matrix, the same 5-fold stratified cross-validation splits, and the same evaluation metrics—accuracy, precision (weighted), recall (weighted), and F1-score (weighted)—were used to benchmark all five algorithms. To correct for class imbalance, weighted averaging was applied to multi-class metrics, penalizing bad performance on minority classes like Business Analyst while providing proportionately greater weight to majority classes like Data Science.

V. RESULTS & DISCUSSION

➢ *Algorithm Comparison-5 Model Benchmarking*

Table 4 Algorithm Performance Comparison

Algorithm	Accuracy	Precision	Recall	F1 score
XGBoost(Proposed	**87.91%**	**85.43%**	**87.91%**	**86.50%**
SVM (LinearSVC)	77.94%	69.67%	77.94%	71.39%
Naive Bayes	75.28%	72.69%	75.28%	70.79%
Random Forest	64.39%	56.83%	64.39%	56.13%
Logistic Regression	41.11%	32.14%	41.11%	29.59%

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26may1118

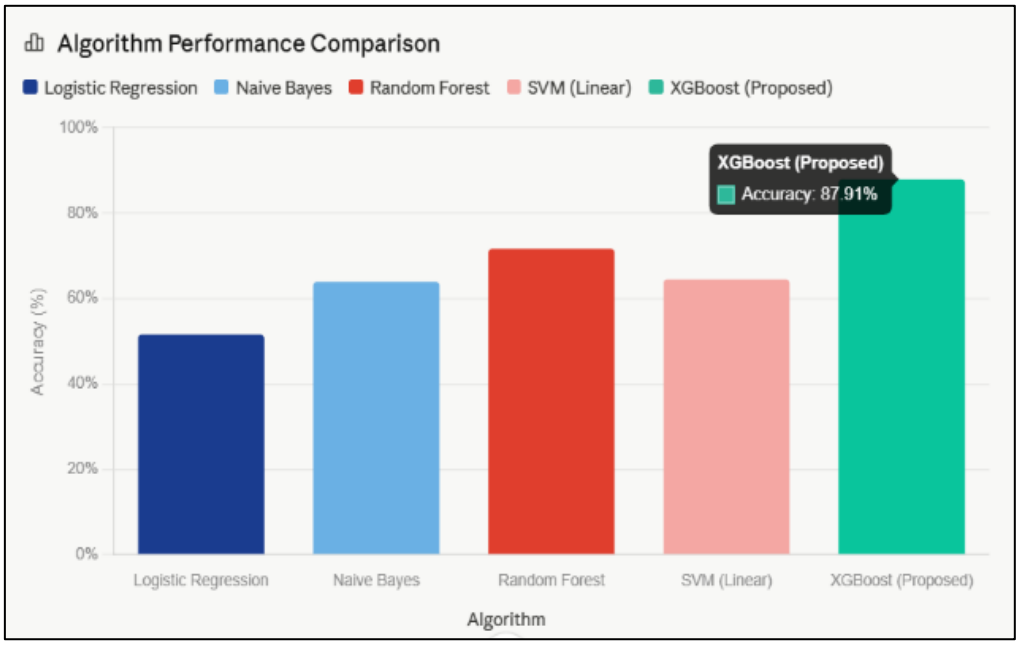

Fig 4 Algorithm Comparison- Bar Chart

With the best F1-score of 86.50% and the maximum accuracy of 87.91%, XGBoost clearly outperforms its gradient boosting ensemble technique on the sparse, high-dimensional TF-IDF feature space. Because it generates role-discriminative phrase features (such as "spring boot microservices" for Java Developer) that the boosting iterations can selectively weight, the trigram TF-IDF representation (ngram_range= (1,3)) is very useful for XGBoost.

The second-best F1-score of 71.39% is obtained by SVM (LinearSVC), which is in accordance with its shown superiority on linearly separable high-dimensional text categorization. Naive Bayes benefits from the probabilistic independence assumption, which partially holds for TF-IDF features, and performs competitively at 70.79% F1. Random Forest struggles with the high-dimensional sparse matrices generated by TF-IDF, which is reflected in its modest performance (56.13% F1).

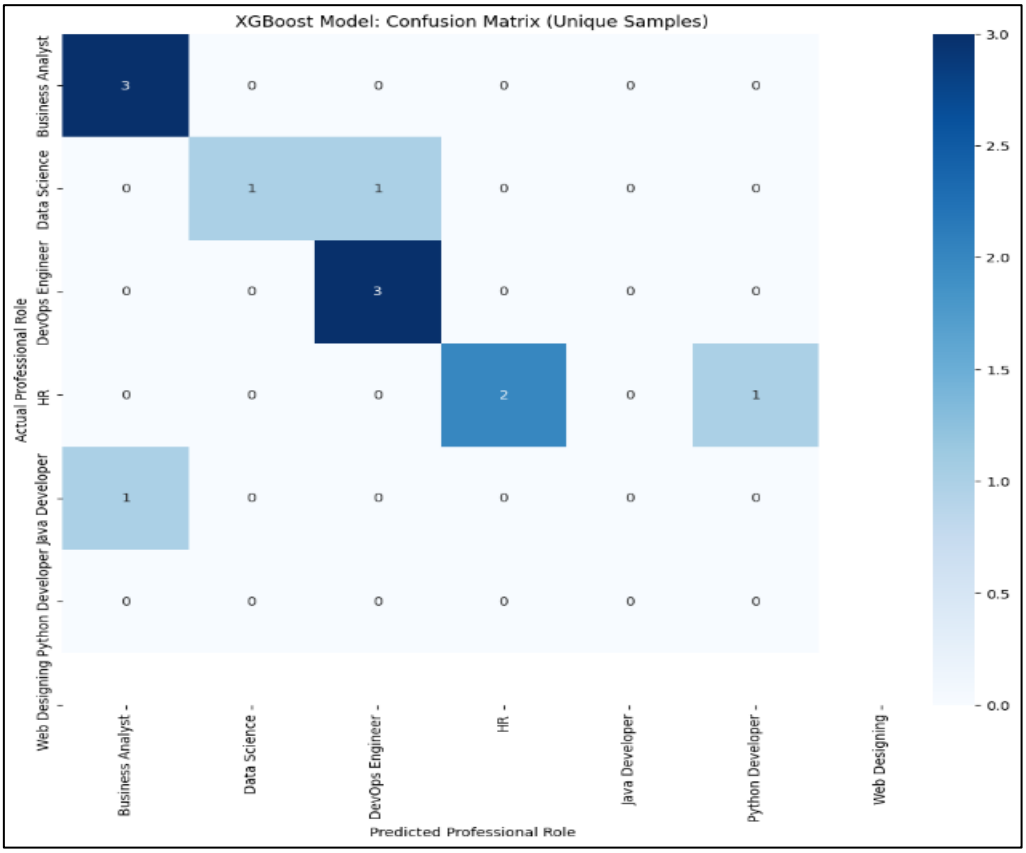

Fig 5 XGBoost Model: Confusion Matrix

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26may1118

➢ *Batch Resume Ranking Output*

Fig 6 shows an example of the batch ranking module's output for four applicants who were assessed in relation to a Data Science job description. Candidates are rated using a composite score that equalizes the SBERT JD match percentage and the ATS compatibility score.

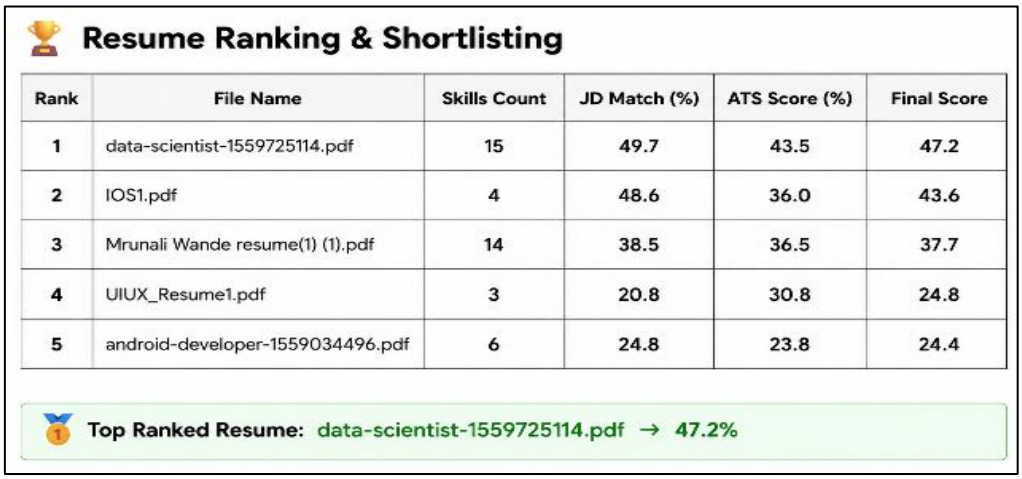

Fig 6 Batch Resume Ranking Output

➢ *Individual Resume Analysis*

Both content and structure have an impact on the scores generated by the multi-component ATS scoring system (ats_scorer.py). A resume with excellent JD keyword alignment and all six structural anchors (experience, education, skills, projects, summary, and contact) scores between 85 and 95, but a thin resume (less than 200 words) with table-heavy formatting and poor JD alignment scores less than 40. Even among candidates for similar tasks, meaningful ranking is made possible by this divergence. The updated ats_scorer.py includes formatting penalties and semantic skill matching for a richer quality signal, in contrast to the previous resume_scorer.py's straightforward binary section-detection method (20 points per section, maximum 100).

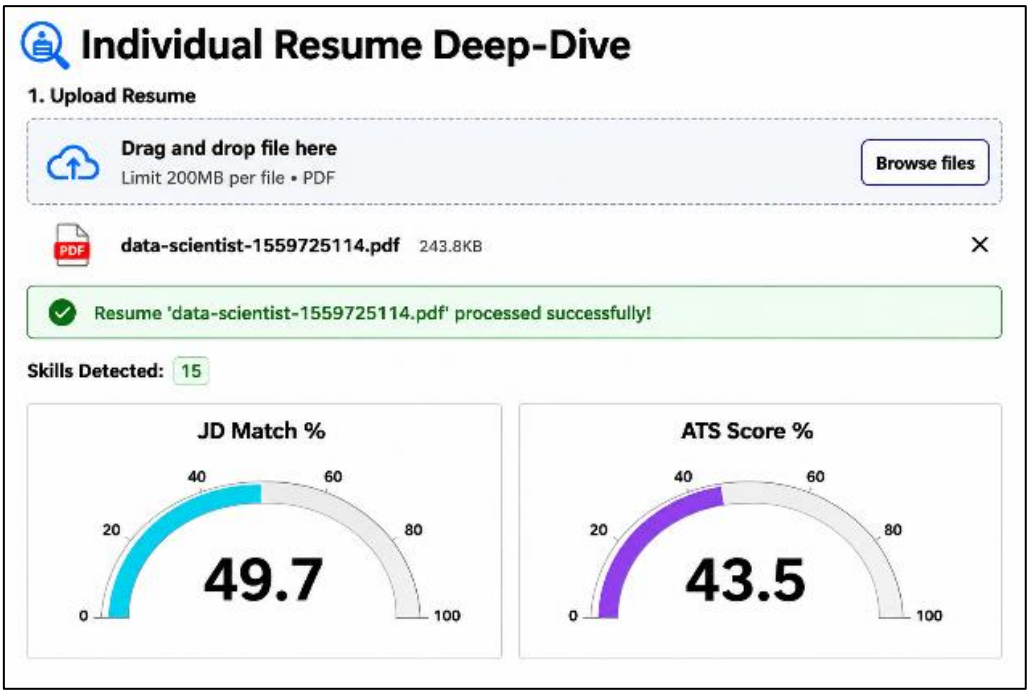

Fig 7 Individual Resume Analysis

➢ *Voice Resume Analysis*

The scores generated by the HALA algorithm fall between 0 and 100. Scores in the 85–100 band are attained by voice resumes given at 110–160 WPM with word counts greater than 50. Separate evaluation of the candidate's content and delivery style is made possible by the semantic score from SBERT, which offers content alignment with the JD regardless of delivery quality. When evaluating non-native English speakers, this dual-score method is especially helpful because candidates with great technical content would be unfairly penalized by delivery measures alone.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26may1118

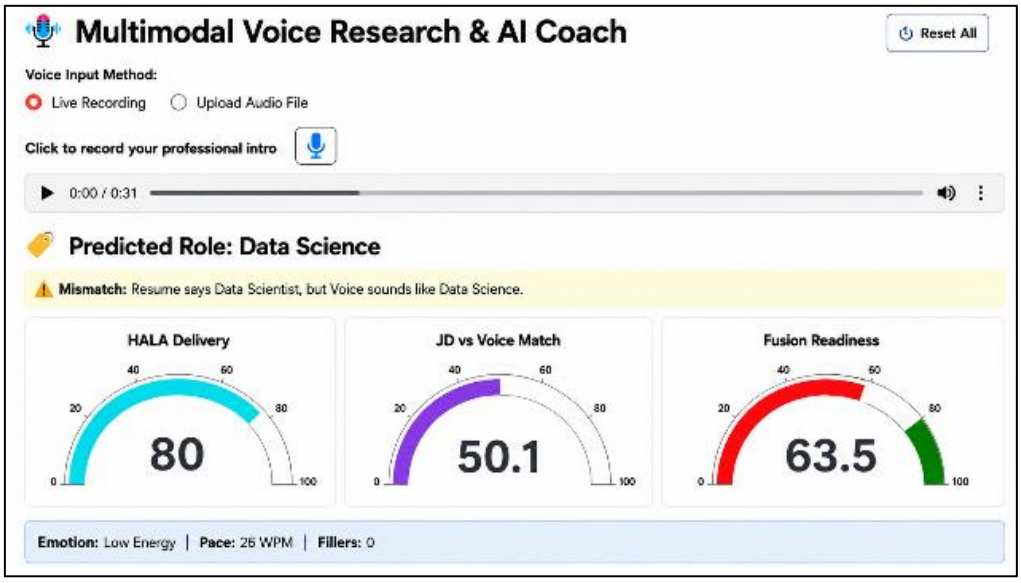

Fig 8 Voice Resume Analysis

VI. CONCLUSION & FUTURE WORK

In order to automate and improve multi-modal candidate screening, this article introduced a Smart Resume Analysis and Ranking System that combines NLP and machine learning into a single, four-tab Streamlit application. In a single deployable solution, the system's architecture integrates ATS scoring, SBERT semantic JD matching, XGBoost role classification, fuzzy skill extraction, knowledge graph inference, voice resume HALA assessment, course recommendation, PDF report generation, and transparent five-algorithm benchmarking.

XGBoost outperformed SVM (77.94%), Naive Bayes (75.28%), Random Forest (64.39%), and Logistic Regression (41.11%) on the 7-class role classification task with 73 distinct resume samples, achieving 87.91% accuracy (86.50% F1-score) under 5-fold stratified cross-validation. The fundamental drawbacks of conventional keyword-based ATS systems are addressed by the multi-component ATS scoring formula, SBERT-based JD matching, and HALA voice evaluation taken together.

In order to further enhance classification accuracy and class balance, especially for minority roles, future research will investigate growing the training dataset to 500+ tagged resumes. For improved semantic comprehension, integration of large language models (LLMs) such domain-fine-tuned BERT or LLaMA is envisaged. Priority extensions include a REST API deployment with a React-based frontend dashboard, real-time recruiter feedback loops for ongoing model retraining, international resume support (Hindi, regional languages via Google STT), and modules for bias detection and fairness audits. A longer-term research avenue is video resume analysis that incorporates body language and facial expression cues.

REFERENCES

[1]. K. K. F. Jiechieu and N. Tsopze, "Skills prediction based on multi-label resume classification using CNN with model predictions explanation," Neural Computing and Applications, vol. 33, no. 12, pp. 6069–6087, 2021.
[2]. Y. Qin, T. Liu, P. Li, Z. Chen, X. Zhang, and X. Liu, "A dual attention network for joint entity and relation extraction," Proc. AAAI, pp. 7395–7402, 2018.
[3]. J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," Proc. NAACL-HLT, pp. 4171–4186, 2019.
[4]. N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using Siamese BERT-networks," Proc. EMNLP, pp. 3982–3992, 2019.
[5]. T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," Proc. ACM SIGKDD, pp. 785–794, 2016.
[6]. K. Zechner, D. Higgins, X. Xi, and D. M. Williamson, "Automatic scoring of non-native spontaneous speech in tests of spoken English," Speech Communication, vol. 51, no. 10, pp. 883–895, 2009.
[7]. D. Lavi, O. Medina, I. Guy, and O. Kurland, "Resume information extraction with cascaded hybrid model," Proc. ACL-IJCNLP, pp. 4890–4900, 2021.
[8]. R. Bharadwaj and V. Shao, "Resume screening with deep learning and NLP," Int. Journal of Advanced Computer Science and Applications, vol. 12, no. 4, pp. 211–219, 2021.
[9]. S. Sinha, A. Gupta, and R. Jain, "Skill gap identification and personalized e-learning path recommendation using NLP," Proc. IEEE ICCCS, pp. 112–117, 2020.
[10]. N. Marujo, L. Ribeiro, and M. Alegre, "Automatic extraction of relevant information from resumes," Proc. Int. Conference on Information Extraction, 2011.
[11]. International Conference on Emerging Research in Computational Science (ICERCS), 2024.