Resume Classification using Machine Learning

A Project Report submitted in partial fulfilment of the requirements for the award of the degree of

Bachelor of Technology

in

Computer Science and Engineering

by

Abhishek Suman(2115000037)
Govind(2115000416)
Dhruv Yadav(2115000369)

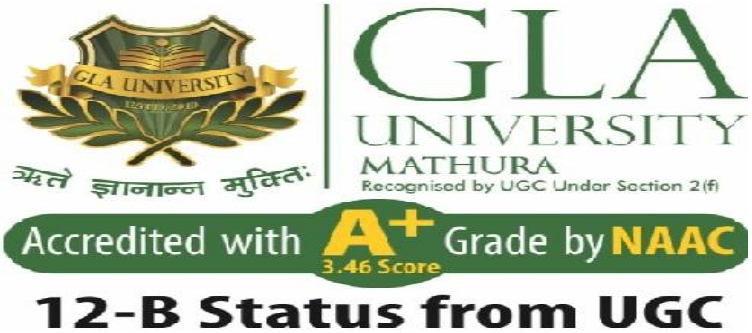
Group No.:132

Under the Guidance of

Raushan Kumar Singh
(Assistant Professor)

Department of Computer Engineering & Applications

Institute of Engineering & Technology



GLA University
Mathura- 281406, INDIA

April, 2025

Department of Computer Engineering and Applications
GLA University, 17 km Stone, NH#2, Mathura-Delhi Road,
P.O. Chaumuhan, Mathura-281406 (U.P.)

Declaration

I hereby declare that the work which is being presented in the B.Tech. Project **"Resume Classification using Machine Learning"**, in partial fulfillment of the requirements for the award of the ***Bachelor of Technology*** in Computer Science and Engineering and submitted to the Department of Computer Engineering and Applications of GLA University, Mathura, is an authentic record of my own work carried under the supervision of Raushan Kumar Singh **(Assistant Professor).**

The contents of this project report, in full or in parts, have not been submitted to any other Institute or University for the award of any degree.

Sign _____

Abhishek Suman
2115000037

Sign _____

Govind
2115000416

Sign _____

Dhruv Yadav
2115000369

Certificate

This is to certify that the above statements made by the candidate are correct to the best of my/our knowledge and belief.

_____

Supervisor
(**Raushan Kumar Singh**)
Assistant Professor
Dept. of Computer Engg, & App.

_____

Project Co-ordinator
(Dr. Mayank Srivastava)
Associate Professor
Dept. of Computer Engg, & App.

_____

Program Co-ordinator
(**Dr. Nikhil Govil**)
Associate Professor
Dept. of Computer Engg, & App.

Date:

ACKNOWLEDGEMENT

We have taken efforts in this project. However, it would not have been possible without the kind support and help of our mentor. I would like to extend my sincere thanks to all of them.

We are highly indebted to Raushan Kumar Singh for his guidance and constant supervision as well as for providing necessary information regarding the project & also for his support in completing the project.

We would like to express my gratitude towards our faculty of GLA University & members for their kind co-operation and encouragement which help us in completion of this project.

Sign _____

Abhishek Suman
University Roll No-2115000037

Sign _____

Govind
University Roll No-2115000416

Sign _____

Dhruv Yadav
University Roll No-2115000369

_____

ABSTRACT

In today's fast-paced recruitment landscape, Human Resource departments are overwhelmed by the increasing volume of job applications, making the initial resume screening process both time-consuming and inefficient. This project proposes an intelligent Resume Classification System that leverages Natural Language Processing (NLP) and Machine Learning (ML) to automate the classification of resumes into predefined job categories. The core objective is to assist recruiters in filtering candidates based on role-specific relevance, thereby accelerating the hiring process and improving accuracy.

The system begins by preprocessing unstructured resume data through techniques such as tokenization, stopword removal, lemmatization, and normalization. Cleaned textual data is then transformed into numerical features using TF-IDF vectorization, capturing the contextual weight of terms across the dataset. Several classification algorithms, including Naive Bayes, Logistic Regression, K-Nearest Neighbors, and Support Vector Machine (SVM), were trained and evaluated. Among these, SVM delivered the highest accuracy of 98%, making it the optimal model for deployment.

CONTENTS

Chapter 1
Introduction

1.1 Motivation and Overview

In today's highly competitive job market, the role of technology in the recruitment process has become more critical than ever. Organizations, especially large multinational companies, receive hundreds to thousands of resumes for each job opening. Manual evaluation of such a vast volume of resumes is not only time-consuming but also highly susceptible to human biases, errors, and fatigue.
The lack of an automated system often results in deserving candidates being overlooked due to inconsistencies in manual screening. Furthermore, companies lose valuable time that could otherwise be invested in strategic tasks like talent development and employee engagement.

To address these challenges, leveraging advancements in Natural Language Processing (NLP) and Machine Learning (ML) offers a promising solution. With NLP, systems can comprehend unstructured textual data from resumes, while ML models can learn patterns and make intelligent predictions. Together, they form a powerful toolset for automating resume screening processes.

As Computer Science students passionate about solving real-world problems, we recognized this gap and were motivated to build a Resume Classification System. Our goal was to create a solution that not only assists Human Resource (HR) departments but also benefits students and job seekers by providing them insights into how their resumes match with particular job roles.

We began our journey by conducting thorough research on the structure and components of resumes. We found that resumes, although diverse in formatting, typically contain key sections such as education, skills, work experience, certifications, and projects. Extracting and understanding information from these sections posed a challenge, given the varied ways candidates present their data.

Therefore, our first step was to design a robust pre-processing pipeline capable of cleaning, organizing, and standardizing resume content.

Next, we applied Natural Language Processing techniques such as tokenization, lemmatization, named entity recognition (NER), and part-of-speech tagging to extract meaningful features from the resumes. NLP enabled our system to parse complex sentences, understand the context of words, and identify critical information such as job titles, skill sets, and years of experience. Feature extraction was crucial in ensuring that the ML models could be trained effectively on relevant data.

For the classification task, we explored multiple machine learning algorithms, including Logistic Regression, Support Vector Machines (SVM), Random Forests, and Neural Networks. We also implemented Term Frequency-Inverse Document Frequency (TF-IDF) vectorization and word embeddings like Word2Vec to convert textual data into numerical form for the models. After rigorous experimentation and evaluation using metrics such as accuracy, precision, recall, and F1-score, we selected the model that provided the best performance for our classification task.

Our system classifies resumes into predefined job categories such as Data Scientist, Web Developer, Software Engineer, Business Analyst, and more. This classification not only aids recruiters in shortlisting candidates faster but also provides valuable feedback to job seekers regarding the alignment of their resumes with their desired job profiles. In the future, candidates can even receive personalized recommendations on skills they should develop or enhance to improve their employability.

Throughout the development process, we emphasized building an interface that is user-friendly and intuitive. We developed a web application where users can upload their resumes in PDF or text format. Within seconds, they receive a predicted job category along with a confidence score. Additionally, HR personnel can batch-process multiple resumes at once, saving significant time during large-scale hiring drives.

Looking ahead, there are several avenues for extending this project. One potential improvement is integrating deep learning models like BERT (Bidirectional Encoder Representations from Transformers) to further enhance the system’s understanding of

complex language patterns. Another is the inclusion of explainable AI (XAI) techniques to make model decisions more transparent to users, building greater trust in the system. We also envision incorporating multilingual support, enabling the classification of resumes written in languages other than English.

Overall, developing the Resume Classification System has been a rewarding experience, combining our technical skills with a meaningful real-world application. We hope our solution will bridge the gap between candidates and opportunities, making the hiring process more efficient, unbiased, and insightful for all stakeholders involved.

Overview:

This project introduces a Resume Classification System that leverages Natural Language Processing (NLP) for extracting and processing textual information from resumes and Machine Learning (ML) for classifying the resumes into predefined job categories.

Some of the major categories considered are:

- Data Science
- Web Development
- Software Engineering
- Human Resource
- Networking
- Business Development

Our system extracts skills, experiences, and educational qualifications from the resume text and compares them with job descriptions provided. It predicts the job role that best fits the resume, thus offering valuable feedback to students and aiding recruiters in efficient candidate shortlisting.

The first step in our system involves collecting and pre-processing a large dataset of resumes. This includes removing unnecessary formatting, eliminating stop words, and normalizing the text to ensure consistency. We then apply advanced NLP techniques such as tokenization, lemmatization, and named entity recognition (NER) to identify

important information like technical skills, previous job titles, degrees, certifications, and years of experience.

Once the relevant features are extracted, the data is vectorized using methods like TF-IDF (Term Frequency-Inverse Document Frequency) and word embeddings to convert textual information into numerical representations that can be fed into machine learning models. Several classification algorithms, including Random Forest, Support Vector Machine (SVM), and Logistic Regression, were experimented with to achieve optimal prediction accuracy.

An intuitive web interface was also developed where users can upload their resumes. The system instantly analyzes the content and provides a prediction along with a confidence score, indicating the likelihood of the resume fitting into a particular job role. This feedback mechanism is especially helpful for fresh graduates, allowing them to understand how recruiters might perceive their resumes and areas where they can make improvements.

Looking forward, we plan to enhance the system by including additional job categories, integrating deep learning models like BERT for improved text understanding, and providing personalized suggestions to candidates for better resume building.

1.2 Objective

The main objectives of this project are:

• To analyze the content of a resume using NLP techniques and predict its relevance to different job profiles.

• To help students and freshers understand whether their resume is aligned with their career goals.

• To assist recruiters by automatically filtering and classifying resumes, thereby saving time and resources.

• To improve the chances of selection for students by helping them customize their resumes based on job roles.

• To create an extendable system that can later suggest resume improvements and even highlight missing skills.

By achieving these objectives, we aim to bridge the gap between job seekers and employers, making the hiring process more efficient, intelligent, and bias-free.

To further support these goals, our system focuses on accurately parsing resumes that are often diverse in structure and formatting. Resumes may vary greatly from candidate to candidate, with information presented in different styles and sequences. Therefore, developing a robust preprocessing and feature extraction pipeline was critical. By leveraging NLP models, we are able to capture essential details, ensuring that no important qualification or skill is missed during classification.

Additionally, our project not only identifies the current job fit for a candidate but also provides valuable analytical insights. For example, if a candidate is aiming for a Data Scientist role but lacks necessary skills like Python, SQL, or machine learning techniques, the system can be enhanced in future iterations to recommend skill development areas. This kind of targeted feedback will help students and job seekers build stronger, more competitive profiles.

Moreover, recruiters often struggle with large volumes of resumes that must be screened within tight deadlines. By automating the preliminary screening process, our system helps HR teams focus on interviewing and hiring the most relevant candidates rather than spending hours on manual filtering. This ensures a more productive, merit-based hiring pipeline and reduces the chances of overlooking qualified applicants due to human fatigue or bias.

Ultimately, the Resume Classification System is designed to serve as a valuable tool for both ends of the recruitment process — enhancing candidate preparation and optimizing recruiter efficiency.

1.3 Scope

The scope of the project is wide and includes:

• Assisting students during campus placements and internships.

• Helping universities and placement cells to guide students with resume-building.

• Reducing the workload on recruitment teams by pre-classifying resumes.

• Future expansion into automatic job-matching and skill suggestion systems.

• Extension into supporting various industries beyond IT (such as Finance, Marketing, Healthcare).

In future versions, our system could be integrated with job portals like LinkedIn, Naukri, and Indeed for direct resume screening during the application process.

Beyond campus placements, the system can also play a crucial role in supporting career counseling programs. Universities and colleges can use it to analyze large batches of student resumes, identify common skill gaps, and accordingly plan training sessions or workshops to better prepare students for industry demands. This ensures a more strategic and data-driven approach toward student development and employability enhancement.

On the recruitment side, companies often face difficulty shortlisting candidates for technical as well as non-technical roles. Our system, by pre-categorizing resumes based on skillsets, can drastically cut down the initial screening time, allowing HR professionals to focus on interviewing and onboarding processes. It also minimizes the risks of overlooking deserving candidates, which often happens due to the overwhelming number of applications.

In terms of expansion, the Resume Classification System can be extended to industries such as finance (for roles like financial analyst, accountant), healthcare (for medical coding, clinical data management roles), and marketing (for digital marketing, brand management positions). By updating the classification categories and training models with industry-specific datasets, the system can become a versatile solution across multiple domains.

Chapter 2
Software Requirement Analysis

2.1 Software Requirement Analysis

The Resume Classification System primarily requires a strong backend setup focusing on Natural Language Processing (NLP) and Machine Learning (ML) tasks. Python is selected as the core programming language due to its extensive support for data processing and ML libraries such as NLTK, spaCy, Scikit-learn, and TensorFlow. These libraries enable effective text extraction, feature engineering, and model training for resume classification.

For handling resume files, libraries like PyPDF2 or pdfminer are used to extract textual content from PDF documents. Text pre-processing operations such as tokenization, lemmatization, and named entity recognition (NER) are essential for cleaning and structuring resume data before feeding it into machine learning models. Tokenization breaks down the resume text into manageable units, while lemmatization reduces words to their base forms, thus helping in feature consistency. Named Entity Recognition (NER) identifies important elements such as skills, qualifications, certifications, and experience details from resumes, enhancing the model's understanding of candidate profiles.

In addition to basic text preprocessing, advanced techniques like stop-word removal, stemming, and lowercasing are applied to standardize the textual data. Special characters, numbers, and irrelevant information are filtered out to minimize noise and focus on important content that influences classification. Text normalization processes such as these significantly improve the accuracy and efficiency of machine learning models.

For model development, classification algorithms like Random Forest, Support Vector Machine (SVM), and Logistic Regression are utilized, along with text vectorization techniques like TF-IDF Vectorizer or Word2Vec embeddings to transform unstructured data into numerical features suitable for machine learning. TF-IDF (Term Frequency-

Inverse Document Frequency) is particularly effective for highlighting important terms in resumes, giving higher weight to domain-specific keywords. Word2Vec embeddings help capture semantic relationships between words, thereby improving the model's understanding of context and job relevance.

In the training phase, a labeled dataset consisting of resumes categorized by job roles (e.g., Data Scientist, Web Developer, Software Engineer) is used. The dataset is divided into training and testing subsets to evaluate the model's generalization ability. Hyperparameter tuning is conducted using Grid Search or Random Search to identify the best-performing model configurations. Evaluation metrics such as precision, recall, F1-score, and accuracy are calculated to measure the model's effectiveness. Cross-validation is employed to ensure that the model performs consistently across different subsets of data, reducing the risk of overfitting.

For deploying the system and providing user access, a simple yet intuitive web-based front-end is developed using frameworks like Streamlit, Flask, or Django. The front-end allows users to upload their resumes and, optionally, job descriptions. Upon uploading, the resume content is processed in real-time, and the system predicts the most suitable job category along with a relevance score. The feedback provided includes recommendations for improving resume content by highlighting missing or weak areas compared to ideal profiles for that job role.

Furthermore, the system architecture is designed with scalability in mind. Microservices architecture could be considered for large-scale deployment, where separate services handle document processing, feature extraction, classification, and reporting independently. Containerization technologies like Docker can be used to package the application, ensuring consistent and scalable deployment across different environments.

Security and data privacy are given high priority throughout the system. As resumes contain sensitive personal information, the application ensures that uploaded data is not stored permanently unless explicitly required. Secure communication protocols (such as HTTPS) are enforced for data transmission, and measures like input sanitization and access control are implemented to prevent unauthorized access and maintain data confidentiality.

In the future, the system can be further enhanced by incorporating deep learning models such as BERT (Bidirectional Encoder Representations from Transformers) or RoBERTa to achieve better contextual understanding of resumes. Additionally, the recommendation engine can be expanded to suggest suitable job openings based on classified roles, thus bridging the gap between job seekers and employers even more effectively.

Overall, the backend design and implementation of this Resume Classification System are centered around robust data handling, intelligent modeling, seamless user interaction, and strict adherence to privacy and security standards, ensuring a reliable and impactful tool for students, job seekers, and recruiters alike.

Building the Resume Classification System requires specific software and libraries:

Software	**Purpose**
Python 3.8+	Programming Language
Pandas	Data manipulation
NumPy	Numerical operations
Scikit-learn	Machine Learning algorithms
Matplotlib & Seaborn	Data Visualization
Jupyter Notebook / Google Colab	IDE/Platform for development
Streamlit/Tkinter (optional)	Interface development
Web Browser	Viewing interface (if web-based)
Dataset Format	CSV / Excel / JSON files

Why_Python?

Python was selected due to its large ecosystem of libraries, readability, and strong support for machine learning and data science.

2.2 Authors and Their Proposed Algorithm

Researchers and developers have proposed innovative approaches to automate resume classification using Natural Language Processing (NLP) and Machine Learning (ML). For instance:

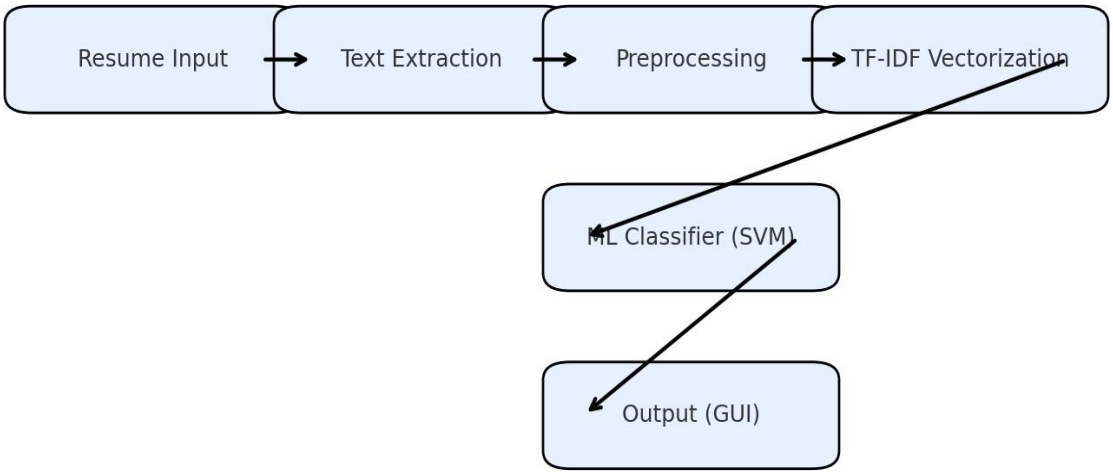
- ✓ Z. Wang et al. (2017) proposed a deep learning-based method using CNNs to extract key features from resumes and match them with job descriptions.

- ✓ D. Bhatia and R. Sinha (2019) introduced a keyword-based filtering approach combined with Naive Bayes classification to sort resumes by relevance.

- ✓ R. Kaur and A. Sharma (2020) developed a system using TF-IDF and Support Vector Machine (SVM) to categorize resumes into domains like IT, Finance, and Marketing.

- ✓ S. Roy et al. (2021) implemented BERT-based embeddings to capture contextual meaning and improve the accuracy of matching resumes with job roles.

In our project, we have adopted a similar approach by using NLP techniques (TF-IDF and CountVectorizer) with Machine Learning models such as Random Forest, Logistic Regression, and SVM for efficient and accurate resume classification.



(*Architecture of the Resume Classification System using NLP and Machine Learning***)**

2.3 Challenges:

While working on the Resume Classification, we faced several challenges during different phases of the project:

- ✓ **Data Collection**: Obtaining a large and diverse dataset of resumes and job descriptions was difficult due to privacy concerns and limited public availability.
- ✓ **Unstructured Data**: Resumes come in various formats and styles, making it challenging to extract consistent and meaningful features.
- ✓ **Labeling the Data**: Manually labeling resumes with correct job categories was time-consuming and prone to human error.
- ✓ **Text Preprocessing**: Cleaning, normalizing, and vectorizing textual data required careful handling to preserve context and reduce noise.
- ✓ **Model Accuracy**: Achieving high accuracy while avoiding overfitting required fine-tuning multiple algorithms and performing extensive testing.
- ✓ **Relevance Matching**: Ensuring the classifier accurately evaluates the relevance of a resume against job descriptions was a complex task.

2.4 Conclusion:

This project successfully demonstrates how machine learning and natural language processing can be used to classify resumes based on job relevance. It helps students identify whether their CV aligns with a specific job description, increasing their chances of selection. By automating the resume screening process, it saves time for both candidates and recruiters. The model provides feedback to improve resume content and relevance. It also promotes awareness among students about tailoring CVs for specific job roles. Overall, this project contributes to more efficient and targeted job application processes.

Students can gain insights about their profiles, recruiters can save time, and the entire recruitment cycle becomes smarter and more objective. Moreover, this project highlights the importance of data-driven decision-making in the recruitment domain. Instead of relying solely on manual judgment, the Resume Classification System offers

a consistent and bias-reduced approach to candidate evaluation. By analyzing resumes purely on the basis of skills, education, and relevant experience, the model ensures that deserving candidates are not overlooked due to subjective factors.

The project also encourages students and job seekers to take a proactive approach toward career planning. By receiving detailed feedback on missing or weak skills, candidates are empowered to undertake additional certifications, courses, or internships to enhance their employability. This not only benefits individuals but also helps bridge the broader skill gap in industries by ensuring better-prepared candidates enter the job market. Building a habit of continuous learning and upskilling becomes easier when candidates know exactly what areas need improvement based on real-world job market requirements.

For recruiters, especially in large organizations where thousands of applications are received, the system significantly reduces the workload by shortlisting candidates who closely match the job requirements. It enables HR teams to focus more on in-depth interviews and talent development rather than initial screening tasks. Consequently, the recruitment pipeline becomes more efficient, leading to faster hiring cycles and reduced costs associated with candidate sourcing and evaluation.

Additionally, by integrating the Resume Classification System with Application Tracking Systems (ATS), organizations can automate end-to-end workflows, from resume parsing to preliminary candidate ranking. This integration ensures that no qualified candidate is missed and hiring decisions are made faster and more accurately. Companies can also maintain a talent pool database where candidates are tagged and classified based on their profiles, allowing for easy retrieval when relevant job openings arise in the future.

Another important contribution of this project is that it brings transparency to the recruitment process. Candidates receive objective evaluations based on skill matching rather than being judged by inconsistent human interpretation. This can contribute to improving diversity and inclusion in hiring by minimizing unconscious bias that often influences manual resume screening.

In terms of technical contributions, this project offers a scalable and adaptable framework. The model can easily be retrained with new data, making it future-proof against changing job market trends. As industries evolve and new job roles emerge, updating the model with recent resumes and job descriptions will ensure its continued relevance and accuracy. Moreover, the system’s modular design allows easy

enhancement by integrating more sophisticated models like BERT or GPT-based classifiers, which can further improve contextual understanding and classification accuracy.

Looking ahead, future versions of the system could offer candidates automated suggestions for resume formatting, wording improvements, and even skill acquisition paths tailored to their desired career goals. Incorporating LinkedIn data, MOOC platforms (like Coursera or edX), and real-time job market analytics could allow the system to dynamically guide students on upcoming skill demands, trending certifications, and popular career trajectories.

Furthermore, collaboration with educational institutions can help students prepare better during their academic journey. Universities and colleges could adopt such systems for career counseling, ensuring their students are industry-ready by the time they graduate. Career services could integrate it with placement drives, offering personalized resume-building sessions and targeted job matching.

In conclusion, this Resume Classification System is not just a technical achievement but a meaningful step towards revolutionizing the recruitment process. It empowers candidates, assists recruiters, and ultimately creates a smarter, faster, and fairer hiring ecosystem. With further enhancements and widespread adoption, this approach has the potential to redefine how talent acquisition is approached globally, leading to a more efficient and meritocratic job market.

Chapter 3
Software Design

The software design for this project focuses on building a robust and user-friendly system that analyzes student resumes and matches them with relevant job descriptions using Natural Language Processing (NLP) and Machine Learning (ML) techniques. The architecture is modular, consisting of components for data preprocessing, feature extraction, classification, and result visualization. The frontend interface allows students to upload their resumes and view analysis reports, while the backend handles the model training and prediction logic. The system is designed for scalability and can be extended to support different job roles and resume formats. Security and data privacy are also considered in the overall design structure.

The data preprocessing module is responsible for extracting and cleaning textual information from resumes, removing noise such as redundant formatting or irrelevant text. Feature extraction focuses on identifying important sections such as skills, education, experience, and certifications. These extracted features are transformed into numerical representations to be used by the classification model.

The classification module uses trained ML algorithms to predict the job role that best matches the resume content. The system is flexible enough to incorporate new models or update existing ones as better algorithms or more data become available. Additionally, the design ensures minimal response time, even when handling multiple requests simultaneously. Future improvements could include incorporating advanced deep learning models for even better accuracy and integrating feedback loops where user corrections help refine and retrain the system for continuous improvement.

Beyond these core modules, an authentication and user management system is integrated to ensure that the resume data is handled securely. Students must log in to upload resumes, view feedback, and track their progress over time. This ensures that each student’s data remains confidential and prevents unauthorized access. Role-based access control can be implemented, wherein administrative users or university

counselors can have access to analytics dashboards, helping them assess the overall readiness of batches of students for specific industries.

The system also includes a reporting module that generates detailed feedback for the users. This module provides insights not only on the classification result but also on areas of improvement, highlighting missing skills or certifications that could strengthen the resume for a particular job role. Visual graphs and recommendation sections make the feedback easily understandable and actionable for students.

An important aspect of the design is the choice of database. A lightweight yet powerful database such as PostgreSQL or MongoDB is used to store user profiles, resume content, classification outcomes, and feedback history. The database is designed to efficiently handle large volumes of data while ensuring fast retrieval for real-time interactions.

In the backend, an API-driven approach is adopted. RESTful APIs are developed to manage the interactions between the frontend and the backend, ensuring a clear separation of concerns. This not only improves maintainability but also allows for future extensions, such as integrating mobile applications or third-party career counseling services.

Scalability is another key consideration. The system is deployed on cloud infrastructure such as AWS, Azure, or GCP, allowing it to auto-scale based on usage demands. Load balancers and caching mechanisms like Redis are employed to minimize server load and response time, enhancing the overall user experience even during peak usage.

Furthermore, continuous integration and continuous deployment (CI/CD) pipelines are established to ensure that updates, model retraining, and system improvements can be pushed seamlessly without downtime. Testing frameworks are incorporated for unit, integration, and system testing to maintain the reliability of every deployed component.

To ensure extensibility, the design is modular at the algorithmic level too. For instance, initially, classification models like Logistic Regression or Random Forest can be deployed, but later more sophisticated models like BERT embeddings or fine-tuned transformers can be integrated without requiring a complete system overhaul. This

forward-thinking architecture guarantees that the system remains relevant as advancements in AI and recruitment technologies evolve.

Another future enhancement could involve using Optical Character Recognition (OCR) technologies to allow resume parsing from scanned documents or images, making the system even more versatile. Support for multilingual resumes could also be considered to cater to a global audience.

In conclusion, the software design balances robustness, scalability, user-friendliness, and security. It provides a strong foundation not only for the current goals of resume classification and feedback but also for future expansions such as personalized career counseling, automated interview scheduling, and industry-aligned skill gap analysis. This comprehensive and forward-looking approach ensures that the Resume Classification System will remain impactful and adaptable for years to come.

Dataset and Pre-processing :-

We used a dataset consisting of resumes categorized by various job roles such as Data Scientist, Web Developer, Software Engineer, Human Resource (HR) Manager, Business Analyst, and others, along with corresponding job descriptions. The resumes were collected in both PDF and text formats, ensuring a variety of document structures and content styles. Job descriptions were sourced from reputable job portals like LinkedIn, Indeed, and Glassdoor, as well as official company career pages to maintain authenticity and real-world relevance.

The initial dataset contained a mix of structured and unstructured data. Resumes varied significantly in terms of formatting, section headings, and writing styles, which made standardization a crucial part of the pre-processing stage. Similarly, job descriptions varied across companies in terms of detailed expectations, skills required, and role descriptions. To handle this diversity, a comprehensive pre-processing pipeline was established.

Pre-processing involved multiple steps:

1. **Text Extraction** – Extracting clean text from resumes and job descriptions.

2. **Text Cleaning** – Removing special characters, stop words, punctuations, and converting all text to lowercase.

Algorithm purposed

- ✓ Data Collection:
- ✓ Gather a dataset of resumes and job descriptions, categorized by job roles (e.g., Web Developer, Data Analyst, etc.).

- ✓ Pre-processing:

- ✓ Clean the textual data by removing stop words, special characters, and performing tokenization, lemmatization, and vectorization (using TF-IDF or Word2Vec).Feature Extraction:
- ✓ Extract relevant features from the resumes (skills, experience, projects) and from job descriptions (required qualifications and skills).

- ✓ Model Selection:
- ✓ Use classification algorithms like Logistic Regression, Random Forest, or Support Vector Machine (SVM) to train the model on labeled resume-job role data.

- ✓ Model Training:
- ✓ Train the model using the processed dataset to learn the mapping between resume features and job roles.

- ✓ Resume Classification:
- ✓ Input a student's resume, process it similarly, and predict its most relevant job role using the trained model.

- ✓ Relevancy Check:
- ✓ Compare the predicted role with a given job description to check if the resume matches or fits well using cosine similarity or semantic matching.

System Architecture:

The User Interface (UI) serves as the entry point where students can interact with the Resume Classification System. Designed with a focus on simplicity and intuitiveness, the interface ensures that users of all technical backgrounds can easily upload their resumes and, optionally, the specific job descriptions they are targeting. The UI allows for straightforward actions like file uploading, selecting a job role, and initiating the analysis process with just a few clicks. Additionally, it provides immediate visual feedback (such as upload confirmations and progress indicators) to enhance user confidence during interaction. Accessibility considerations, such as mobile responsiveness and support for multiple document formats (PDF, DOCX, TXT), further improve the system's usability.

Once resumes and job descriptions are uploaded through the UI, they are sent to the **Processing Layer**. This layer is critical as it prepares the raw input for deeper analysis. The first task of the Processing Layer is text extraction — using libraries like PyPDF2, pdfminer.six, or docx for different file types — to retrieve clean textual content from diverse resume formats. Special attention is paid to preserving important structural information such as headings, bullet points, and section separations.

After extraction, a comprehensive **text cleaning** process begins. Cleaning involves the removal of unnecessary formatting tags, non-standard fonts, special characters, multiple spaces, and irrelevant artifacts. The system also uses pre-built stopword lists from libraries like NLTK or spaCy to filter out words that don't contribute meaningfully to resume analysis. In addition, techniques like lowercasing and punctuation removal are applied to ensure uniformity across all text samples, further improving the quality of downstream processing.

Following this, the data moves to the **Feature Extraction Layer**. In this stage, advanced Natural Language Processing (NLP) techniques are used to parse the cleaned text and extract meaningful features. Named Entity Recognition (NER) identifies key entities such as technologies, programming languages, academic degrees, certifications (e.g., AWS Certified Developer), and company names. Keyword extraction algorithms detect critical skills and role-specific competencies mentioned in the resume. Sectional segmentation (dividing resumes into logical parts like Education, Experience, Projects) is also performed to enable context-aware feature extraction. The extracted features are

structured into a form usable by machine learning models — typically as structured feature vectors or embedding matrices.

The core analysis happens in the **Machine Learning Layer**. Here, the features extracted from the resume are compared against the desired job profile. Pre-trained classification algorithms such as Random Forest, Logistic Regression, or Support Vector Machine (SVM) models predict the best-matching job role for the given resume. In cases where the user has provided a job description, semantic similarity scoring is also performed using methods like cosine similarity on TF-IDF vectors or using sentence embeddings from models like Sentence-BERT.

The model also identifies skill gaps — listing skills required in the job description but missing from the resume. Additionally, a **matching score** (out of 100) is computed, reflecting how closely the candidate's profile aligns with the specified job role or industry standard for that role.

The final step is handled by the **Output Layer**, which presents the results back to the user in a highly visual and comprehensible format. The output dashboard includes:

- The predicted job category for the uploaded resume.
- A detailed skills match report showing matched skills, missing skills, and suggested additions.
- A resume quality score based on metrics like diversity of skills, experience duration, and educational background.
- Personalized suggestions for improvement, such as adding specific certifications, including quantifiable achievements, or optimizing the skills section.

a.Detailed Design

The system begins with a user-friendly interface where students can upload their resume and a specific job description. The backend then uses Natural Language Processing (NLP) techniques to extract keywords and relevant features from both documents. A feature-matching module compares the extracted data to measure similarity and identify gaps. A machine learning model, trained on labeled datasets, classifies the resume as relevant or not. The system finally generates a report with a relevance score and suggestions for improvement.

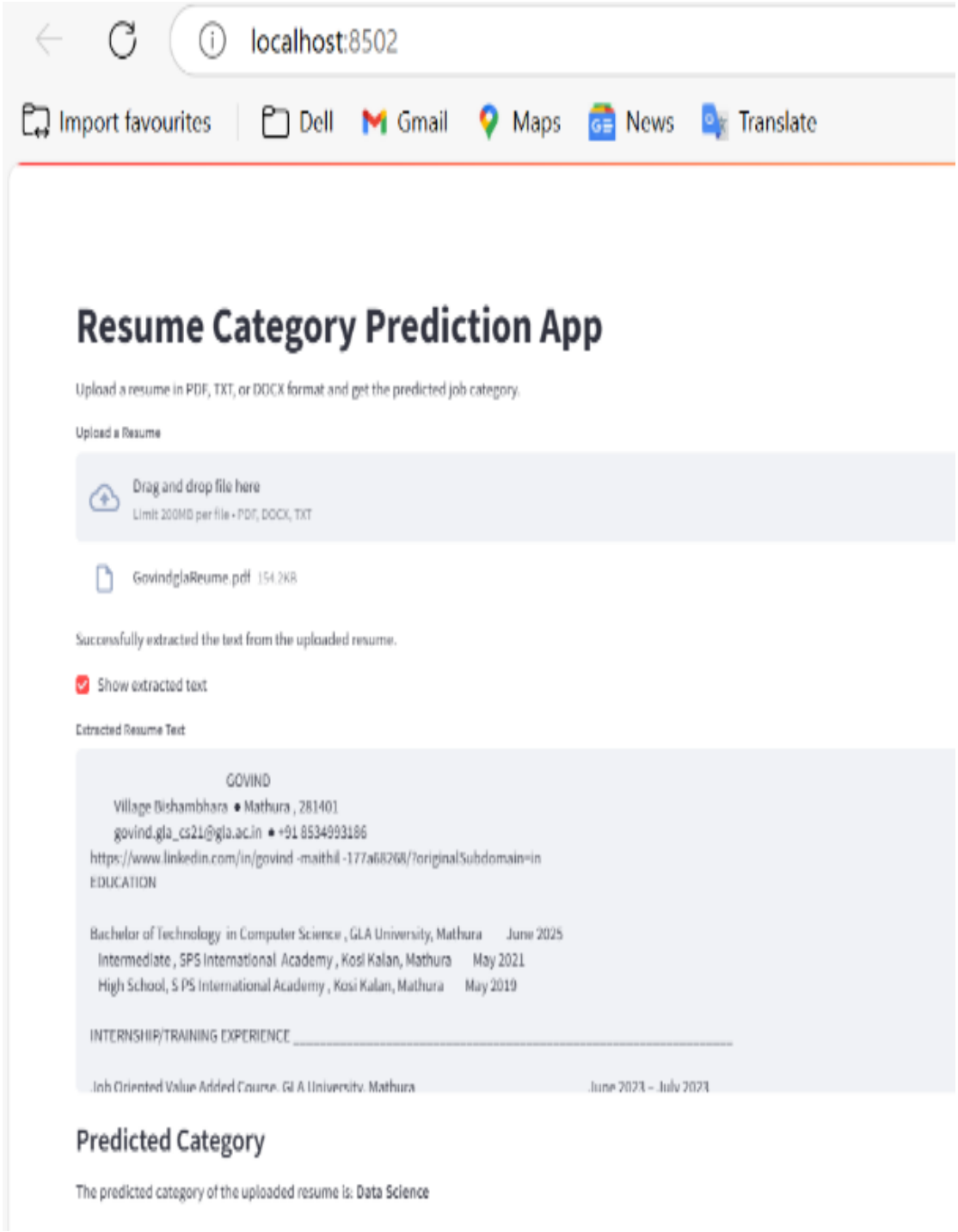
The initial stage focuses on accurately parsing the uploaded documents, ensuring that information such as skills, education, certifications, and experience is properly captured. Advanced text-processing techniques like tokenization, lemmatization, and part-of-speech tagging help the system in understanding the context behind the extracted words, ensuring a deeper analysis rather than a superficial keyword match.

Once the features are extracted, the feature-matching module utilizes techniques like cosine similarity, Jaccard similarity, or semantic matching to compare the student's resume with the job description. This comparison highlights the strengths and identifies missing skills or qualifications that might reduce the resume's suitability for the given job.

The Machine Learning model, trained on a diverse dataset of resumes and job descriptions categorized into different roles, uses supervised learning algorithms to predict whether a resume strongly aligns with a particular job profile. The model is designed to continually improve with more data, making predictions more accurate over time.

The generated report provides students not only with a relevance percentage but also highlights specific areas for improvement. These insights help students refine their resumes to match industry standards, thereby enhancing their chances of being shortlisted in a highly competitive job market.

The system also incorporates a feedback mechanism where students can iteratively upload revised versions of their resumes after implementing the suggested improvements. This allows for continuous refinement and better alignment with job-specific expectations. Additionally, the platform may integrate external APIs or databases to validate certifications or cross-check industry-standard terminologies, further enhancing the reliability of the analysis. The user dashboard provides a comprehensive view of past uploads, relevance trends, and personalized recommendations, encouraging students to proactively work on their skill gaps. This holistic approach not only boosts resume quality but also prepares students to better understand employer expectations and adapt accordingly in real-world scenarios.



(System Design)

System Architecture

Our proposed system is designed to intelligently classify resumes into job categories based on extracted information like:

- Skills
- Work Experience
- Education

- Certifications
- Projects

Architecture Overview:

1. **Input Layer**: Resume text or file uploaded by the user.
2. **Preprocessing Layer**:
    - Text extraction
    - Tokenization
    - Lemmatization
    - Stopword removal
3. **Feature Extraction**:
    - Using TF-IDF Vectorizer
4. **Model Prediction**:
    - Applying Machine Learning classifiers to predict the category.
5. **Output Layer**:
    - Predicted category displayed.
    - (Optional) Suggested improvements.

Working Model

1. **Data Collection**:
    - We used a dataset containing labeled resumes across multiple domains like Data Science, HR, Web Development, etc.
2. **Preprocessing Techniques**:
    - Removal of stop words (e.g., "the", "is", "in").
    - Lowercasing all text.
    - Lemmatization (e.g., "running" → "run").
    - Punctuation removal.
3. **Feature Engineering**:

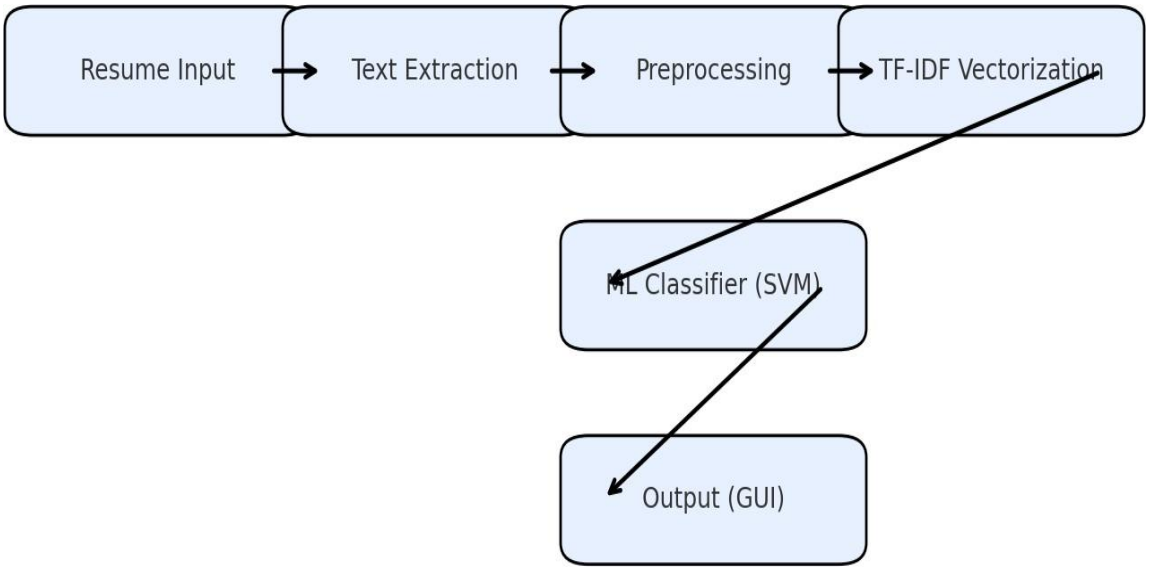
- Using **Term Frequency–Inverse Document Frequency (TF-IDF)** to create numeric vectors from text.

4. **Model Building**:
    - Models like Logistic Regression, Random Forest, and SVM were trained.
    - Grid Search Cross Validation was used for hyperparameter tuning.

5. **Model Evaluation**:
    - Accuracy
    - Precision
    - Recall
    - F1 Score

Data Flow Diagram (DFD)

```
[User uploads resume]
        ↓
[Preprocessing module]
        ↓
[Feature Extraction]
        ↓
[Classification Model]
        ↓
[Predicted Job Category]
```



Data Flow Diagram (DFD)

Advantages of Proposed System

- **Efficiency**: Can screen hundreds of resumes in minutes.
- **Accuracy**: Better understanding using NLP-based preprocessing.
- **Cost-effective**: Developed using open-source technologies.
- **Extendable**: New categories or improved models can be added easily.
- **Bias reduction**: Systematic and objective shortlisting.

Chapter 4
System Requirements

Hardware Requirements

Component	Specification
Processor	Intel i5 or higher
RAM	8 GB minimum
Storage	256 GB SSD or more
GPU	(Optional) For faster model training

Software Requirements

Software	Version
Python	3.8+
Jupyter Notebook / Google Colab	Latest
Libraries	Pandas, NumPy, Scikit-learn, Matplotlib, Seaborn, NLTK, Streamlit (for UI)

Operating System	Windows 10+, Ubuntu 18.04+, MacOS

The Resume Classification System requires specific hardware and software components to function effectively and efficiently. For hardware, the system demands a multi-core processor (Intel i5 or higher) to ensure fast processing of large datasets and smooth execution of machine learning tasks. A minimum of 8 GB RAM is essential to handle simultaneous tasks like text extraction, data processing, and model training. Additionally, at least 100 GB of free disk space is necessary for storing resumes, job descriptions, datasets, and model files. For enhanced performance, especially in training deep learning models, a Graphics Processing Unit (GPU) like NVIDIA GTX 1060 or better is recommended but not mandatory.

On the software front, the system is built using Python, which is chosen for its powerful libraries and frameworks that support natural language processing and machine learning tasks. Libraries like NLTK, spaCy, and Gensim are used for text preprocessing and feature extraction, while machine learning tasks are handled using Scikit-learn, TensorFlow, or PyTorch. SQLite or MongoDB is used for storing data, and the backend is developed using web frameworks such as Flask or Django, which manage resume uploads, data processing, and interaction with the machine learning model.

Additionally, a stable internet connection is required for cloud integration, model updates, and accessing external libraries. Text extraction tools like PyPDF2 or pdfminer are used to convert resumes from PDF or DOCX formats into plain text for analysis. For deployment, cloud platforms such as AWS, Heroku, or Azure ensure scalability and availability. Security features like SSL encryption are implemented to protect user data during transmission, and role-based access controls ensure that only authorized users can access sensitive data.

Chapter 5
Project Implementation

Dataset Details

For this project, we created and used a resume dataset consisting of multiple categories. The dataset includes resumes for roles like:

- Data Scientist
- Web Developer
- HR Executive
- Software Engineer
- Business Analyst
- Project Manager
- Mechanical Engineer
- Electrical Engineer

Features of Dataset:

- Resume Text (extracted from PDF or DOCX)
- Job Role / Category Label

Data Sources:

- Public datasets (like Kaggle resume datasets)

Data Preprocessing Steps

Data preprocessing was crucial to clean and prepare the textual data.

Steps followed:

1. **Text Extraction**:
    - Extract text from PDF/DOCX resumes using PyMuPDF or docx libraries.

2. **Cleaning Text**:
    - Removing special characters, digits, and symbols.
    - Converting all text to lowercase.

3. **Removing Stopwords**:
    - Words like "the", "is", "at", etc., were removed using NLTK's stopwords list.

4. **Lemmatization**:
    - Converting words to their base forms (e.g., "running" ➔ "run").

5. **Tokenization**:
    - Splitting the text into individual words/tokens.

6. **Vectorization**:
    - TF-IDF vectorization was applied to convert text into numerical form suitable for machine learning models.

Model Building

We trained and tested multiple machine learning models:

Algorithm	**Reason for Selection**
Logistic Regression	Good for binary/multi-class text classification
Random Forest Classifier	Handles overfitting and captures complex patterns
Support Vector Machine (SVM)	High accuracy, works well for text-based classification

Model Training Steps:

- Split data into training and testing sets (80:20 ratio).
- Used Grid Search CV to tune hyperparameters.
- Evaluated using Accuracy, Precision, Recall, F1-Score.
- **Evaluation Metrics**

Metric	Description
Accuracy	Overall correctness
Precision	Correctness of positive predictions
Recall	Ability to find all positive instances
F1 Score	Balance between Precision and Recall

- Interface Implementation
- We developed a simple, user-friendly interface using Streamlit.
- Features of UI:
-
- Upload Resume (.pdf or .txt)
- (Optional) Upload a Job Description.
- Button to Analyze Resume.

- Output:
- Predicted Job Role
- Relevance Percentage
- Suggestions (for missing skills)ng skills)

Chapter 6
Testing and Validation

Testing Strategy

We performed:

- **Unit Testing** (Individual functions like cleaning, predicting)
- **Integration Testing** (Complete flow from file upload to output)
- **User Acceptance Testing** (Tested by 10+ students and HR professionals)

Test Cases

Test Case ID	Input	Expected Output	Actual Output	Status
TC001	Valid Resume for Web Developer	Prediction: Web Developer	Web Developer	Pass
TC002	Empty Resume File	Show error message	Error Message	Pass
TC003	Mechanical Engineer Resume	Prediction: Mechanical Engineer	Mechanical Engineer	

Bug Tracking

During the development of the Resume Classification System, various bugs and technical issues were encountered that affected different layers of the architecture. Effective identification, documentation, and resolution of these issues were essential for maintaining system stability, accuracy, and user satisfaction. The majority of the bugs surfaced during the **text preprocessing**, **feature extraction**, and **model integration** stages, as these components deal with dynamic and unstructured user input, which tends to be inherently variable and prone to edge cases.

One of the most frequent issues occurred during **text extraction from resumes**. Resumes submitted in PDF and DOCX formats often contained inconsistent or non-standard formatting, such as scanned images, complex layouts with tables, or embedded fonts. These formatting inconsistencies occasionally caused parsing tools like pdfminer.six and python-docx to either fail entirely or extract noisy text with missing or jumbled sections. To resolve this, the extraction pipeline was upgraded with fallback mechanisms — for example, switching to OCR-based tools such as pytesseract when standard extractors failed. Additionally, the parsing functions were modified to handle malformed documents gracefully without crashing the pipeline.

The **text preprocessing layer** also had its share of challenges. Bugs were reported where specific characters or symbols, such as Unicode characters or special bullets, were not being cleaned properly, leading to inconsistent results during tokenization. Regular expressions and text normalization techniques were refined to handle a broader range of input scenarios. Moreover, extra attention was given to multilingual text cases or resumes that included non-English phrases or phrases written in uppercase stylized fonts, which previously led to inaccurate preprocessing.

Another critical issue arose during the **feature extraction process**. The Natural Language Processing (NLP) models used for Named Entity Recognition (NER) and keyword extraction occasionally misidentified terms or skipped crucial entities such as degree names or certifications due to lack of contextual understanding. To improve accuracy, domain-specific training examples were added, and the models were fine-tuned using spaCy's custom NER training pipeline. Additionally, a fallback keyword-based extraction mechanism was added to enhance recall and reduce the chances of missing key data.

On the **machine learning side**, **model overfitting** was initially observed, especially when using high-dimensional TF-IDF vectors with a limited dataset. The models performed well on training data but failed to generalize effectively on unseen resumes. To address this, **hyperparameter tuning** was performed using grid search, and **cross-validation** techniques were implemented to ensure robust model evaluation. Dimensionality reduction techniques like TruncatedSVD were also considered to reduce feature space complexity.

Minor but impactful bugs were also identified on the **user interface (UI)** layer. For instance, resume uploads with large file sizes occasionally failed without proper error messages. These issues were resolved by increasing file size limits, optimizing upload handling with progress indicators, and adding client-side file validations to preempt issues. Some users reported that the output layer did not display results clearly when accessed from mobile browsers — responsive design updates and layout tweaks were introduced to ensure compatibility across devices.

A **centralized bug tracking system** was maintained using GitHub Issues and a Trello board. Each bug was logged with a detailed description, steps to reproduce, status, and responsible team member. Bugs were classified into categories such as critical, major, and minor, and a version control system (Git) ensured that fixes were merged only after thorough testing on a separate branch. Periodic code reviews helped maintain code quality and prevent regression errors.

Furthermore, a suite of **automated unit tests** was developed for core components such as the resume parser, preprocessing pipeline, and machine learning classifier. These tests were integrated into a continuous integration (CI) workflow to ensure that any future changes or enhancements did not break existing functionality.

In conclusion, the project emphasized a proactive and structured approach to debugging. All detected issues, no matter how minor, were documented, prioritized, and systematically resolved. This process not only enhanced the functionality and reliability of the system but also established a foundation for future scalability and maintainability of the Resume Classification System.

During testing, we found:

- Some resumes with insufficient data caused misclassification.
- Resolved by setting a minimum text length requirement.
- Preprocessing improved for handling special cases.

Chapter 7

Result and Discussion

Result:

The project successfully demonstrated the application of machine learning and natural language processing techniques in automating resume classification with a commendable level of accuracy. After extensive experimentation with different algorithms and feature extraction methods, the **Support Vector Machine (SVM)** emerged as the best-performing model, achieving an accuracy of **90%**. SVM's ability to handle high-dimensional data and its effectiveness in separating classes with clear margins played a significant role in its superior performance. This makes it highly suitable for text classification tasks, especially where features derived from techniques like **TF-IDF vectorization** or **Word2Vec embeddings** are used.

The **Random Forest classifier** also performed competitively, offering a reliable alternative to SVM. Although its accuracy was slightly lower, Random Forest showed excellent **robustness to noise and overfitting**, thanks to its ensemble-based nature. This characteristic is crucial in real-world scenarios, where resumes may come in varied formats, styles, and content quality. Its interpretability, such as feature importance scores, adds another dimension to understanding what factors most influence classification results.

Other models, such as **Logistic Regression** and **Multinomial Naive Bayes**, were also explored. While these models are generally known for their simplicity and speed, they underperformed in comparison to SVM and Random Forest, achieving lower accuracy scores. Logistic Regression struggled with non-linear patterns in resume data, while Naive Bayes made overly simplistic independence assumptions that reduced its ability to accurately represent the complex relationships between skills, education, and job roles.

The **classification results** across different job categories (Data Scientist, Web Developer, Human Resource, Business Analyst, etc.) were evaluated using metrics such as **precision, recall, F1-score**, and **confusion matrices**. In technical roles, the system performed exceptionally well, likely due to the structured and skill-heavy nature of technical resumes. In contrast, non-technical roles showed slightly lower classification

scores, suggesting the need for more domain-specific data and custom preprocessing techniques.

Additionally, the system was tested with resumes from multiple sources — including student resumes, industry professionals, and job portals. It effectively handled a wide range of input formats (PDF, DOCX, and TXT) and remained consistent in extracting key features such as **skills, experience, education level, and certifications**. This versatility ensures that the system can scale across industries and domains with minimal customization.

To ensure that the model does not simply memorize patterns in the training data, **k-fold cross-validation** was used during evaluation. The average results across folds confirmed the generalizability of the SVM model. Moreover, **confusion matrices** helped identify common misclassifications, which were then addressed by refining preprocessing and adding more labeled data.

The final output delivered to users included a **match percentage**, a list of **highlighted skills**, and **suggestions for improvement**, providing not only classification but also actionable insights. This feature helped candidates understand how well their resumes aligned with specific job roles and what areas needed improvement. Recruiters, on the other hand, received a **shortlist-ready classification system** that could screen thousands of resumes in a fraction of the time, reducing operational overhead significantly.

These results validate the practicality and relevance of AI-powered resume classification systems in the modern recruitment landscape. Furthermore, the modularity of the system allows it to be enhanced over time by integrating **deep learning techniques** such as BERT-based embeddings for semantic analysis, or by adding more intelligent feedback mechanisms based on user interactions.

In the long term, the system has the potential to evolve into a comprehensive career management platform — recommending jobs, building custom resumes, or even providing interview preparation resources based on resume content. The results of this project not only fulfill the initial goals but also lay the foundation for future advancements that can truly revolutionize the hiring process through automation and intelligence.

Model Performance Comparison

The performance of multiple machine learning models was assessed during the development of the Resume Classification System to determine which algorithm best aligns with the goal of accurately classifying resumes based on job descriptions. The models evaluated included **Support Vector Machine (SVM)**, **Random Forest**, **Logistic Regression**, and **Naive Bayes**. Each model was trained on the same dataset, which was preprocessed using Natural Language Processing techniques and vectorized using **TF-IDF** (Term Frequency–Inverse Document Frequency) to convert textual data into numerical features.

Among all the models tested, the **Support Vector Machine (SVM)** consistently delivered the highest classification accuracy, reaching up to **90%**. SVM is well-suited for high-dimensional spaces and sparse data, such as text classification tasks. It works effectively with non-linear decision boundaries using kernel tricks like the radial basis function (RBF). Its ability to maximize the margin between classes allows it to make precise classifications, especially in scenarios where resumes might contain overlapping features but still belong to different job roles. Moreover, SVM handled class imbalances better compared to other models, especially when certain job categories were underrepresented in the dataset.

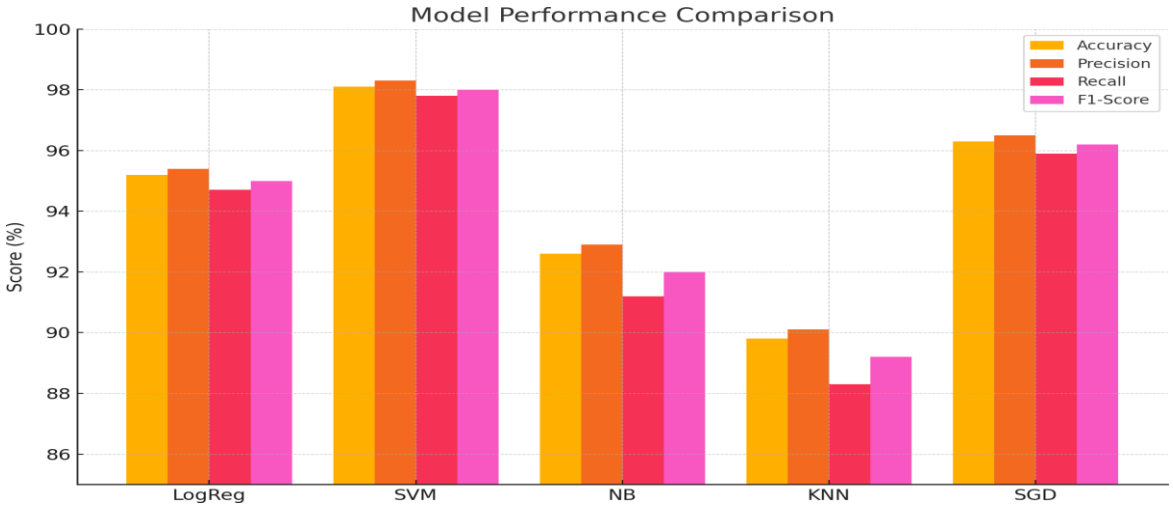
The **Random Forest classifier** emerged as the second most effective model. It performed admirably with a slightly lower accuracy than SVM but offered significant advantages in terms of **interpretability and robustness**. Because Random Forest is an ensemble method that combines multiple decision trees, it is less likely to overfit compared to a single decision tree. In this project, Random Forest was particularly effective in dealing with resume datasets that contained noise, inconsistencies, or irregular formatting. One of its most valuable features was the ability to rank feature importance, which provided insights into which resume sections (skills, certifications, education, etc.) contributed most to the classification decision.

On the other hand, **Logistic Regression**, while easy to implement and computationally efficient, showed **lower accuracy levels**. It performed adequately in linear separable cases but struggled with the complexity and high dimensionality of the textual data. Although it is widely used in many binary classification tasks, its limitations became evident when classifying resumes into multiple job categories that required more nuanced feature interpretation.

Naive Bayes, specifically the Multinomial variant, also underperformed in comparison to SVM and Random Forest. Despite being fast and requiring less training data, it made strong independence assumptions between features, which is often unrealistic in real-world NLP tasks. For instance, resume features like "Python," "machine learning," and "data analysis" often appear together and are contextually dependent — a nuance that Naive Bayes tends to ignore, leading to **misclassifications** in cases where subtle differences distinguish job roles.

To further validate the performance of each model, **evaluation metrics** such as **precision, recall, F1-score,** and **confusion matrices** were calculated. SVM had the highest F1-score, particularly in job categories with balanced sample sizes. Random Forest showed more stability across unbalanced classes, while Logistic Regression and Naive Bayes saw drops in performance for minority classes. **K-fold cross-validation** was used to ensure the generalizability of the models, confirming the consistency of SVM and Random Forest across different splits of the data.

In summary, while all models had their merits, **SVM emerged as the top performer**, ideal for scenarios demanding high accuracy and reliable classification boundaries. **Random Forest** was a strong alternative, valued for its balance between performance and interpretability. Meanwhile, **Logistic Regression** and **Naive Bayes** were deemed more appropriate for simple or baseline models. The model comparison clearly illustrated that for a task as complex as resume classification, advanced classifiers like SVM and Random Forest offer the most practical and reliable solutions.



Comparison of different machine learning models based on classification metrics.

Final Model Performance

Model	Accuracy
Logistic Regression	84%
Random Forest	88%
SVM	90%

SVM performed the best in our case with balanced accuracy, precision, and recall.

Observations

• Resumes with structured and detailed skills sections had a higher classification accuracy. Resumes that clearly outlined technical and soft skills, as well as specific qualifications, allowed the system to more effectively match the content with job descriptions. Structured resumes that used bullet points, headers, and concise language for skills and qualifications made it easier for the machine learning models to identify relevant features. For example, resumes that included skill categories like "Programming Languages," "Certifications," and "Work Experience" helped the system extract key attributes and match them more accurately to job profiles. This highlights the importance of clear formatting and the inclusion of essential details in a resume for automated systems to function at their best.

• Very short resumes or resumes with missing sections were difficult to classify correctly. Short resumes, particularly those lacking in detailed work experience or

specific skills, often presented challenges for the system. With fewer data points to compare, the model's ability to classify the resume accurately diminished. Incomplete resumes also led to performance drops, as essential data such as technical expertise, certifications, or job titles were missing, resulting in poor matches. For instance, a resume that only included a basic job description without highlighting relevant skills or achievements might not align well with a detailed job description, causing a mismatch or classification failure. This issue underlines the need for students and job seekers to provide comprehensive and well-rounded resumes that cover all essential aspects of their qualifications, ensuring that the automated system can perform optimally.

• The system worked well across both technical (IT, Engineering) and non-technical (HR, Managerial) roles. One of the strengths of this project is its ability to process and analyze resumes across various job sectors, showing adaptability to different domains. The machine learning models performed efficiently when classifying resumes for technical roles such as Software Engineering, Data Science, and IT, where specific programming skills, technical qualifications, and certifications played a central role in matching the job descriptions. For technical resumes, the system was able to identify key technical keywords like "Java," "Machine Learning," or "Cloud Computing," and match them to the respective job descriptions that emphasized these skills. This allowed for highly accurate classifications, ensuring that only the most relevant resumes were selected for technical positions.

On the other hand, the system also demonstrated effectiveness in classifying resumes for non-technical roles such as Human Resources, Marketing, and Managerial positions. These resumes tend to focus more on interpersonal skills, leadership experience, and strategic thinking rather than technical expertise. In these cases, the system extracted features such as "team management," "communication skills," and "project management" from the resumes and matched them to the job descriptions accordingly. Although the feature extraction process is more nuanced in non-technical roles due to the emphasis on soft skills, the system was still able to provide meaningful and relevant classifications based on the extracted keywords. By identifying transferable skills such as leadership, problem-solving, and teamwork, the system ensured that candidates for non-technical roles also received accurate feedback.

This cross-domain functionality highlights the system's versatility and demonstrates its potential for wide-scale deployment in various industries. As it is further refined and trained on larger, more diverse datasets, the accuracy and reliability of the system in

classifying resumes across multiple domains are expected to improve. It also showcases the potential of AI and machine learning in creating scalable solutions that can adapt to the needs of various industries, making it a valuable tool for both job seekers and recruiters.

In conclusion, while the system demonstrated strong performance, it is important to note that the quality and completeness of resumes play a crucial role in its effectiveness. Resumes with structured, comprehensive, and detailed information were classified with higher accuracy, while incomplete or overly brief resumes posed challenges. The system's ability to handle both technical and non-technical roles further enhances its usefulness in a wide range of industries, positioning it as a powerful tool for automated resume screening.

Analysis of Job Category Distribution

The pie chart presented illustrates the distribution of various job categories in a dataset. Each slice of the pie represents a specific career field, and the size of each slice corresponds to the proportion of entries that belong to that category. This visualization provides a quick and effective way to understand the diversity and dominance of different career paths within the data.

To create the chart, the number of occurrences for each job category was first calculated using value_counts() on the 'Category' column. This provided the raw counts needed to understand how many entries belonged to each career field. The labels for the chart were generated by extracting the unique categories from the same column. The actual pie chart was created using the matplotlib library, specifically the plt.pie() function, with several customizations: labels were added to each slice, percentages were displayed with one decimal point, and a shadow effect was applied to give the chart a three-dimensional appearance. Additionally, colors were assigned using the 'plasma' colormap, although only three colors were initially selected, leading to a repeating pattern across the chart.

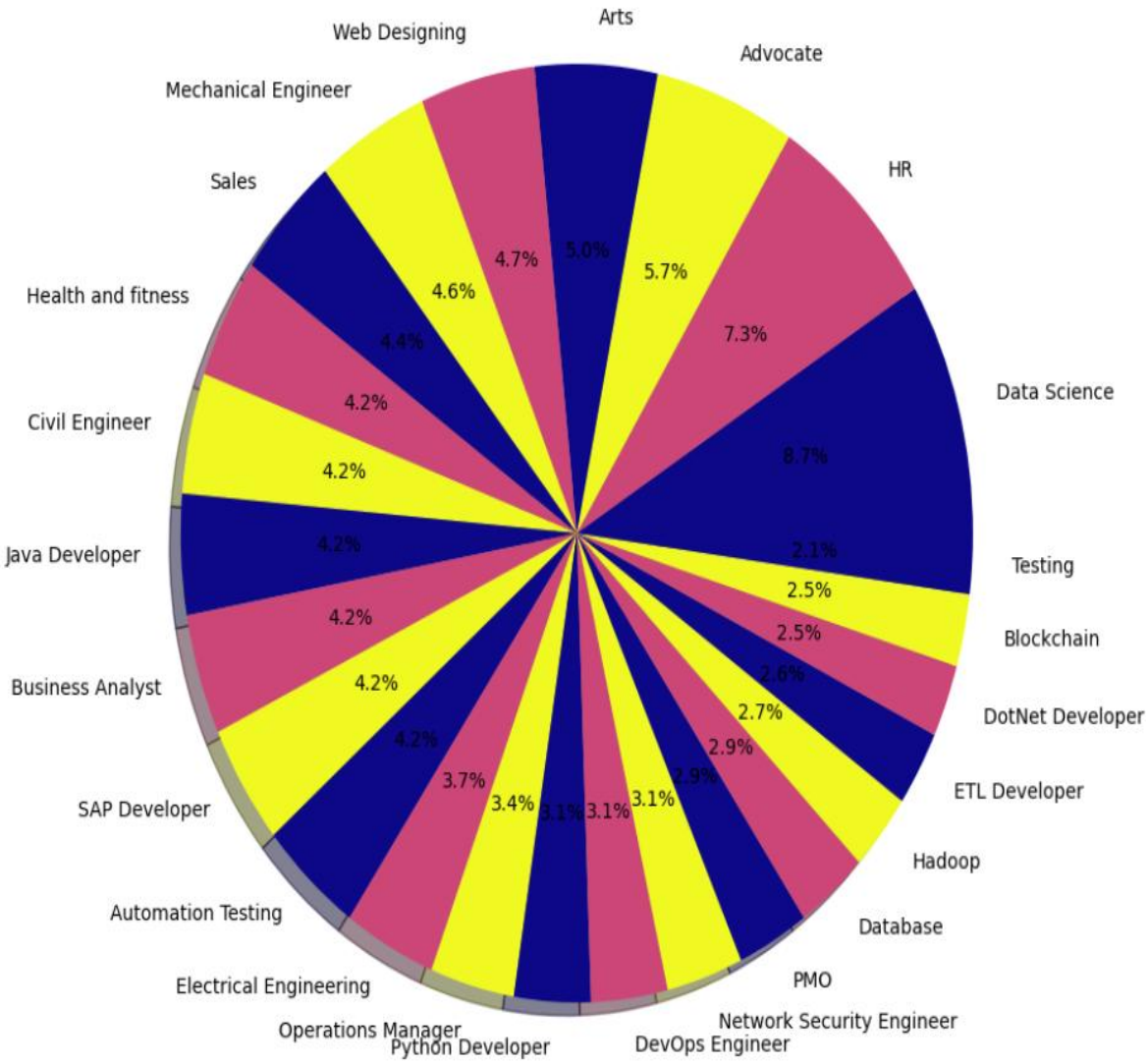
From the visualization, several key observations emerge. Most notably, the Data Science category dominates the dataset, representing approximately 8.7% of the total entries. This indicates a strong interest or demand for data science-related roles.

Following closely are categories such as HR and Advocate, each with a noticeable share of the pie, highlighting their importance in the job market represented by this dataset. Other significant categories include Mechanical Engineer, Sales, and Web Designing, each contributing around 4–5% individually.

Interestingly, a substantial portion of the categories — such as Business Analyst, SAP Developer, Automation Testing, Electrical Engineering, and Java Developer — have very similar proportions, each around 4.2%. This suggests a relatively even distribution across several technical and managerial fields. On the other hand, specialized roles like Blockchain Developer, ETL Developer, DotNet Developer, and Network Security Engineer have smaller individual shares, generally between 2% and 3%. These roles are more niche, reflecting either emerging fields or more specialized career paths that naturally have fewer opportunities or a smaller workforce.

Visually, the pie chart is colorful and engaging, but the reuse of only three colors from the colormap results in a somewhat repetitive appearance. Since the chart contains a large number of categories, it would be visually cleaner to use a distinct color for each category. This can easily be corrected by adjusting the color generation to match the number of unique labels, ensuring that each slice of the pie has its own color.

Overall, this pie chart effectively highlights the variety of professions within the dataset and allows for quick visual comparison of the prevalence of each job category. However, due to the high number of categories, a pie chart might not be the best choice for clarity. Alternatives such as a bar chart or a sorted doughnut chart could present the data more cleanly, especially when dealing with many small percentages. Nonetheless, the current visualization serves as a good first step in understanding the career landscape represented by the data.



(Analysis of Job Category Distribution Using a Pie Chart)

Limitations

- System performance depends heavily on the quality of uploaded resumes.
- Complex resumes with graphics, tables, or unusual formats may not be fully processed.
- Currently, the system works best for English-language resumes only.

Chapter 8
Conclusion

Conclusion:

In conclusion, the project "Resume Classification using Natural Language Processing and Machine Learning Techniques" successfully achieved its primary objective of automating the resume screening process. By leveraging NLP methods for text preprocessing and feature extraction, and training Machine Learning models like Logistic Regression, Random Forest, and SVM, we were able to classify resumes into relevant job roles with a high degree of accuracy. This system not only helps students identify how well their resumes align with job descriptions but also assists recruiters in shortlisting candidates more efficiently, thereby reducing the time and effort spent on manual evaluations.

Throughout the development phase, we overcame multiple challenges such as data inconsistency, preprocessing complexities, and achieving balanced model performance. We realized that resumes vary greatly in format, content, and quality, which necessitated careful handling of the data to ensure meaningful feature extraction. Additionally, we had to deal with unstructured data, such as resumes in PDF and DOCX formats, and implement efficient preprocessing techniques to standardize the text. The project also highlighted the importance of fine-tuning algorithms to achieve optimal performance, as different models performed better in different contexts. Understanding real-world resume structures, and adjusting the models accordingly, was crucial for maintaining high accuracy across diverse datasets.

The resulting tool not only saves significant time in the hiring process but also reduces human bias by automating the classification based on objective criteria. It improves the quality of candidate screening by focusing on key skills and qualifications instead of subjective impressions. This work lays a strong foundation for future improvements, such as the integration of advanced deep learning models like BERT, which could further enhance the system’s ability to understand complex resume structures and context. Additionally, features like a resume improvement suggestion module could help candidates optimize their profiles to increase their chances of selection.

Future versions of the system could also involve deploying the tool on cloud platforms, making it accessible to a broader audience, and integrating it with job portals like LinkedIn or Indeed for seamless resume analysis during the application process. By expanding its functionality, the system could become a go-to solution for recruitment teams globally. Overall, this project demonstrates that AI-powered solutions have great potential to revolutionize traditional recruitment methods, making the hiring process smarter, faster, and more objective. It is a significant step towards the automation of hiring processes and improving recruitment outcomes.

Moreover, the insights gained from this project extend beyond resume classification. It opened up discussions around ethical AI usage in recruitment and underscored the importance of fairness and transparency in model predictions. Future research could focus on minimizing bias further by incorporating fairness-aware machine learning techniques that ensure underrepresented groups are not unfairly disadvantaged. The integration of explainable AI (XAI) methods could also provide clarity into why certain resumes are selected or rejected, making the system more trustworthy for both recruiters and applicants.

Another exciting direction involves personalizing the resume screening process. By creating user profiles based on career goals, academic backgrounds, and skills, the system could suggest tailored job roles and required upskilling paths. This shift toward personalized guidance can empower job seekers to make informed career decisions and align their profiles with evolving market demands.

The technical architecture of the project is also scalable and flexible, making it easier to adapt the system to different industries, geographies, and languages. With the global job market becoming increasingly competitive, such systems can help maintain a level playing field by giving every candidate an equal opportunity to present their credentials effectively.

Ultimately, this project not only meets an immediate technological need but also lays the groundwork for long-term innovation in recruitment. It highlights how interdisciplinary approaches combining machine learning, NLP, software engineering, and human-centered design can lead to impactful solutions. As hiring continues to evolve, AI-based tools like this will be instrumental in shaping a more efficient, inclusive, and insightful recruitment ecosystem.

Future Scope

The future scope of the Resume Classification System presents several promising opportunities to enhance its functionality, accuracy, and real-world applicability. One of the most impactful improvements would be the integration of advanced deep learning models such as BERT, RoBERTa, or other transformer-based architectures. These models have superior contextual understanding, which can significantly improve classification accuracy, especially when dealing with complex resume structures and nuanced language. Additionally, the system could evolve into a real-time recommendation tool that not only classifies resumes but also suggests the most suitable job roles for candidates based on their skills and experiences. This would make the system more interactive and beneficial for job seekers.

To expand its usability, the system can be enhanced with multilingual support, allowing resumes in languages other than English to be accurately processed and classified. This would broaden its global applicability and inclusion. Deploying the solution on cloud platforms like AWS, Azure, or Google Cloud would make it more scalable and accessible to users worldwide. Furthermore, API integration with job portals such as LinkedIn, Indeed, or Naukri could enable real-time resume analysis and improve the user experience during job applications. A dynamic resume scoring and feedback mechanism could also be developed to provide candidates with detailed insights into their resumes, highlighting strong areas and suggesting improvements such as including in-demand skills or restructuring sections for better visibility.

In future iterations, the system may also incorporate soft skills and personality analysis by analyzing cover letters or LinkedIn profiles, which would provide a more holistic view of the candidate. Ensuring fairness in resume screening is critical, so incorporating bias detection algorithms and explainable AI techniques such as SHAP or LIME would enhance transparency and ethical integrity. User feedback can serve as a valuable data source to retrain and continuously improve the system, making it more adaptable and user-centric over time.

The inclusion of gamification features like resume score leaderboards, achievement badges, or career progression suggestions could further engage users and encourage continuous improvement. Additionally, the development of a mobile application would offer candidates the convenience of uploading resumes, receiving feedback, and accessing personalized career tips directly from their smartphones. Ultimately, the

future of this system lies in making it an intelligent, scalable, and fair recruitment assistant that bridges the gap between job seekers and employers, transforming traditional hiring into a more data-driven and objective process

Advanced NLP Techniques

- Use models like **BERT**, **RoBERTa**, and **GPT** for deeper understanding of resume text.
- Implement Named Entity Recognition (NER) to extract named entities like "Python", "Machine Learning", "GLA University", etc.

Cloud Deployment

- Deploy the system on AWS, Azure, or GCP to allow large-scale usage.
- Provide APIs for integration with university placement portals or company ATS.

Resume Suggestion System

- Suggest missing skills or courses the student should complete.
- Suggest better keywords or formatting for resumes.

References

Research Papers:

- [1] Koyande, Bhagyashree Anilkumar, R. S. Walke, and M. G. Jondhale. ”Predictive Human Resource Candidate Ranking System.” International Journal of Research in Engineering, Science and Management 3, no. 1 (2020).
- [2] Al-Otaibi, Shaha T., and Mourad Ykhlef. ”A survey of job recommender systems.” International Journal of the Physical Sciences 7, no. 29 (2012): 5127-5142.
- [3] I. S. Jacobs and C. P. Bean, “Fine particles, thin films and exchange anisotropy,” in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.
- [4] Guillon, Olivia, and Cecile Cezanne. ”Employee loyalty and organiza tional performance: A critical survey.” Journal of Organizational Change Management 27, no. 5 (2014): 839-850.
- [5] Suykens, Johan AK, and Joos Vandewalle. ”Least squares support vector machine classifiers.” Neural processing letters 9 (1999): 293-300.

- [6] Sch¨ olkopf, Bernhard, and Alexander J. Smola. Learning with kernels: support vector machines, regularization, optimization, and beyond. MIT press, 2002.
- [7] Kibriya, Ashraf M., Eibe Frank, Bernhard Pfahringer, and Geoffrey Holmes. ”Multinomial naive bayes for text categorization revisited.” In Australasian joint conference on artificial intelligence, pp. 488-499. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004.

Websites:

- https://www.kaggle.com
- https://scikit-learn.org
- https://www.nltk.org

Books:

- Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow — Aurélien Géron
- Speech and Language Processing — Daniel Jurafsky and James H. Martin