

Capstone Project Phase B

**Understanding of AI-Based Recruitment Outcomes**

**25-1-R-6**

[GitHub](https://github.com/YoavKatz99/Understanding-of-AI-Based-Recruitment-Outcomes)

[GUI](https://understanding-of-ai-based-recruitment-4i26.onrender.com/) QR Code:

Supervisors:

Dr. Julia Sheidin

Dr. Avital Shulner Tal

Alin Migdalovich [alin.migdalovich@e.braude.ac.il](mailto:alin.migdalovich@e.braude.ac.il)

Yoav Katz [yoav.katz@e.braude.ac.il](mailto:yoav.katz@e.braude.ac.il)

Table of Content

[**1 Introduction** 2](#_Toc203727553)

[**2 Background and Related Work** 3](#_Toc203727554)

[**2.1 Recruiting Process and Parameters** 3](#_Toc203727555)

[**2.2 Chosen Dataset and Algorithm** 4](#_Toc203727556)

[**2.3 Explanations for Text-Based Recruitment Decisions** 5](#_Toc203727557)

[**2.3.1 XAI Tools** 5](#_Toc203727558)

[**2.3.2 Textual Data vs. Visual Data Explanations** 6](#_Toc203727559)

[**2.4 Summary** 6](#_Toc203727560)

[**3 Expected Achievements** 7](#_Toc203727561)

[**4 Research Process** 8](#_Toc203727562)

[**4.1 XAI type selection** 8](#_Toc203727563)

[**4.3 Architecture and tools** 9](#_Toc203727564)

[**4.3.1 Backend (AI and Data Processing)** 9](#_Toc203727565)

[**4.3.2 Frontend (User Interaction and Visualization)** 10](#_Toc203727566)

[**4.4 GUI** 12](#_Toc203727567)

[**4.5 Challenges** 14](#_Toc203727569)

[**4.5.1 Difficulty in finding relevant academic literature** 14](#_Toc203727570)

[**4.5.2 Familiarize with SHAP, LIME, CARLA and DiCE** 14](#_Toc203727571)

[**4.5.3 Explaining Why the Tool Made Its Decision** 14](#_Toc203727572)

[**4.5.4 Compatibility Issues with CARLA** 14](#_Toc203727573)

[**4.5.5 The Need to Use Two Different Models** 15](#_Toc203727575)

[**4.5.6 Getting People to Answer Our Survey** 15](#_Toc203727577)

[**5 Evaluation Plan** 16](#_Toc203727579)

[**6 Results and Conclusions** 19](#_Toc203727580)

[**6.1 Kruskal-Wallis Test results with Bonferroni Correction** 19](#_Toc203727581)

[**6.2 System Testing Results Summary** 19](#_Toc203727582)

[**6.3 Future Work** 20](#_Toc203727583)

[**7 User Guide** 21](#_Toc203727584)

[**8 Maintenance Manual** 25](#_Toc203727585)

[**References** 27](#_Toc203727586)

# **1 Introduction**

The recruitment process in the software industry is a challenge for both job seekers and recruiters. Candidates often struggle to showcase their skills and experience in the best way. Many resumes are poorly written or don’t match the job descriptions well, which causes strong candidates to be rejected even when they are qualified [5].

On the other hand, recruiters have a hard time too. Each job opening brings in hundreds of resumes, and recruiters only have a few minutes to review each one. They often focus on specific parts like education and experience, but this can lead to missing great candidates. Automated tools, while helpful, sometimes reject good resumes because they don’t understand the context well enough [5]. This gap between candidates and recruiters shows how much we need a better solution.

Current tools like resume templates or professional writing services give some help, but they are limited. Many automated systems don’t explain how they make decisions, which makes it hard for people to trust them [6].

In this research project, we aim to investigate ways to represent the data contained in CVs. This includes designing clear, intuitive visual and textual explanations of the data so candidates will be able to understand how they should create a proper CV.

We hope that the impact of this project will be significant for job seekers, who will get useful insights to make their resumes stronger and better suited for specific roles and have a better chance to showcase their potential. In the end, our goal is to help candidates to create more efficient CVs.

# 

# 

# 

# **2 Background and Related Work**

## **2.1 Recruiting Process and Parameters**

Recruitment plays a critical role in organizational success, with HR departments tasked with identifying the most suitable candidates from pools of applicants [1]. HR departments face several challenges, including managing the vast quantity of applications [4] and ensuring fair and unbiased candidate evaluations [7]. Traditional processes often result in qualified candidates being overlooked due to human error or biases, while the need for speed in filling positions can lead to suboptimal hiring decisions [4]. However, with the advent of AI, the dynamics of recruitment have shifted dramatically. AI-driven systems now provide recruiters with powerful tools to streamline candidate selection by automating resume evaluations, skill matching, and even clustering candidates based on their suitability for specific roles [4].

For example, [**HireVue**](https://www.hirevue.com/) uses AI to analyze resumes in conjunction with video interviews. Its CV analysis component focuses on matching qualifications with job requirements and scoring candidates based on experience and skills.

According to Koivunen et al. (2019), recruitment decision-making is often conceptualized as a process involving four main stages: establishing requirements, identifying alternatives, comparing alternatives, and selecting the most suitable match [2]. The initial stage defining what constitutes a good match. Organizations frequently lack clarity about their goals and long-term needs, leading to opportunistic hiring decisions that may result in suboptimal outcomes [2].

Identifying alternatives is identifying and attracting individuals who would meet the requirements [2]. Platforms like LinkedIn and other e-recruitment tools are widely used to attract and identify candidates. However, these tools often fail to capture the deeper, unique qualities of candidates, such as personality traits, which are essential for team long-term success. Moreover, the reliance on algorithms to pre-screen candidates may limit diversity and reinforce biases, further complicating the decision-making process [2].

Comparing alternatives is assessing the candidates in relation to the requirements and each other [2]. Decision-makers often rely on superficial information like job titles or personal impressions about suitable people, which provide an incomplete picture of a candidate’s capabilities [2]. This stage is also influenced by self-reported description, where candidates present themselves in ways they believe align with recruiters' expectations.

The final stage, selecting the most suitable candidate, is frequently constrained by time pressures and organizational practices that prioritize speed over quality [2]. To address these challenges, iterative approaches, including trial periods or phased hiring processes, are recommended to ensure better alignment between candidates and organizational needs [2].

Key recruitment parameters often begin with academic qualifications. Education credentials, including relevant coursework and GPA, serve as foundational criteria in the evaluation process [5]. A strong academic background is frequently associated with cognitive ability, motivation, and work ethic [5]. However, it is not the sole determinant of employability. Professional experience—whether through internships, previous roles, or industry exposure—holds even greater weight in predicting a candidate’s potential contributions [5].

Another critical parameter is the alignment of a candidate’s skill set with job requirements. AI systems such as IntelCV leverage predefined patterns to extract and compare skills from resumes against those outlined in job descriptions [4]. By applying clustering techniques, these systems not only enhance accuracy but also provide recruiters with actionable insights into the quality of matches within their applicant pools [4].

Technology has the potential to enhance decision-making in HR by addressing some of these challenges. Existing job technologies have been found to bring issues to the hiring decision-making, such as biased hiring, lack of job match quality, and cognitive overload [2].

## **2.2** [**Chosen Dataset and Algorithm**](https://www.kaggle.com/datasets/sauravsolanki/hire-a-perfect-machine-learning-engineer/data)

The dataset that we intend to work with is entitled: “Hire A Machine Learning Engineer” or “HireAMLE” which is owned by Saurav Solanki. Our dataset contains 150 CVs of various titles such as ML Engineer, MERN Stack Developer, AWS Engineer, Computer Vision Engineer, etc. The CVs include unique id, job roles, preferred skills and education. The CVs are divided into test and train groups. The dataset was designed to predict the match percentage for a specific ML job description which is also included in the dataset. For each CV in the training group, Match percentages are given. For the test group, we can predict the match percentages using compatible analysis code, provided by the user “REMI LEGRAND” on Kaggle.com.

The code evaluates candidate CVs to find the best match for a job based on their technical skills. It begins with pre-processing of lemmatize and removing all the stopwords. It checks each CV for specific skills from a predefined list, counts how many skills match, and calculates a score. The score considers how many skills a CV matches compared to the average and gives extra points to CVs with many skills. A chart shows how often each skill appears in the CVs. Finally, the CVs are ranked by their scores, and the results are compared to an existing dataset for validation. We will work on top of this code in order to fit it to our needs.

## **2.3 Explanations for Text-Based Recruitment Decisions**

When a system matches job seekers with job descriptions, it looks at many kinds of information—like skills, education, experience, and the meaning behind the words used in resumes. These systems often use complex models that are hard to understand. As explained in the research by Barrak (2021), it’s very important to make these decisions clear and easy to explain why a match was made or why someone wasn’t selected. Tools from Explainable AI (XAI) can show which parts of a resume or job post were most important in the decision, like a missing skill or a strong match in experience. Without explanations, people may not trust the system or know how to improve their chances.

## **2.3.1 XAI Tools**

Explainable Artificial Intelligence (XAI) methods are critical in understanding how AI systems make decisions, particularly when dealing with complex models often perceived as "black boxes" [6].

The paper elaborates on SHAP (SHapley Additive exPlanations) and its reliance on game theory to assign importance to each feature in a dataset [6]. SHAP determines the contribution of each feature to the model's output using Shapley values, which provides a fair distribution of importance by calculating the marginal contributions of all feature subsets.

Similarly, LIME (Local Interpretable Model-Agnostic Explanations) creates interpretable surrogate models that approximate the predictions of a complex AI system locally (around a specific instance of interest) [6]. LIME confuses the input data and observes how the predictions change, thereby identifying the features that are most influential in the decision-making process. This method is particularly valuable for explaining specific decisions of the AI model, such as why a particular prediction was made, in a straightforward and understandable way [6].

CARLA is used in supervised machine learning when a model gives an undesired prediction for a specific data point. They identify the minimal changes to the input features that would result in a desired model output [11].

### **2.3.2 Textual Data vs. Visual Data Explanations**

Visual explanations excel at displaying item connections but demand greater user engagement for interpretation [10]. Textual explanations were consistently rated as more persuasive and easier to understand compared to visual formats [10]. Visual explanations were perceived as less convincing, particularly by users without prior familiarity with complex visualizations [10]. This suggests that while visual formats can be powerful, their utility is limited by the user's ability to interpret the presented graphics. User familiarity with visualizations had a positive correlation with the acceptance of visual formats but did not exceed the overall preference for textual explanations [10].

## **2.4 Summary**

In conclusion, effective recruitment is essential for organizational success, but HR departments face challenges such as handling large applicant pools, ensuring unbiased evaluations, and making fast yet valid hiring decisions. AI-driven solutions like HireVue offer tools to address these concerns by automating processes: from resume analysis and skill matching to candidate clustering. The four-stage framework (requirements, alternatives, comparison, and selection) highlights the need for clear job descriptions. Various data explanation methods, including visualizations offer ways to explore and communicate insights. Explainable AI techniques, such as SHAP, LIME, and Counterfactual explanations further enhance transparency by clarifying AI models results.

# 

# **3 Expected Achievements**

As part of this project, we conducted an in-depth analysis of the collected dataset, focusing on parameters that affect job seekers, such as education, skills, work experience, and match percentages for specific roles. This analysis helped us identify how these factors influenced a candidate’s chances of being matched to a job and revealed patterns that can guide job seekers in improving their resumes.

Next, we integrated SHAP, LIME, and CARLA (Counterfactual and Recourse Library) into the existing matching algorithm. These tools provided explanations of how the match percentage was calculated and why a specific CV received a specific match score. We utilized these advanced tools to present the data both visually and textually, incorporating additional visualization techniques to ensure a clear and intuitive representation of the findings.

Finally, we evaluated the effectiveness of these explanations by gathering feedback from a relevant target audience, such as recruiters and students. This feedback helped us refine our approach and ensure that the explanations were clear and valuable.

# **4 Research Process**

The process began with an extensive literature review, which involved examining academic and industry publications to gather insights on the challenges in the recruitment process, such as how recruiters evaluate candidates and in manual and automated screening methods. The review also looked at how visual and textual explanations can make the results more understandable. Following this, we proceeded to explore and analyze our dataset, ensuring its relevance and quality to our research objectives. Next, we shifted our focus to understand how explainable AI tools like SHAP, LIME, and Counterfactual explanations. Finally, we focused on finding ways to combine these insights to create a system that presents and explains the match percentage of CVs and we evaluated it using various methods.

## **4.1 XAI type selection**

We've decided to select SHAP since it's more accurate, and supports an explanation of the model and the decision making over a specific CV. In addition, it has tools such as Force Plot and Summary Plot visual graphs, which is part of what we are focusing on in this project.

We also used LIME to provide quick, interpretable explanations for individual predictions in a model-agnostic manner. It has the ability to create a simple, interpretable surrogate model around the specific prediction. Additionally, it supports diverse data types like text, tabular, and images.

Finally, we run CARLA on our dataset. CARLA provides insights by answering "what if" scenarios. This approach is particularly valuable in applications where end-users need to know what adjustments could lead to a positive decision. Those explanations are intuitive, user-centric, and focus on transparency by highlighting model behavior in specific contexts. They are also useful for ensuring fairness, as they can uncover biased decision patterns in the model.

We had some problems using CARLA because it didn’t work well with the Python versions required for SHAP and LIME. Because of that, we decided to use DiCE instead. DiCE creates counterfactual explanations, which means it shows small changes that could be made to a CV in order to improve the match score [14]. For example, it can suggest that if the candidate had one more skill or different experience, the result would be better.

**4.2 Activity Diagram**

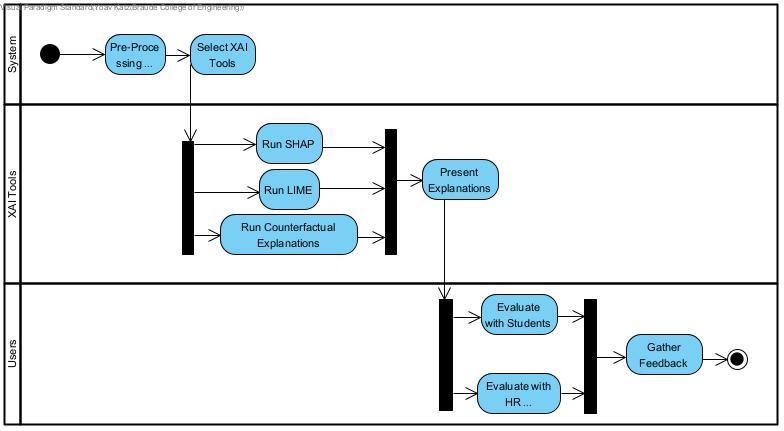
The following activity diagram presents the stages of our system: 

Figure – Activity Diagram

*Figure - Activity Diagram*

## **4.3 Architecture and tools**

The following tools and technologies will be used in our project:

### **4.3.1 Backend (AI and Data Processing)**

#### Programming Language:

* Python: Widely used for machine learning, data processing, and API integration.

Framework:

* Flask: A lightweight web framework used to build the backend API. It handles resume uploads, triggers model predictions, and returns explanations to the frontend.

#### AI Models:

Main model (for SHAP and LIME):

* pdfminer.six: Extract raw text from PDF resumes.
* TF-IDF to extract features from the resume text.
* XGBoost Regressor is trained to predict the match percentage between a resume and a job description.
* This model supports SHAP and LIME explanations.

DiCE-Compatible Model:

* We trained a separate XGBoost Regressor using binary features based on the presence of specific technical skills in the resume text.

#### XAI tools libraries:

* SHAP (SHapley Additive exPlanations): Provides insights into individual predictions by calculating the contribution of each feature using principles from cooperative game theory.
* LIME (Local Interpretable Model-Agnostic Explanations): Explains the predictions of any classifier by approximating it locally with an interpretable model, helping users understand model behavior in specific instances.
* DiCE: Replaced CARLA due to compatibility issues. Generates counterfactual examples (minimal skill changes) that would improve a candidate’s score.

### **4.3.2 Frontend (User Interaction and Visualization)**

#### Programming Language:

* JavaScript/TypeScript: Standard for interactive web applications.

#### Frameworks/Libraries:

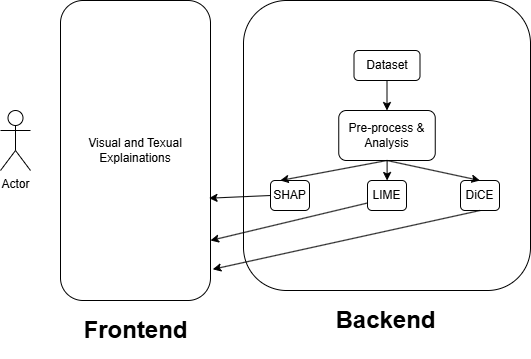
* React.js: For building responsive, component-based web interfaces.

#### Visualization Tools:

* matplotlib: Used in the backend for generating static bar charts and explanation visuals.

#### Styling:

* Tailwind CSS: For responsive and visually appealing designs.



*Figure 2 - System Architecture*

## **4.4 GUI**

Figure 3 – GUI and SHAP output

*Figure – GUI with SHAP explanation*

This is what our system looks like: Upload a resume – The candidate submits their resume, which is then processed using a TF-IDF vectorizer and analyzed by an XGBoost regression model. A match score is created – The system calculates a match score that reflects how well the resume aligns with a given job. Feedback is given – The system provides visual and written explanations, helping candidates understand and improve their resumes: For SHAP, a bar chart showing the impact of different features on the score (Figure 5). For LIME, visualizations that highlight influential words directly in the resume text (Figure 6) and DiCE-based suggestions offering actionable skill changes to boost the match score (Figure 7).

## 

Figure 5 - DiCE text output

Figure 4 - LIME output

*Figure 6 - LIME explanation*

*Figure 7 - DiCE explanation snapshot*

## **4.5 Challenges**

### **4.5.1 Difficulty in finding relevant academic literature**

One of the challenges we faced was finding articles that covered both the technical side of our project and the HR side. While there’s a lot of research on tools like SHAP and LIME and how they work, there weren’t many studies about the use of SHAP, LIME or CARLA/DICE in resume matching or recruitment.

### **4.5.2 Familiarize with SHAP, LIME, CARLA and DiCE**

Neither of these tools had been part of our prior studies, and their functionality requires research and experimentation. We discovered that SHAP, LIME, CARLA or DiCE use different approaches to generating explanations global vs. local which added to the complexity of integrating them with our matching algorithm. Moreover, we need to understand the technical limitations of these tools.

### **4.5.3 Explaining Why the Tool Made Its Decision**

It’s hard to explain why the tool decided on specific match percentages because the process it uses is very complex. Tools like SHAP, LIME and CARLA help show which parts of the resume, like skills or education, influenced the result, but their explanations are often technical and not easy to understand. The challenge is figuring out how to take these detailed, complicated outputs and turn them into clear, simple explanations that users can actually use. If we’re not careful, the explanations might end up being too confusing or too vague to be helpful.

### **4.5.4 Compatibility Issues with CARLA**

### While we originally planned to use CARLA for counterfactual explanations, we encountered compatibility problems. CARLA required older Python and library versions that did not match our working environment. These issues made installation and integration very difficult. Since we couldn’t get it to run properly, we switched to DiCE, which provided a more accessible and updated solution for generating "what-if" scenarios.

### **4.5.5 The Need to Use Two Different Models**

### Another challenge was that we couldn’t use the same model for all explanation types. SHAP and LIME required a complex model based on TF-IDF and XGBoost, while DiCE (used instead of CARLA) needed a simpler rule-based model. This meant we had to maintain and test two separate models—one for prediction and interpretability, and another for generating counterfactual suggestions. This added extra work and increased system complexity.

### **4.5.6 Getting People to Answer Our Survey**

### We also faced difficulties when trying to evaluate our system with real users. It was hard to distribute the survey and get enough people to answer it. Many people we're struggling to read or understand the XAI tools explanations in English. This made it challenging to collect enough feedback to improve the system based on real user opinions. We also haven’t reached a sufficient number of HR professionals through the survey distribution channels we used to review our results. This made it harder to evaluate the system from a professional recruitment point of view.

# **5 Evaluation Plan**

To ensure the system operates correctly and meets its intended goals, we conducted a user-centered evaluation by distributing surveys that included our system’s explanation outputs. These surveys were shared with real users, particularly focusing on two groups: students and HR recruiters. The purpose was to assess how users perceived the system’s explanations in terms of causality (causability), explainability, trust, and satisfaction. This evaluation strategy aimed to validate not only the technical performance of the system but also how understandable, transparent, and helpful it was from the user’s perspective. To measure perception, we used a validated questionnaire inspired by Shin (2021) [13], which included Likert-scale statements assessing users' agreement with statements as described later in the text.

The survey included four groups of questions: causability, explainability, trust, and satisfaction. Causability means how well users could understand why the system made a certain decision, and if the explanation helped them see the connection between input and output. Explainability checked if users found the algorithm easy to understand and if the system made sense overall. Trust looked at whether users believed the system’s results and felt confident in its suggestions. Satisfaction measured how happy users were with the system and if it met their expectations [13].

**Causability:**

* I understood the explanations in the context of the task.
* I did not need help to understand the explanations.
* I found the explanations helped me understand cause and effect.

**Explainability:**

* I found the match algorithm easy to understand.
* I think the match algorithm is interpretable.
* I can figure out how the machine learning model works.

**Trust:**

* I trust the recommendations made by the algorithm.
* I believe the suggestions from the algorithm are trustworthy.
* I believe the algorithm’s results are reliable.

**Satisfaction:**

* Overall, I am pleased with the algorithm’s service.
* The algorithm meets my initial expectations.
* In general, I am happy with what the algorithm provides.

Each question was rated on a 7-point scale, from “Strongly disagree” (1) to “Strongly agree” (7). We then calculated an average score for each category based on its three questions.

We also asked demographic and categorical questions, such as age, gender, current employment-seeking status, and field of interest, to better interpret user feedback across different backgrounds. All responses were analyzed to identify common patterns, evaluate the clarity and usability of each explanation method (SHAP, LIME, DiCE and No explanation), and determine areas for improvement. This comprehensive evaluation helps ensure that the system not only functions accurately but also delivers transparent and satisfying explanations to its users.

To ensure the system operates correctly and as intended, we will evaluate it through the following steps:

| Test ID | Description | Expected Result | Precondition Comments | Comments |
| --- | --- | --- | --- | --- |
| 1 | Matching a CV to a job description and calculating a match percentage | The system will display a clear match percentage based on the analyzed parameters. | A job description is entered, and a matching CV is uploaded. | Local |
| 2 | Generating a textual explanation | The system will provide a simple and clear textual explanation of how the match percentage was calculated. | A successful match process has been completed. | Ensure the explanation is easy to understand for non-technical users. |
| 3 | Generating a visual explanation | The system will produce visual graphs or charts to illustrate the key factors affecting the match percentage. | The match analysis is completed successfully. | Evaluate the clarity and usability of the visual outputs. |
| 5 | Generating SHAP explanations | SHAP-based explanations will highlight the most impactful factors contributing to the match. | Match analysis is completed. | .Ensure SHAP results are intuitive and align with user expectations. |
| 6 | Generating LIME explanations | LIME will generate localized explanations for specific CV-job matches. | Match analysis is completed. | Test on both high and low match percentages for variety. |
| 7 | Generating Counterfactual explanations | The system will provide "what-if" scenarios by showing the minimal changes needed to improve a match percentage. | Match analysis is completed. | Test for clarity in counterfactual outputs and relevance of suggested changes. |

# **6 Results and Conclusions**

### **6.1 Kruskal-Wallis Test results with Bonferroni Correction**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **XAI TOOL** | **Count** | **CAUSABILITY** | **EXPLAINABILITY** | **TRUST** | **SATISFACTION** |
| **NON** | 9 | -0.44 (2.46) | -0.48 (2.26) | -0.11 (1.56) | -0.52 (1.72) |
| **DICE** | 10 | 1.90 (0.77) | 1.77 (0.92) | 1.40 (0.93) | 1.80 (1.31) |
| **LIME** | 5 | 0.40 (0.80) | 1.07 (0.76) | 1.13 (0.69) | 0.60 (0.89) |
| **SHAP** | 10 | 0.80 (1.53) | 0.77 (1.32) | 0.60 (1.31) | 0.63 (1.78) |

\*\*\*The table presents the various XAI tools and the Mean (STD) results for each tested variable.

Significant differences were found between **NON** and **DICE** in **CAUSABILITY** (H=2.4483, p-value=0.01435) **EXPLAINABILITY** (H=2.6153, p-value=0.00891) **SATISFACTION** (H=2.8835, p-value=0.00393) No other significant differences were found. This means that **DICE** enhances users’ **CAUSABILIT**, **EXPLAINABILITY** and **SATISFACTION** compared to **NON.**

Based on the user feedback results and the Kruskal-Wallis test, we found that the DiCE explanation tool significantly improved user experience in several areas. Users who received DiCE-based explanations reported higher scores in Causability, Explainability, and Satisfaction compared to users who received no explanation (NON group). These differences were statistically significant, showing that DiCE helped users better understand why the system made certain decisions, how it worked, and led to a more satisfying overall experience. In contrast, LIME and SHAP also showed positive average scores, but the differences compared to the NON group were not statistically significant. This suggests that while LIME and SHAP may be somewhat helpful, DiCE was the most effective explanation method for supporting user understanding and trust in the system.

### **6.2 System Testing Results Summary**

As part of our testing process, we created a Python script named test\_app.py to automatically check that the system works as expected. This script sends requests to the main Flask server and verifies the responses. It tests three main functions of the system: the /explain endpoint, which returns a match score along with a SHAP explanation and saves a visual SHAP plot; the /explain\_lime\_text endpoint, which generates a LIME explanation showing the most important words from the resume; and the /explain\_carla endpoint, which uses DiCE to give counterfactual suggestions—small changes that could improve the candidate’s match score. Each test simulates real user actions using the requests library, and checks that the server returns valid and useful results. The script also prints simple confirmation messages, helping us quickly see if everything works correctly.

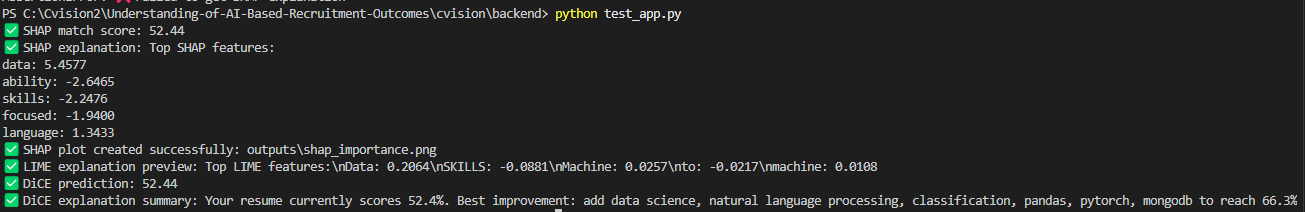
The system successfully returned a match score (for example, 52.4%) and gave SHAP explanations as both a visual bar plot and a short text showing the top five important features. The SHAP plot was saved as an image file in the outputs folder. LIME also gave explanations in the form of important words and their weights. DiCE worked well too, giving suggestions like adding new skills to improve the score along with a helpful summary. All the automatic tests we ran passed successfully, and the explanations were clear and easy to understand.

Figure 6 - app\_test.py output

### **6.3 Future Work**

In the future, we aim to make the system more robust by allowing it to adapt to different job descriptions, not just one fixed example. This would make the tool more flexible and useful for a wider range of job seekers. Another important step is to develop a mobile-friendly version of the system, so users can upload resumes and receive feedback directly from their phones. This would improve accessibility and convenience, especially for people who prefer using mobile devices.

# **7 User Guide**

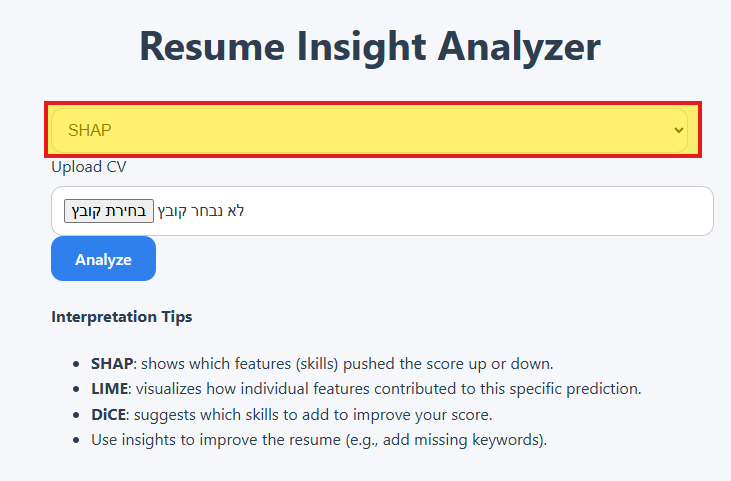


Figure 7 - Step 1

1. Select Explanation Tool

At the top of the page, choose the explanation method:

SHAP – Displays which features (skills or keywords) most influenced your match score, globally.

LIME – Highlights the specific words in your resume that affected the score for this prediction.

DiCE – Provides suggestions for changes (e.g., missing skills) that could improve your score.

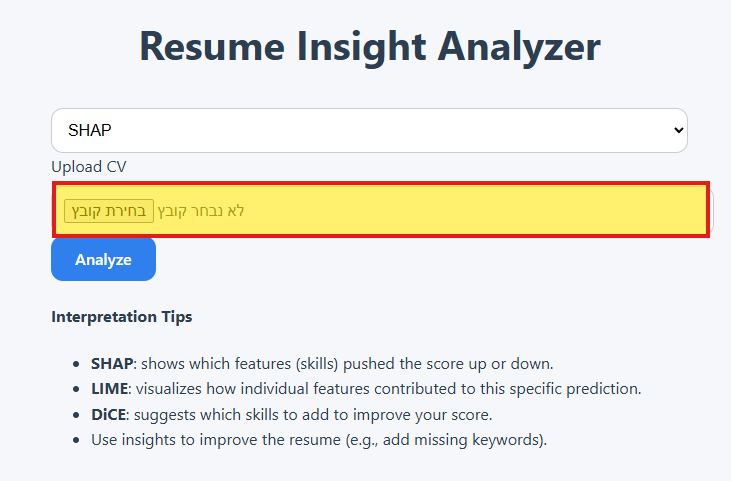


Figure 8 - Step 2

1. Upload Resume

Click the “Choose File” button to upload a PDF of your resume.

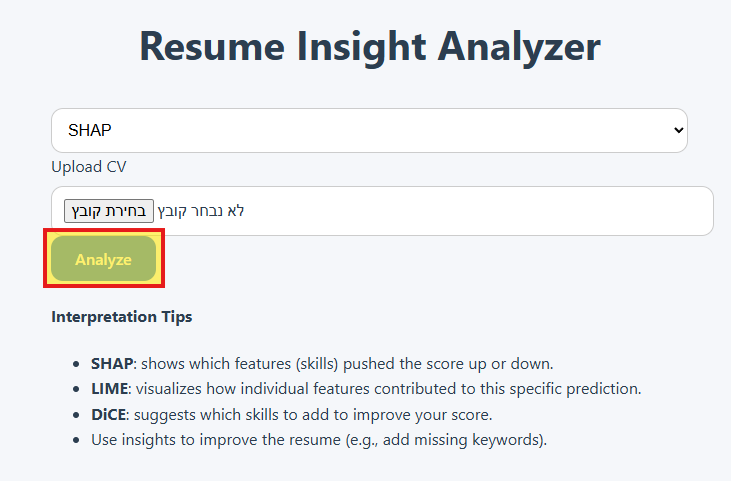


Figure 9 - Step 3

1. Click Analyze

Once your file is selected, press the “Analyze” button to run the XAI analysis.

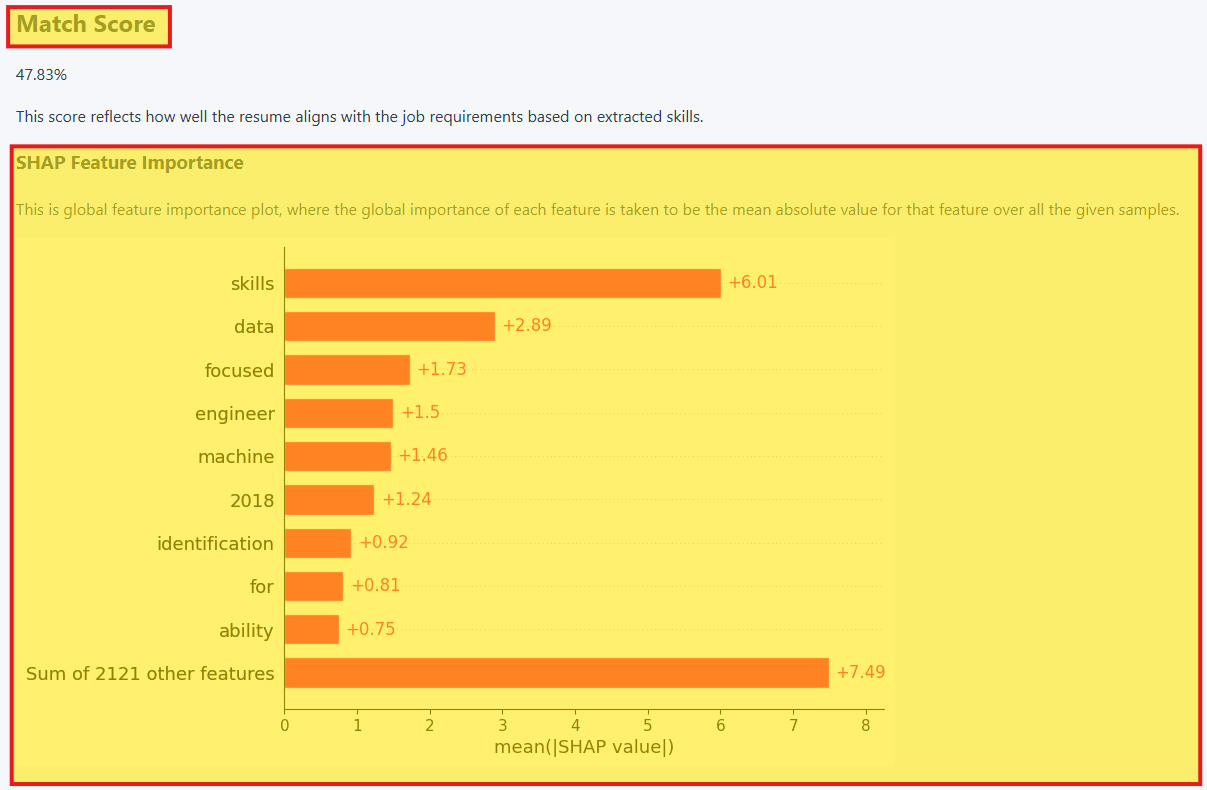


Figure 10 - Step 4

1. Review Results

Match Score – A percentage that indicates how closely your resume matches the job description.

Explanation Output – Depending on the selected tool, you’ll see either a bar chart (SHAP), highlighted text (LIME), or improvement suggestions (DiCE).

Interpretation Tips – These help you understand the visuals and how to act on them.

# **8 Maintenance Manual**

**System Requirements**

The system is built with a Python-based backend using Flask, and a React-based frontend. It requires Python 3.8–3.10 for compatibility with all libraries. Core dependencies include Flask, joblib, xgboost, scikit-learn, pdfminer.six, spacy, lime, and dice-ml for counterfactuals. For local testing or deployment, it is recommended to create a virtual environment and install dependencies via requirements.txt. Node.js and npm are required for running or building the frontend React application.

**Architecture Overview**

The application follows a modular architecture split into two layers: **frontend** and **backend**.

* The **frontend** (React) handles user interactions. It allows users to upload a resume, select an explanation method (SHAP, LIME, or DiCE), and visualize the output.
* The **backend** (Flask) serves as the processing engine. When a resume is uploaded, the text is extracted using pdfminer.six. The text is then processed using a pre-trained TF-IDF vectorizer and passed to a regression model (XGBoost) to generate a match score.
* Depending on the selected explanation tool, the system invokes one of the explanation modules (explain\_resume.py, explain\_lime\_text.py, or explain\_simple\_dice.py) to generate feedback.

The explanation is returned in the form of PNG (for SHAP), saved HTML (for LIME), or a dictionary of suggested changes (for DiCE). The outputs files are located in the folder "outputs" inside "backend" folder.

**Model Management**

The system uses two machine learning models:

1. A **text-based XGBoost regressor**, trained on TF-IDF features extracted from resume documents. This model is used for both SHAP and LIME explanations.
2. A **binary-skill-based XGBoost regressor**, trained on structured skill presence features. This model is used by DiCE to generate counterfactuals.

Models and vectorizers are saved using joblib:

* tfidf\_vectorizer.pkl: TF-IDF transformer used to vectorize resume text.
* xgb\_text\_model.pkl: Regression model for scoring resumes based on textual similarity.
* xgb\_regressor\_model.pkl: DiCE-compatible skill-based model.
* feature\_list.pkl: List of expected binary features (technical skills) used by the DiCE model.

If models are retrained or updated, the corresponding .pkl files must be replaced in the project directory. It is important that the feature space (especially for DiCE) remains consistent between training and inference.

**App Testing**

Automated backend tests are included in the test\_app.py script, which uses Python’s requests module to simulate API calls and validate system functionality. It must be run locally while the Flask server is running on http://localhost:5000.

The test suite covers:

• SHAP Explanation Test: Validates that uploading a resume with the SHAP tool returns both a prediction score and explanation, and confirms the SHAP plot image is saved and non-empty.

• LIME Explanation Test: Confirms that the /explain\_lime\_text endpoint returns a valid explanation string.

• DiCE Explanation Test: Sends a resume to the /explain\_carla endpoint and checks for both a prediction score and counterfactual suggestions.

To execute the test suite: python test\_app.py

Ensure that:

• The Flask backend is running.

• test\_resume.pdf is present in the same directory as test\_app.py.

• The outputs/ directory is writable and cleared if you want fresh output.

# **References**

[1] Sürücü, S., Küçük, B., & Aydın, M. K. (2024). An example of the application of artificial intelligence models in human resources processes. Afyon Kocatepe Üniversitesi İktisadi ve İdari Bilimler Fakültesi Dergisi, 26(Özel Sayı), 101-116.‏

[2] Koivunen, S., Olsson, T., Olshannikova, E., & Lindberg, A. (2019). Understanding decision-making in recruitment: Opportunities and challenges for information technology. Proceedings of the ACM on human-computer interaction, 3(GROUP), 1-22.‏

[3] Midway, S. R. (2020). Principles of effective data visualization. Patterns, 1(9).‏

[4] Moumita Kabir, "IntelCV: An Intelligent Skilled Resume Selection Method for Job Purposes," United International University, 2024.

[5] Pina, A.; Petersheim, C.; Cherian, J.; Lahey, J.N.; Alexander, G.; Hammond, T., "Using Machine Learning with Eye-Tracking Data to Predict if a Recruiter Will Approve a Resume," Machine Learning & Knowledge Extraction, 2023.

[6] Angelov, P. P., Soares, E. A., Jiang, R., Arnold, N. I., & Atkinson, P. M. (2021). Explainable artificial intelligence: An analytical review. *WIREs Data Mining and Knowledge Discovery*, 11(5), e1424.<https://doi.org/10.1002/widm.1424>

[7] Barrak, A., Zouaq, A., & Adams, B. (2021). Toward a traceable, explainable, and fair JD/Resume recommendation system. *Research Proposal*, Polytechnique Montréal, Québec, Canada.

[8] Zenkert, J., Klahold, A., & Fathi, M. (2018). Knowledge discovery in multidimensional knowledge representation framework: An integrative approach for the visualization of text analytics results. Iran Journal of Computer Science, 1(4), 199-216.‏

[9] Milev, P. (2023). The Role of Data Visualization in Enhancing Textual Analysis. Trakia Journal of Sciences, 21(1), 248-253.‏

[10] Kouki, P., Schaffer, J., Pujara, J., O'Donovan, J., & Getoor, L. (2019, March). Personalized explanations for hybrid recommender systems. In Proceedings of the 24th international conference on intelligent user interfaces (pp. 379-390).‏

[11] Verma, S., Dickerson, J., & Hines, K. (2020). Counterfactual explanations for machine learning: A review. arXiv preprint arXiv:2010.10596, 2, 1.‏

[12] Conner, C., Samuel, J., Kretinin, A., Samuel, Y., & Nadeau, L. (2020). A picture for the words! textual visualization in big data analytics. *arXiv preprint arXiv:2005.07849*.‏

[13] Shin, D. (2021). The effects of explainability and causability on perception, trust, and acceptance: Implications for explainable AI. *International journal of human-computer studies*, *146*, 102551.‏

[14] Raufi, B., Finnegan, C., & Longo, L. (2024, July). A comparative analysis of shap, lime, anchors, and dice for interpreting a dense neural network in credit card fraud detection. In World conference on explainable artificial intelligence (pp. 365-383). Cham: Springer Nature Switzerland.‏