WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal Homepage: www.ijrpr.com ISSN 2582-7421

AI-Powered Fake Resume Analyser and Candidate Credibility Detection System Using Machine Learning

Hemalatha M A[1], Dharshini G[2] , Archana Devi M[3], Deeksha P[4], Dharani R S[5]

[1] Assistant Professor, Department of AIDS, Sri Shakthi Institute of Engineering and Technology, Coimbatore, Tamil Nadu, India.
E-mail: hemalathaaids@siet.ac.in
[2] Student, Department of AIDS, Sri Shakthi Institute of Engineering and Technology, Coimbatore, Tamil Nadu, India .
E-mail: dharshinig25ad@srishakthi.ac.in
[3] Student, Department of AIDS, Sri Shakthi Institute of Engineering and Technology, Coimbatore, Tamil Nadu, India
E-mail: archanadevim25ad@srishakthi.ac.in
[4] Student, Department of AIDS, Sri Shakthi Institute of Engineering and Technology, Coimbatore, Tamil Nadu, India.
E-mail:deekshap25ad@srishakthi.ac.in
[5] Student, Department of AIDS, Sri Shakthi Institute of Engineering and Technology, Coimbatore, Tamil Nadu, India.
E-mail: dharanirs25ad@srishakthi.ac.in

ABSTRACT

Recruiters routinely receive hundreds of digital resumes for a single vacancy, and verifying every claim manually is slow, inconsistent and prone to oversight. Fabricated qualifications, overlapping employment periods, inflated experience and vague or templated descriptions can therefore pass the first screening stage unnoticed. This paper presents the design of an AI-powered, web-based Fake Resume Analyser and Candidate Credibility Detection System that supports recruiters in finding potentially misleading or inconsistent information in candidate resumes. The system accepts resumes in PDF and DOCX formats, extracts and preprocesses the text, and applies Natural Language Processing (NLP), rule-based consistency checks and Machine Learning techniques, with the Scikit-learn library as the modelling framework, to detect timeline conflicts, suspicious employment and education details and other anomalies. The findings are combined into an explainable candidate risk score that is presented on a recruiter dashboard together with the specific indicators that raised it, and a job-description matcher and an audit history support everyday screening work. The system is intentionally a decision-support tool: it flags items for human verification and never decides that a candidate is fraudulent. The expected benefits are reduced screening effort, structured and reproducible analysis and a clearer starting point for verification. The limitations include dependence on document quality, the possibility of bias in trained models and the fact that automated analysis cannot prove fraud. Quantitative evaluation results are not reported in this manuscript and are left as clearly marked placeholders to be completed after testing. Future work includes transformer-based language models, larger verified datasets, improved explainability, multi-language support and secure cloud deployment.

Keywords: *Artificial Intelligence, Fake Resume Detection, Machine Learning, Natural Language Processing, Resume Analysis, Candidate Credibility, Anomaly Detection, Recruitment Automation.*

1. INTRODUCTION

Recruitment is one of the most consequential activities of any organisation, because the people who are hired determine the quality of its work, its culture and its long-term costs. The first practical step of almost every hiring process is resume screening, in which a recruiter reads a candidate's document and decides whether the person deserves further attention. With online job portals, professional networking sites and one-click applications, resumes are now created, edited and submitted entirely in digital form, and a single advertisement can attract hundreds or thousands of applicants.

The same convenience that makes digital applications easy also makes embellishment easy. Studies of resume deception report that a meaningful share of applicants admit to, or are found to have, inflated their responsibilities, shifted employment dates to hide gaps, claimed qualifications they did not complete or listed skills they do not possess [1, 2]. Such claims are rarely obvious. A degree year that is slightly inconsistent with the first job, two full-time positions that overlap for several months, or a long list of advanced skills attached to a very short career are signals that are easy to miss when a recruiter has only a minute or two for each document.

Manual verification does not scale. Checking every claim by calling institutions and employers is expensive and is normally reserved for shortlisted candidates, so the initial reading of the resume is the only cheap opportunity to notice irregularities. At this stage, however, human attention is limited, fatigue produces inconsistency between reviewers, and the process is difficult to document or audit afterwards.

Artificial Intelligence (AI) and Machine Learning (ML) have already been applied to several stages of recruitment, including resume parsing, candidate-job matching and ranking [3, 4, 5]. Natural Language Processing (NLP) converts unstructured resume text into structured fields, and statistical learning and anomaly-detection methods can highlight patterns that deviate from what is normal [6]. These technologies can also introduce risks of bias and over-reliance, which is why their use in hiring is now the subject of regulation and research on fairness and explainability [7, 8].

The motivation for this project is to give recruiters a practical assistant for the very first reading of a resume. The proposed AI-powered Fake Resume Analyser and Candidate Credibility Detection System extracts the content of an uploaded resume, checks it for inconsistencies and suspicious patterns, and presents a transparent risk score with the reasons behind it. It is deliberately designed as a recruiter-support tool: it points to what should be verified and leaves every hiring decision to a human.

The scope of the work is limited to the analysis of the textual content of a single resume at a time. The system does not contact employers or institutions, does not scrape social networks and does not make employment decisions. The main contributions of the paper are: (i) a modular design that combines rule-based consistency checks with ML-based anomaly scoring; (ii) an explainable risk score in which every contributing factor is visible to the recruiter; (iii) a methodology and evaluation plan that avoids unsupported claims; and (iv) a discussion of the ethical and fairness issues that arise when such a tool is used in hiring.

The remainder of this paper is organised as follows. Sections 2 and 3 state the problem and objectives, Section 4 reviews related work, Sections 5 to 9 describe the proposed system, technologies, ML methodology, architecture and modules, Section 10 reports the implementation and the evaluation plan, and Sections 11 to 14 discuss advantages, limitations, future enhancements and conclusions.

2. PROBLEM STATEMENT

Recruiters face difficulties in identifying misleading or inconsistent information in resumes. Manual screening is time-consuming and may fail to identify suspicious details, especially when the number of applications is large. Therefore, an automated system is required to analyse resume content, identify potential irregularities and assist recruiters in evaluating candidate credibility, while keeping the final judgement with the recruiter.

3. OBJECTIVES

The objectives of the project are as follows.

1. To develop an AI-powered resume analysis web application.
2. To extract text from PDF and DOCX resumes.
3. To identify inconsistencies in candidate information.
4. To detect suspicious employment and education details.
5. To generate candidate risk scores.
6. To provide a recruiter-friendly dashboard.
7. To support comparison of a resume with a job description.
8. To maintain analysis history and audit information.
9. To improve screening efficiency while maintaining human oversight.

4. RELATED WORK

4.1 Resume Deception

Research in organisational psychology treats resume fraud as a measurable behaviour. Henle et al. developed and validated a measure of intentional resume deception and related it to personality and attitudinal factors [1], and Guillory and Hancock examined how the visibility of a profile to other people affects the honesty of the claims made in it [2]. These works establish that embellishment is common enough to justify systematic screening, but they study the behaviour of applicants rather than automated detection.

4.2 Resume Parsing and Screening

Early resume information extraction used cascaded hybrid models that first segmented a resume into blocks and then extracted fields such as name, education and employment [3]. More recent work uses pre-trained transformers such as BERT [9] to parse resumes and match candidates against job descriptions [4]. Black and van Esch discuss how AI-enabled recruiting tools are used and what managers should consider when relying on them [5]. Most of this literature targets parsing accuracy and relevance ranking, not the credibility of the content.

4.3 NLP and Text Representation

Classical text representation based on term weighting [10] remains a strong and interpretable baseline for document classification, while transformer encoders [9] provide contextual representations at a higher computational cost. Practical toolkits [11] and textbooks [12] describe the tokenisation, normalisation and feature-extraction steps used in this project.

4.4 Anomaly Detection

Anomaly detection identifies observations that deviate from expected behaviour; Chandola et al. classify its techniques and application domains [6], and Pang et al. review deep-learning approaches [13]. Isolation Forest isolates unusual points through random partitioning and is efficient for tabular features [14], and comparative studies show that no single unsupervised detector is best on all data [15]. Because verified fake resumes are scarce, unsupervised or semi-supervised techniques combined with explicit rules are attractive for this problem.

4.5 Fairness and Explainability

Hiring is a high-stakes domain. Raghavan et al. analyse claims and practices of vendors of algorithmic hiring tools and warn that bias mitigation is often poorly documented [7]; Köchling and Wehner review discrimination and fairness in algorithmic HR decisions [16]; and Mehrabi et al. survey sources of bias in machine learning [17]. Model-agnostic explanation methods such as LIME [18] and SHAP [19] help users understand individual predictions, and the need for rigorous evaluation of interpretability is discussed in [20].

4.6 Research Gap

Existing work concentrates on parsing and matching. A recruiter-facing tool that combines transparent consistency checks, anomaly-based indicators and a documented human-oversight workflow for resume credibility is less commonly reported, and this project addresses that gap at undergraduate-project scale.

4.7 Positioning of the Present Work

In contrast to parsers that aim to recover fields accurately [3, 4], the present work treats the extracted fields as evidence to be cross-checked. In contrast to fully automated screening products criticised for opacity [7], it exposes the reason for every flag and keeps the recruiter in control. In contrast to purely statistical detectors [6], it uses explicit domain rules for which labelled examples are not needed, and employs learned models only where data are available. This hybrid, human-centred positioning is appropriate for a final-year project because it is implementable with open-source tools and can be evaluated honestly with limited data.

5. PROPOSED SYSTEM

The proposed system is a web application used by a recruiter or HR professional. Its workflow, shown in Fig. 2 together with the learning pipeline, has nine steps.

1. The recruiter opens the web application in a browser.
2. The recruiter uploads a candidate resume in PDF or DOCX format.
3. The backend validates the file and extracts its text.
4. The extracted content is cleaned and preprocessed.
5. NLP and ML techniques analyse the resume information.
6. The system checks for inconsistencies, suspicious patterns and possible anomalies.
7. A risk score is generated from the implemented detection rules and models.
8. The results are shown on the recruiter dashboard together with the reasons for each flag.
9. The recruiter reviews the findings and makes the final decision.

The system does not label a candidate as fake. A high score only indicates that more items deserve verification, and every flag carries a human-readable explanation so that the recruiter can judge it.

5.1 Design Principles

Four principles guide the design. Transparency: every flag is accompanied by the evidence from the resume that triggered it. Human oversight: the interface uses cautious wording such as “verify” and “clarify” rather than “fake”. Privacy by design: only the data needed for analysis are processed and retention is limited. Modularity: extraction, analysis, scoring and presentation are separate components, so that any of them can be replaced, for example a better PDF extractor or a stronger model, without changing the rest.

5.2 Illustrative Walkthrough

Consider a synthetic resume in which a candidate lists a bachelor’s degree ending in one year but a full-time job starting two years earlier, and two jobs that overlap by six months. After upload, the system extracts the education and experience entries, parses the dates and runs the checks. It raises an education-timeline flag and an employment-overlap flag, each showing the conflicting dates, and computes a score that falls into a higher band. The recruiter reads the two explanations and may decide to ask the candidate about part-time study or concurrent contracts. The system has therefore directed attention without drawing a conclusion. No numerical result is implied by this example.

6. TECHNOLOGIES USED

Table 1 lists the technologies and their purpose. In line with academic honesty, only libraries present in the source code should be retained in the final submission; items that are not confirmed are highlighted.

Table 1: Technologies and their purpose

Technology	**Purpose in the project**	**Status**
Python	Main programming language for backend logic, text processing and ML	Python 3.10
Flask	Lightweight backend web framework: routing, file upload handling and API responses	Flask 3.0
HTML, CSS, JavaScript	Frontend: upload form, dashboard layout, score visualisation	HTML5,CSS3, ES6 JavaScript
Scikit-learn	ML algorithms, feature extraction, model selection and evaluation utilities	Scikit-learn 1.3
NLP techniques	Tokenisation, normalisation, pattern and date extraction, similarity	spaCy /NLTK/Regex
PDF extraction library	Extracts text from PDF files	PyMuPDF
python-docx	Reads paragraphs and tables from DOCX resumes	Python-docx 1.1
NumPy	Numerical operations on feature vectors and scores	NumPy 1.26
Joblib	Saves and loads trained ML models and vectorisers	Joblib 1.3
ReportLab	Generates PDF analysis reports	ReportLab 4.0
Pytest	Automated unit and integration testing	Pytest 8.0
GitHub	Version control and collaboration	Github
Vercel or other platform	Deployment of the web application	Vercel

Python is used throughout because of its mature text-processing and ML ecosystem. Flask [21] receives the uploaded file, calls the extraction and analysis functions and returns the results to the browser; its minimal design suits a project of this size. The frontend uses HTML for structure, CSS for styling and JavaScript for interactions such as file selection and rendering the risk indicator. Text is extracted with a PDF library and with python-docx, and numerical operations use NumPy. Trained models are serialised with Joblib so that they are loaded once instead of being retrained for every request.

Deployment on a serverless platform such as Vercel is attractive for demonstrations, but package-size and execution-time limits can make large ML dependencies difficult to host; a container-based or conventional server host may be required, depending on backend compatibility vercel.

NLP techniques are used for tokenisation, normalisation and the extraction of dates, organisations and qualifications from free text [11, 12]; regular expressions complement them for structured patterns such as date ranges. Pytest provides a uniform way of writing small, repeatable tests that can be run automatically before every commit, and GitHub supports version history, issue tracking and collaboration between the five team members. Dependencies should be pinned in a requirements file so that the project can be reproduced on another machine.

7. MACHINE LEARNING METHODOLOGY

This section describes the intended methodology. Items that depend on the actual implementation or on experimental data are marked as placeholders; no dataset, algorithm or performance figure is claimed here without confirmation.

7.1 Data Collection and Preparation

Verified labelled data on fake resumes is difficult to obtain because of privacy and ground-truth problems. The training and testing data for the project therefore consist of [describe dataset: source, size, licence, labelling method]. If synthetic or publicly available resumes are used, they must be anonymised, personal identifiers must be removed and the labelling procedure must be documented. The dataset is cleaned by removing duplicates and unreadable files.

7.2 Text Preprocessing

Extracted text is normalised by converting it to a consistent case, removing control characters and page artefacts, standardising date formats and splitting the document into sections such as education, experience and skills. Tokenisation, stop-word handling and optional lemmatisation [11, 12] prepare the text for feature extraction.

7.3 Feature Extraction

Two groups of features are considered. Textual features use term-frequency inverse-document-frequency (TF-IDF) weighting [10] to represent descriptions. Structured features are derived from the content, for example the number of employment entries, total claimed experience, overlap in months between roles, gap length, time between graduation and first job, skills-per-year-of-experience ratio, proportion of vague statements and similarity between sections. The final feature set is [confirm implemented features].

7.4 Training and Testing

The labelled data are divided into training and testing subsets, for example 80% and 20% with stratification , and cross-validation is used for model selection so that the test set remains untouched until the final evaluation.

7.5 Algorithm Selection

Candidate algorithms are Logistic Regression and Support Vector Machines [22] for supervised text classification, Random Forest [23] for mixed structured features, and Isolation Forest [14] for unsupervised anomaly scoring. The algorithm finally used is; the others remain proposed alternatives.

7.6 Scikit-learn in Brief

Scikit-learn [24] is an open-source Python library that provides a consistent interface to many ML methods. Every model follows the same pattern: it is created, trained on examples with a fit step and applied to new data with a predict step, while transformers such as the TF-IDF vectoriser follow a fit-transform pattern. Pipelines chain preprocessing and a model into one object so that exactly the same steps are applied during training and prediction, and utilities for data splitting, cross-validation and metrics support fair evaluation. This uniformity allows algorithms to be compared with little code change.

7.7 Evaluation, Prediction and Risk Scoring

Models are evaluated using accuracy, precision, recall, F1-score and a confusion matrix; for imbalanced data, recall and precision on the suspicious class are more informative than accuracy. The results will be reported in Table 4 after experiments. At prediction time the model output and the rule-based indicators are combined into a bounded risk score, $R = \Sigma(w_i \cdot s_i)$, where s_i is the normalised severity of indicator i and w_i is its weight, with R scaled to the range 0–100 . The score is mapped to the bands Low, Medium and High

7.8 Rule Engine and Score Design

Each rule produces an indicator with a severity level (low, medium or high) and a short explanation. Severities are mapped to numeric values and multiplied by weights that reflect how strongly the indicator usually relates to a genuine problem; for example, an end date earlier than a start date is

more serious than a missing employer name. To avoid double counting, indicators that describe the same underlying fact are merged, and each category contributes at most a fixed share of the total. The final value is clipped to the range 0–100. The weights and thresholds are initially set from domain reasoning and should later be calibrated on a validation set so that the bands correspond to a meaningful review workload. Low means that no significant irregularity was detected, Medium means that a few items deserve a quick check, and High means that several independent indicators were found; none of the bands means that a resume is false.

7.9 Evaluation Protocol

For the suspicious class treated as positive, precision is TP/(TP+FP), recall is TP/(TP+FN) and F1 is their harmonic mean, where TP, FP and FN denote true positives, false positives and false negatives. In this application the two error types have different costs: a false positive wastes a recruiter's time and may unfairly cast doubt on an honest candidate, while a false negative lets an irregularity pass unnoticed. The decision threshold should therefore be chosen explicitly on validation data and not left at a default. Stratified k-fold cross-validation reduces the variance of estimates when the number of suspicious samples is small, and metrics should additionally be reported per subgroup, such as resume format or industry, to reveal uneven performance [17]. Model-agnostic explanation tools [18, 19] can be used to check that predictions rely on meaningful features and not on artefacts of the dataset.

7.10 Limitations and Bias Considerations

Models trained on historical data may reproduce historical bias. Employment gaps may reflect study, illness or caregiving; non-traditional career paths or resume formats may be penalised unfairly; and names, gender, age, photographs and institution prestige must not be used as features [17, 7]. Feature importance and subgroup error rates should be examined, and flags are presented as questions for the recruiter and not as accusations.

8. SYSTEM ARCHITECTURE

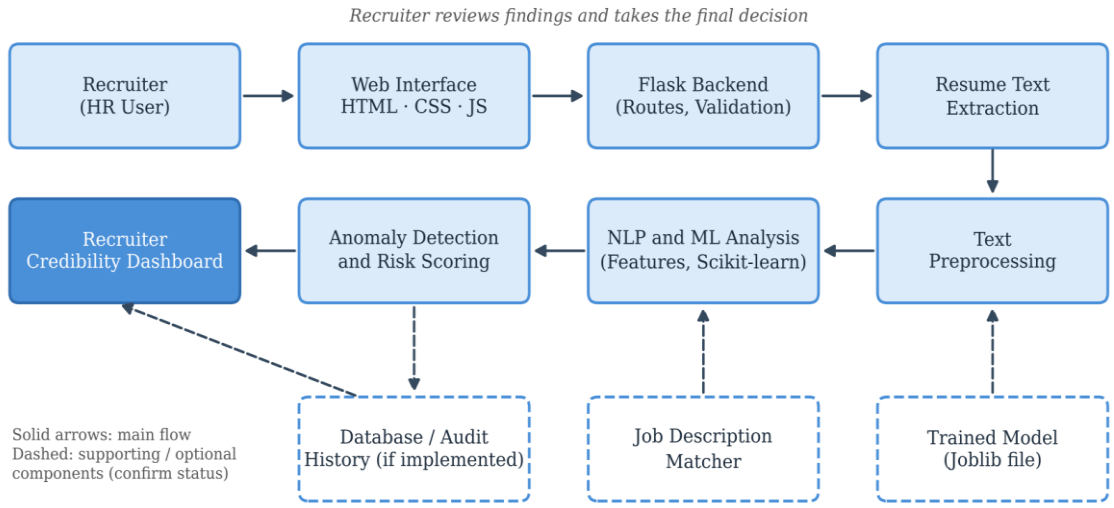


Fig 1: System architecture of the proposed system

Fig. 1 shows the architecture. Each component is explained below.

Recruiter and Web Interface

The recruiter is the only user of the analysis. The web interface, built with HTML, CSS and JavaScript, provides the upload form and the dashboard, and sends the file to the backend.

Flask Backend

The backend validates the file type and size, coordinates the processing pipeline, calls the analysis functions and returns structured results as a web page or JSON response.

Resume Text Extraction

This component reads PDF and DOCX files and returns plain text with basic structure. Scanned images require optical character recognition and are not covered unless implemented

Text Preprocessing

The text is cleaned, normalised and segmented into sections so that later stages work on consistent input.

NLP and ML Analysis

Entities and dates are extracted, features are computed and the trained Scikit-learn model, loaded through Joblib, produces predictions or anomaly scores.

Anomaly Detection and Risk Scoring

Rule-based checks and model outputs are combined to flag inconsistencies and to compute the overall risk score with its contributing factors.

Database / Audit History

If implemented, each analysis (file reference, time, score and flags) is stored so that decisions can be reviewed later

Recruiter Dashboard

The dashboard displays the score, the band, the flagged items with explanations, and the job-description match, leaving the final decision to the recruiter.

Request Lifecycle

A request begins when the browser posts the file to the upload route. The backend checks the extension, size and content type, and stores the file temporarily. The extraction component returns text, the preprocessing component returns normalised sections, and the analysis component returns features and indicators. The scoring component aggregates them into a result object that is rendered by the dashboard. If the audit history is enabled, the result is saved before the response is returned, and the temporary file is deleted according to the retention policy . Errors at any stage, such as an unreadable file, are reported to the recruiter as a clear message instead of a misleading score.

Deployment View

The application runs as a single web service in which the model files are loaded at start-up. This keeps the architecture simple for a prototype; larger deployments could separate the analysis service from the web layer and add a queue for batch processing.

9. MAIN MODULES

Table 2 lists the modules and distinguishes those that are core to the described workflow from those that are optional or proposed. The status of each module must be verified against the source code before submission.

Table 2: Main modules and implementation status

Module	Function	Status
1. Resume Upload	Accepts PDF/DOCX files, checks type and size	Implemented
2. Text Extraction	Extracts text from PDF and DOCX documents	Implemented
3. Resume Analysis	Preprocessing, section detection, entity and date extraction	Implemented
4. Inconsistency Detection	Timeline, education, skills and wording checks (Table 3)	Implemented
5. Risk Scoring	Combines indicators and model output into a 0–100 score	Implemented
6. Credibility Dashboard	Recruiter view of score, flags and explanations	Implemented
7. Job Description Matcher	Similarity between resume and job description	Implemented
8. Audit History	Stores past analyses for review	Implemented
9. Ethical AI and Documentation	Human-in-the-loop notice, bias notes, user guide	Implemented

Table 3: Categories of inconsistency indicators

Category	Example check	Status
Employment timeline	Overlapping full-time roles; end date before start date; unexplained long gaps	Implemented
Education timeline	Graduation year inconsistent with first job; implausible course duration	Implemented
Experience vs. skills	Large number of advanced skills versus total years of experience	Implemented
Seniority progression	Abrupt jumps in title relative to experience	Implemented
Wording patterns	Vague or generic descriptions; repeated sentences across roles	Implemented
Completeness	Missing dates, institutions or employer names	Implemented

The Resume Upload Module provides the entry point and rejects unsupported or oversized files. The Text Extraction Module converts the document to text, and the Resume Analysis Module structures it into sections and fields. The Fake Information and Inconsistency Detection Module applies the checks in Table 3 and, where a trained model is available, anomaly scoring [14]. The Candidate Risk Scoring Module aggregates the evidence and normalises it. The Recruiter Credibility Dashboard displays the score and each reason; the Job Description Matcher computes the similarity between resume and job text, for example using TF-IDF cosine similarity [10]; and the Audit History records the analyses. The Ethical AI and Documentation module covers the on-screen notice that results are advisory, the bias considerations and a data-handling note in line with applicable data-protection requirements [25, 8].

Module Inputs and Outputs

The modules communicate through simple data structures. The upload module outputs a validated file; the extraction module outputs plain text and a list of warnings; the analysis module outputs a dictionary of sections, entities, dates and numeric features; the detection module outputs a list of indicators, each with a category, severity, explanation and supporting text; and the scoring module outputs the score, the band and the ranked list of contributing factors. The dashboard only consumes the final result object, so its design is independent of how the indicators are produced. Documenting these interfaces also allows each module to be tested alone with prepared inputs, which supports the testing approach described in Section 10 [confirm interfaces with source code].

9.1 Ethical AI Considerations

Because hiring decisions affect livelihoods, the system adopts several safeguards. First, it is advisory: the dashboard states that results are indicators requiring human verification. Second, it avoids protected or proxy attributes such as gender, age, religion, photographs and the name of the institution when computing scores [7, 16]. Third, explanations are shown for every flag so that errors can be spotted and challenged. Fourth, resumes contain personal data and should be processed lawfully, for a stated purpose and for no longer than necessary, in line with the Digital Personal Data Protection Act, 2023 in India [25]; recruitment systems are also classified as high-risk in the EU Artificial Intelligence Act, which requires human oversight and documentation [8]. Finally, candidates should be told that automated analysis is used and should have a way to correct mistakes in their data. These aspects are documented in the project user guide .

10. IMPLEMENTATION AND RESULTS

10.1 Development Environment

The project is developed in Python 3.14 using Pythonon Window, with dependencies listed in a requirements file and source code managed in a GitHub repository [URL].

10.2 Frontend and Backend

The frontend consists of the home page, the upload interface and the dashboard. The Flask backend exposes routes for uploading a file and returning the analysis [list actual routes].

10.3 Upload and Analysis Workflow

After upload the file is checked, text is extracted, the preprocessing and feature steps are applied, and the rule engine and model produce indicators and a score. The returned page lists the flags with explanations and the score visualisation.

10.4 Synthetic Demonstration Scenarios

Because real candidate resumes cannot be shared, the system is demonstrated with fictional, synthetic resumes created by the team, which contain no real person's data. Each scenario contains one deliberately introduced irregularity, such as overlapping employment, a graduation year after the first job, an implausible skills-to-experience ratio, or copied job descriptions, together with one clean control resume (Table 5).

10.5 Testing Methodology

Unit tests written with Pytest check text extraction, date parsing, each rule and the bounds of the risk score; integration tests use the Flask test client to confirm that uploads, error handling and responses behave correctly [confirm number of tests and pass status]. Usability is reviewed manually with sample users [optional].

The test types are: functional tests of every module with valid inputs; boundary tests with very short and very long resumes and with the maximum permitted size; negative tests with empty, corrupted, file password-protected and wrongly named files; and regression tests that re-run all scenarios after every change so that earlier behaviour is not broken. Because the rules are deterministic, their expected output can be written down in advance and compared automatically.

10.6 Performance Evaluation

Performance will be measured as classification quality and as system efficiency (processing time per resume and failures on unusual formats). Table 4 is intentionally empty; no experimental values are claimed.

10.7 Security and Privacy Measures

Table 6 summarises the measures that the implementation should provide. Uploaded files are untrusted input, so type and size checks, safe file names and temporary storage are essential; resumes are sensitive documents, so access should be restricted to authorised recruiters and the audit history should store the minimum necessary data.

10.8 Expected Outcomes and Threats to Validity

The expected outcome is a prototype that reliably extracts text from well-formed resumes, flags the deliberately introduced irregularities of the synthetic scenarios and presents them clearly. Conclusions about real-world detection ability cannot be drawn from synthetic data alone. Threats to validity include a small or synthetic dataset, labels that reflect the authors' assumptions about what is suspicious, over-fitting of rule thresholds to the demonstration cases, and variation in resume layout across countries and industries. Future evaluation should therefore use independent reviewers and, where permitted, anonymised real resumes.

Table 4: Evaluation results (to be completed after testing)

Model / Method	Accuracy	Precision	Recall	F1-score
[Algorithm 1]	[ ]	[ ]	[ ]	[ ]
[Algorithm 2]	[ ]	[ ]	[ ]	[ ]
Rule-based checks only	[ ]	[ ]	[ ]	[ ]
Hybrid (rules + ML)	[ ]	[ ]	[ ]	[ ]

Table 5: Synthetic demonstration scenarios

ID	Scenario (synthetic resume)	Expected system behaviour	Observed result
S1	Clean, consistent resume (control)	Low risk band; few or no flags	[to be tested]
S2	Two full-time jobs overlapping by several months	Employment-overlap flag; raised score	[to be tested]
S3	Graduation year later than first job start	Education-timeline flag	[to be tested]

ID	Scenario (synthetic resume)	Expected system behaviour	Observed result
S4	Many advanced skills with under one year of experience	Skills-vs-experience flag	[to be tested]
S5	Repeated generic descriptions across roles	Wording-pattern flag	[to be tested]
S6	Unusual layout / multi-column PDF	Extraction warning; no false accusation	[to be tested]

Table 6: Security and privacy measures

Area	Measure	Status
File upload	Allow only PDF/DOCX; enforce size limit; sanitise file names	[confirm]
Storage	Delete temporary files after analysis; avoid storing full resume text	[confirm]
Access control	Restrict dashboard and history to authorised recruiters	[Implemented / Proposed]
Transport	Serve the application over HTTPS in deployment	[confirm]
Auditability	Log time, file reference, score and flags without protected attributes	[confirm]

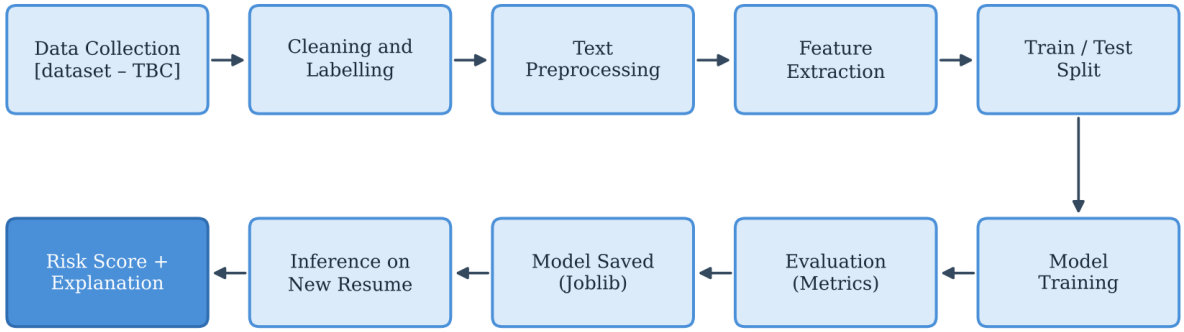


Fig 2: Methodology flowchart of the machine learning pipeline

10.9 Discussion

The design reflects a deliberate trade-off between sensitivity and trust. A system that flags too much trains recruiters to ignore it, whereas a system that flags too little offers little value. Presenting evidence-based explanations makes each flag cheap to check, which keeps the cost of a false positive low. The hybrid of rules and learned models also gives a practical migration path: rules provide dependable behaviour from the first day, and models can be introduced gradually as verified data accumulate. Finally, the audit history makes screening practices reviewable, which is useful both for quality improvement and for demonstrating fair treatment of candidates.

11. ADVANTAGES

1. Reduces the manual effort spent on first-stage resume screening.
2. Helps identify suspicious or inconsistent information that is easily overlooked.
3. Provides structured, repeatable analysis with explanations for every flag.
4. Supports recruiters with risk indicators rather than opaque decisions.
5. Improves the screening workflow through history and job matching.
6. Offers a user-friendly web interface that needs no technical knowledge.

Together these advantages make the system suitable as a first-pass assistant in small and medium-sized organisations, college placement cells and recruitment agencies that cannot afford large manual screening teams, provided that its outputs are always read as advice.

12. LIMITATIONS

1. Accuracy depends on document quality and the data available.
2. Automated detection cannot prove fraud; it can only indicate what to verify.
3. Incorrect or incomplete resume information may affect the analysis.
4. ML models may contain bias inherited from the training data [17].
5. Human verification is necessary before any decision.
6. Some PDF and DOCX formats, such as scanned or multi-column layouts, may not extract correctly.
7. Evaluation is limited by the absence of a large verified dataset [confirm].

These limitations imply that the system should be introduced gradually, monitored for unexpected behaviour, and evaluated periodically with fresh data and independent reviewers, and that recruiters should be trained to interpret flags correctly.

13. FUTURE ENHANCEMENT

1. Advanced NLP models such as transformer encoders [9].
2. Larger, verified and anonymised resume datasets.
3. Improved anomaly detection, including deep approaches [13].
4. Integration with applicant tracking and recruitment management systems.
5. Secure database integration with access control and retention policies.
6. Multi-language resume analysis.
7. Improved explainable AI using methods such as LIME and SHAP [18, 19].
8. Cloud deployment and scalability.

Beyond these items, a longer-term goal is a feedback loop in which recruiters can mark a flag as useful or mistaken; such feedback, stored with consent, would provide verified labels for improving the models and calibrating the score thresholds.

14. CONCLUSION

This paper presented the design of an AI-powered Fake Resume Analyser and Candidate Credibility Detection System that combines resume text extraction, NLP, rule-based consistency checking and Machine Learning to help recruiters identify potentially suspicious or inconsistent resume information. By turning an unstructured resume into explainable indicators and a risk score, the system aims to make first-stage screening faster, more consistent and easier to audit, and its job-description matcher and history features support day-to-day recruitment work.

The system is a decision-support tool. It does not establish that any candidate is fraudulent, and its outputs must be interpreted together with verification of documents and references. Final recruitment decisions must remain with human recruiters, and the system must be used in a fair, transparent and privacy-respecting manner. Quantitative evaluation, confirmed implementation details and screenshots are to be added before submission, after which the work can be extended with stronger language models, larger verified datasets and improved explainability.

ACKNOWLEDGEMENT

The authors thank [guide name] and the Department of [department name], [college name], for guidance and facilities provided during this project. [Add funding or other acknowledgements, or delete this section.]

REFERENCES

[1] C. A. Henle, G. Dineen, and M. K. Duffy, "Assessing intentional resume deception: Development and nomological network of a resume fraud measure," J. Bus. Psychol., vol. 34, pp. 87–106, 2019, doi: 10.1007/s10869-017-9527-4.

[2] J. Guillory and J. T. Hancock, "The effect of LinkedIn on deception in resumes," Cyberpsychol. Behav. Soc. Netw., vol. 15, no. 3, pp. 135–140, 2012, doi: 10.1089/cyber.2011.0389.

[3] K. Yu, G. Guan, and M. Zhou, "Resume information extraction with cascaded hybrid model," in Proc. 43rd Annu. Meeting Assoc. Comput. Linguistics (ACL), 2005, pp. 499–506, doi: 10.3115/1219840.1219902.

[4] V. Bhatia, P. Rawat, A. Kumar, and R. R. Shah, "End-to-end resume parsing and finding candidates for a job description using BERT," arXiv:1910.03089, 2019.

[5] J. S. Black and P. van Esch, "AI-enabled recruiting: What is it and how should a manager use it?" Bus. Horizons, vol. 63, no. 2, pp. 215–226, 2020, doi: 10.1016/j.bushor.2019.12.001.

[6] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," ACM Comput. Surv., vol. 41, no. 3, Art. no. 15, 2009, doi: 10.1145/1541880.1541882.

[7] M. Raghavan, S. Barocas, J. Kleinberg, and K. Levy, "Mitigating bias in algorithmic hiring: Evaluating claims and practices," in Proc. ACM Conf. Fairness, Accountability, Transparency (FAT*), 2020, pp. 469–481, doi: 10.1145/3351095.3372828.

[8] Regulation (EU) 2024/1689 of the European Parliament and of the Council laying down harmonised rules on artificial intelligence (Artificial Intelligence Act), Official Journal of the European Union, Jun. 2024.

[9] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in Proc. NAACL-HLT, 2019, pp. 4171–4186, doi: 10.18653/v1/N19-1423.

[10] K. Spärck Jones, "A statistical interpretation of term specificity and its application in retrieval," J. Documentation, vol. 28, no. 1, pp. 11–21, 1972, doi: 10.1108/eb026526.

[11] S. Bird, E. Klein, and E. Loper, Natural Language Processing with Python. Sebastopol, CA, USA: O'Reilly Media, 2009.

[12] D. Jurafsky and J. H. Martin, Speech and Language Processing, 3rd ed. (draft). Stanford, CA, USA, 2024. [Online]. Available: https://web.stanford.edu/~jurafsky/slp3/

[13] G. Pang, C. Shen, L. Cao, and A. van den Hengel, "Deep learning for anomaly detection: A review," ACM Comput. Surv., vol. 54, no. 2, Art. no. 38, 2021, doi: 10.1145/3439950.

[14] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in Proc. 8th IEEE Int. Conf. Data Mining (ICDM), 2008, pp. 413–422, doi: 10.1109/ICDM.2008.17.

[15] M. Goldstein and S. Uchida, "A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data," PLoS ONE, vol. 11, no. 4, Art. no. e0152173, 2016, doi: 10.1371/journal.pone.0152173.

[16] A. Köchling and M. C. Wehner, "Discriminated by an algorithm: A systematic review of discrimination and fairness by algorithmic decision-making in the context of HR recruitment and HR development," Bus. Res., vol. 13, pp. 795–848, 2020, doi: 10.1007/s40685-020-00134-w.

[17] N. Mehrabi, F. Morstatter, N. Saxena, K. Lerman, and A. Galstyan, "A survey on bias and fairness in machine learning," ACM Comput. Surv., vol. 54, no. 6, Art. no. 115, 2021, doi: 10.1145/3457607.

[18] M. T. Ribeiro, S. Singh, and C. Guestrin, "'Why should I trust you?': Explaining the predictions of any classifier," in Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, 2016, pp. 1135–1144, doi: 10.1145/2939672.2939778.

[19] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in Proc. Adv. Neural Inf. Process. Syst. (NIPS), vol. 30, 2017, pp. 4765–4774.

[20] F. Doshi-Velez and B. Kim, "Towards a rigorous science of interpretable machine learning," arXiv:1702.08608, 2017.

[21] Pallets Projects, "Flask documentation." [Online]. Available: https://flask.palletsprojects.com/

[22] C. Cortes and V. Vapnik, "Support-vector networks," Mach. Learn., vol. 20, no. 3, pp. 273–297, 1995, doi: 10.1007/BF00994018.

[23] L. Breiman, "Random forests," Mach. Learn., vol. 45, no. 1, pp. 5–32, 2001, doi: 10.1023/A:1010933404324.

[24] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," J. Mach. Learn. Res., vol. 12, pp. 2825–2830, 2011.

[25] Government of India, The Digital Personal Data Protection Act, 2023 (Act No. 22 of 2023), Ministry of Law and Justice, New Delhi, India, 2023.