CV ANALYSIS AND OPTIMIZING THE RECRUITMENT PROCESS IN THE IT INDUSTRY USING MACHINE LEARNING TECHNIQUES

Project Final (Draft) Report

De Silva M. – IT20207854

Zoysa E.S. - IT20231200

Maldeniya M.M.D. – IT20203726

De Silva S.R. – IT20216900

BSc (Hons) Degree in Information Technology

Specializing in Data Science

Department of Information Technology

Sri Lanka Institute of Information Technology

Sri Lanka

September 2023

CV ANALYSIS AND OPTIMIZING THE RECRUITMENT PROCESS IN THE IT INDUSTRY USING MACHINE LEARNING TECHNIQUES

De Silva M. – IT20207854

Zoysa E.S. - IT20231200

Maldeniya M.M.D. – IT20203726

De Silva S.R. – IT20216900

The dissertation was submitted in partial fulfillment of the requirements for the

BSc (Hons) in Information Technology Specializing in Data Science

Department of Information Technology

Sri Lanka Institute of Information Technology

Sri Lanka

September 2023

DECLARATION

We declare that this is our own work, and this dissertation does not incorporate without acknowledgment any material previously submitted for a degree or diploma in any other university or Institute of higher learning, and to the best of my knowledge and belief, it does not contain any material previously published or written by another person except where the acknowledgment is made in the text. Also, we hereby grant to Sri Lanka Institute of Information Technology the non-exclusive right to reproduce and distribute my dissertation in whole or part in print, electronic, or other medium. We retain the right to use this content in whole or part in future works (such as articles or books).

Name	Student ID	Signature
De Silva M.	IT20207854	
Zoysa E.S.	IT20231200	
Maldeniya M.M.D.	IT20203726	
De Silva S.R.	IT20216900	

…… ……

Signature of Supervisor Date

(Dr. Anuradha Karunasena)

ACKNOWLEDGEMENTS

We would like to offer our sincere gratitude to everyone who supported us in making our final year research project a success. Firstly, we would like to extend our sincere gratitude to our project supervisor Dr.Anuradha Karunasena, and co-supervisor Dr. Lakmini Abeywardhane, for their guidance and the invaluable knowledge shared throughout the year to make the research successful. The success of this research would not have been possible without their support and constant guidance.

We would also like to express our gratitude to Sri Lanka Institute of Information Technology and the CDAP team for providing us with the guidance and the expertise to finish the project successfully.

Group members – 2023-098

Faculty of Computing,

Sri Lanka Institute of Information Technology.

ABSTRACT

In the rapidly changing IT industry, selecting candidates with the right technical, educational, professional, and personality attributes is increasingly challenging. This research presents an innovative recruitment approach that thoroughly assesses candidates in these aspects.

The study develops a custom RASA-AI chatbot for automated job description creation. Candidate suitability is evaluated by calculating matching percentages between resumes and job descriptions, with TF-IDF achieving high accuracy. In academic transcript analysis, Named Entity Recognition (NER) models identify module titles and grades. Skill area prediction, led by the SVM model, attains high accuracy. The research introduces a novel method for measuring programming language proficiency through GitHub data analysis and compares candidates using a knowledge graph. LinkedIn skill data predicts job categories via an SVM multiclass classification machine learning model, and Referee feedback is analyzed for improved recommendations. The study also assesses candidates' Big Five personality traits and their alignment with job roles, with the XGBoost model excelling in predicting personality clusters.

In summary, this research offers a comprehensive and innovative candidate recruitment approach, leveraging advanced machine learning and natural language processing techniques to address the evolving challenges of IT industry recruitment, ultimately enhancing efficiency and accuracy for organizations in this dynamic sector.

Keywords *– Recruitment, Academic transcript, Big Five traits, skills, Job description*

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

NLP	Natural Language Processing
FWHR	Facial width-to-height ratio
PPM	Personality Prediction Model
BOW	Bag of Words
BERT	Bidirectional Encoder Representations from
DM	DM Data Mining
API	Application Programming Interface
ML	Machine learning
HR	Human Resource
PLP	Programming Language Proficiency
IT	Information Technology
CV	Curriculum Vitae
NER	Name Entity Recognition
SVM	Support Vector Machine
TF-IDF	Term Frequency - Inverse Document Frequency
TSRM	TSRM Technical Skills Recognition Model
SMOTE	Synthetic Minority Oversampling Technique
RE	Regular Expressions
OCR	Optical Character Recognition

1. INTRODUCTION

1.1. Background

Recruitment plays a vital role in any organization's quest to bring in the right talent to meet its objectives. However, in today's IT industry, the traditional recruitment process often appears time-consuming, inefficient, and costly. Tasks like reviewing resumes and academic transcripts, alongside lengthy interviews to gauge skills, can lead to a waste of resources and even the hiring of candidates who may not be the best fit.

Moreover, a significant challenge lies in the absence of a well-defined approach to evaluate critical aspects like technical expertise and personality traits in candidates and their alignment with job requirements. This underscores the need for a more streamlined and systematic recruitment approach that considers these essential factors, ultimately leading to more efficient and effective hiring processes.

To address these issues, many organizations have embraced automated recruitment systems powered by machine learning, data extraction, and natural language processing. These systems aim to accelerate recruitment, enhance accuracy, and reduce costs. For instance, according to data from the Sri Lanka Information and Communication Technology Agency (ICTA) [1], the country had set an ambitious target of employing 200,000 individuals in the IT sector alone by 2022. This underscores the urgency for a recruitment system that can facilitate faster and more cost-effective hiring practices, especially for IT roles.

This research project centers on the criteria involved in selecting new candidates and aims to develop an automated system capable of comprehensively assessing candidates across all these criteria. The ultimate goal is to identify and recruit the most qualified candidates for specific job positions. In essence, this research seeks to revamp and optimize the recruitment process, making it more attuned to the demands of the modern IT industry.

1.2. Literature Survey

In recent years, numerous studies have focused on exploring methods to go beyond evaluating only CVs and academic skills during candidate recruitment, with the aim of optimizing the overall recruitment process. The research paper titled "DevFlair: A Framework for Streamlining the Pre-screening Process in Software Engineering Job Applications" authored by Jayasekara et al. [2] presents a novel system designed to automate the evaluation of job candidates' suitability. The system leverages data from various sources such as social media, GitHub, and open-ended questionnaires to predict the compatibility of candidates for specific job positions. In this study, the authors use a Personality Prediction Model (PPM) to forecast the Big Five personality traits of candidates. The PPM utilizes data extracted from the candidates' LinkedIn profiles through the ProxyCurl LinkedIn API. This data is processed and fed into a predictor to receive a compound probability score, which serves as the final output representing the candidate's personality.

A study done in 2020 titled “Candidate Selection for the Interview using GitHub Profile and User Analysis for the Position of Software Engineer” [3] focuses on candidate selection for an interview for the position of Software engineer, the analysis of the academic transcript played a pivotal role in the candidate selection process for software engineer positions. The researchers recognized the importance of assessing both technical skills and soft skills derived from the academic curriculum. To achieve this, they categorized the modules within the academic transcript into these skill categories. By doing so, they gained a comprehensive understanding of the candidate's capabilities in technical areas, such as programming, as well as their proficiency in soft skills like communication and problem-solving. Furthermore, the researchers devised a grading criterion to evaluate the candidate's performance in each module. Candidates with grades above C+ were deemed eligible for further consideration, as this threshold indicated a

satisfactory level of achievement and competence in the respective subject matter. On the other hand, candidates with grades below this threshold were not considered suitable for the job role. This research also proposes a screening process that utilizes phone call transcripts containing candidate responses to open-ended questions to assess their personality traits. The study uses feature extraction techniques to identify the Big Five traits of candidates from the responses. Multiple supervised learning classification algorithms are utilized to evaluate the model, and it is found that Logistic regression achieves the highest level of accuracy. Additionally, this research has proposed a pre-screening solution to screen the applicants for the position of Software Engineer where candidates are screen-based GitHub profile data along with CV data, matching the total number of LinkedIn profile skills with job position with academic transcripts and, Recommendation letters. For recommendation letters they have applied a multi-class classification machine learning approach to a candidate as 2 recommend, highly recommend, neutral, and weak.

Extensive research has been conducted in the field of candidate selection processes, aiming to identify the most qualified candidate from a pool of applicants through a comprehensive assessment of their skills and qualifications. In a study conducted by Ransing et al.[4] , advanced machine learning techniques were employed to conduct the research. The primary objective of this study was to cater to the specific requirements of recruiters by effectively screening resumes based on their predefined criteria. The data utilized in this study was in the form of a CSV file, with carefully assigned labels corresponding to various roles within the IT industry. A sophisticated stacked classification approach was implemented to systematically rank resumes based on their suitability for different job roles. As a result of this comprehensive research effort, the study successfully obtained a good understanding of the relevance of specific job roles in the IT industry. The findings are more focused on crucial insights that can significantly aid recruiters in making well-informed decisions when evaluating applicants' resumes. According to Muntaha Mehboob [5], the research utilized a novel approach to extract relevant information from resumes and effectively organize it into distinct categories, such

as education, experience, and talents, based on values. This specialized solution was specifically designed to tackle the challenges of unstructured CVs, which often exhibited varying formats and structures. The researchers developed a sophisticated tool capable of automatically shortlisting and ranking candidates based on their job profiles. By employing cosine similarity, the tool efficiently evaluated each resume against the corresponding job description. The implementation of this novel method held considerable potential in enhancing the recruitment process, saving valuable time, and ultimately improving the quality of candidate selection for diverse job positions. Yan Wang et al. [6] implemented a search engine for recruitment and staffing, employing knowledge graphs and Bidirectional Encoder Representations from Transformers (BERT). In this research, the utilization of the Machine Learning Model (MLM) and Named Entity Recognition (NER) of pre-trained BERT allowed for the identification of competence keywords from the corpus of CVs and Natural Language Processing (NLP). The knowledge graph, composed of Concept Map (CMAP) and competency keywords, provided effective recommendations in the neighborhood domain, yielding positive outcomes. The authors of the paper titled 'Resume Classification using Elite Bag-of-Words Approach' [7] explored the use of elite keywords, for selecting significant keywords individually for each class in the context of resume classification. The method involved shortlisting class-specific keywords, removing redundancies, and concatenating them across classes to ensure that relevant class-specific keywords were included in the feature columns. The stability of the method was ensured by the entropy partitioning method. The features were trained on a random forest classifier using a grid search for hyperparameter optimization. After conducting a thorough evaluation of various models, it was found that the elite keyword subset demonstrated the highest reliability and accuracy for classifying different categories of resumes in the benchmark dataset.

Academic transcripts have been extremely important when hiring a candidate for a certain job position. Certain studies have explored the appropriateness of IT graduates for careers without relying on their academic transcripts. In the research paper titled 'Machine Learning Approach for Predicting Career Suitability, Career Progression, and Attrition of

IT Graduates' [8], the authors explored the field of Information Technology (IT) career development, focusing on Sri Lanka's IT industry. The primary aim was to help both fresh graduates entering the IT workforce and existing IT employees in achieving their career goals while addressing some critical challenges in the industry. One of the main challenges identified was that IT graduates often struggle to find the most suitable career path and skills alignment upon entering the industry. Additionally, IT employees face obstacles in career progression, leading to potential attrition issues for employers. To tackle these challenges, the researchers collected data from IT professionals in Sri Lanka through surveys. They used various machine learning classification algorithms, including XGBoost, Random Forest, Support Vector Machine, K-Nearest Neighbors, Decision Tree, and Naive Bayes, to build predictive models for career suitability, initial salary, career and salary progression, professional course recommendations, employee attrition, and remedial actions. The models' performance was evaluated using accuracy, precision, recall, and F1-Score metrics. Notably, XGBoost emerged as the top-performing algorithm for several predictions, demonstrating the potential of machine learning in addressing career-related challenges in the IT industry. This research approach effectively addressed the pressing issues in the IT sector. For fresh graduates, the study provided a valuable career mentoring system that predicts suitable job roles and initial salaries based on their skills and qualifications, offering them guidance and direction in a competitive job market. Additionally, existing IT employees benefited from predictions related to career progression, salary advancement, and professional course recommendations, aiding them in achieving their career goals. Furthermore, the study addressed employers' concerns about attrition by predicting employee attrition and suggesting remedial actions to retain valuable staff. The comprehensive methodology involved data collection, preprocessing, model training, evaluation, and the development of a web-based application to deliver these predictions. This research not only contributes to the IT industry in Sri Lanka but also demonstrates the power of data-driven solutions in overcoming career-related challenges, making it a valuable addition to the field of career development and predictive analytics.

Personality traits can be described as patterns of thought, feeling, and behaviors that are characteristic of an individual. Aligning candidate personality traits with job role requirements is important as it ensures employees possess the necessary skills for success, promotes better job fit, and boosts job satisfaction and performance. Additionally, it reduces employee turnover benefitting both the organization and the employee. Several studies have examined the relationship between personality traits and job performance consistently indicating a strong correlation between the two. Studies have found that certain personality traits are more important for some jobs than others. Stephen P. and Timothy A. [9] suggest that conscientiousness has the most significant influence on job performance. Employees with high conscientiousness tend to be knowledgeable, organized, and reliable. They are also more likely to be leaders and take on additional responsibilities. Employees who are high in neuroticism are more likely to experience stress, anxiety, and burnout. They may also be more likely to take sick leave. Extroverts are more likely to be outgoing, assertive, and enthusiastic. They are also more likely to be willing to take on leadership roles and be involved in social activities. Employees who are open to new experiences are more likely to be creative, innovative, and adaptable. They are also more likely to be supportive of change. Agreeable employees are more likely to be cooperative, helpful, and friendly. They are also more likely to be satisfied with their jobs. According to M. Aqeel Iqbal et al. [10] researchers have identified that the Big Five Model (BFM) also known as the Five-factor Model is the most prominent and trusted personality classification model. Dr. S.K. Nivetha et al. [11] conducted a study that uses the facial-width-to-height ratio (fWHR) to predict personality traits in interview candidates. The researchers utilize a Convolutional Neural Network (CNN) to extract the fWHR from candidate images. Subsequently, this fWHR measurement is used to predict a person's personality traits with an impressive accuracy rate of 92%.

Traditional hiring processes rely heavily on resumes and interviews, these methods may not be enough to fully evaluate a candidate's professional skills. It is aimed to critically analyze the existing literature on the use of LinkedIn, GitHub profiles, and Employee reference checking in evaluating professional skills during the candidate hiring process.

The paper explores the use of GitHub REST API to gather information on potential candidates from their user profiles. This research paper discusses how using novel methods to inspect public GitHub repositories can help manage the influx of candidate profiles during the recruitment process. Research conducted in 2022 [2] to automate the prescreening process of job candidates used TSUAM (Technical Skill Usage Analysis Model) which focuses on creating a scoring system to assess technical skills usage on GitHub based on the number of repositories. Another interesting research was done in 2021 [12] to predict user interest based on social network profile data using machine learning. This research paper outlines a system designed to analyze data from Facebook and Twitter by utilizing scraping tools to assess user interests and offering public opinion and gaining insights into people's sentiments. When it comes to Employee reference checking, [13], [14] mentions reference checking in the hiring process can be hugely helpful for choosing the most suitable candidates by confirming that the candidates have all the professional skills required or uncovering something disappointing. The current research on employee personality traits focuses primarily on identifying individual traits rather than assessing their suitability for specific job roles. This research aims to bridge this gap by considering the compatibility between an employee's traits and a particular job role. In recent years' competitive job market, identifying the right candidate for a job is crucial for an organization's success. While traditional hiring processes rely heavily on resumes and interviews, these methods may not be enough to fully evaluate a candidate's professional skills. In this research, we aim to critically analyze the existing literature on the use of LinkedIn, GitHub profiles, and Reference check forms in evaluating professional skills during the candidate hiring process. In the following sections, we will discuss the methodology followed in developing the proposed solution, the data collection and preprocessing procedures, the machine learning algorithms utilized, and the evaluation metrics utilized to assess the performance. We will then present and discuss the study's results, followed by the conclusion and suggestions for future research.

1.3. Research Gap

According to the literature survey done above the following issues were identified as research gaps:

- The absence of a well-defined job description structure makes it challenging to accurately rank resumes.
- There is difficulty achieving high accuracy in resume ranking due to the lack of an established method.
- There is difficulty achieving high accuracy in resume ranking due to the lack of an established method.
- The existing research in this field provides valuable insights but lacks a specific focus on individual job roles within IT as they have only focused on Software Engineer job role. This research dives deeper by analyzing academic transcripts in detail, mapping course descriptions and module outlines to identify high-level technical skills. This helps bridge the gap between academic qualifications and real-world job demands, allowing universities to better prepare students for IT careers.
- In academic transcript analysis, a novel aspect is introduced by using graphical representations and skill proficiency scores derived from academic transcripts. These visuals make data more accessible and enable quick comparisons between candidates, while skill proficiency scores provide a quantitative measure of a candidate's readiness for a specific IT job role. This innovative approach enhances the hiring process for IT positions.
- The ability to gain comprehensive insights about a candidate is constrained by the restricted tools available for extracting data from platforms like GitHub and LinkedIn, beyond the information provided in their curriculum vitae (CV).
- Many recruitment systems treat LinkedIn and GitHub separately, missing the opportunity for comprehensive candidate analysis. Integrating and analyzing data from both platforms can provide a more holistic view.

- An automated system for predicting a candidate's job category before screening would improve the efficiency of the hiring process.
- Manual reference checks are time-consuming and subject to bias. An automated system for collecting and analyzing reference feedback would save time and provide more reliable insights into candidates.
- Prior research has mainly centered around predicting how job candidates exhibit the Big Five Personality traits. However, there is a gap in research when it comes to assessing whether a candidate is a good fit for a particular job based on their personality traits. This study aims to fill this gap by not only predicting the distribution of candidates' Big Five Personality traits but also by determining how well their personalities align with the specific job they are applying for. The goal is to make hiring decisions more informed and focused on the individual's suitability for the role, adding a more human touch to the process.

1.4. Research Problem

In today's fiercely competitive IT job market, organizations and job seekers alike face a multifaceted challenge that calls for a comprehensive and innovative approach to candidate selection and assessment. The traditional methods of identifying the ideal candidates often fall short in meeting the evolving demands of employers seeking the best talent while job seekers aim to differentiate themselves in the highly competitive landscape.

organizations are increasingly turning to personality prediction methods to enhance their recruitment and selection processes. These methods hold the promise of providing valuable insights into a candidate's personality traits, facilitating better job fit. However, this promising avenue also presents the challenge of identifying and implementing accurate and tailored assessment techniques that align with the specific needs and goals of each organization. Achieving a successful personality-job fit is essential for long-term employee success and organizational performance.

The effective analysis of academic transcripts poses another significant hurdle in the candidate assessment process. These transcripts contain a wealth of information about a person's educational history, including the courses they've taken and the grades they've earned. However, they often present lengthy and complex narratives, making it crucial to determine how to extract the most pertinent insights, particularly regarding technical skills. A fair and reliable method for assessing a candidate's qualifications based on their academic record is paramount to making informed hiring decisions.

Also, this aim to reveal the true professional identity of candidates by scrutinizing their online presence on platforms like GitHub and LinkedIn. Additionally, it seeks to assess the viability of streamlining the recruitment process through automation and emerging technologies. This entails the utilization of automated tools for extracting pertinent details

from online profiles and gathering input from references to enhance the quality of hiring decisions.

In summary, the multifaceted challenge in IT candidate selection and assessment encompasses the need to leverage personality prediction methods effectively, analyze academic transcripts comprehensively, implement automated CV ranking systems accurately, and harness candidates' professional media accounts for deeper insights. These challenges require innovative solutions to optimize the hiring process, reduce bias, and ultimately secure the best-fit candidates for IT roles, all within the context of a dynamic and competitive job market.

1.5.Research Objectives

The primary objective is to develop a systematic approach that comprehensively evaluates candidates across all critical areas while simultaneously streamlining the often complex and time-consuming recruitment process. By doing so, we aim to enhance the efficiency and effectiveness of candidate selection, ensuring that the right fit for a specific job role is identified with precision. This entails incorporating innovative methods for assessing personality traits, effectively analyzing academic transcripts to extract key qualifications, implementing automated CV ranking systems for swift and accurate candidate screening, and harnessing candidates' professional media accounts to gain in-depth insights. The ultimate goal is to create a well-rounded and data-driven recruitment process that not only saves valuable time but also significantly improves the chances of securing the most suitable candidates for IT positions in today's highly competitive job market.

- The first objective is to create a highly efficient system for evaluating and ranking job applicants' resumes based on the specific job descriptions provided by recruiters. This system will leverage machine learning techniques to analyze and compare the content of resumes with the job requirement.

- Another key aspect of this research is the development of a method to assess a candidate's expertise in a particular area by analyzing their academic transcripts, particularly from a specific university. The objective here is to extract meaningful insights from these transcripts, such as the courses taken, grades earned, and specialized knowledge gained.

- This research also focuses on predicting a candidate's personality traits and evaluating their suitability for a particular job role, again employing machine learning techniques. By analyzing various aspects of a candidate's background, behavior, and responses, the system aims to provide recruiters with valuable

insights into how well a candidate's personality aligns with the demands of a specific job.

- The final objective entails the utilization of extracted candidate LinkedIn profile data through the LinkedIn REST API to summarize the skill set mentioned in user profiles. Additionally, the GitHub REST API will be employed to assess their coding proficiency, thereby providing valuable insights into their technical abilities. This holistic approach allows for a comprehensive understanding of candidates' skills through LinkedIn. Furthermore, it involves conducting effective and thorough employee reference checks. To further enhance candidate evaluation, machine learning techniques will be used to predict the ideal job category for each candidate based on their skill sets.

This research aims to enhance the entire recruitment process by efficiently evaluating and ranking resumes, assessing expertise through academic transcripts, predicting personality traits for job suitability, and extracting professional skills and preferences from candidates' digital media profiles. These innovative approaches leverage machine learning techniques and data analysis to provide recruiters with a more comprehensive and data-driven basis for making informed hiring decisions in the competitive job market.

2. METHODOLOGY

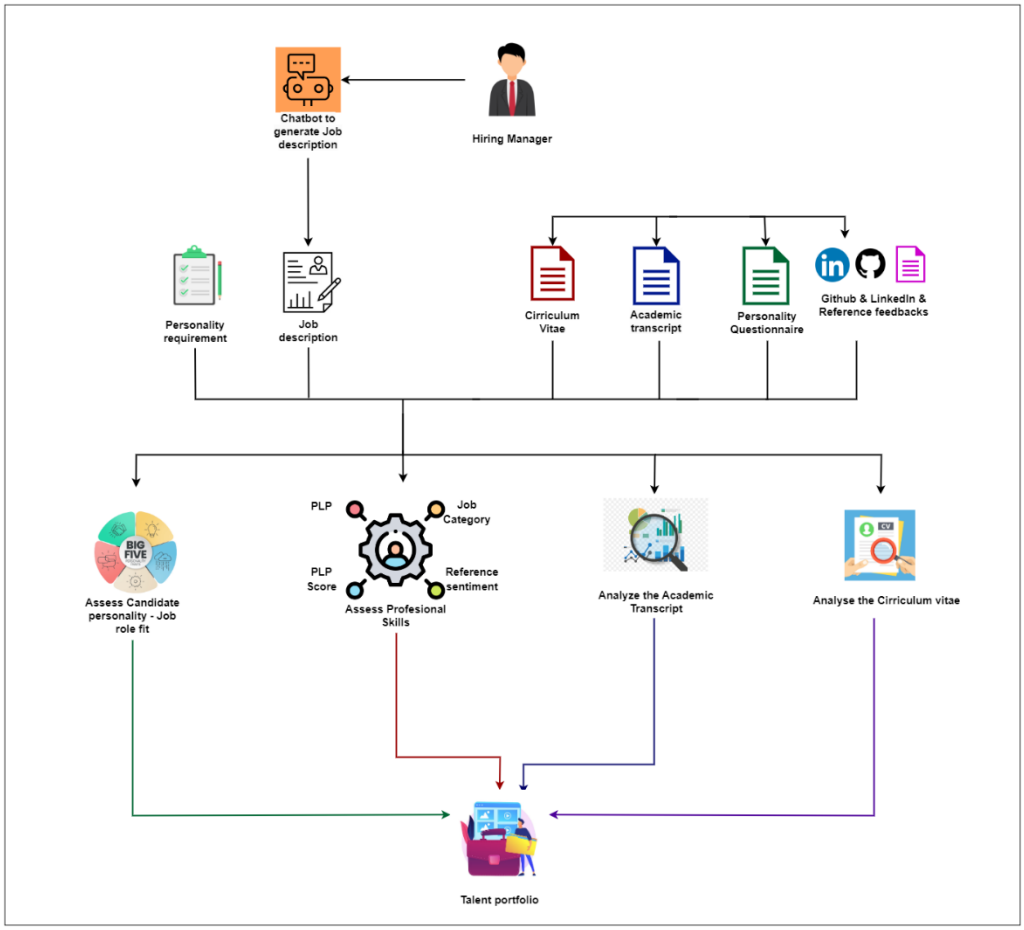


Figure 1: System architecture diagram

Figure 1 above illustrates the architectural diagram of our proposed approach, which comprises four crucial components aimed at comprehensively evaluating a candidate's suitability for organizational success and alignment with an ideal job role. These four key components encompass:

- Analyzing and aligning the candidate's CV with the job description.
- Examining academic transcripts to identify the candidate's skill areas.
- Evaluating the candidate's professional skills by aggregating data from GitHub, LinkedIn, and employee reference checking.

- Assessing a candidate's distribution of the Big Five Traits and determining their compatibility with the personality traits required for a specific job role.

2.1. CV Ranking and Job description Matching

This component mainly consists of three subcomponents.Firstly, an AI chatbot generates a job description prompting questions to the user and secondly, it computes matching percentages by comparing the job description with resumes. Lastly, it provides a summary representation of the skillproficiency of each candidate.

2.1.1. Architecture of Chatbot

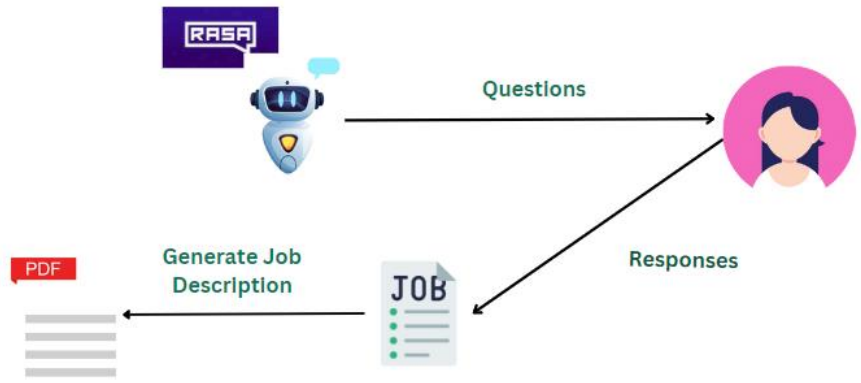


Figure 2: Architecture of chatbot job description

During the job description chatbot's development, we used intent classification to train the Rasa chatbot for accurate user message understanding. As the chatbot started responding, these clear intents, supported by practical examples, played a crucial role in shaping the bot's answers. This methodical use of intents and examples provided a structured foundation that directed the chatbot's interactions, ensuring accurate and relevant

responses, especially for job-related questions and sharing information. This approach emphasized the chatbot's capacity to provide customized and contextually relevant answers, enhancing the overall user experience.

relevant answers, enhancing the overall user experience.

```
- intent: request_job_description
  examples: |
    - i want to create a job description
    - give job description
    - job description
    - how to create a job description
    - make job description
    - job
    - description

- intent: again
  examples: |
    - Start over
    - Begin again
    - Restart
    - Refresh
    - Run it one more time
```

Figure 3 : Intents of JD Bot

To guide the chatbot through its conversation flow and obtain the necessary responses, stories were created. In the use case of developing a job description chatbot, the aim was to prompt specific questions and capture the responses efficiently, storing them in designated slots for future reference. The actions in Rasa were implemented to both save the responses and facilitate the generation of a PDF.

```
slot_values = [
    first_name,
    last_name,
    job_title,
    job_summery,
    job_Responsibilities,
    job_Qualifications,
    job_Education,
    job_Employment_Type,
    job_Work_Schedule,
    job_Location,
    job_Salary,
    job_Application_Process,
]

pdf_filename = "output.pdf"
doc = SimpleDocTemplate(pdf_filename, pagesize=letter)
styles = getSampleStyleSheet()
story = []
for slot_name, slot_value in zip(tracker.slots.keys(), slot_values):
    story.append(Paragraph(f"{slot_name}: {slot_value}", styles['Normal']))
doc.build(story)

dispatcher.utter_message(text=f"I've created a PDF with the slot values. ")
```

Figure 4: PDF generation of JD bot

2.1.2. Predicting matching percentage of resume and job description

Dataset collection

A dataset [10] containing 90 resumes along with job descriptions and matching percentages was obtained from Kaggle for the model-building process. The resumes were originally in PDF forms and were related to IT industry job positions. For matching purposes, the generated job description is used.

Data Extraction of Resumes

Since all the resumes and job descriptions are in PDF format, we utilized a PyPDF2 parser to extract the data.

```
def pdf2Text(filename):
    ''' load pdf and return the text'''
    text = ''
    # open the pdf file
    with open(filename, 'rb') as file:
        # create a PDF reader object
        reader = PyPDF2.PdfReader(file)

        # loop over each page
        for page in reader.pages:
            # extract text from the page and concat
            text += page.extract_text()

    # return all texts
    return text
```

Figure 5: Data extraction from resumes and JD

Matching resumes with job descriptions

This component is focused on matching the extracted resume data and job description. Here the final score will be the matching percentage between resume and job description.

The dataset is initially cleaned to remove any noise.

```
# convert education degrees like B.Tech or BTech to a specified form
x = re.sub(r"\s+b[.]?[ ]?tech[(. /]{1}", " btech bachelor of technology ", x)
x = re.sub(r"\s+m[.]?[ ]?tech[(. ]{1}", " mtech master of technology ", x)
x = re.sub(r"\s+b[.]?[ ]?a[(. ]{1}", " ba bachelor of arts ", x)
x = re.sub(r"\s+m[.]?[ ]?a[(. ]{1}", " ma master of arts ", x)
x = re.sub(r"\s+b[.]?[ ]?sc[(. ]{1}", " bsc bachelor of science ", x)
x = re.sub(r"\s+m[.]?[ ]?sc[(. ]{1}", " msc master of science ", x)
x = re.sub(r"\s+b[.]?[ ]?e[(. ]{1}", " beng bachelor of engineering ", x)
x = re.sub(r"\s+m[.]?[ ]?e[(. ]{1}", " meng master of engineering ", x)
x = re.sub(r"\s+b[.]?[ ]?c[.]?[ ]?a[(. ]{1}", " bca bachelor of computer applications ", x)
x = re.sub(r"\s+m[.]?[ ]?c[.]?[ ]?a[(. ]{1}", " mca master of computer applications ", x)
x = re.sub(r"\s+b[.]?[ ]?b[.]?[ ]?a[(. ]{1}", " bba bachelor of business administration ", x)
x = re.sub(r"\s+m[.]?[ ]?b[.]?[ ]?a[(. ]{1}", " mba master of business administration ", x)
```

Figure 6:Cleaning resume data

Next, concatenation was implemented.

```
def concat(s):
    '''Concatenate words like "D A T A  S C I E N C E" to get "DATA SCIENCE"'''

    # add spaces at both end for better processing

    s = ' '+s+' '

    while True:

        # search if more than two alphabets are separated by space

        x = re.search(r"(\s[a-zA-Z]){2,}\s", s)

        if x==None:

            break
        # replace to get the concatenation

        s = s.replace(x.group(),' '+x.group().replace(' ','')+' ')

    return s
```

Figure 7: Concatenation of words in resumes

Data splitting

```
X_train, X_test, y_train_, y_test = train_test_split(X, y, test_size=0.30, random_state=1)
```

Figure 6: Data Splitting of resume dataset

Feature Vectorization

Finally, the extracted text was vectorized under two methods. Count vectorizer and tf-idf vectorizer was used in Scikit learn for this.

1. Feature vectorization using Count Vectorizer
2. Feature vectorization using TF-IDF Vectorizer.

```
vectorizer1 = CountVectorizer()

job_transformed = vectorizer1.fit_transform(X['job_description'])
resume_transformed = vectorizer1.fit_transform(X['processed_resume'])

# Combine the transformed data horizontally
X_transformed = hstack((job_transformed, resume_transformed))
```

Figure 7: Feature Vectorizer of resume dataset

Model Training

After the features were extracted, several machine learning models were used to identify the best models among them.

```
vectorizer1 = TfidfVectorizer()

job_transformed = vectorizer1.fit_transform(X['job_description'])
resume_transformed = vectorizer1.fit_transform(X['processed_resume'])

# Combine the transformed data horizontally
X_transformed = hstack((job_transformed, resume_transformed))
```

Figure 8 Combining Transformations

Following are the machine learning models used,

1. Linear Regression Model
2. KNN Regression Model
3. Decision Tree Regression Model
4. Support Vector Regression Model – Linear Kernal
5. Support Vector Regression Model – RBF Kernal
6. Random Forest Regression Model

To optimize their performance, hyperparameter tuning was conducted using the GridSearchCV technique, allowing for the identification of the best parameters and the selection of the most optimal model.

```
from sklearn.model_selection import GridSearchCV
param_grid = {
    'C': [0.1, 1, 10],
    'epsilon': [0.01, 0.1, 1]
}
grid_search =GridSearchCV(estimator=svr_model_linear ,param_grid =param_grid ,cv =5)
grid_search.fit(X_train,y_train)
best_param =grid_search.best_params_
grid_search.score(X_test,y_test)
```

Figure 9: Implementing Grid searchCV.

```
print( grid_search.score(X_train,y_train))
best_model = grid_search.best_estimator_
print(best_model)
print(best_param)
```

Figure 10: Choosing Best Model

Stacked Ensemble Model

To further enhance accuracy, the stacked method was utilized as an ensemble technique. This approach involved combining the individual models that demonstrated favorable accuracies, resulting in a single unified model with significantly improved accuracy. This methodology was applied to the data features obtained from TF-IDF vectorization and Bag-of-Words vectorization techniques.

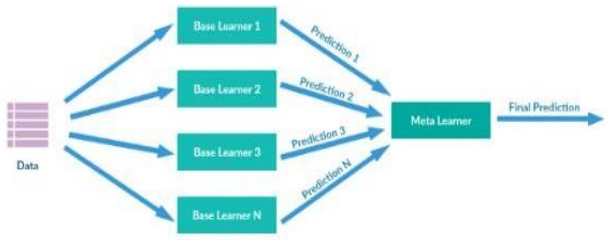


Figure 11: Stacked Model

For the implementation of the stacked model individual model with the best accuracies were taken as base models.

```
# Step 3: Build the Base Models
base_models = [
    ('Ridge', Ridge(alpha=0.1, fit_intercept=False, solver='lsqr')),
    ('svr_model_linear' , SVR(C=10, epsilon=0.01, kernel='linear')),
    ('rfr_model',  RandomForestRegressor(max_depth=20, n_estimators=50)),
    ('dt_model', DecisionTreeRegressor(max_depth=10, min_samples_leaf=4) ),
]

# Step 4: Build the Stacking Model
stacking_model = StackingRegressor(
    estimators=base_models,
    final_estimator = KNeighborsRegressor(metric='manhattan', n_neighbors=9, p=1, weights='distance')
)
```

Figure 12:Stacked Model Implementation

2.1.3. Keyword extraction from resumes

As the initial step, the resume data was subjected to preprocessing steps. The clean text function serves the purpose of cleaning input text data by removing non-ASCII special characters. It takes a text string as input and returns a cleaned version of the text.

```
def clean_text(text):

    import re
    cleaned_data = re.sub(r'[^a-zA-Z0-9\s\/]', '', text)
    cleaned_data = cleaned_data.replace('/', ' ')
    return cleaned_data
```

Figure 13:Resume Data Cleaning

First, the cleaned data was tokenized using the NLTK library in Python.

```
def nltk_tokenizer(text):

    nltk.download('punkt')
    from nltk import word_tokenize
    tokens = word_tokenize(text)
    #tokens = text.split()
    return tokens
```

Figure 1 4: Tokenization of resume words

Then the tokenized data is fed into the below methods to get token lists

```
def nltk_pos_tag(token_list):

    nltk.download('averaged_perceptron_tagger')
    from nltk import pos_tag
    tagged_list = pos_tag(token_list)
    return tagged_list
```

Figure 15: Getting NLTK POS Tags

Stop words present in tokenized words are removed.

```
def nltk_stopwords_removal(token_list):

    nltk.download('stopwords')
    from nltk.corpus import stopwords
    stop_words = set(stopwords.words('english'))
    stopwords_filtered_list = [w for w in token_list if w not in stop_words]
    return stopwords_filtered_list
```

Figure 16: Stop Word Removal

After subjecting to preprocessing of cleaning tokenization and stop word removal unique tokens were obtained. The code snippet assigns the variable 'keywords' with a filtered list of tokens from 'pos_tagged_tokens', where tokens are retained if they belong to specific

grammatical categories: proper nouns (NNP), nouns (NN), verbs in non-3rd person singular present tense (VBP), and adjectives (JJ).

```
def nltk_keywords(data):

    data = clean_text(data)
    tokens = nltk_tokenizer(data)
    pos_tagged_tokens = nltk_pos_tag(tokens)
    keywords = filter_token_tag(pos_tagged_tokens, ['NNP', 'NN', 'VBP', 'JJ'])
    keywords = nltk_stopwords_removal(keywords)
    keywords = unique_tokens(keywords)
    #print('NLTK Keywords: ', keywords)
    return keywords
```

Figure 13: Getting Keywords using NLTK

Finally, the extracted keywords from a resume are compared with the keywords from a job description. The code iterates through each word in the job description (referred to as a word). It checks if the current word from the job description is also present in the keywords_resume list, which likely contains keywords extracted from the resume.

```
#Creating a table showing Match Result between JD and Resume
jd_keywords_in_resume_table = []
for word in keywords_jd:
    if word in keywords_resume:
        match_result = [word, 'Match']
    else:
        match_result = [word, 'No Match']
    jd_keywords_in_resume_table.append(match_result)

from tabulate import tabulate
print(f'Comparing Resume and Job Description:')
print(tabulate( jd_keywords_in_resume_table, headers=['Words', 'Keywords in JD', 'JD-Resume Match Result'], showindex='al

#Write output in a result file
write_string = 'Match Result between JD and Resume Keywords' + '\n' + tabulate( jd_keywords_in_resume_table, headers=['S
```

Figure 14: Final keyword matching

2.1.4. CV skills matching

The skills provided by the recruiter are compared with the skills extracted from the CV. Unique skills and their frequencies are then collected to create a skills summary chart for further analysis.

```
def get_skill_frequency(text, skills_list):
    skill_counts = Counter()

    for skill in skills_list:
        pattern = r"\b{}\b".format(re.escape(skill))
        matches = re.findall(pattern, text, re.IGNORECASE)
        skill_counts[skill] = len(matches)

    return skill_counts
```

Figure 15: Skills Analysis

2.2. Academic Transcript Analysis

This research component bridges the gap between academic qualifications and professional suitability. Firstly, module keywords and phrases are systematically extracted from module outlines at a specific university using n-grams, providing a comprehensive foundation for data analysis. Next, data is extracted from academic transcripts via Optical Character Recognition (OCR) techniques, allowing for the identification of module names and associated grades using a custom Named Entity Recognition (NER) model. Following this, skill areas are categorized utilizing machine learning models, offering a nuanced understanding of a candidate's skill profile. Further enhancing this assessment, a proficiency-based score is computed, considering both the categorized skills and specific job role requirements. Finally, these skill areas are visually represented through graphical visualization techniques, providing employers with an intuitive means to understand a candidate's suitability.

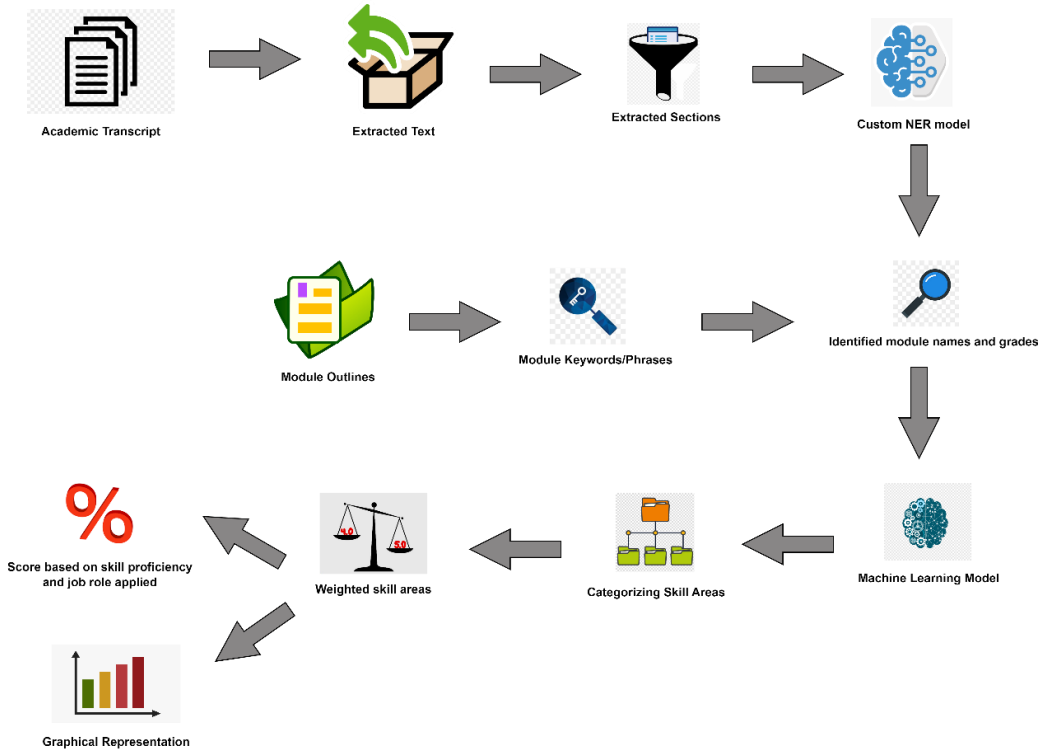


Figure 16: High level system diagram of analyzing academic transcript component

2.2.1. Extracting module keywords and phrases from module outlines

In this section, methodology related to extracting module keywords and phrases will be discussed.

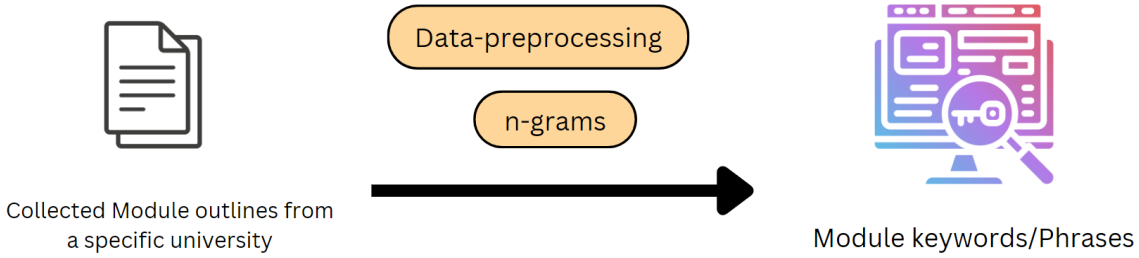


Figure 17: Extracting module keywords and phrases from module outlines

Data Pre-processing

In Natural Language Processing (NLP), a crucial step is cleaning up the text data, and we achieve this through data pre-processing. By using the NLTK library, stop words are removed, and any unwanted characters are taken out. Additionally, all words are changed to lowercase to ensure consistency throughout the text. This pre-processing is vital because it sets the stage for more effective NLP tasks and model training, by enhancing the overall robustness and accuracy of the research framework.

```
[ ]  # Create a list to store the words in each row
     words_per_row = []

     # Iterate over the rows and split the module description into words
     for desc in module_desc_column:
         if isinstance(desc, str):
             words = desc.lower().split()  # Split the description into lowercase words

             # Remove stop words
             stop_words = set(stopwords.words("english"))
             words = [word for word in words if word not in stop_words]

               # Remove unwanted characters using regular expressions
             words = [re.sub(r"[^a-zA-Z0-9]", "", word) for word in words]

             # Remove empty strings
             words = [word for word in words if word]

             words_per_row.append(words)  # Add the words to the list

     # Print the lists of words per row
     for words in words_per_row:
         print(words)
```

Figure 18: Dataset pre-processing

Figure 19: Preprocessed Dataset

Furthermore, WordNet lemmatizer is used which is a tool that simplifies words to their base forms. A list is created to hold these simplified words for each row of text. While going through the rows, the lemmatizer processes the words, converting them into their basic forms. This step ensures that words with similar meanings are treated consistently, enhancing the accuracy and clarity of our analysis. It's similar to simplifying complex math problems to better understand their solutions.

Keywords and phrases extraction using n-grams

Once the preprocessing was completed, the TF-IDF vectorizer was applied to these sentences to capture the importance of individual words. The most significant words for each sentence were identified and presented. Additionally, n-grams (word combinations) were extracted from the sentences using a similar TF-IDF approach, highlighting meaningful word sequences. These steps collectively improved the understanding and representation of the module description, aiding in the research's accuracy and comprehensibility.

2.2.2. Extracting the data from academic transcripts

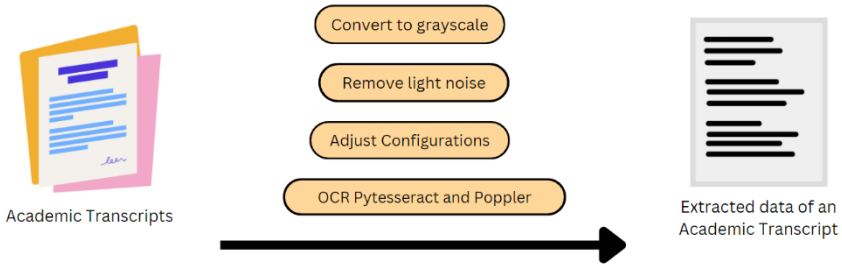


Figure 20: Academic Transcript extraction overview diagram

Data Pre-processing and extracting data

When a file is uploaded, it is temporarily saved, and then optical character recognition (OCR) technology is applied to convert the content from PDF format into text format. This involves adjusting the image quality, converting it to grayscale, and eliminating any background noise in order to improve the accuracy of extracting the text. The extracted text is then stored in a text file, for analysis.

```
def extract_text():

    uploaded_file = request.files['file']

    if uploaded_file.filename != '':
        # Save the uploaded file temporarily
        uploaded_file.save('files/academic_transcript/uploaded_file.pdf')

        pdf_paths = glob.glob("files/academic_transcript/uploaded_file.pdf")
        text_file_path = "files/academic_transcript/extracted_text.txt"

        pytesseract.pytesseract.tesseract_cmd = r'C:\Program Files\Tesseract-OCR\tesseract.exe'  # Provide the path to your Tesseract executable

        with open(text_file_path, "w", encoding="utf-8") as text_file:
            for pdf_path in pdf_paths:
                pages = convert_from_path(pdf_path, dpi=300)  # Adjust DPI as needed

                for pageNum, imBlob in enumerate(pages):
                    im = Image.frombytes('RGB', imBlob.size, imBlob.tobytes())  # Corrected line
                    im = im.convert('L')  # Convert to grayscale
                    im = im.point(lambda x: 0 if x < 200 else x)  # Threshold to remove light noise
                    text = pytesseract.image_to_string(im, lang='eng', config='--psm 6')  # Adjust configuration
                    text_file.write(text)
```

Figure 21:Academic transcript pre-processing and extracting data

2.2.3. Name entity recognition model to identify module names and grades

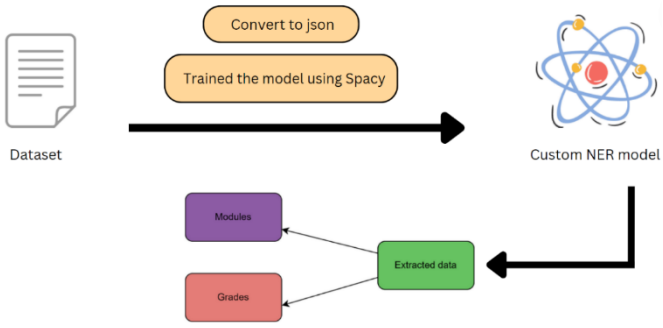


Figure 22: Name entity recognition model overview diagram

Custom NER model

Dataset and Data Exploratory Analysis

To build the custom Named Entity Recognition (NER) model, a dummy dataset was created. This dataset consisted of 8298 records, each containing specific columns such as "Code," "Module Title," "Semester," "Period," "Credits," and "Grade." These columns mimic the structure of academic transcripts, providing a foundation for training the NER model to recognize and extract essential information, such as module names and associated grades, from textual data resembling academic records.

	A	B	C	D	E	F
1	Code	Module Title	Semester	Period	Credits	Grade
2	EC1430	Data Communications & Computer Network	1	16-Apr	4	B+
3	EL1200	English Language Skills - I	1	16-Apr	4	B+
4	IT1000	Computer Fundamentals	1	16-Apr	4	A
5	IT1010	Introduction to Programming Environments	1	16-Apr	5	A+
6	MA140	Mathematics for Information Technology	1	16-Apr	4	A-
7	EL1210	English Language Skills- II	2	16-Apr	3	A
8	IT1020	Software Technology - I (Data Structures)	2	16-Apr	4	A+
9	IT1030	Database Management Systems - I	2	16-Apr	4	A+
10	IT1040	Internet Technology and Applications	2	16-Apr	4	A
11	MA1410	Foundations of Computer Science	2	16-Apr	4	A-
12	CG2000	Computer Graphics & Multimedia	1	16-Oct	4	B+
13	IT2000	Software Technology - II (OOP)	1	16-Oct	4	B+
14	IT2020	Database Management Systems - II	1	16-Oct	4	B+
15	IT2200	Software Engineering- I	1	16-Oct	4	B+
16	MA220	Probability & Statistics	1	16-Oct	4	A
17	EC2440	Data Communications & Computer Network	2	16-Oct	4	A
18	IT2010	Systems Programming and Design	2	16-Oct	4	A
19	IT2210	Software Engineering - II	2	16-Oct	4	B+
20	IT2220	Information Technology Project	2	16-Oct	4	A-
21	IT2400	Design and Analysis of Algorithms	2	16-Oct	4	A
22	IT3050*	Employability Skills Development - Seminar	1	16-Oct	1	B
23	SE3010	Software Engineering Process & Quality Man	1	16-Oct	4	B
24	SE3020	Distributed Systems	1	16-Jun	4	A
25	SE3030	Software Architecture	1	16-Jun	4	A-

Figure 23: Dataset for custom NER model

Data preprocessing for model training

In data preprocessing of the custom Named Entity Recognition (NER) model, a new Spacy model was loaded, and the custom NER component was integrated into the pipeline. Distinct entity labels, including MODULE_CODE, MODULE_TITLE, SEMESTER, PERIOD, CREDITS, and GRADE, were defined to represent the different elements in the academic transcripts. The dataset containing the relevant information was then imported, and text along with entity annotations were structured into a compatible format, typically in JSON (JavaScript Object Notation) objects, which served as a standardized way to represent the annotated data.

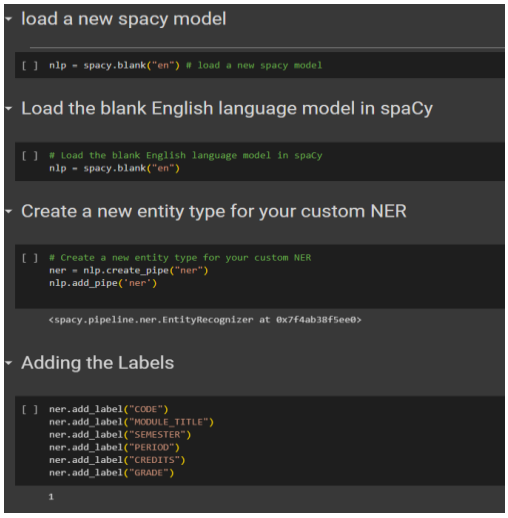


Figure 24: Dataset preprocessing for custom NER model

Figure 25: Conversion of objects to JSON format

Training the model to identify module names and their corresponding grades

The NER model was then trained using this annotated training data, with multiple iterations and mini-batch training to optimize its performance. Finally, the trained NER model was saved for future use, allowing it to accurately recognize and extract module-related information from textual data.

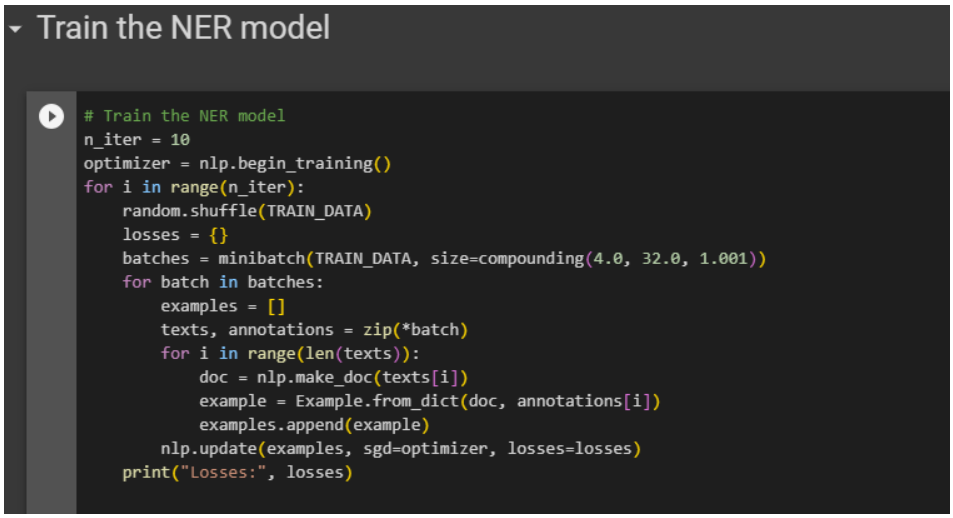


Figure 26: Training custom NER model

Pre-trained NER model

Dataset and data preprocessing

The dataset, initially in a text format, was annotated and transformed into structured JSON objects, as illustrated below.

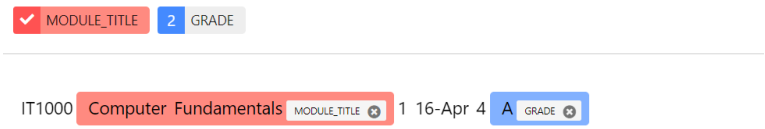


Figure 27: Dataset annotations

Model training

Initially, a blank Spacy model is loaded, and a DocBin object is created to store the training data. The training data, obtained from a JSON file, contains text annotations and entity labels. The text and entity annotations are processed, and the entities are added to the DocBin object. This training data is then saved to a file. Next, a configuration file is

initialized for the NER pipeline, followed by training the NER model using the training data. The trained model is saved and loaded for testing purposes.

```
from spacy.util import filter_spans

#for training_example  in training_data:
text = training_data['annotations'][0][0]
labels = training_data['annotations'][0][1]['entities']
doc = nlp.make_doc(text)
ents = []
for start, end, label in labels:
    span = doc.char_span(start, end, label=label, alignment_mode="contract")
    if span is None:
        print("Skipping entity")
    else:
        ents.append(span)
filtered_ents = filter_spans(ents)
doc.ents = filtered_ents
doc_bin.add(doc)

doc_bin.to_disk("train.spacy") # save the docbin object
```

Figure 28: docBin object to save the model

```
! python -m spacy train config.cfg --output ./ --paths.train ./train.spacy --paths.dev ./train.spacy
```

Figure 29: Training the model

2.2.4. Skill area categorization and graphical representation of skill areas

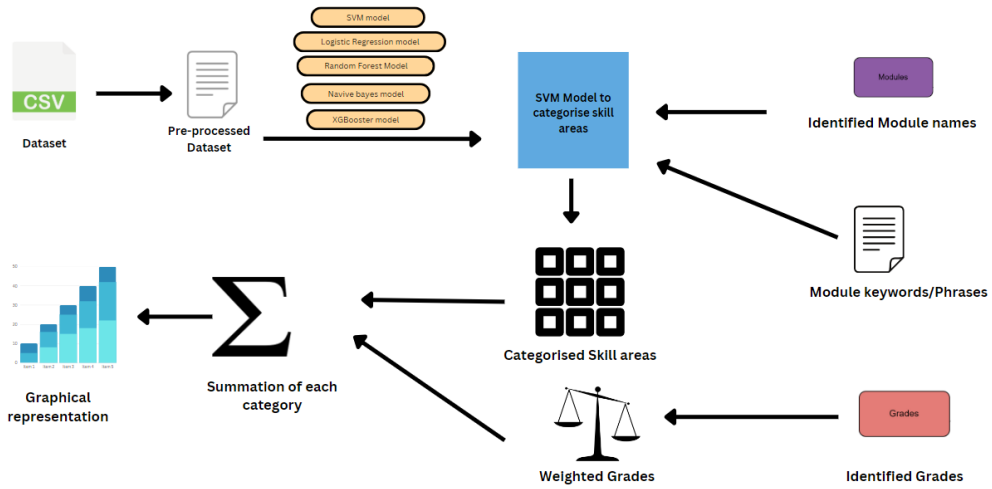


Figure 30: Skill area categorization and graphical representation of skill areas

Dataset and Data Exploratory Analysis

The dataset used for predicting skill areas comprises three main columns: "Module Title," "Module Keywords," and "Skill Area," encompassing a total of 7776 records. The identification of the ten skill areas within this dataset was informed by comprehensive research drawn from various online sources [1] [2] Furthermore, the population who was considered for this survey were the IT employees from the industry. From the survey below, we can see that the majority of the respondents have selected the first 10 categories.

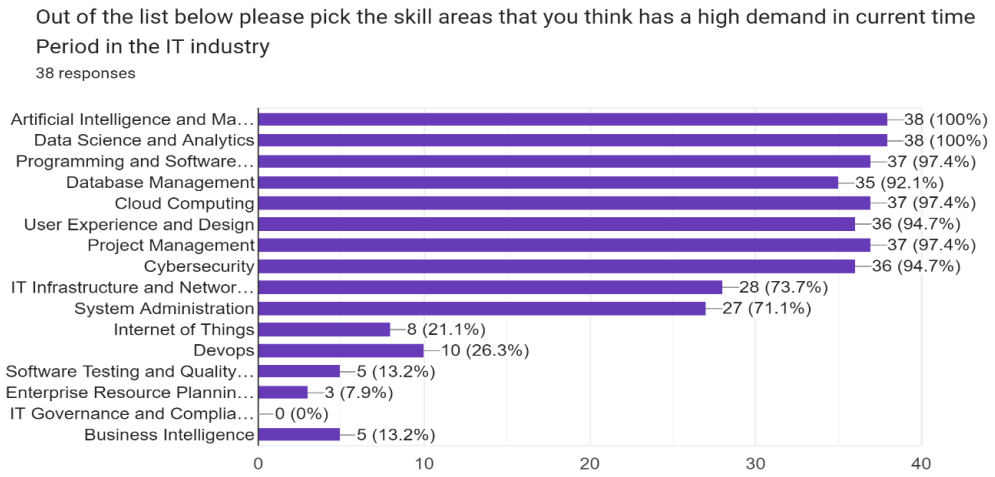


Figure 31:Survey results of identifying the high level skill areas

Thus, the categories are,

- System Administration
- IT Infrastructure and Networking
- Data Science and Data Analytics
- Artificial Intelligence and Machine Learning
- Cloud Computing
- Cybersecurity
- Database Management
- Project Management
- Programming and Software Development
- User Experience, and Design.

These skill categories were derived from real-world industry insights and serve as the basis for our skill area prediction model, enabling us to categorize module titles and keywords effectively in alignment with these crucial skill domains.

Data preprocessing for model training

In the data preprocessing phase, several key tasks were performed to prepare the dataset for analysis. Initially, stopwords, which are common and non-informative words in English, were removed to focus on more meaningful content. Next, text data within the "Module Title" and "Module Keywords" columns were converted to lowercase for uniformity and to reduce text variations. Additionally, special characters and symbols were eliminated from the text, ensuring that only relevant alphanumeric characters and spaces remained. These data preprocessing steps collectively contributed to cleaning and standardizing the dataset, making it more suitable for subsequent analysis and modeling within the research framework.

Model Building for Skill Area Prediction

In this phase of the research methodology, Advanced natural language processing techniques were used to analyze textual data, and multi-class text classification models of supervised learning machine learning models were used to predict the skill area. Considering the accuracies of all models, The Support Vector Machine (SVM) model was selected as the best option because it consistently achieved the highest accuracy in our classification tasks. This means that SVM was the most reliable choice for accurately categorizing skill areas from textual data, making it the preferred model for the research

```python
import gensim
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression

# Train Word2Vec model
sentences = [text.split() for text in x_train]
word2vec_model = gensim.models.Word2Vec(sentences, vector_size=100, window=5, min_count=1, workers=4)

# Create document vectors using Word2Vec
document_vectors = []
for text in sentences:
    vectors = [word2vec_model.wv[word] for word in text if word in word2vec_model.wv]
    if vectors:
        document_vectors.append(sum(vectors) / len(vectors))
    else:
        document_vectors.append([0] * 100)

import numpy as np

# Create document vectors using Word2Vec
document_vectors = [np.mean([word2vec_model.wv[word] for word in sentence], axis=0) for sentence in sentences]

# Convert document vectors to numpy array
X_train = np.array(document_vectors)

# Encode class labels
label_encoder = LabelEncoder()
y_train_encoded = label_encoder.fit_transform(y_train)


# Split into training and testing data
X_train, X_test, y_train_encoded, y_test_encoded = train_test_split(X_train, y_train_encoded, test_size=0.2, random_state=42)

# Train a logistic regression classifier
classifier = LogisticRegression()
classifier.fit(X_train, y_train_encoded)

# Predict and evaluate
X_test_vectors = []
for text in x_test:
    vectors = [word2vec_model.wv[word] for word in text.split() if word in word2vec_model.wv]
    if vectors:
        X_test_vectors.append(sum(vectors) / len(vectors))
    else:
        X_test_vectors.append([0] * 100)
X_test_vectors = np.array(X_test_vectors)

y_pred_encoded = classifier.predict(X_test_vectors)
y_pred = label_encoder.inverse_transform(y_pred_encoded)

accuracy = accuracy_score(y_pred, y_test)
print(f'Accuracy: {accuracy:.4f}')
```

Figure 32: Logistic regression algorithm for skill area prediction

```
[ ]  from sklearn.naive_bayes import MultinomialNB
     from sklearn.pipeline import Pipeline
     from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer
     from sklearn.metrics import accuracy_score
     from sklearn.preprocessing import LabelEncoder

     # Encode class labels
     label_encoder = LabelEncoder()
     y_train_encoded = label_encoder.fit_transform(y_train)

     # Create the pipeline
     naivebayes = Pipeline([
         ('vect', CountVectorizer()),
         ('tfidf', TfidfTransformer()),
         ('clf', MultinomialNB()),
     ])

     # Fit the pipeline
     naivebayes.fit(x_train, y_train_encoded)

     # Predict and evaluate
     y_pred_encoded = naivebayes.predict(x_test)

     # Decode predicted labels
     y_pred = label_encoder.inverse_transform(y_pred_encoded)

     accuracy = accuracy_score(y_pred, y_test)
     print(f'Accuracy: {accuracy}')
```

Figure 33: Naive Bayes algorithm for skill area prediction

```
[ ]  from sklearn.preprocessing import LabelEncoder
     from xgboost import XGBClassifier
     from sklearn.pipeline import Pipeline
     from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer
     from sklearn.metrics import accuracy_score

     # Encode class labels
     label_encoder = LabelEncoder()
     y_train_encoded = label_encoder.fit_transform(y_train)

     xgboost = Pipeline([
         ('vect', CountVectorizer()),
         ('tfidf', TfidfTransformer()),
         ('clf', XGBClassifier())
     ])

     xgboost.fit(x_train, y_train_encoded)

     y_pred_encoded = xgboost.predict(x_test)

     # Decode predicted labels
     y_pred = label_encoder.inverse_transform(y_pred_encoded)

     accuracy = accuracy_score(y_pred, y_test)
     print(f'Accuracy: {accuracy}')
```

Figure 34: XGBoost algorithm for skill area prediction

```
[ ]  from sklearn.feature_extraction.text import CountVectorizer
     from sklearn.ensemble import RandomForestClassifier
     from sklearn.metrics import accuracy_score

     # Preprocess the data
     nltk.download('stopwords')
     STOPWORDS = set(stopwords.words('english'))

     def clean_text(text):
         text = text.lower()
         text = re.sub('[/(){}\[\]\|@,;]', ' ', text)
         text = re.sub('[^0-9a-z #+_]', '', text)
         text = ' '.join(word for word in text.split() if word not in STOPWORDS)
         return text

     df['Module Title'] = df['Module Title'].apply(clean_text)
     df['Module Keywords'] = df['Module Keywords'].apply(clean_text)

     # Split the dataset into training and testing
     x = df['Module Title'].str.lower() + ' ' + df['Module Keywords'].str.lower()
     y = df['Skill Area']
     x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)

     # Initialize CountVectorizer
     countvect = CountVectorizer()

     # Prepare the feature matrix
     x_train_encoded = countvect.fit_transform(x_train.tolist())
     x_test_encoded = countvect.transform(x_test.tolist())

     # Train the Random Forest model
     random_forest = RandomForestClassifier(n_estimators=100, random_state=42)
     random_forest.fit(x_train_encoded, y_train)  # Use y_train instead of y_train_encoded

     # Predict on the test set
     y_pred = random_forest.predict(x_test_encoded)

     # Evaluate the model
     accuracy = accuracy_score(y_pred, y_test)
     print(f'Accuracy: {accuracy}')
```

Figure 35: Random forest algorithm for skill area prediction

```
[ ] from imblearn.over_sampling import RandomOverSampler

    # Splitting the dataset into training and testing
    from sklearn.model_selection import train_test_split
    x = df['Module Title'].str.lower() + ' ' + df['Module Keywords'].str.lower()
    y = df['Skill Area']
    x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, stratify=y, random_state=42)

    # Initialize CountVectorizer
    countvect = CountVectorizer()

    # Transform text data into vectors
    x_train_vect = countvect.fit_transform(x_train)
    x_test_vect = countvect.transform(x_test)

    # Create an instance of RandomOverSampler
    random_oversampler = RandomOverSampler(sampling_strategy='auto', random_state=0)

    # Apply SMOTE to the vectorized data
    X_train = pd.DataFrame(x_train_vect)
    x_resample, y_resample = SMOTE().fit_resample(x_train_vect, y_train)

    # Initialize and train the SVM model
    svm_model = SVC(kernel='rbf', random_state=0)
    svm_model.fit(x_resample, y_resample)

    # Predict and evaluate
    prediction = svm_model.predict(x_test_vect)
    accuracy = accuracy_score(y_test, prediction)
    print("Accuracy:", accuracy)

    # Visualize confusion matrix
    sns.heatmap(confusion_matrix(y_test, prediction), annot=True, fmt="g")
    plt.xlabel("Predicted")
    plt.ylabel("Actual")
    plt.show()

    # Display classification report
    print(classification_report(y_test, prediction))
```

Figure 36: SVM for skill area prediction

In the process of fine-tuning the SVM model, to improve its performance we used a linear kernel and applied cross-validation for robust evaluation. The mean score of the cross-validation results was calculated to assess the model's effectiveness. Next, the new SVM model was fitted on the resampled training data, ensuring it was optimized for the research's objectives. Finally, the model was saved to a file using the Pickle library, making it available for future use in skill area prediction tasks.

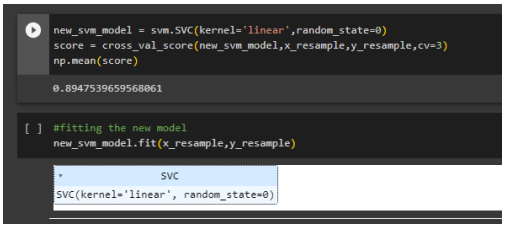


Figure 37: Fine-tuned SVM model

2.2.5. Skill area categorization and graphical representation

In the research methodology, several critical steps were taken to process academic transcripts and predict skill areas. Initially, the extracted text from the academic transcripts was divided into sections based on academic years using regular expressions. The custom Named Entity Recognition (NER) model that was built was used to extract module titles and their corresponding grades from these sections.

Each identified module title was combined with its corresponding module keywords and phrases which were extracted from the module outline using n-grams and passed into the SVM model to categorize them into different skill areas.

Then, the weighted grades were calculated for each module title. A grading scheme was defined, assigning weights to grades from 'A+' to 'E'.

- A+ : 10
- A : 9
- A- : 8
- B+ : 7
- B : 6
- B- : 5
- C+ : 4
- C: 3
- C- : 2
- D+ : 1
- D : 1
- E : 1

These weighted grades were used to evaluate the academic performance associated with each skill area. Next, a table was constructed, summarizing skill areas, module titles, and their corresponding weighted grades.

Furthermore, the research analyzed the distribution of weighted grades across different skill areas. The skill areas were grouped, and the total weighted grades for each category were calculated. These categories were sorted in descending order based on their total

weighted grades, providing insights into the relative importance of various skill areas within the academic transcripts.

In order to present a concise overview of the skill areas a graphical representation of the skill categories was used, which serves as a valuable tool for hiring managers, offering a clear and concise view of candidates' skill profiles.

```python
# Predict skill areas using the function
predicted_skill_areas = predict_skill_area(module_titles, module_title_to_keywords)

# Map the module titles from the extracted data to the module titles in the module outline
module_titles_df['Category'] = predicted_skill_areas

# Define the grade weighting dictionary
grade_weighting = {
    'A+': 10,
    'A': 9,
    'A-': 8,
    'B+': 7,
    'B': 6,
    'B-': 5,
    'C+': 4,
    'C': 3,
    'C-': 2,
    'D+': 1,
    'D': 1,
    'E': 1
}

# Calculate the weighted grades based on the grade weighting dictionary
module_titles_df['Weighted Grade'] = module_titles_df['Grade'].map(grade_weighting)


# Calculate the weighted grades based on the grade weighting dictionary
module_titles_df['Weighted Grade'] = module_titles_df['Grade'].map(grade_weighting)

# Group the skill area table by the 'Category' column and calculate the sum of the 'Weighted Grade' column for each category
category_totals = module_titles_df.groupby('Category')['Weighted Grade'].sum()


# Sort the categories by their total weighted grades in descending order
category_totals = category_totals.sort_values(ascending=False)
```

Figure 38: Skill area prediction

```python
# Predict skill areas using the function
predicted_skill_areas = predict_skill_area(module_titles, module_title_to_keywords)

# Map the module titles from the extracted data to the module titles in the module outline
module_titles_df['Category'] = predicted_skill_areas

# Define the grade weighting dictionary
grade_weighting = {
    'A+': 10,
    'A': 9,
    'A-': 8,
    'B+': 7,
    'B': 6,
    'B-': 5,
    'C+': 4,
    'C': 3,
    'C-': 2,
    'D+': 1,
    'D': 1,
    'E': 1
}

# Calculate the weighted grades based on the grade weighting dictionary
module_titles_df['Weighted Grade'] = module_titles_df['Grade'].map(grade_weighting)


# Calculate the weighted grades based on the grade weighting dictionary
module_titles_df['Weighted Grade'] = module_titles_df['Grade'].map(grade_weighting)

# Group the skill area table by the 'Category' column and calculate the sum of the 'Weighted Grade' column for each category
category_totals = module_titles_df.groupby('Category')['Weighted Grade'].sum()


# Sort the categories by their total weighted grades in descending order
category_totals = category_totals.sort_values(ascending=False)
```

Figure 39: Skill area categorization

2.2.6. Academic Transcript score prediction model

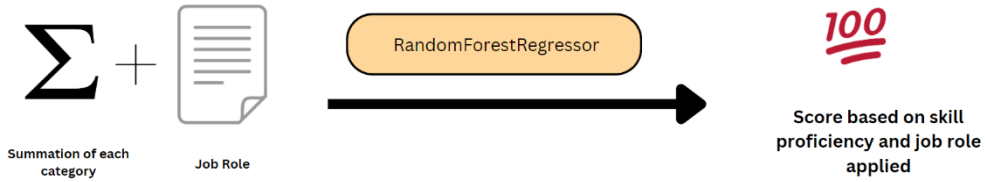


Figure 40: Academic Transcript score prediction model overview diagram

Data preprocessing and feature engineering

In data preprocessing and feature engineering, duplicate entries based on job roles were removed. Then, to address any missing values within the dataset, a consistent approach was adopted by replacing them with zero values. Additionally, to facilitate the model's understanding of job roles, a one-hot encoding technique was applied to the "Job role" column, converting categorical job roles into a numerical format. Simultaneously, to convert the "job description" column into a numerical format, a text preprocessing method known as CountVectorizer was employed.

```
# Define a column transformer for one-hot encoding the "Job Role" column
column_transformer = ColumnTransformer(
    transformers=[
        ('encoder', OneHotEncoder(), ['Job role'])
    ],
    remainder='passthrough'
)

# Transform the X data including CountVectorizer encoding
X_encoded = column_transformer.fit_transform(X)

# After the transformation
print("X_encoded shape:", X_encoded.shape)
```

Figure 41: Job role conversion to numeric using One hot encoder

```
# Remove duplicate rows
data.drop_duplicates(inplace=True)

# Fill null values with 0
data.fillna(0, inplace=True)
```

Figure 42: Drop duplicates and full missing values

```
# Define the CountVectorizer for the "Job Description" column
vectorizer = CountVectorizer()

# Fit and transform the "Job Description" column
job_description_matrix = vectorizer.fit_transform(data['Job Description'])

# Convert the result to a DataFrame
job_description_df = pd.DataFrame(job_description_matrix.toarray(), columns=vectorizer.get_feature_names_out())

# Concatenate the new DataFrame with the original X_encoded
X_encoded = pd.concat([pd.DataFrame(X_encoded, columns=column_transformer.get_feature_names_out()), job_description_df], axis=1)
```

Figure 43: Job description conversion to numeric using countvectorizer

Model building

The dataset was divided into two distinct sets: a training set and a testing set, enabling the assessment of model performance. Four regression models were considered for this task: the Random Forest Regressor, Linear Regression, Support Vector Regression (SVR), and Gradient Boost Regressor. Initially, the dataset underwent feature transformation, involving the addition of polynomial features and standardization to enhance the models' predictive capabilities. Next, the models were trained, and the models' performance was evaluated by two metrics, Mean Squared Error (MSE), which quantifies the average squared difference between predicted and actual scores, and the R-squared (R2) score, which serves as an indicator of how effectively the models elucidate the variations in the data.

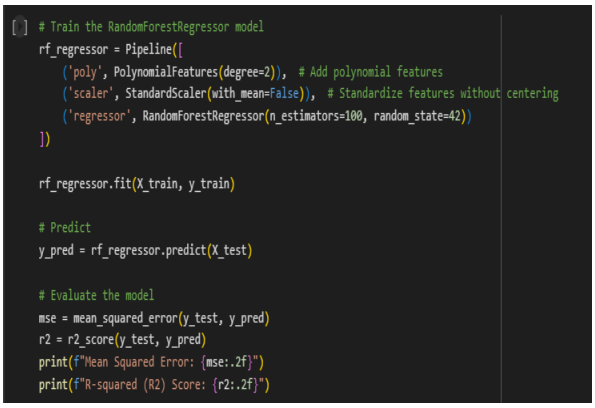


Figure 44: Random Forest Regressor model to predict score

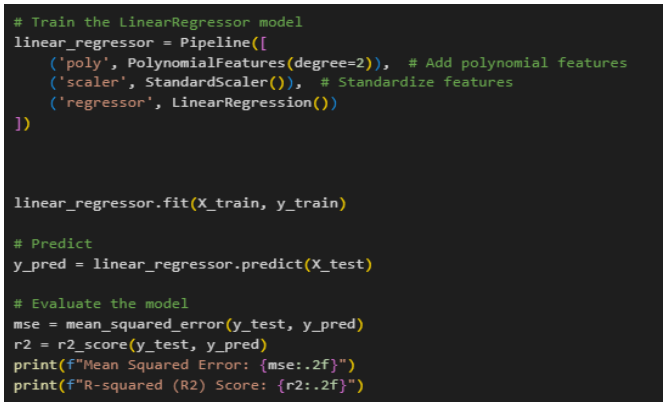


Figure 45: Linear Regressor model to predict score

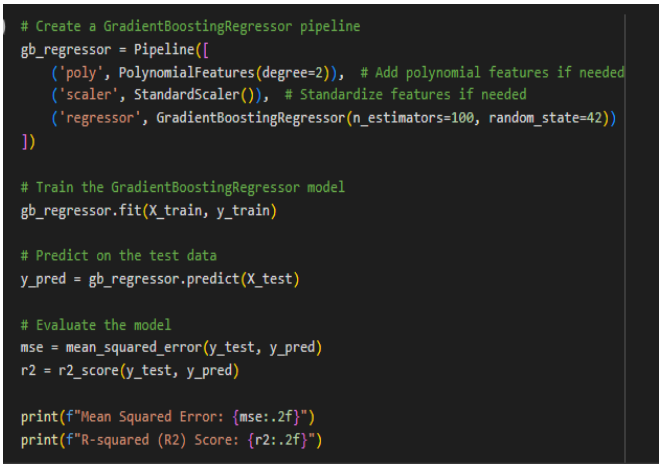


Figure 46: Gradient Boost Regressor model to predict score

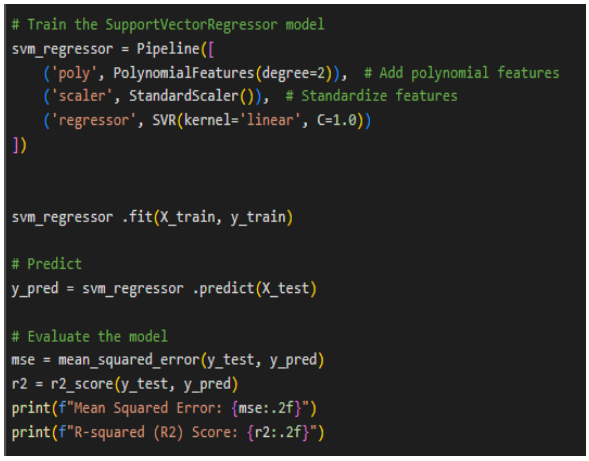


Figure 47: Support Vector Regressor model to predict score

1.2.1. Frontend Development

The frontend of this research component was built using HTML, CSS, and JavaScript. Flask was used as the engine to connect the frontend with the backend(Machine learning models and NLP algorithms).

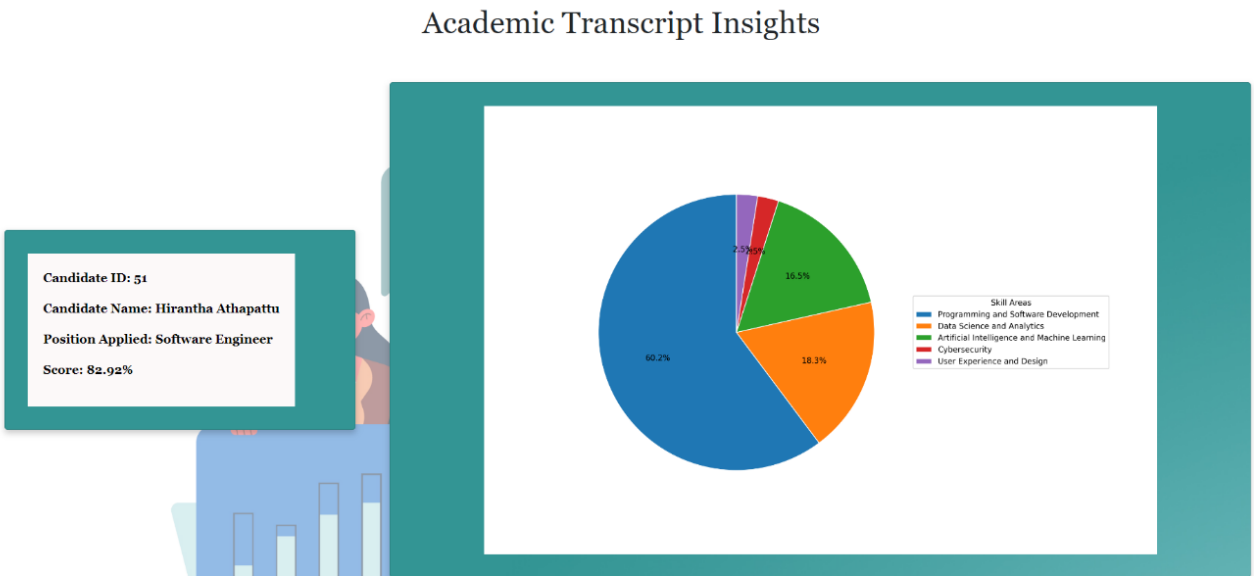


Figure 48: Academic Transcript user interface

2.3.Professional Skills Analysis

2.3.1. LinkedIn skill-based job category prediction

Model development

Dataset - This dataset, vital for our IT job research, was methodically constructed by combining data from Kaggle, job descriptions, advertisements, vacancies, and IT recruiter validation. And it is crucial for the research on IT job categories, roles, and skills. Through careful data cleaning and integration, it has established a solid research foundation while upholding data privacy and reliability. Beyond the research, this dataset offers valuable insights for the broader IT community, facilitating workforce planning and industry analysis.

A multi-class text classification model under supervised machine learning was used to predict the matching or classifying of job candidates into specific job categories.

Text Processing

This involves converting the text into numerical vectors through the Term Frequency-Inverse Document Frequency (TF-IDF) method. This technique allows us to assess the significance of individual words within the entire corpus. To achieve this, we first ensure uniformity by converting all text to lowercase and eliminating punctuation. Subsequently, we calculate the importance of each word based on its frequency within the text. In essence, TF-IDF transforms our text data into a format that the classification algorithm can comprehend, highlighting the importance of words in the broader context of the text collection. The TfidfVectorizer function was used with the parameters below.

- min_df - Exclude words that appear in fewer than 'min_df' documents.
- Sublinear_tf – Scale the frequency in logarithmic scale when it is true.
- Stop_words- remove pre-defined English words.

```
tfidf = TfidfVectorizer(sublinear_tf=True, min_df=5,
                        ngram_range=(1, 2),  # consider both unigrams and bigrams
                        stop_words='english')
# Transform each skill into a vector
features = tfidf.fit_transform(df1.clean_skills).toarray()
labels = df1.category_id
```

Figure 49: TfidfVectorizer function

Then identify the terms that exhibit the highest correlation with each of the specified product categories. This process involves pinpointing words or phrases that are strongly associated with each category, shedding light on the key features or attributes that distinguish them from one another.

```
# Finding the three most correlated terms with each of the job categories
N = 3 # top 3 most correlated terms
for category, category_id in sorted(category_to_id.items()):
  features_chi2 = chi2(features, labels == category_id)
  # computes the chi-squared statistic and p-values for each feature against the sel
  indices = np.argsort(features_chi2[0])
  feature_names = np.array(tfidf.get_feature_names_out())[indices]  # chi-squared scc

  unigrams = [v for v in feature_names if len(v.split(' ')) == 1]
  bigrams = [v for v in feature_names if len(v.split(' ')) == 2]
  print("n==> %s:" %(category))
  print("  * Most Correlated Unigrams are: %s" %(', '.join(unigrams[-N:])))
  print("  * Most Correlated Bigrams are: %s" %(', '.join(bigrams[-N:])))
```

Figure 50: Finding the most correlated words

Explore Text Multi-Class Classification Models

Classification models that were used were,

- Random Forest
- Linear Support Vector Machine
- Multinomial Naive Bayes
- Logistic Regression.
- XGBoost Classifier

Next divided the data into two subsets: a training set and a test set allocating 75% of the data for training purposes, and the remaining 25% will be used for testing. In this split, the 'required skills' column will serve as our input data (X), while the 'Job Category' column will be our output (Y). This division allows us to train our model on a substantial portion of the data and then assess its performance on unseen data to evaluate its effectiveness in predicting job categories based on candidate LinkedIn-based skills.

All five models were kept using in list iterating the list for each model to get mean accuracy and standard deviation to compare the performance of each model. For the best-performing model cross-validation and hyperparameter tuning were done to tweak model performance for optimal results.

Comparing Text Classification Model performance.

Compare the Mean Accuracy and the Standard Deviation for each of the five text classification algorithms. Then identified the best performing model to train model multi-class classification tasks.

- **Text Classification Model Evaluation -** Trained the model using the best-performing model to evaluate and check unseen data.
- **Prediction -** LinkedIn profile data was extracted using a third-party paid API service called 'ProxyCurl' to retrieve candidate skills added to the LinkedIn profile under "**skills**". The candidate's LinkedIn profile must be public to extract the skills for prediction. This process was done using the function below.

```python
def scrape_linkedin_skills(linkedin_profile_url):
    api_key = 'MxVwlMuCI00hrmsugxWLjA'
    api_endpoint = 'https://nubela.co/proxycurl/api/v2/linkedin'
    headers = {'Authorization': 'Bearer ' + api_key}

    response = requests.get(api_endpoint,
                            params={'url': linkedin_profile_url, 'skills': 'include'},
                            headers=headers)

    profile_data = response.json()

    if 'skills' in profile_data:
        return profile_data['skills']
    else:
        return []
```

Figure 51:Scrape LinkedIn Function

Finally, run the prediction for the skills scraped through the above function.

2.3.2. GitHub Programming Language Comparator

2.3.2.1. Language proficiency for a single candidate

- GitHub user profile data was used to evaluate Programming language proficiency in this research.
- GitHub REST API with PyGitHub modules was used to fetch data from candidate profiles using a personal access token to authenticate with GitHub.
- Then fetched the user's repositories belonging to the user with a GET request.

'/users/{username}/repos'

```python
# Fetch user information
current_username = "Maldeniya99"
current_user = g.get_user(current_username)
```

Figure 52: Fetch GitHub user data

- Iterate through the repositories returned in the response and generate a repository object for each repository by using a loop.
- Retrieve the repository language proficiency using the **'get_languages()'** method which returns the dictionary where the keys are languages used in the languages used in the repository and the values are the number of bytes written in that language. If there are languages present, it will iterate over the dictionary of languages and their respective line counts and print them. If no languages are detected, it will print "Languages: None" for that repository.
- Calculated the language proficiency by iterating over all the user's repositories and retrieving the languages used in each repository by accumulating the line counts for each language.

```
# Calculate language proficiency
language_proficiency = {}
repository_count = user.public_repos

for repo in user.get_repos():
    languages = repo.get_languages()
    for language, value in languages.items():
        if language in language_proficiency:
            language_proficiency[language] += value
        else:
            language_proficiency[language] = value
```

Figure 53: Calculate language proficiency

- Next calculate the weighted language proficiency scores by multiplying the language proficiency value by the reciprocal of the user's repository count.

```
# Calculate weighted language proficiency scores
weighted_scores = {}

for language, value in language_proficiency.items():
    weighted_score = value * (1 / repository_count)
    weighted_scores[language] = weighted_score
```

Figure 54: Calculate weighted language scores

- Then the total score by summing the weighted score.

- Normalized the weighted scores by dividing each score by the total score and multiplying by 100 to convert them to percentages. The normalized percentage scores are stored in the **percentage_scores** dictionary.

```
# Normalize and convert scores to percentages
total_score = sum(weighted_scores.values())

percentage_scores = {
    language: (score / total_score) * 100
    for language, score in weighted_scores.items()
}
```

Figure 55: Language proficiency percentage calculation

2.3.2.2. Language proficiency comparator with peer candidates

- In the comparator, it stores the weighted scores in separate dictionaries for each user and then compares the language proficiency for common languages assuming that a user with more repositories may have a broader range of language proficiency.

```
# Compare language proficiency
common_languages = set(current_user_language_proficiency.keys()) & set(other_user_language_proficiency.keys())
```

Figure 56: Common language proficiency

2.3.3. Sentiment Analysis on Reference Checking

- A pre-created Google form was sent to referees mentioned in the CV. This Google form was created surveying how to do a proper and reliable reference checking on candidates from internet resources and in recruiters' view covering all the aspects of a candidate when it comes to reference checking.

- Google Form Link here - https://forms.gle/1HCVJ1F7efaM2vhb8

- This Google form was sent to a referee to answer the questions related to candidate reference checking.
- Responds will saved as CSV files.
- “pandas” and library were used to read CSV files and pre-process the text.
- “SentimentIntensityAnalyzer’” from VADER sentiment analysis tool to perform sentiment on the text.
- ‘SentimentIntensityAnalyzer’ from NLTK calculates the sentiment score which is a compound value that ranges from -1 (negative sentiment) to 1 (positive sentiment).
- CSV data was read into a DataFrame and then sentiment analysis was performed on all the columns except the first column as it contains the Timestamp of the Google form.
- Then sentiment scores were calculated for each response column referring “SentimentIntensityAnalyzer” function.

```
# Step 2: Perform sentiment analysis
sid = SentimentIntensityAnalyzer()

# Calculate sentiment scores for each column
sentiment_scores = []
for column in columns_to_analyze:
    column_scores = [sid.polarity_scores(str(response)) for response in data[column]]
    sentiment_scores.extend(column_scores)
```

Figure 57: Calculate sentiment score

- Finally, visualize the sentiment distribution across all columns collectively in a single pie chart.

```
# Calculate overall sentiment distribution
labels = ['Positive', 'Negative', 'Neutral']
sentiment_distribution = [0, 0, 0]

for score in sentiment_scores:
    max_sentiment = max(score, key=score.get)

    if max_sentiment == 'pos':
        sentiment_distribution[0] += 1
    elif max_sentiment == 'neg':
        sentiment_distribution[1] += 1
    else:
        sentiment_distribution[2] += 1
```

Figure 58: Calculate overall sentiment distribution

This function is designed to retrieve the latest entry in a CSV file, with the assumption that it represents the most recent candidate's referee who has submitted their information through a Google Form.

2.4.Personality Assessment of Candidates

Candidates will be assessed based on their responses to a questionnaire comprising two question types:

- Self-rating questions, where candidates rate their responses on a scale ranging from 1 (strongly disagree) to 5 (strongly agree).
- Open-ended questions, allowing candidates to freely express their thoughts and ideas.

This questionnaire serves as the foundation for determining the distribution of the Big Five personality traits in each candidate.

Simultaneously, the recruiter will outline the specific personality criteria essential for the job position that the candidate is seeking. These criteria will cover ten distinct

personality traits, which will be mapped to the Big Five personality traits to determine the desired Big Five trait distribution for the job role.

Ultimately, the candidate's distribution of Big Five traits will be compared against the expected trait distribution, resulting in a score that reflects the alignment. to the candidate indicating the degree to which the candidate suits the job role.

2.4.1. Data collection

Two datasets were primarily utilized for this research:

- Self-rated questionnaire responses

  This dataset was obtained from a publicly available dataset collection on 'Kaggle.' It comprises responses to twenty-five questions, with ten questions corresponding to each of the traits in the Big Five Model. The candidates' responses are rated on a scale ranging from 1 to 5.

- Open-ended questionnaire responses

  This dataset was created by collecting responses through a Google Form survey. The survey consists of five questions, each designed to assess one of the Big Five traits.

2.4.2. Model training and keyword extraction

2.4.2.1. Self-rating responses dataset

The dataset, which includes responses to self-rating questions, was used to determine the potential personality clusters to which a candidate might belong to. Initially, the dataset

underwent preprocessing, followed by exploratory data analysis to gain insights into its characteristics.

To identify personality clusters within the dataset, the K-means clustering algorithm was used. K-means is an unsupervised machine-learning technique that groups data points into clusters based on similarity. In this context, each respondent's self-rating responses are used as data points, and the algorithm aims to group them into clusters of similar responses.

Before applying the K-means algorithm, it is essential to determine the optimal number of clusters, often referred to as the 'k-value'. This is done to ensure that the clustering results are meaningful. The 'Elbow Visualization' technique was used for this purpose. It involves running the K-means algorithm with different values of k (from 2 to 15 in this case) and plotting the variance explained by each number of clusters. The 'elbow' point on the graph is typically chosen as the optimal k-value, as it represents a balance between maximizing within-cluster similarity and minimizing the number of clusters.

```
[ ]  # Visualize the elbow
     from sklearn.cluster import KMeans
     from yellowbrick.cluster import KElbowVisualizer

     kmeans = KMeans()
     visualizer = KElbowVisualizer(kmeans, k=(2,15))
     visualizer.fit(df_sample)
     visualizer.poof()
```

Figure 59: Implementing the Elbow Visualization Technique

```
[ ]  # creating the K-means Cluster Model
     from sklearn.cluster import KMeans

     df_model = data_df.drop('country', axis=1)

     # define 6 clusters and fit the model
     kmeans = KMeans(n_clusters=6)
     k_fit = kmeans.fit(df_model)
```

Figure 60: Fitting the K-means model

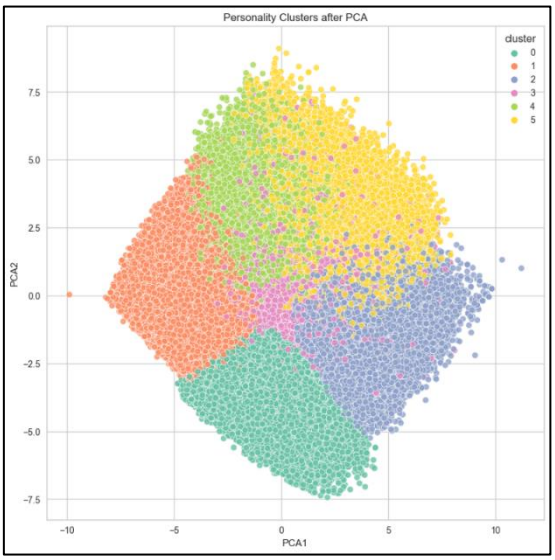


Figure 61: Visualization of the six personality clusters

Once the K-means model was trained, it was used to predict clusters for each response in the dataset. These cluster predictions underwent validation through the application of supervised learning algorithms like Random Forest, Naïve-Bayes, and XGBoost.

	EXT1	EXT2	EXT3	EXT4	EXT5	...	OPN2	OPN3	OPN4	OPN5	cluster
0	4.0	1.0	5.0	2.0	5.0	...	1.0	4.0	1.0	4.0	1
1	3.0	5.0	3.0	4.0	3.0	...	2.0	4.0	2.0	3.0	0
2	2.0	3.0	4.0	4.0	3.0	...	1.0	2.0	1.0	4.0	0
3	2.0	2.0	2.0	3.0	4.0	...	2.0	5.0	2.0	3.0	4
4	3.0	3.0	3.0	3.0	5.0	...	1.0	5.0	1.0	5.0	1
5	3.0	3.0	4.0	2.0	4.0	...	1.0	5.0	1.0	3.0	1
6	4.0	3.0	4.0	3.0	3.0	...	2.0	4.0	3.0	4.0	0
7	3.0	1.0	5.0	2.0	5.0	...	1.0	3.0	1.0	5.0	1
8	2.0	2.0	3.0	3.0	4.0	...	1.0	5.0	1.0	4.0	0
9	1.0	5.0	3.0	5.0	2.0	...	1.0	3.0	1.0	3.0	5

Figure 62: Cluster predictions for dataset

The average answer to each question group is obtained to identify the Big Five trait distribution for each cluster. This will be used to identify the Big Five trait distribution of candidates belonging to each cluster.

```
data = pd.DataFrame()
data['openness'] = df_model[opn].sum(axis=1)/5
data['conscientiousness'] = df_model[csn].sum(axis=1)/5
data['extraversion'] = df_model[ext].sum(axis=1)/5
data['agreeableness'] = df_model[agr].sum(axis=1)/5
data['neuroticism'] = df_model[est].sum(axis=1)/5
data['cluster'] = predictions
data.groupby('cluster').mean()
```

cluster	openness	conscientiousness	extraversion	agreeableness	neuroticism
0	3.083385	3.117457	2.994411	2.708733	3.256719
1	3.124081	3.102679	3.253143	2.829100	2.898948
2	3.091929	3.122257	2.769948	2.927402	3.347435
3	3.165879	3.088749	3.129183	3.165834	3.399048
4	3.173568	3.173206	3.121892	2.971011	3.269334
5	2.959557	2.882248	2.823555	2.880840	2.754992

Figure 63: Big Five trait distributions for the six personality clusters

2.4.2.2. Open-ended responses dataset

The open-ended responses are processed using Natural Language Processing techniques to identify the keywords that can be used to determine the presence of each trait from candidate responses.

The first step in the process of keyword extraction is text cleaning which includes converting the text to lowercase and removing non-word and whitespace characters as well as digits.

```
# convert to lowercase
responses_df = responses_df.applymap(lambda x: x.lower() if isinstance(x, str) else x)

# remove non-word and non-whitespace characters
responses_df = responses_df.replace(to_replace=r'[^\w\s]', value='', regex=True)

# remove digits
responses_df = responses_df.replace(to_replace=r'\d', value='', regex=True)
```

Figure 64: Text cleaning of open-ended responses

Next, the responses undergo tokenization, where they are split into individual words using the ‘word_tokenize’ function.

```
[ ]  # Tokenize the text data
     responses_df['Openness'] = responses_df['Openness'].apply(word_tokenize)
     responses_df['Concientiousness'] = responses_df['Concientiousness'].apply(word_tokenize)
     responses_df['Extraversion'] = responses_df['Extraversion'].apply(word_tokenize)
     responses_df['Agreeableness'] = responses_df['Agreeableness'].apply(word_tokenize)
     responses_df['Neuroticism'] = responses_df['Neuroticism'].apply(word_tokenize)
```

Figure 65: Text tokenization of open-ended responses

Next, stopwords are removed from the tokenized words to remove commonly used words such as ‘the’, ‘an’, and ‘a’ that do not provide meaning.

```
[ ]  # Remove stop words
     stop_words = set(stopwords.words('english'))

     responses_df['Openness'] = responses_df['Openness'].apply(lambda x: [word for word in x if word not in stop_words])
     responses_df['Concientiousness'] = responses_df['Concientiousness'].apply(lambda x: [word for word in x if word not in stop_words
     responses_df['Extraversion'] = responses_df['Extraversion'].apply(lambda x: [word for word in x if word not in stop_words])
     responses_df['Agreeableness'] = responses_df['Agreeableness'].apply(lambda x: [word for word in x if word not in stop_words])
     responses_df['Neuroticism'] = responses_df['Neuroticism'].apply(lambda x: [word for word in x if word not in stop_words])
```

Figure 66: Stopword removal from open-ended responses

The next step of extracting keywords from the dataset is Lemmatization. Lemmatization helps in improving the accuracy of text analysis by reducing words to their base or dictionary form [11]. This makes it easier to identify and analyze words that have similar meanings. The ‘WordNetLematizer’ function of the nltk library is used for this purpose.

```
import nltk
from nltk.stem import WordNetLemmatizer
from nltk.corpus import wordnet
import pandas as pd

# initialize lemmatizer
lemmatizer = WordNetLemmatizer()

# define function to lemmatize tokens
def lemmatize_tokens(tokens):
    # convert POS tag to WordNet format
    def get_wordnet_pos(word):
        tag = nltk.pos_tag([word])[0][1][0].upper()
        tag_dict = {"J": wordnet.ADJ,
                    "N": wordnet.NOUN,
                    "V": wordnet.VERB,
                    "R": wordnet.ADV}
        return tag_dict.get(tag, wordnet.NOUN)

    # lemmatize tokens
    lemmas = [lemmatizer.lemmatize(token, get_wordnet_pos(token)) for token in tokens]

    # return lemmatized tokens as a list
    return lemmas

# apply lemmatization function to column of dataframe
responses_df['Openness_lemmatized'] = responses_df['Openness'].apply(lemmatize_tokens)
```

Figure 67: Lemmatization of open-ended responses

After completing the above steps, the resulting data consists of a set of words in each response.

Openness_lemmatized	Concientiousness_lemmatized	Extraversion_lemmatized	Agreeableness_lemmatized	Neuroticism_lemmatized
[strongly, believe, seek, new, experience, tak...	[prioritize, work, accord, deadline, level, im...	[im, quite, comfortable, social, situation, al...	[try, view, feedback, criticism, opportunity, ...	[face, difficult, situation, try, remain, calm...
[someone, thrives, novelty, excitement, make, ...	[believe, accuracy, attention, detail, critica...	[im, outgo, person, nature, social, situation,...	[initially, tough, receive, criticism, negativ...	[experience, one, effective, way, handle, stre...
[naturally, inclined, seek, new, experience, t...	[manage, time, effectively, planning, ahead, a...	[someone, value, personal, relationship, im, a...	[handle, feedback, stay, openminded, actively,...	[deal, stressful, situation, like, take, proac...
[personal, professional, growth, two, top, pri...	[ensure, work, accurate, complete, time, follo...	[wouldnt, necessarily, describe, extroverted, ...	[take, feedback, seriously, also, try, take, p...	[find, deal, stressful, situation, communicati...
[natural, risktaker, recognize, value, seek, n...	[believe, set, realistic, goal, key, complete,...	[someone, naturally, extroverted, thrive, soci...	[view, feedback, opportunity, strengthen, rela...	[experience, effective, way, handle, stressful...
[someone, enjoys, take, risk, seek, new	[accuracy, timeliness, essential,	[ocial, situation, always,	[take, proactive, approach,	[face, difficult, situation,

Figure 68: Set of words from the open-ended responses

The final step is to extract the keywords for each trait from the above data. These keywords will be the indicators of the presence of each trait. Keyword extraction was done using 3 methods. The top 40 keywords with the highest word frequencies were considered.

a) Keyword extraction using the Bag of Words (BOW) method

This method considers the term frequency of each word occurring in the dataset. The 40 words with the highest term frequency will be returned as the keywords.

```
# OPENNESS

# Create a bag of words representation of the text data
vectorizer = CountVectorizer()
bag_of_words = vectorizer.fit_transform(responses_df['Openness_lemmatized'].apply(lambda x: ' '.join(x)))

# Get the sum of the counts of each word in the corpus
sum_words = bag_of_words.sum(axis=0)

# Get the frequency of each word in the corpus
word_freq = [(word, sum_words[0, idx]) for word, idx in vectorizer.vocabulary_.items()]

# Sort the list of words by frequency in descending order
word_freq = sorted(word_freq, key=lambda x: x[1], reverse=True)

# Print the top 40 most common words
print('Bag of words representation for Openness trait:\n')

Openness_top40 = []

for word, freq in word_freq[:40]:
    print(word, freq)
    Openness_top40.append(word)
```

Figure 69: Keyword extraction using Bag of Words

b) Keyword extraction using the TF-IDF method

This method considers both term frequency and inverse document frequency to identify keywords.

```
# create a TfidfVectorizer
tfidf = TfidfVectorizer()

# fit and transform the text column
tfidf_matrix = tfidf.fit_transform(df['Openness'])

# get the sum of the TF-IDF scores for each word
tfidf_sum = tfidf_matrix.sum(axis=0)

# convert the sum to a list of tuples (word, score)
word_scores = [(word, tfidf_sum[0, idx]) for word, idx in tfidf.vocabulary_.items()]

# sort the list by score (in descending order)
word_scores.sort(key=lambda x: x[1], reverse=True)

# print the top 40 most frequent words
print('Most frequently used words for Openness trait:\n')

Openness_tokens = []

for word, score in word_scores[:40]:
    #print(f'{word}: {score}')
    Openness_tokens.append(word)
```

Figure 70: Keyword extraction using TF-IDF

c) Keyword extraction using the KeyBERT method

KeyBERT is a keyword extraction technique that uses BERT embeddings to identify the keywords that best capture the essence of the given text document. [12].

```python
from keybert import KeyBERT

def extract_keywords_from_dataframe(df, text_column, num_keywords=5):
    # Initialize the KeyBERT model
    model = KeyBERT()

    # Concatenate all text phrases in the column into a single string
    text = ' '.join(df[text_column])

    # Extract keywords from the concatenated text
    keywords = model.extract_keywords(text, keyphrase_ngram_range=(1, 1), stop_words='english', top_n=num_keywords)

    # Return the extracted keywords
    return [keyword for keyword, _ in keywords]
```

Figure 71: Keyword extraction using KeyBERT

2.4.3. Determining candidate Big Five trait distribution

After extracting the keywords, a keyword list was prepared for each keyword extraction method. These lists consist of 5 columns, each consisting of keywords that will be used to identify each of the Big Five traits.

	Openness_tokens	Concientiousness_tokens	Extraversion_tokens	Agreeableness_tokens	Neuroticism_tokens
0	new	work	new	feedback	stay
1	risk	time	connection	try	situation
2	take	ensure	people	take	try
3	life	task	make	improve	help
4	im	complete	try	make	take
5	professional	project	others	believe	problem
6	personal	accurately	relationship	use	make
7	try	team	social	receive	find
8	believe	include	meeting	stay	stress
9	learn	also	situation	approach	solution
10	opportunity	make	able	understand	manage
11	potential	deadline	building	ask	difficult

Figure 72: Keyword extraction using KeyBERT

The candidate responses to the open-ended questions were subjected to the process of text cleaning, tokenization, stopword removal, and lemmatization. Afterward, keywords were extracted from the candidate responses using the three methods mentioned above.

The keywords extracted from the candidate responses are compared against the prepared keyword lists to identify common words. The score for each trait will be calculated as follows,

$$Openness\_score(opn\_ended) = (number\ of\ common\ words\ /\ 40) * 5$$

Equation 1: Equation to calculate candidate's open-ended score

A score was calculated for each of the Big Five traits for the keywords extracted using all 3 keyword extraction methods. By looking at the scores, it was determined that TF-IDF keywords extraction method was the most successful in identifying the keywords that can be used to identify the presence of Big Five traits in candidates.

The score from the self-rating responses will be calculated by taking the average value for each question group as follows,

$$Openness\_score(self\_rating) = (OPN1 + OPN2 + OPN3 + OPN4 + OPN5)/5$$

Equation 2: Equation to calculate candidate's self-rating score

Finally, a single score is obtained by considering both the self-rating and open-ended scores of the candidate. The self-rating score is given a weight of 40% to the final score, whereas the open-ended score is given a weight of 60%.

$$Candidate\_score(openness) = (self\_rating(openness) * 0.4) + (open\_ended(openness) * 0.6)$$

Equation 3: Equation to calculate candidate's personality score

2.4.4. Determining the expected Personality Requirement for the Job Role

The expected personality traits are taken as a list of ten traits needed in the workplace. These traits are rated on a scale of 1 to 5 by the recruiter to indicate the importance of each trait for a specific job role (Figure 20). Then, these ten traits are mapped to the Big Five traits [13] to obtain the expected Big Five trait distribution.

```
Innovative                  5
Fast learner                4
Organization skills         3
Attention to detail         4
Assertiveness               2
Leadership skills           2
Team Player                 5
Communication skills        3
Confidence                  3
Adaptability to changes     4
Name: 0, dtype: object
```

Figure 71: Traits required for the job role

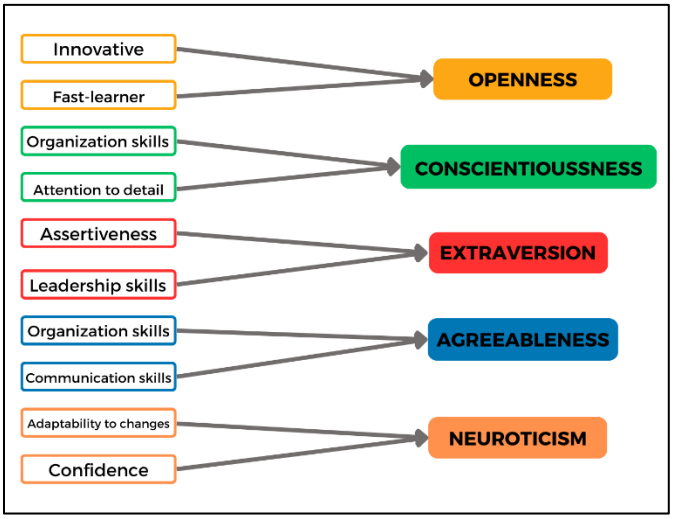


Figure 74: Mapping the personality traits to Big Five traits

```
# Map the personality requirements with Big Five Traits

expected['openness'] = [(data['Fast learner'] + data['Innovative'])/2]
expected['conscientiousness'] = [(data['Attention to detail'] + data['Organization skills'])/2]
expected['extraversion'] = [(data['Assertiveness'] + data['Leadership skills'])/2]
expected['agreeableness'] = [(data['Team Player'] + data['Communication skills'])/2]

# expected (anti-)neuroticism score
expected['neuroticism'] = [(data['Confidence'] + data['Adaptability to changes'])/2]

# Show the updated DataFrame
print(expected)
```

```
   openness  conscientiousness  extraversion  agreeableness  neuroticism
0       4.5                3.5           2.0            4.0          3.5
```

Figure 75: Obtaining the expected Big Five trait distribution

Figure 22 above depicts how the expected Big Five trait distribution was obtained. This distribution will be compared against the candidate's Big Five trait distribution to evaluate how closely the candidate's personality aligns with the requirements of the job role.

2.4.5. Evaluating candidate personality - job role fit

The final personality score for the candidate is calculated by taking into consideration the candidate's Big Five Trait distribution and the expected Big Five trait distribution obtained in the sections discussed above. The final score is calculated as follows,

$$\text{Score} = \text{S}\left(\frac{trait(candidate)}{trait(expected)}\right) * \frac{100}{5}$$

Equation 4: Equation to calculate candidate's personality-job role fit

2.4.6. Frontend development for personality assessment

The frontend development utilized HTML, CSS, and JavaScript technologies, while Flask served as the backend engine to facilitate the connection between the frontend and the Machine Learning and NLP algorithms.

The primary objective in the development of user interfaces was to create user-friendly interfaces that enable recruiters to quickly gain comprehensive insights into a candidate's personality.

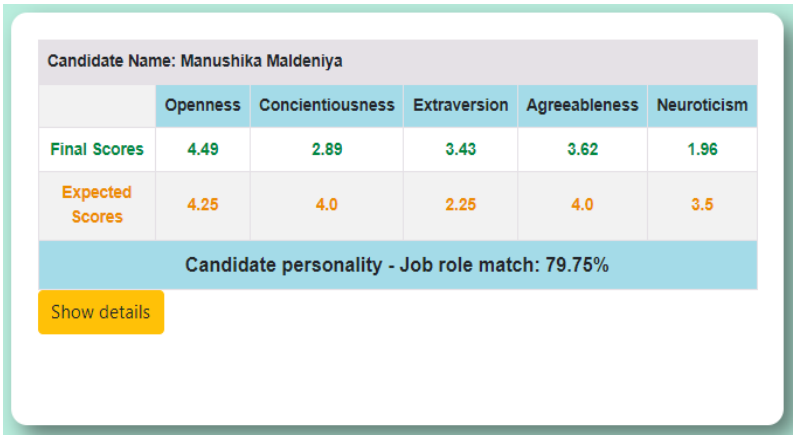


Figure 76: User interface – screenshot 01

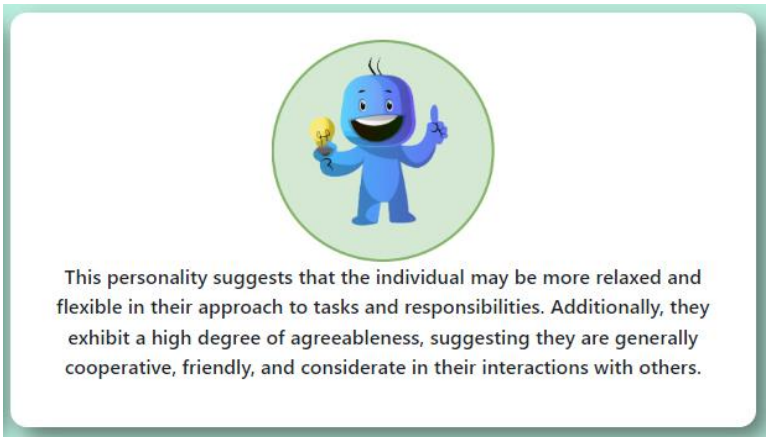


Figure 77: User interface – screenshot 02

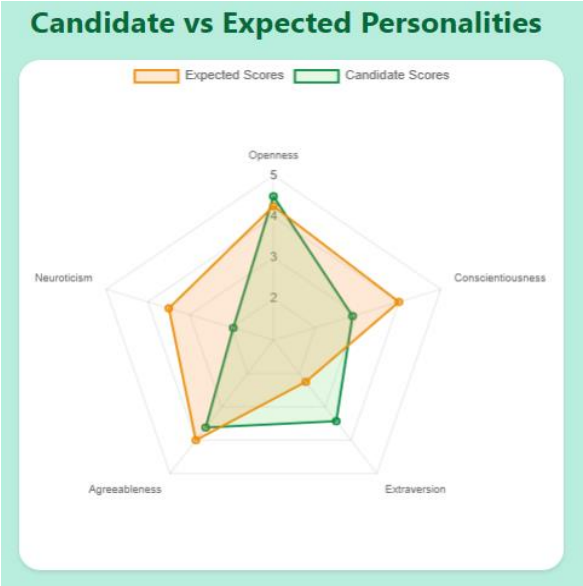


Figure 78: User interface – screenshot 03

2.5.Candidate Profile Creation

After evaluating the candidate's resume, academic transcript, professional competencies, and personality traits, a comprehensive profile is generated to provide a concise overview of the candidate's qualifications. This profile serves as a valuable tool for recruiters to gauge the candidate's skill set and suitability for the desired job position.

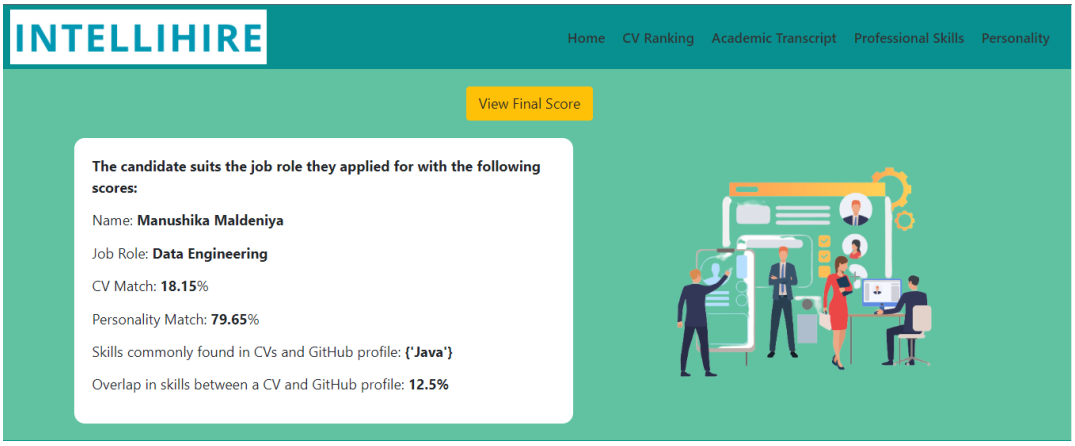


Figure 79: Candidate profile

Furthermore, a cross-validation process is conducted to compare the skills listed in the candidate's resume with those found in their GitHub profile. As illustrated in the figure above, the cross-validated results are displayed in the candidate's profile, empowering recruiters to enhance the precision of their hiring choices.

2.6. Commercialization Aspect of the Product

‘Intellihire’ is a recruitment software created by ‘SMMS Software Solutions’ to simplify the hiring process in IT companies. Intellihire helps to streamline the recruitment process by using advanced machine learning algorithms and natural language processing techniques to analyze candidate profiles, job requirements, and other relevant data to identify the most suitable candidates for a particular job

Firstly, LinkedIn Skill-based Job Category Prediction, GitHub Programming Language Comparator, and Sentiment Analysis on Reference Checking features cater to the specific needs of IT companies, making the hiring process more efficient. For instance, LinkedIn Skill-based Job Category Prediction helps match candidates to job categories based on their skills, while GitHub Comparator assists in evaluating a candidate's programming skills. Sentiment Analysis on Reference Checking customizes reference checks to provide deeper insights into a candidate's background.

Another critical aspect is the Personality Prediction feature, which simplifies the evaluation of a candidate's personality traits, saving time for hiring managers and ensuring a better fit for the organization. This is especially crucial in the fast-paced IT industry, where finding the right candidate quickly is essential. Moreover, Intellihire offers a user-friendly interface and can be customized for different organizations, enhancing its usability.

Furthermore, the Academic Transcript Analysis and CV Analysis and Ranking components offer unique value. The Academic Transcript Analysis streamlines the

identification of top talent among recent graduates, while the CV Analysis and Ranking system uses AI to match CVs with job descriptions accurately. Additionally, the introduction of a chatbot for generating job descriptions enhances the entire recruitment process by providing consistent and accurate job listings.

Intellihire presents a comprehensive and user-friendly solution for IT companies to streamline their recruitment processes. It offers various features like LinkedIn and GitHub Analysis, personality prediction, academic transcript analysis, and advanced CV analysis, making it a valuable tool for the modern HR industry. The customization options, user interface, and efficiency improvements make it a promising product for commercialization, addressing the specific needs of IT recruitment while simplifying the process for professionals in the field.

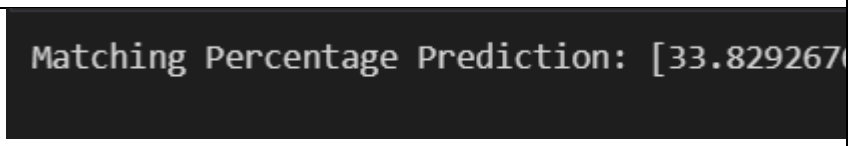
Figure 80: Logo - Intellihire

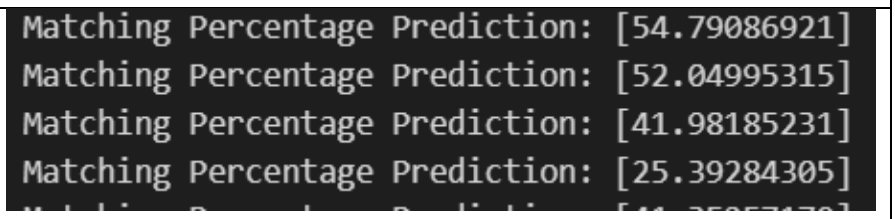
Figure 81: Logo - SMMS software solutions

2.7. Testing and Implementation

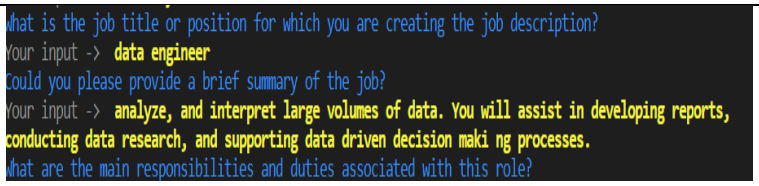
In the context of optimizing the recruitment process, rigorous testing plays a pivotal role in ensuring the effectiveness and reliability of the system. This section of the report delves into the comprehensive testing efforts undertaken to evaluate the system's functionality and performance. By systematically assessing various facets of the recruitment optimization system, this testing phase aims to provide critical insights into its capabilities, identify potential areas for enhancement, and ultimately contribute to the system's successful deployment within the recruitment workflow.

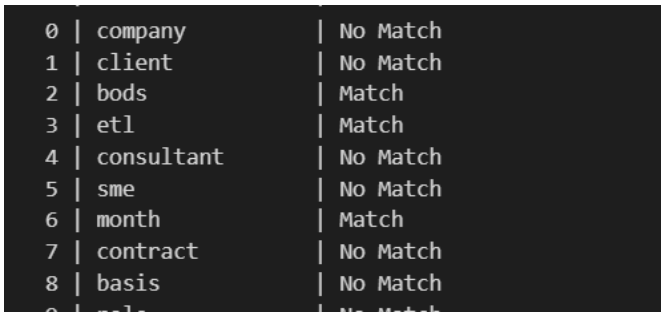
2.7.1. CV Ranking and Job description Matching

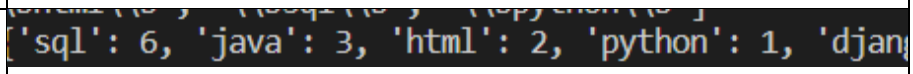
Test Case No	**01**
Description	Matching a resume with job description
Input	Input resume of a candidate Input the related job description for the position candidate applied for. Input required skills for the position.
Expected Output	Matching percentage
Actual Output	
Result	Pass

Test Case No	**02**
Description	Rank resumes of candidates according to the matching percentages
Input	1. Input resumes of candidates 2. Input the related job description for the position candidate applied for. 3. Input required skills for the position.
Expected Output	Ranked matching percentages of candidates from the highest score to the lowest
Actual Output	
Result	Pass

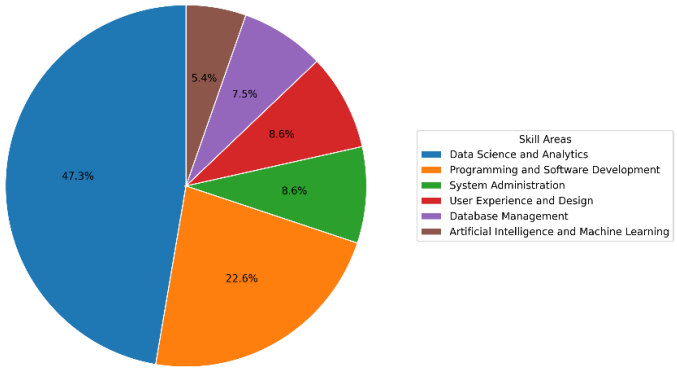
Test Case No	**03**
Description	Create a job description for a specific job position
Input	Input responses for the prompted questions of the chatbot.
Expected Output	Job description

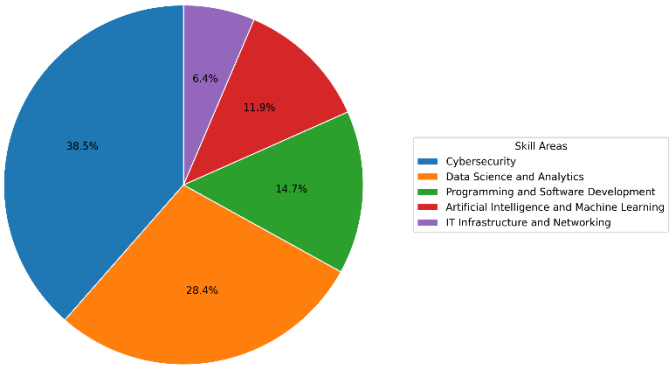
Actual Output	
Result	Pass

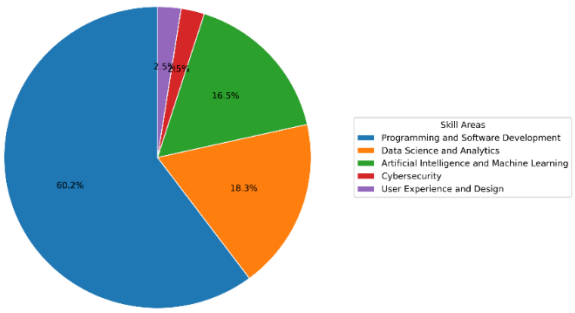
Test Case No	**04**
Description	Get Matching keywords
Input	Input candidate resume Input job description
Expected Output	Matching and Not Matching keywords
Actual Output	
Result	Pass

Test Case No	**05**
Description	Match and get skills of resume
Input	Input candidate resume Input job description
Expected Output	Matching skills in resume and job description
Actual Output	
Result	Pass

2.7.2. Academic Transcript Analysis

Test Case 01	
Description	When the academic transcript is uploaded, the skill areas should be categorized and shown in a pie chart
Input	An academic transcript of a data science specialized candidate
Expected output	Categorized skill areas with the higher weight for Data Science and Analytics.
Actual Output	
Test Status	Pass

Test Case 02	
Description	When the academic transcript is uploaded, the skill areas should be categorized and shown in a pie chart
Input	An academic transcript of a cyber security specialized candidate
Expected output	Categorized skill areas with the higher weight for Cyber-security
Actual output	
Test Status	Pass

Test Case 03	
Description	When the academic transcript is uploaded, the skill areas should be categorized and shown in a pie chart
Input	An academic transcript of a software engineering specialized candidate
Expected output	Categorized skill areas with the higher weight for Programming and Software development
Actual output	
Test Status	Pass

Test Case 04	
Description	When the academic transcript is uploaded, the skill areas should be categorized and shown in a pie chart. The score is given based on the summation of weighted grades of all skill categories
Input	An academic transcript of a data science specialized candidate who applied for the Data Scientist position
Expected output	A score greater than 70%
Actual Output	**Position Applied: Data Scientist** **Score: 71.51%**
Test Status	Pass

Test Case 05	
Description	When the academic transcript is uploaded, the skill areas should be categorized and shown in a pie chart. The score is given based on the summation of weighted grades of all skill categories
Input	An academic transcript of a data science specialized candidate who applied for the Software Engineer position
Expected Output	A score between 50% to 75%
Actual output	**Position Applied: Software Engineer** **Score: 59.03%**
Test Status	Pass

Test Case 06	
Description	When the academic transcript is uploaded, the skill areas should be categorized and shown in a pie chart. The score is given based on the summation of weighted grades of all skill categories
Input	An academic transcript of a cyber security specialized candidate who applied for the Cyber Security Engineer position
Expected output	A score greater than 70%
Actual output	**Position Applied: Cyber Security Engineer** **Score: 77.43%**
Test Status	Pass

2.7.3. Professional Skills Analysis

Test Case	**01**
Description	Testing LinkedIn skills-based job category classification for famous personality
Input	`# Example LinkedIn profile URL` `linkedin_profile_url = 'https://www.linkedin.com/in/chanuxbro/'`
Expected Outcome	Executive Leadership / Infrastructure and Operations

Actual output	`# Example LinkedIn profile URL`
	`linkedin_profile_url = 'https://www.linkedin.com/in/chanuxbro/'`
	
	`# Call the function to get skills`
	`skills = scrape_linkedin_skills(linkedin_profile_url)`
	
	`predicted_category = model.predict(fitted_vectorizer.transform(skills))`
	`result = f"Predicted Job Category: {predicted_category[0]}"`
	
	`print(result)`
	
	`Predicted Job Category: Insfastructure and Operations`

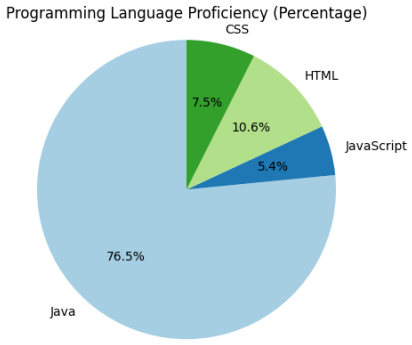
Result	Pass

Test Case	**02**
Description	Testing LinkedIn skills-based job category classification for a team member
	`# Example LinkedIn profile URL` `linkedin_profile_url = 'https://www.linkedin.com/in/maleesha-de-silva-37a26a221/'`
Expected Outcome	Software Engineering

Actual output	`# Example LinkedIn profile URL`
	`linkedin_profile_url = 'https://www.linkedin.com/in/maleesha-de-silva-37a26a221/'`
	
	`# Call the function to get skills`
	`skills = scrape_linkedin_skills(linkedin_profile_url)`
	
	`predicted_category = model.predict(fitted_vectorizer.transform(skills))`
	`result = f"Predicted Job Category: {predicted_category[0]}"`
	
	`print(result)`
	`✓ 0.8s`
	
	`Predicted Job Category: Software Development and Engineering`

Result	Pass

Test Case	03
Description	Testing programming language proficiency of students in SLIIT
Input	`# Fetch user information` `username = "Maldeniya99"`
Expected Outcome	Proficient in Java

Actual output	
	`mming Language Proficiency (Percentage):`
	`: 76.55%`
	`Script: 5.38%`
	`: 10.61%`
	`7.45%`

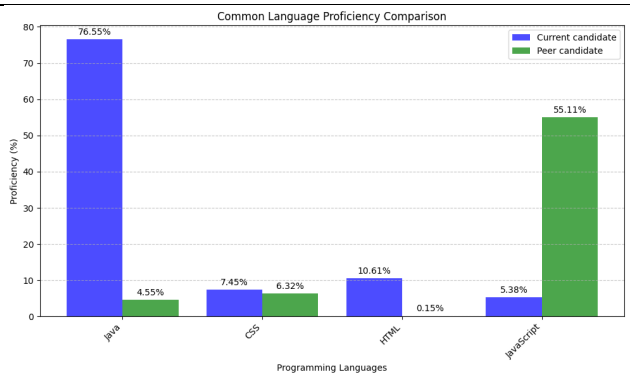
Result	Pass

Test Case	04
Description	Testing programming language proficiency of students in SLIIT
Input	`# Fetch user information` `username = "Maldeniya12345"`
Expected Outcome	There should not be a user account under the name: Maldeniya12345

Actual output	1] ✓ 0.7s · User 'Maldeniya12345' not found.
Result	Pass

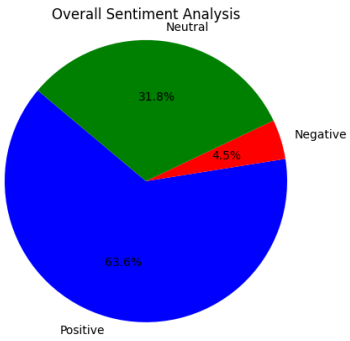
Test Case	**05**
Description	Testing Programming language proficiency for two peer candidates.

Input	`# Fetch user information`
	`current_username = "Maldeniya99"`
	`current_user = g.get_user(current_username)`
	
	`other_username = "IT20207854"`
	`other_user = g.get_user(other_username)`

Expected Outcome	Common languages – Java, CSS, HTML, Javascript
Actual output	
Result	Pass

Test Case	**06**
Description	Testing and applying sentiment analysis truly filled form.

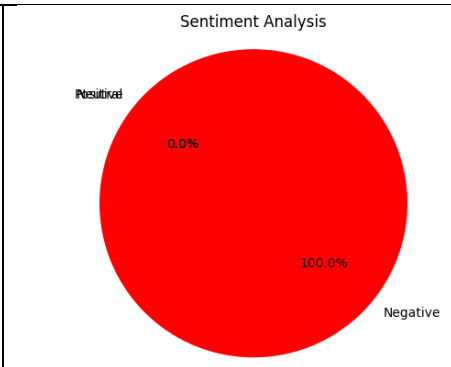
Input	A CSV file containing referee responses.
	
	`# Read CSV file into a DataFrame`
	`csv_file_path = 'Responses.csv'`
	`data = pd.read_csv(csv_file_path)`

Expected Outcome	There should be real-world output with a mix of all sentiments.
Actual output	
Result	Pass

Test Case	**07**
Description	Testing and applying sentiment analysis purposely filling the Google form with negative.

Input	A CSV file containing the referee's responses (intentionally negatively filled)
	
	`# Read CSV file into a DataFrame`
	`csv_file_path = 'Responses.csv'`
	`data = pd.read_csv(csv_file_path)`

Expected Outcome	Negative Sentiment should be the highest one.

Actual output	
Result	Pass

2.7.4. Personality Assessment of Candidates

Test case Number	**01**
Description	Testing the predictions of the K-means clustering model – when the candidate provides answers to the self-rating questions(on a scale of 1 to 5) the personality cluster the candidate belongs to should be displayed.
Input	3,2,1,1,2,1,2,2,1,4,2,5,4,4,4,5,5,4,4,4,5,3,3,4,5
Expected Output	A number from [0] to [5] (to represent the six clusters)
Actual Output	[3]
Test Result	Pass

Test case Number	02

Description	Testing the keyword extraction from open-ended response
Input	“I am someone who enjoys taking on new challenges and seeking out new experiences.”
Expected Output	'someone', 'enjoy', 'take', 'new', 'challenge', 'seek', 'new', 'experiences.'
Actual Output	['someone', 'enjoy', 'take', 'new', 'challenge', 'seek', 'new', 'experiences.']
Test Result	Pass

Test case Number	03
Description	Testing the mapping of job role requirements to Big Five traits – when the recruiter provides the requirement for the job role it should be mapped to the Big Five traits
Input	Innovative 5 Fast learner 4 Organization skills 3 Attention to detail 4 Assertiveness 2 Leadership skills 2 Team Player 5 Communication skills 3 Confidence 3 Adaptability to changes 4
Expected Output	Openness 3.5 Conscientiousness 4.4 Extraversion 2.0 Agreeableness 4.0 Neuroticism 3.5

Actual Output	openness conscientiousness extraversion agreeableness neuroticism 4.5 3.5 2.0 4.0 3.5
Test Result	Pass

3. RESULTS AND DISCUSSION

3.2. Results

3.2.1. CV Ranking and Job description Matching.

The main objective of this fucntion is to choose the best candidate by analyzing cv with the job description. The following will show the results obtained .

Generation of Job description using a chatbot

During the development of the chatbot, a crucial step involves prompting users with questions to gather their inputs, which are then validated based on the length of the provided responses. The implementation of this methodology underscores the chatbot's commitment to quality user interactions and accurate information retrieval. By adhering to a sequential questioning strategy and employing length-based validation, the chatbot can better comprehend user requirements and provide more effective, relevant, and personalized responses.

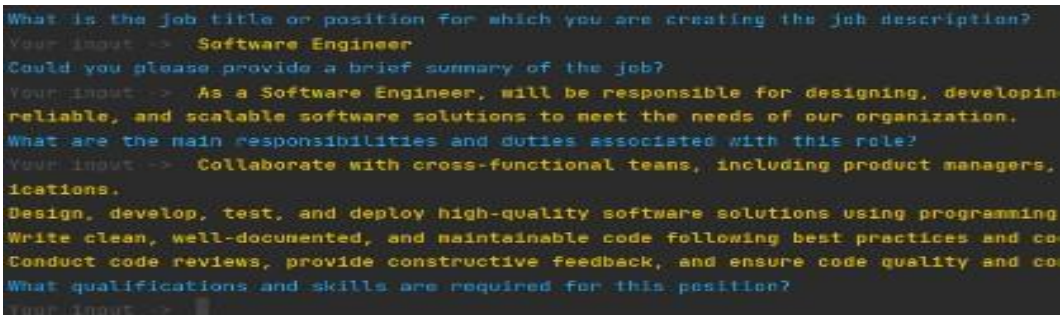


Figure 72: Generating Job Description through Bot

The test involved a meticulously designed set of 35 questions, strategically arranged in a sequential order to simulate a job-related inquiry. Remarkably, the chatbot demonstrated exceptional proficiency, as it accurately responded to all questions with a perfect accuracy score of 1.00.

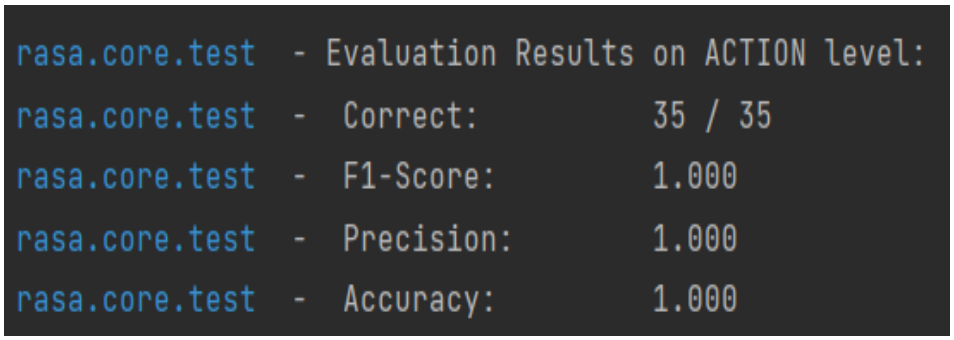


Figure 73: Accuracy of generating bot

Predicting matching percentage of resume and job description

In the initial attempts to match resumes with job descriptions using machine learning models, the achieved accuracies fell short of expectations. Consequently, a novel approach

was used by developing a stacked ensemble model that combined multiple base models for both TD-IDF and BOW vectors. Through the implementation of hyperparameter tuning and advanced cross-validation techniques, remarkable performance was accomplished.

The models and their accuracies for both TD-IDF vectors and BOW are in the table below.

Model	**Accuracy before Hyper-parameter tuning**	**Accuracy After Hyper-parameter tuning**
The Stacked Model with Count Vectorizer	0.6035	0.6810
The Stacked Model with TF-IDF Vectorizer	0.5371	0.6128

Table 1: Stacked Model Accuracies

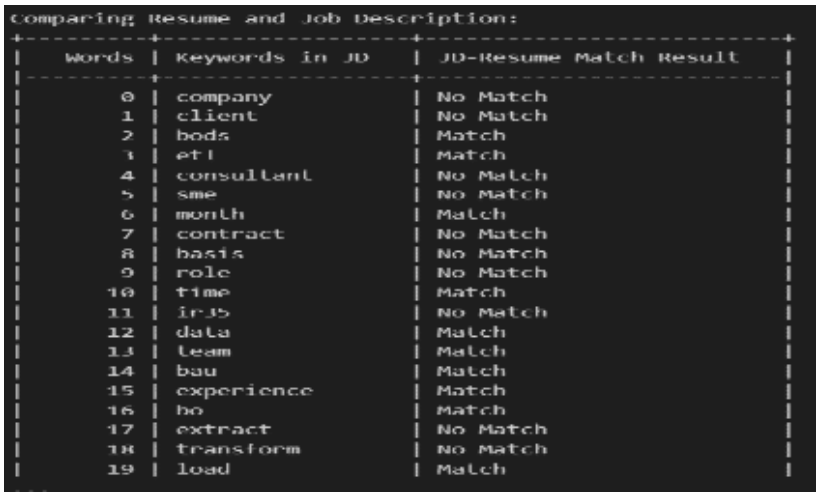
Model	**Hyper parameters at the Initial Stage**	**Hyper parameters after Tuning**
Linear Regressor	alpha=0.1	alpha=0.1, fit_intercept=False, solver='lsqr'
Support Vector Regressor RBF Kernal	kernel='linear'	C=10, epsilon=0.01, kernel='linear'

Random Forest Regressor	n_estimators=50	max_depth=20, n_estimators=50
Decision Tree Regressor	min_samples_split=10	min_samples_split=10
XGBoost	learning_rate=1	learning_rate=1, max_depth=3, n_estimators=150, reg_lambda=1000

Table 2: Hyper Parameter Tunning

Keyword matching and skills extraction

The resumes were matched with job descriptions based on keywords that persisted in them. The results were categorized as 'Match' and 'No Match. As a result of this approach, matching percentages based on keywords and cosine similarity were obtained for each job description.



Comparing Resume and Job Description:

Words	Keywords in JD	JD-Resume Match Result
0	company	No Match
1	client	No Match
2	bods	Match
3	etl	Match
4	consultant	No Match
5	sme	No Match
6	month	Match
7	contract	No Match
8	basis	No Match
9	role	No Match
10	time	Match
11	irj5	No Match
12	data	Match
13	team	Match
14	bau	Match
15	experience	Match
16	bo	Match
17	extract	No Match
18	transform	No Match
19	load	Match

Figure 74 : CV skill Matching

3.2.2. Academic Transcript Analysis

Extracting module keywords and phrases using n-grams

Figure 75: Extracted module keywords and phrases from module outline descriptions

Extracting data from academic transcripts

The text extracted from OCR engine is shown as follows.

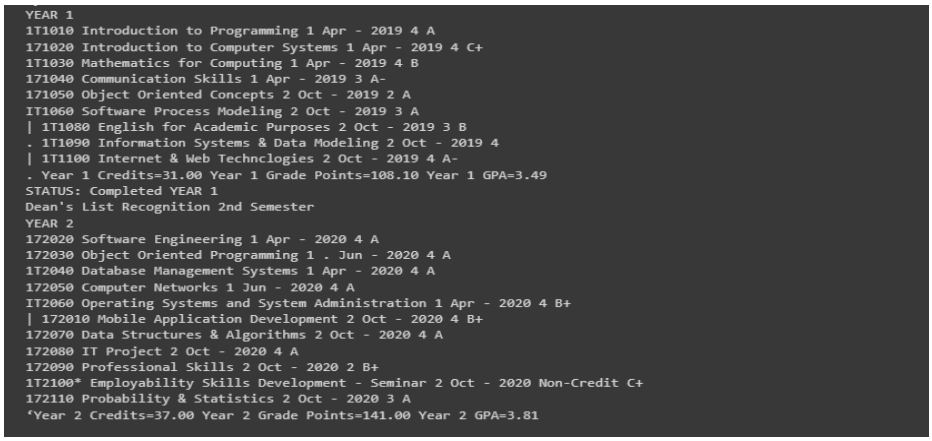


Figure 76: Extracted text from an academic transcript using OCR

Named entity recognition model

Custom NER model

The custom NER model was trained with 10 epochs and had a loss of 2.2673915734306062e-07.

```
Losses: {'ner': 1080.9641413251313}
Losses: {'ner': 81.69892219707003}
Losses: {'ner': 26.13685030728511}
Losses: {'ner': 32.97153806471643}
Losses: {'ner': 2.39907462387285}
Losses: {'ner': 22.977387887101763}
Losses: {'ner': 4.599581692964621e-06}
Losses: {'ner': 1.9882955234192774e-08}
Losses: {'ner': 5.26225924263131e-09}
Losses: {'ner': 2.2673915734306062e-07}
```

Figure 77: Custom NER losses

```
IT2010 Mobile Application Development MODULE_TITLE
2 SEMESTER
Oct - 2020 PERIOD
4 CREDITS
B+ GRADE
IT2070 CODE
Data Structures & Algorithms MODULE_TITLE
2 SEMESTER
Oct - 2020 PERIOD
4 CREDITS
A GRADE
IT2080 CODE
IT Project MODULE_TITLE
2 SEMESTER
Oct - 2020 PERIOD
4 CREDITS
A GRADE
IT2090 CODE
Professional Skills MODULE_TITLE
2 SEMESTER
Oct - 2020 PERIOD
2 CREDITS
B+ GRADE
```

Figure 78: Tested output of custom NER

Pre-trained NER model

The pre-trained Named Entity Recognition (NER) model that was built using the Spacy library was trained with 200 epochs, 128 as the batch size, a learning rate of 0.001, and a loss of 84.75 with a score of 1.00.

E	#	LOSS TOK2VEC	LOSS NER	ENTS_F	ENTS_P	ENTS_R	SCORE
0	0	0.00	47.20	0.00	0.00	0.00	0.00
1	200	258.80	1839.42	99.42	99.47	99.37	0.99
2	400	13.28	35.24	99.61	99.61	99.61	1.00
4	600	21.16	30.47	99.64	99.56	99.71	1.00
6	800	34.70	43.84	99.68	99.66	99.71	1.00
8	1000	38.80	52.18	99.78	99.81	99.76	1.00
12	1200	92.93	73.50	99.66	99.66	99.66	1.00
16	1400	329.55	91.60	99.83	99.85	99.81	1.00
20	1600	174.63	81.22	99.88	99.90	99.85	1.00
26	1800	76.73	59.74	99.85	99.85	99.85	1.00
33	2000	72.33	61.78	99.90	99.90	99.90	1.00
42	2200	76.58	60.44	99.88	99.81	99.95	1.00
53	2400	131.30	81.35	99.90	99.90	99.90	1.00
63	2600	194.17	81.45	99.90	99.90	99.90	1.00
74	2800	296.60	67.01	99.88	99.81	99.95	1.00
84	3000	127.52	75.52	99.90	99.90	99.90	1.00
95	3200	198.12	88.36	99.88	99.81	99.95	1.00
105	3400	489.22	91.21	99.90	99.90	99.90	1.00
116	3600	976.41	84.75	99.88	99.81	99.95	1.00

```
✓ Saved pipeline to output directory
model-last
```

Figure 79: Pre-trained NER model losses and score

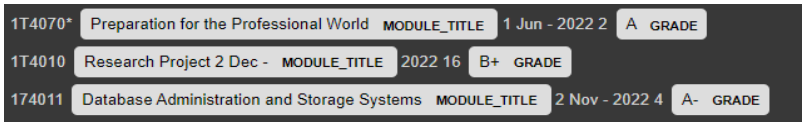


Figure 80: Tested output of pre-trained NER

Thus, the custom NER model was chosen over the pre-trained custom NER model as the loss that was obtained after running 10 epochs was low in the custom NER model.

Model Building for skill area categorization

The SVM model demonstrated the highest accuracy among the evaluated models, achieving an impressive accuracy rate of 0.8810. The Random Forest model closely followed with an accuracy of 0.8784, while the XGBoost model achieved an accuracy of 0.8623. Additionally, the Logistic Regression model yielded an accuracy of 0.7524, and the Naïve Bayes model achieved an accuracy of 0.7980. The hyper-parameters were tuned for a better result. Based on these results, the SVM model was determined to be the most effective and reliable model for skill area prediction.

Model	**Accuracy**
Logistic Regression	75.24%
Naïve Bayes	79.80%
XGBoost	86.23%
Random Forest	87.84%
Support Vector Machine	89.47%

Table 3: Model accuracies of score prediction model

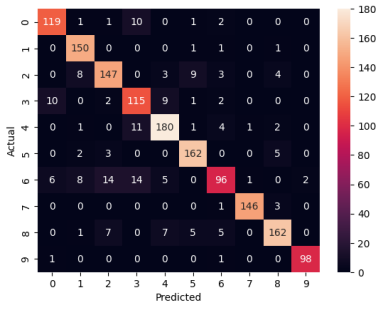


Figure 81: Confusion matrix of the new SVM model

	precision	recall	f1-score	support
Artificial Intelligence and Machine Learning	0.88	0.89	0.88	134
Cloud Computing	0.88	0.98	0.93	153
Cybersecurity	0.84	0.84	0.84	174
Data Science and Analytics	0.77	0.83	0.80	139
Database Management	0.88	0.90	0.89	200
IT Infrastructure and Networking	0.90	0.94	0.92	172
Programming and Software Development	0.83	0.66	0.74	146
Project Management	0.99	0.97	0.98	150
System Administration	0.92	0.87	0.89	187
User Experience and Design	0.98	0.98	0.98	100
accuracy			0.88	1555
macro avg	0.89	0.89	0.88	1555
weighted avg	0.88	0.88	0.88	1555

Figure 82: Classification report of the new SVM model

Graphical representation of skill areas

After categorizing the skills of a candidate, they are visually presented using a graphical representation. In order to visualize, 3 graphical representations were considered. Bar chart, Pie Chart and Radar graph as shown below.

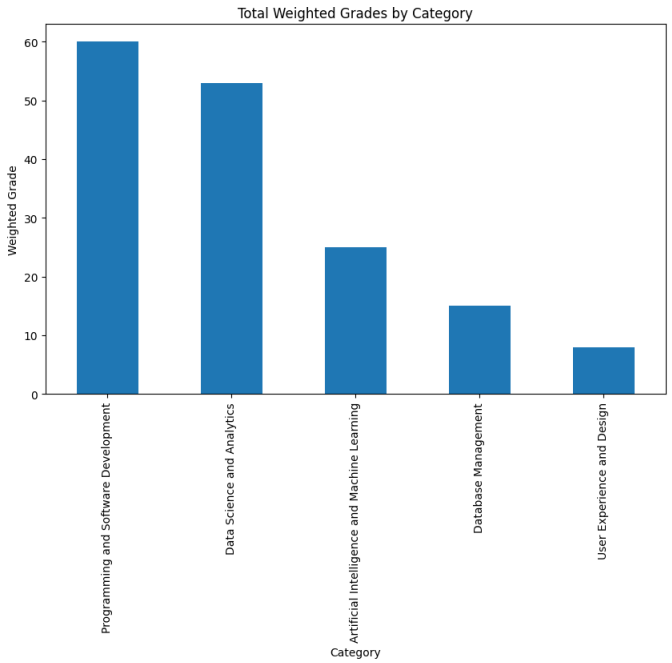


Figure 83: Bar chart representation of skill areas

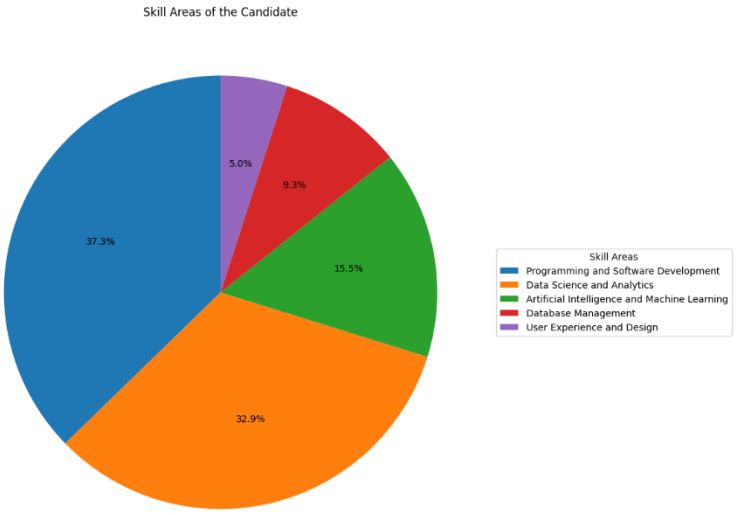


Figure 84: Pie chart representation of skill areas

Figure 85: Radar graph representation of skill areas

Here a pie chart is used to visualize the skill areas. The following are some justifications to why this approach was used.

- Compositional View: Pie charts excel at showing the relative proportions of different categories within a dataset. In the context of skill areas, a pie chart can provide a quick, at-a-glance view of how various skills contribute to the whole.

- Simplicity: Pie charts are straightforward and easy to understand so that the hiring managers can make well-defined decisions.

- Percentage Representation: Each "slice" of the pie can be labeled with the percentage it represents, giving the hiring managers an immediate sense of the distribution. As we need to show the audience the percentages as a whole, a pie chart can be more appropriate.

- Limited Categories: Pie charts work best when you have a relatively small number of categories or skill areas to display. If you have numerous skills, a bar chart or radar graph may become cluttered and less effective.

Reasons to Eliminate the Bar Chart:

- Not Suitable for Composition: Bar charts are better suited for comparing different categories or values, rather than showing the composition of a whole.

- Less Efficient for Few Categories: Since there graph represents a maximum of 10 categories, it would be ideal to display it in a pie chart rather than a bar graph.

- Less Intuitive for Percentages: [9]When using a bar chart, the hiring may need to rely on labels or tooltips to understand the proportion of each skill area relative to the whole, making it less intuitive than a pie chart. Thus, the hiring managers won't

be able to get an understanding at a glance as the bar chart does not represent each category.

Reasons to Eliminate the Radar Graph:

- Complexity: Radar graphs are most suitable for comparing multiple data points across different categories simultaneously. [10]After inputting all the measurements onto the chart, the filled-in area can create distortions in the data since the shaded region becomes a visual indicator of magnitude and favorable performance, potentially leading to misinterpretation. If you only want to represent the distribution of skills without emphasizing the relationships between them, a radar graph can make the data appear unnecessarily complex.

- Difficulty in Interpretation: Radar graphs can be challenging for some viewers to interpret, especially if they are not familiar with this type of chart. [10]And also, the distance on the radii is quite difficult to read thus it can be difficult to quantify. They require more cognitive effort to understand compared to a pie chart, which is more straightforward.

- Not Ideal for Skill Representation: Radar graphs are often used for multi-dimensional data with varying degrees of performance across multiple attributes. Since the primary objective is to display the distribution of skills, a pie chart will be a more appropriate choice.

Due to these reasons, a pie chart was used to graphically represent the skill areas.

Academic transcript score prediction model

The MSE and R-squared(R2) for each regression model is as follows:

Table 4: Mean square errors of academic transcript score prediction model

Model	MSE	R2 score
Random Forest Regressor	35.21	0.65
Logistic Regressor	48.44	0.54
Support Vector Regressor	60.98	0.47
Gradient boost Regressor	30.67	0.71

Among the various models considered, we can see that the Gradient Boost Regressor has the lowest MSE and thus is taken as the best choice and was selected to predict the score. The model was saved and loaded to ensure its availability for predicting academic scores based on job roles. Candidate's job role, job descriptions, and weighted grades for specific skill categories were taken and finally, by using the model, a score was generated based on the skill proficiency and the position the candidate applied for

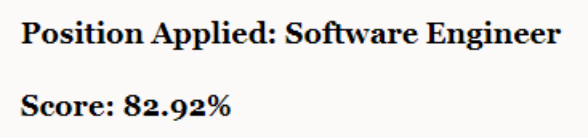


Figure 86: Score based of the skill proficiency and the position

1.2.2. Professional Skill Analysis

1.2.2.1. LinkedIn-based Job Category Prediction

To predict the job category five models were tested for accuracy to select the best performing classification model. These models took the skills as the input, it analyzed the skills, and identified the job category. Further, the hyperparameters for these classifiers are tuned to get better after cross-validation for each. The following tables show the accuracy of each model.

Model	RandomForest Classifier	LinearSVC	MultinomialNB	Logistic Regression	XGB Classifier
Mean Accuracy	0.5790	**0.6638**	0.5505	0.5790	0.4371
SD	0.1199	0.1314	0.1094	0.1432	0.0963

Table 5: Job category prediction accuracy scores

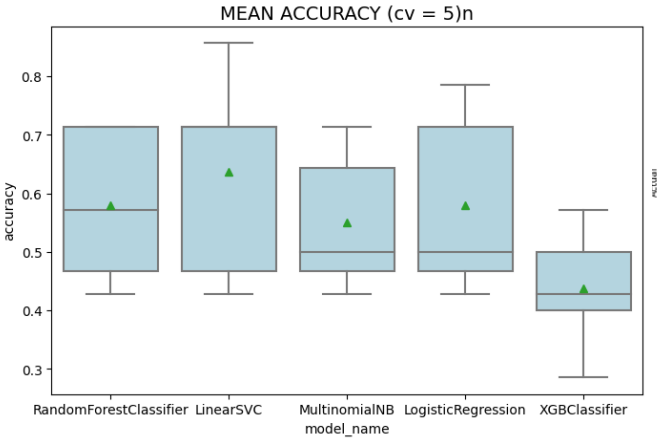


Figure 88: Box plot for accuracy

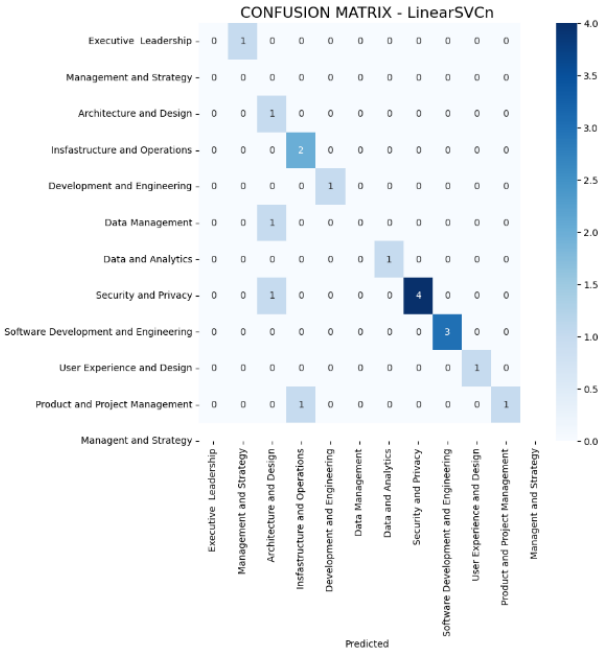
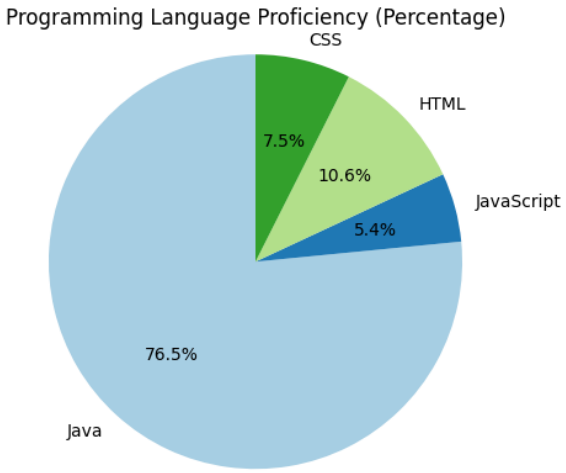


Figure 87: Heatmap for model accuracy

1.2.2.2. GitHub Programming Language Proficiency



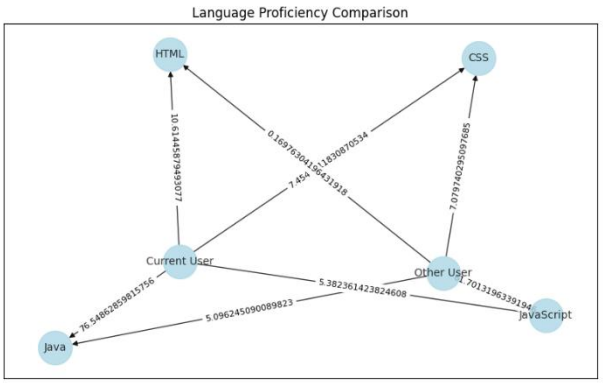


Figure 89: peer candidate comparison

Figure 90:Single candidate PLP visualization

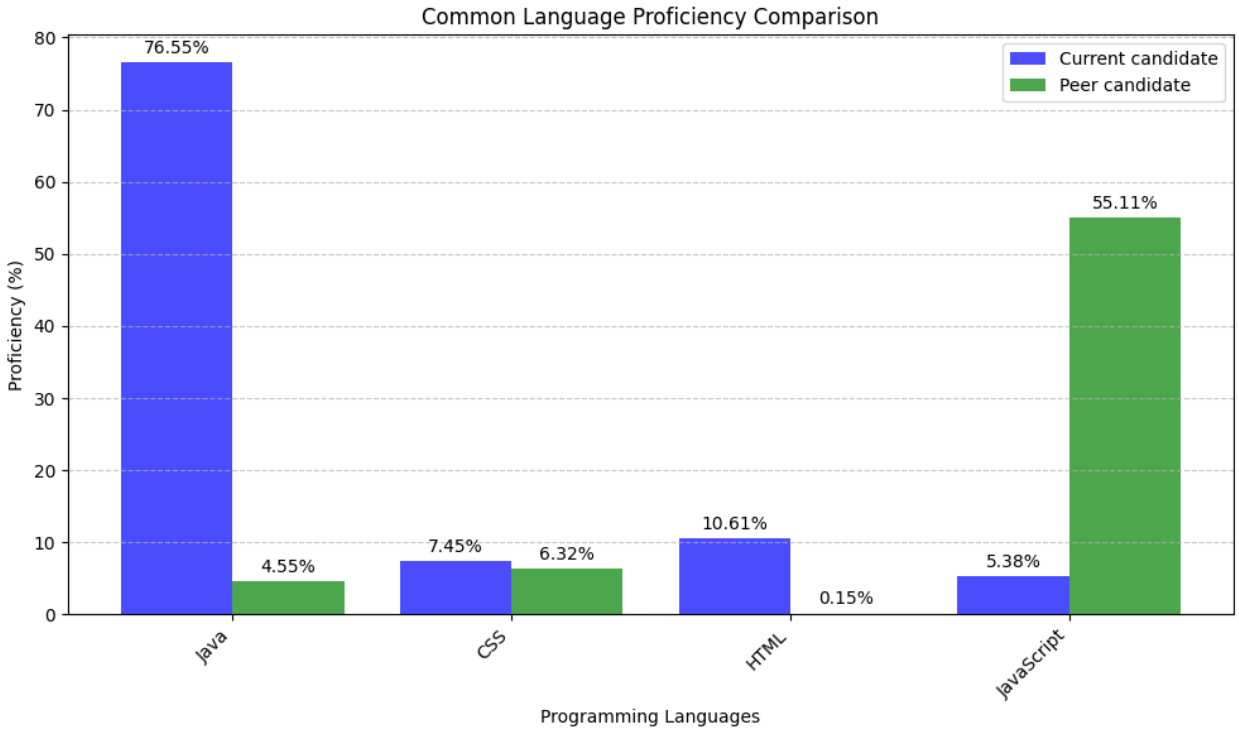


Figure 91: Bar chart visualization for comparison

1.2.2.3. Sentiment analysis on reference checking

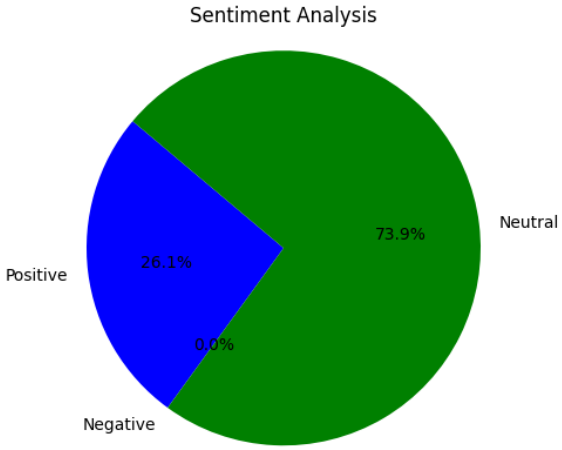


Figure 92:sentiment results for recommendation letters

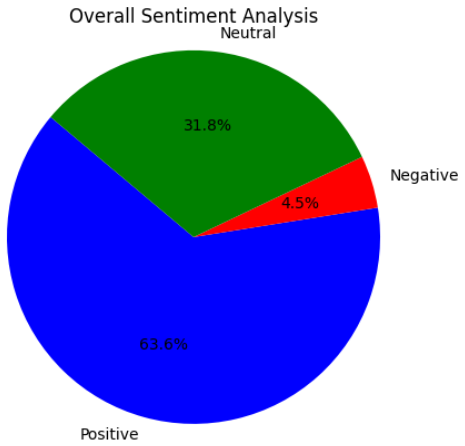


Figure 93: sentiment results for google form responses.

1.2.3. Personality Assessment of Candidates

The Elbow Visualization technique revealed a k-value of 6 indicating that the candidates can be grouped into six distinct personality clusters.

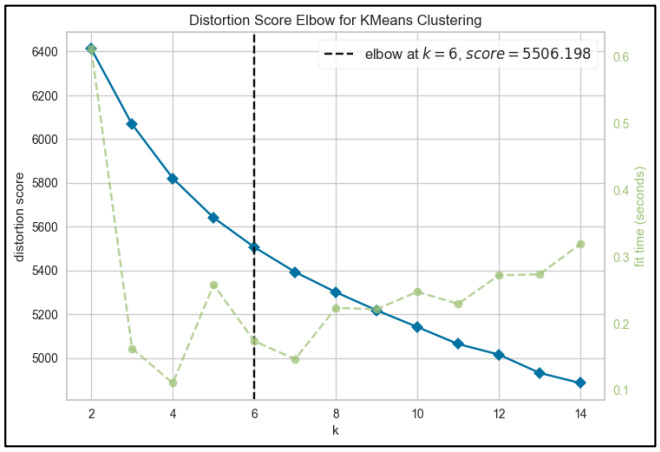


Figure 94: Elbow Visualization for K-means clustering

The Big Five trait distribution for the six clusters was obtained as shown below.

	Openness	Conscientiousness	Extraversion	Agreeableness	Neuroticism
Cluster 0	3.08	3.11	2.99	2.70	3.25
Cluster 1	3.12	3.10	3.25	2.83	2.89
Cluster 2	3.09	3.12	2.77	2.93	2.34
Cluster 3	3.16	3.08	3.12	3.17	2.39
Cluster 4	3.17	3.17	3.12	2.97	3.26
Cluster 5	2.96	2.88	2.82	2.88	2.75

Table 6: Big Five trait distribution for the five personality clusters

Identifying which cluster, a candidate belongs to would help recruiters to determine what kind of personality a candidate has.

The cluster prediction was validated using using RandomForest, Naïve-Bayes, and XGBoost algorithms. The performance parameters of the algorithms are shown below.

Model	Accuracy	Precision	Recall	F1 score
RandomForest	0.78	0.78	0.77	0.77
Naive-Bayes	0.78	0.78	0.78	0.78
XGBoost	0.76	0.75	0.75	0.75

Table 7: Model performance parameters – Personality prediction

A radar graph is utilized to visually represent and compare the candidate's Big Five trait distribution with the expected Big Five trait distribution for the job role. This visualization provides recruiters with a quick and intuitive understanding of how well the candidate's personality aligns with the requirements of the job role.

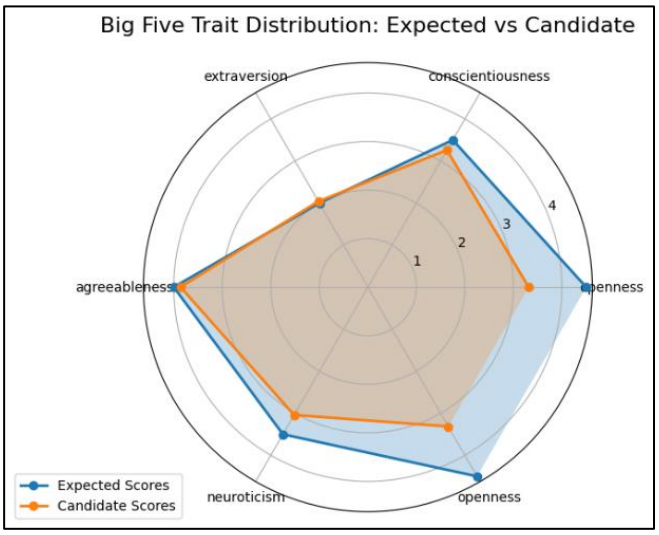


Figure 95: Radar graph - Candidate vs Expected traits

1.3. Research Findings

This research indicates a comprehensive approach to candidate evaluation in the context of professional recruitment. To assess the reliability of a candidate's technical skills, GitHub-extracted skills are meticulously cross-checked with skills extracted from their CV. This cross-referencing process ensures a robust evaluation of the candidate's technical prowess. Additionally, the study employs personality prediction techniques to derive an overall personality match score, contributing to a holistic understanding of the candidate's potential cultural fit within the organization. Furthermore, through CV analysis, an overall job matching score and skill proficiency score is computed. These individual scores are then aggregated to determine a final composite score for each candidate. This multifaceted evaluation approach enhances the precision and effectiveness of the candidate selection process, enabling organizations to make more informed hiring decisions.

1.3.1. CV Ranking and Job description Matching

When it comes to generating job descriptions, utilizing an AI-powered bot streamlines the process, ensuring that all relevant information is comprehensively included. In contrast, crafting job descriptions manually is time-consuming and challenging. Moreover, the AI chatbot can seamlessly compile the content into a professional PDF format, simplifying the task of uploading it to various professional platforms.

Based on the conducted tests and the obtained results, it became evident that the individual models were providing moderate levels of accuracy. Consequently, the next step involved the fusion of these models into a stacked ensemble. Upon thorough analysis of the outcomes, it was established that the stacked model constructed using the Count Vectorizer outperformed its counterpart, which employed the TF-IDF Vectorizer, in terms of predicting matching percentages. Furthermore, the accuracy of these models underwent enhancement through the process of hyperparameter tuning. This optimization effort significantly improved the overall performance of the models

1.3.2. Academic Transcript Analysis

A comprehensive approach was taken, beginning with the extraction of module keywords and phrases from university course outlines, forming the foundation for our data analysis. Through the use of Optical Character Recognition (OCR) techniques, the extraction of valuable data from academic transcripts was a success. Next, identifying module names and associated grades using a custom Named Entity Recognition (NER) model was also successful.

One of the key findings of this research component was the effectiveness of different machine learning models in categorizing skill areas based on module titles and keywords. We compared several models, including Logistic Regression, Naïve Bayes, XGBoost, Random Forest, and Support Vector Machine (SVM). The results indicated that the SVM model had the highest accuracy, achieving an accuracy rate of 89.47%. This finding underscores the importance of choosing the right model for the task, as the SVM model excelled in accurately categorizing skills, which is crucial for evaluating job candidates effectively.

In addition to skill categorization and graphical representation, this research ventured into the realm of generating a score based on skill proficiency and job role. The key insight from this part of this research component was the successful development of a predictive model that could generate scores reflecting a candidate's skill proficiency relative to the requirements of a particular job role. The Gradient Boost Regressor was our tool of choice for this task. By inputting a candidate's skill profile and the skill requirements of a job role into the model, we could produce a numerical score that quantified how well the candidate's skills aligned with the job's demands.

Furthermore, A graphical representation is used to represent the candidate’s skill distribution. A pie chart was used as it was an excellent choice for displaying the relative proportions of different skill categories within a candidate's profile.

1.3.3. Professional Skills Analysis

In LinkedIn skill-based job category prediction considering the results of accuracy testing, among all five models Linear Support Vector Classifier outperforms the best one giving a mean accuracy of 66.38%. This result was delivered after following a fivefold cross-validation and hyperparameter tuning for each model.

When it comes to LinkedIn scraping LinkedIn is generally not allowed according to LinkedIn's User Agreement and robots.txt file. LinkedIn has implemented measures to prevent scraping and unauthorized access to its platform, and it takes measures to protect the privacy and security of its users' data. However, LinkedIn allows a third-party API called ProxyCurl for limited scraping in LinkedIn.

GitHub PLP analysis often involves tracking the number of repositories, commits, and contributors for each language. It allows to extract these data by using GitHub API's. it has been able to reliable calculations on the overall language proficiency of a GitHub user considering all the repositories.

Sentiment analysis on recommendation letters and reference letters gives overall highly positive feedback while sentiment on Google form responses done reference checking gives mixed sentiments regarding the candidate.

1.3.4. Personality Prediction

Based on the results (Table 28), the Naive-Bayes algorithm showed the most promising results in validating the cluster predictions.

The sum of ratios between the candidate and the expected scores for each of the Big Five traits is taken as a percentage. As neuroticism is negatively correlated with job performance [2], the probability that the candidate does not possess the neuroticism personality trait is considered.

$$Candidate\_score = OPN + CSN + EXT + AGR + (1 - NEU)$$

Equation 3.1: Neuroticism score calculation

1.4. Discussion

This research project indeed has significant social, security, and ethical implications in the context of modern recruitment and talent acquisition.

The research project presents several notable social advantages. Firstly, it streamlines the hiring process by automating the evaluation of candidates' qualifications and skills, saving valuable time for recruiters. This efficiency can result in speedy recruitment and job placements, which is beneficial for both job seekers and employers. Moreover, by focusing on skill-based assessments and personality-job role alignment, the research enhances the likelihood of candidates finding positions that align with their abilities and interests, potentially leading to higher job satisfaction and career fulfillment. This can contribute to a more productive and content workforce in the IT industry, benefiting society at large.

Ethical aspects are paramount in deploying such technologies. It's crucial to ensure fairness and mitigate bias in the algorithms used for candidate evaluation. Ethical guidelines should be established to handle sensitive candidate data, including personal information and reference feedback, with utmost privacy and security. Transparency in the use of AI and ML models is also vital, ensuring that candidates understand how their profiles and skills are being assessed. Additionally, a fair feedback mechanism should be implemented to address any concerns or disputes that may arise from the automated assessments. Ethical practices should always prioritize fairness, diversity, and inclusion in the recruitment process.

Ensuring data security is fundamental. Robust security protocols should be in place to protect the confidentiality and integrity of candidate data. This includes safeguarding academic transcripts, reference feedback, and any personal information collected during the recruitment process. Using encryption techniques and secure storage methods is

essential to prevent data breaches. Moreover, cybersecurity measures should be implemented to protect the recruitment platform from potential threats and vulnerabilities in order to compromise the integrity of the system. Regular security audits and updates should be conducted to maintain a high level of protection.

The ethical impacts of this research project lie in its potential to disrupt traditional hiring practices. While automation can streamline processes, it may also lead to concerns about job displacement among HR professionals. It is crucial to strike a balance between automation and the human touch in recruitment, recognizing that automated systems are tools to assist rather than replace human decision-makers. Moreover, transparency in how AI algorithms make hiring decisions is essential to maintain trust and ensure fairness.

This research project has the potential to bring about positive social changes by making recruitment more efficient and job-focused. However, it also raises ethical considerations related to fairness, privacy, and transparency, which must be addressed with a strong commitment to responsible and ethical AI practices. Security measures are imperative to protect sensitive candidate data and the integrity of the system. Overall, this research aims to enhance recruitment processes, providing benefits to job seekers, employers, and society, while upholding the highest ethical standards.

2. SUMMARY OF EACH STUDENTS CONTRIBUTION

4.1. CV Ranking and Job description Matching

In this research component, the primary aim was to revolutionize the resume screening process using advanced technology and innovative methodologies. The traditional method of manually screening resumes is often tedious and time-consuming, especially for large organizations dealing with a high volume of job applications. To address this challenge, the project proposed the development of a CV ranking system. Firstly, the research introduced the concept of an AI-powered chatbot that assists in generating structured job descriptions. This chatbot streamlines the process of creating job descriptions by collecting essential information from recruiters and standardizing the format. By ensuring consistency in job descriptions, this tool enhances the accuracy and efficiency of the recruitment process. Secondly, this component focused on automating the ranking of candidate resumes based on job descriptions. Machine learning models were used to assess the relevance of each resume to a specific job. The introduction of a stacked ensemble model significantly improved accuracy, ensuring a more precise ranking of candidates. Thirdly, keyword extraction from resumes and matching these keywords with job descriptions was done. This approach helped identify the extent to which a candidate's resume aligns with a job's requirements, providing a matching percentage. Lastly, the project involved skill analysis by comparing the skills mentioned in resumes with those listed in job descriptions. This analysis generated skill summaries, aiding recruiters in identifying the most qualified candidates. Throughout the project, various tools and technologies were employed, including Python, Scikit-learn, NLTK, PDF parsing libraries, and Rasa for chatbot development. These technologies played a vital role in the implementation of the proposed methodologies. This research project aimed to streamline and enhance the recruitment process by combining AI-driven chatbot technology, machine learning algorithms, and keyword analysis to create a comprehensive CV ranking system. The methods and technologies used have the potential to revolutionize the way organizations handle job applications, making the process more efficient, accurate, and fair.

4.2. Academic Transcript Analysis

This research component represents a significant step forward in the world of education and employment by addressing the gap between academic qualifications and job suitability. First, important keywords and phrases from university course outlines were extracted, creating a strong foundation for the analysis. Then, data from academic transcripts was extracted using Optical Character Recognition (OCR) techniques, identifying module names and associated grades using a custom Named Entity Recognition (NER) model. One of the significant findings of this research component was the effectiveness of different machine learning models in categorizing skill areas based on module titles and keywords. We compared various models and found that the Support Vector Machine (SVM) had the highest accuracy. This means that SVM was the most reliable choice for accurately categorizing skills, which is crucial for evaluating job candidates effectively. Additionally, a predictive model was developed that generates scores reflecting a candidate's skill proficiency relative to the requirements of a specific job role. The Gradient Boost Regressor was the tool of choice for this task. It allowed to produce a numerical score that quantifies how well a candidate's skills align with a job's demands, providing a more objective way to assess candidates. To visually represent a candidate's skill distribution, a pie chart was used, making it easy for employers to understand at a glance how different skills contribute to the overall profile. This approach not only provided a more objective and data-driven assessment of candidates but also demonstrated the potential for automation in the hiring process. It offered a systematic and standardized way to evaluate job applicants, enabling employers to make more informed decisions about candidate suitability. Overall, the research findings emphasize the potential for data-driven academic transcript analysis to revolutionize the recruitment and hiring processes, enhancing objectivity and efficiency in talent acquisition.

4.3. Personality Prediction

This research component aimed to make hiring in the IT industry more efficient and fairer by using machine learning and natural language processing to understand a candidate's personality and how well it matches a specific job. This is achieved by creating a comprehensive methodology that involved the use of questionnaires, data processing, and advanced algorithms. First, the questionnaire was designed with two types of questions: self-rating questions and open-ended questions. Candidates would rate themselves on various traits and provide written responses. The data was collected from both publicly available sources and a survey. Then, this data was used to build machine-learning models that could identify personality clusters among candidates. The models were validated to ensure their accuracy. Next, NLP techniques were used to extract keywords from the candidates' written responses to determine the presence of specific personality traits. Three different methods for keyword extraction were used and found that one method, called TF-IDF, worked particularly well. To evaluate how well a candidate's personality suited a job role, job requirements were matched to the Big Five personality traits, and calculated a compatibility score. A user-friendly front-end interface using HTML, CSS, JavaScript, and Flask was also developed to make the system accessible to recruiters. Through testing and validation, the effectiveness of this component was confirmed in predicting personality traits, extracting keywords, mapping job requirements, and calculating personality-job fit scores. The findings demonstrated that the Naive-Bayes algorithm performed best in predicting personality clusters. This research component represents a significant contribution by introducing a practical and accurate way to assess a candidate's personality and match it with job requirements using state-of-the-art technology and methodologies. It goes beyond traditional personality assessment by evaluating the alignment between a candidate's personality and a specific job role, providing valuable insights for the recruitment process. This work has the potential to enhance hiring decisions and improve the overall efficiency of talent acquisition in various industries.

4.4. Professional Skills Analysis

This research component tackles some of the key challenges in modern job recruitment processes through a multifaceted approach. It introduces three innovative components that aim to revolutionize how candidates are evaluated and matched with job opportunities. Firstly, it leverages the power of GitHub analysis to assess a candidate's proficiency in programming languages. By analyzing a candidate's GitHub repositories, this component provides a quick and reliable measure of their technical skills, saving time and ensuring a better alignment between a candidate's abilities and job requirements. Secondly, the project introduces a machine learning-based model for LinkedIn skills-based job category prediction. This model automatically categorizes candidates based on their LinkedIn profiles, making the initial screening process more efficient and accurate. Recruiters can quickly identify candidates who are the best fit for a particular job, reducing manual effort and improving the recruitment process's overall efficiency. Lastly, the integration of sentiment analysis in reference checking using a Google form offers a fresh approach to evaluating candidates beyond traditional CV assessments. This component adds objectivity to the reference-checking process and helps identify candidates with not only technical skills but also the soft skills necessary for success within an organization. The methodology behind these involves the use of various tools and technologies, including Python, PyGitHub, NLTK, and machine learning models. These tools are used to extract and analyze data from GitHub and LinkedIn, perform sentiment analysis on reference checks, and predict job categories based on skills. By combining methodologies with cutting-edge technologies, it provides recruiters with a more efficient and comprehensive solution for candidate evaluation, ultimately leading to more informed hiring decisions. However, it is crucial to remain mindful of ethical considerations and potential biases in the automated processes to ensure fairness and transparency in recruitment practices.

5. CONCLUSION

In conclusion, this research strives to transform the recruitment process within the IT industry by embracing automation. The introduced concept, ‘Intellihire’, aims to revolutionize how candidates are evaluated. In contrast to traditional methods that often emphasize qualifications and experience alone, this approach delves deeper.

Candidate assessment should extend beyond mere paper qualifications. ‘Intellihire’ takes into account a candidate's resume, academic records, professional skills, and even personality traits. This comprehensive approach enables recruiters to gain a profound understanding of not only qualifications but also a candidate's potential for success and suitability within the organization.

The objective is to bridge the gap between conventional recruitment practices and a more sophisticated, nuanced approach. The ultimate goal is to cultivate a workforce that possesses not only the necessary skills but also the capacity to thrive in the long term, thereby contributing to the growth and innovation of the IT industry. This approach recognizes the unique qualities and potential of each candidate and represents the future of recruitment.

In future work, the incorporation of validation methods for candidate information sourced from resumes, academic transcripts, platforms like LinkedIn, and responses from questionnaires could significantly enhance the precision of the outcomes generated by ‘Intellihire’.

REFERENCE LIST

[1] “Sri Lanka aiming 200,000 ICT workforce by 2022,” *ICTA*, Aug. 08, 2019. https://www.icta.lk/news/sri-lankaaiming-200000-ict-workforce-by-2022/

[2] R. T. R. Jayasekara, K. A. N. D. Kudarachchi, K. G. S. S. K. Kariyawasam, D. Rajapaksha, S. L. Jayasinghe, and S. Thelijjagoda, "DevFlair: A Framework to Automate the Pre-screening Process of Software Engineering Job Candidates," in *2022 4th International Conference on Advancements in Computing (ICAC)*, Colombo, Sri Lanka: IEEE, Dec. 2022, pp. 288–293. doi: 10.1109/ICAC57685.2022.10025337.

[3] R. G. U. S. Gajanayake, M. H. M. Hiras, P. I. N. Gunathunga, E. G. Janith Supun, A. Karunasenna, and P. Bandara, "Candidate Selection for the Interview using GitHub Profile and User Analysis for the Position of Software Engineer," in *2020 2nd International Conference on Advancements in Computing (ICAC)*, Malabe, Sri Lanka: IEEE, Dec. 2020, pp. 168–173. doi: 10.1109/ICAC51239.2020.9357279.

[4] R. Ransing, A. Mohan, N. B. Emberi, and K. Mahavarkar, "Screening and Ranking Resumes using Stacked Model," in *2021 5th International Conference on Electrical, Electronics, Communication, Computer Technologies and Optimization Techniques (ICEECCOT)*, Mysuru, India: IEEE, Dec. 2021, pp. 643–648. doi: 10.1109/ICEECCOT52851.2021.9707977.

[5] M. Mehboob, M. S. Ali, S. Ul Islam, and S. Sarmad Ali, "Evaluating Automatic CV Shortlisting Tool For Job Recruitment Based On Machine Learning Techniques," in *2022 Mohammad Ali Jinnah University International Conference on Computing (MAJICC)*, Karachi, Pakistan: IEEE, Oct. 2022, pp. 1–4. doi: 10.1109/MAJICC56935.2022.9994112.

[6] Y. Wang, Y. Allouache, and C. Joubert, "Analysing CV Corpus for Finding Suitable Candidates using Knowledge Graph and BERT," May 2021.

[7] M. Sharma, G. Choudhary, and S. Susan, "Resume Classification using Elite Bag-of-Words Approach," in *2023 5th International Conference on Smart Systems and Inventive Technology (ICSSIT)*, Tirunelveli, India: IEEE, Jan. 2023, pp. 1409–1413. doi: 10.1109/ICSSIT55814.2023.10061036.

[8] B. M. D. E. Bannaka, D. M. H. S. G. Dhanasekara, M. K. Sheena, A. Karunasena, and N. Pemadasa, "Machine learning approach for predicting career suitability, career progression and attrition of IT graduates," in *2021 21st International Conference on Advances in ICT for Emerging Regions (ICter)*, Colombo, Sri Lanka: IEEE, Dec. 2021, pp. 42–48. doi: 10.1109/ICter53630.2021.9774825.

[9] T. A. Judge and C. P. Zapata, "The Person–Situation Debate Revisited: Effect of Situation Strength and Trait Activation on the Validity of the Big Five Personality Traits in Predicting Job Performance," *AMJ*, vol. 58, no. 4, pp. 1149–1179, Aug. 2015, doi: 10.5465/amj.2010.0837.

[10] M. A. Iqbal, F. A. Ammar, A. R. Aldaihani, T. K. U. Khan, and A. Shah, "Building Most Effective Requirements Engineering Teams by Evaluating Their Personality Traits Using Big-Five Assessment Model," in *2019 IEEE 6th International Conference on Engineering Technologies and Applied Sciences (ICETAS)*, Kuala Lumpur, Malaysia: IEEE, Dec. 2019, pp. 1–5. doi: 10.1109/ICETAS48360.2019.9117561.

[11] S. K. Nivetha, M. Geetha, R. S. Latha, K. Sneha, S. Sobika, and C. Yamuna, "Personality Prediction for Online Interview," in *2022 International Conference on*

Computer Communication and Informatics (ICCCI), Coimbatore, India: IEEE, Jan. 2022, pp. 1–4. doi: 10.1109/ICCCI54379.2022.9740980.

[12] H. K. S M, S. Hegde, S. G, S. M, S. R, and S. L. N, "User Interest Prediction based on Social Network Profile with Machine Learning," in *2021 6th International Conference for Convergence in Technology (I2CT)*, Maharashtra, India: IEEE, Apr. 2021, pp. 1–6. doi: 10.1109/I2CT51068.2021.9418126.

[13] "How Do You Conduct A Great Reference Check?" https://www.personio.com/hr-lexicon/applicant-screening/

[14] B. M. Anderson, "13 Questions You Should Ask When Checking References," Jun. 23, 2021. https://www.linkedin.com/business/talent/blog/talent-acquisition/reference-check-questions

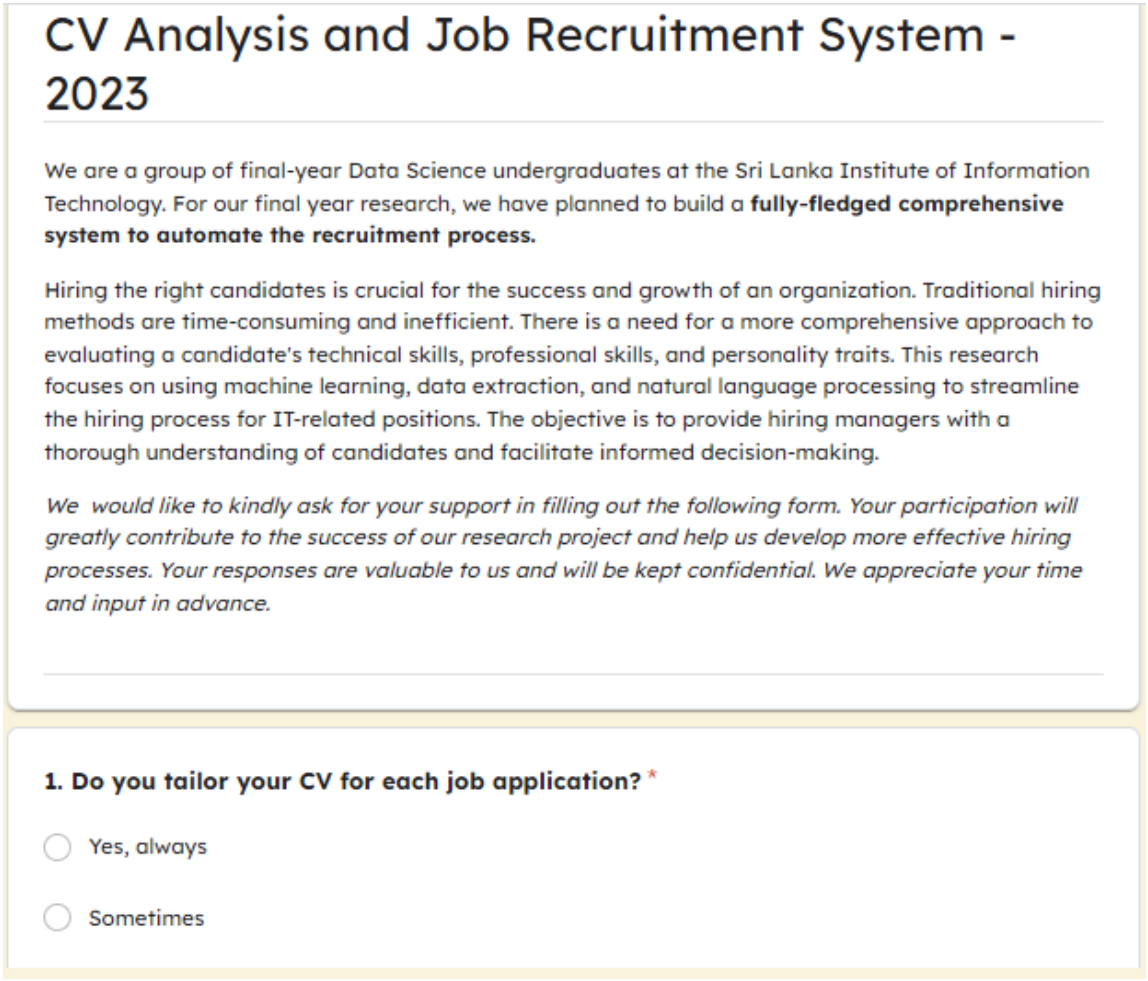
[15] Y. R, "Python | Lemmatization with NLTK." https://www.geeksforgeeks.org/python-lemmatization-with-nltk/

[16] M. Q. Khan *et al.*, "Impact analysis of keyword extraction using contextual word embedding," *PeerJ Computer Science*, vol. 8, p. e967, May 2022, doi: 10.7717/peerj-cs.967.

[17] J. Darby, "HR Blog," *thomas.co*, May 16, 2023. https://www.thomas.co/resource-type/hr-blog

6. LIST OF APPENDICES

Appendix A: Google Form Survey



CV Analysis and Job Recruitment System - 2023

We are a group of final-year Data Science undergraduates at the Sri Lanka Institute of Information Technology. For our final year research, we have planned to build a **fully-fledged comprehensive system to automate the recruitment process.**

Hiring the right candidates is crucial for the success and growth of an organization. Traditional hiring methods are time-consuming and inefficient. There is a need for a more comprehensive approach to evaluating a candidate's technical skills, professional skills, and personality traits. This research focuses on using machine learning, data extraction, and natural language processing to streamline the hiring process for IT-related positions. The objective is to provide hiring managers with a thorough understanding of candidates and facilitate informed decision-making.

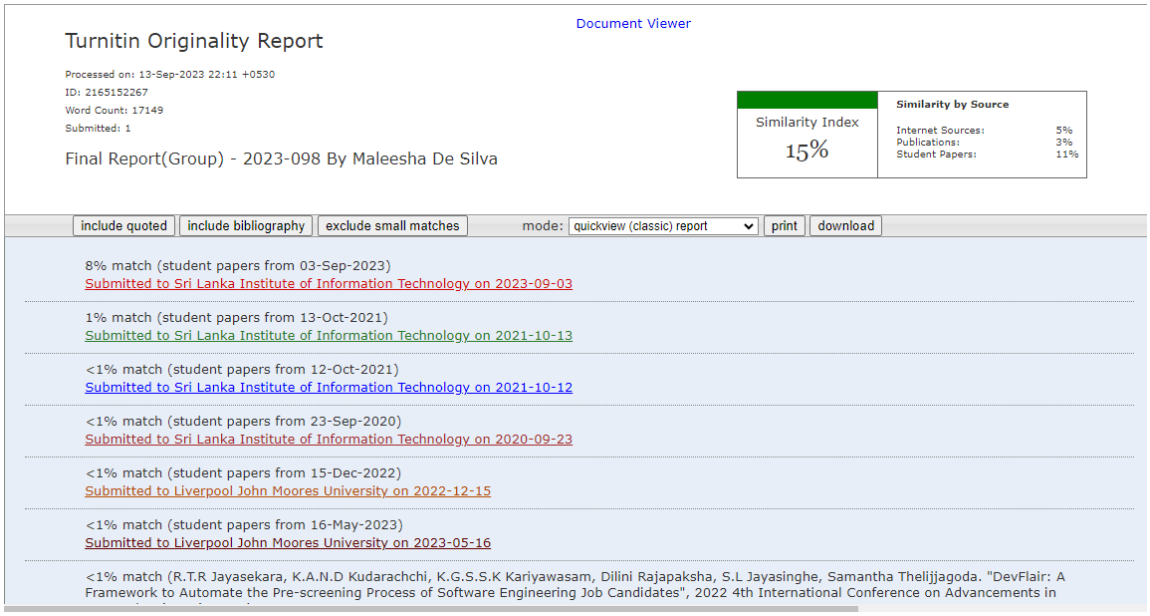
We would like to kindly ask for your support in filling out the following form. Your participation will greatly contribute to the success of our research project and help us develop more effective hiring processes. Your responses are valuable to us and will be kept confidential. We appreciate your time and input in advance.

1. Do you tailor your CV for each job application? *

- Yes, always
- Sometimes

Link:
https://docs.google.com/forms/d/1rMXemkTNLyp9QDe3ZRPQIbC8_WcvK0Sl8AoDX0u2O14/edit?usp=forms_home&ths=true

Appendix B: Turnitin Report



Document Viewer

Turnitin Originality Report

Processed on: 13-Sep-2023 22:11 +0530
ID: 2165152267
Word Count: 17149
Submitted: 1

Final Report(Group) - 2023-098 By Maleesha De Silva

Similarity Index
15%

Similarity by Source
Internet Sources: 5%
Publications: 3%
Student Papers: 11%

include quoted | include bibliography | exclude small matches | mode: quickview (classic) report | print | download

8% match (student papers from 03-Sep-2023)
Submitted to Sri Lanka Institute of Information Technology on 2023-09-03

1% match (student papers from 13-Oct-2021)
Submitted to Sri Lanka Institute of Information Technology on 2021-10-13

<1% match (student papers from 12-Oct-2021)
Submitted to Sri Lanka Institute of Information Technology on 2021-10-12

<1% match (student papers from 23-Sep-2020)
Submitted to Sri Lanka Institute of Information Technology on 2020-09-23

<1% match (student papers from 15-Dec-2022)
Submitted to Liverpool John Moores University on 2022-12-15

<1% match (student papers from 16-May-2023)
Submitted to Liverpool John Moores University on 2023-05-16

<1% match (R.T.R Jayasekara, K.A.N.D Kudarachchi, K.G.S.S.K Kariyawasam, Dilini Rajapaksha, S.L Jayasinghe, Samantha Thelijjagoda. "DevFlair: A Framework to Automate the Pre-screening Process of Software Engineering Job Candidates", 2022 4th International Conference on Advancements in