

School of Science (SofS)

College of Engineering, Environment and Computing (EEC)

**6000CEM Individual Project Preparation** 2024

**Project Proposal**

**NAME: Jaidon Froggatt-Morton**

**SID: 11191168**

Contents

[1 ETHICS 3](#_Toc184236177)

[2 RESEARCH QUESTION 4](#_Toc184236178)

[3 INTRODUCTION 4](#_Toc184236179)

[4 LITERATURE REVIEW 4](#_Toc184236180)

[5 RESEARCH METHODS 9](#_Toc184236181)

[5.1 Research Methods and Tools 9](#_Toc184236182)

[5.1.1 Experimental Research 9](#_Toc184236183)

[5.1.2 Simulation and Application Development 9](#_Toc184236184)

[5.1.3 Summary 10](#_Toc184236185)

[5.2 Standard Practices 10](#_Toc184236186)

[5.2.1 Software Development Approach 10](#_Toc184236187)

[5.2.2 Technology Stack 10](#_Toc184236188)

[5.3 Project Plan 11](#_Toc184236189)

[5.4 Conclusion 12](#_Toc184236190)

[6 TECHNICAL ASPECTS 12](#_Toc184236191)

[6.1 Artefact Overview 12](#_Toc184236192)

[6.2 Hardware Requirements 13](#_Toc184236193)

[6.3 Software Stack 13](#_Toc184236194)

[6.4 Data Source 14](#_Toc184236195)

[6.5 Software Versions 14](#_Toc184236196)

[6.6 Deployment and Access 15](#_Toc184236197)

[7 ACKNOWLEDGEMENTS 15](#_Toc184236198)

[8 OPTIONAL EXTRAS 15](#_Toc184236199)

[8.1 RISK ANALYSIS 15](#_Toc184236200)

# ETHICS



# 2 RESEARCH QUESTION

Can Machine Learning algorithms be used to efficiently classify CVs and provide practical skill gap analysis for university students?

# 3 INTRODUCTION

The increasing competitiveness of the job market has shown the need for new tools to help students enhance their employability. The research will explore and compare the performance of four prominent machine learning models, **K-Nearest Neighbors (KNN)**, **Support Vector Machines (SVM)**, **Naive Bayes**, and **Decision Trees**, to determine which model is most suitable for classifying CVs. This project will also develop a machine learning-based system for CV classification and skill gap analysis, aiming to provide students with recommendations to close the gap between their current skills and the industry requirements.

These models will be evaluated based on key metrics such as **accuracy**, **efficiency**, and **scalability**.

**Aims and Objectives**

* Develop and evaluate **KNN, SVM, Naive Bayes, and Decision Tree** models for CV classification.
* Compare the models based on **accuracy**, **efficiency**, and **scalability**.
* Build a prototype web application using **Flask** that integrates the best-performing model.

Finally, this project will deliver a comprehensive evaluation of the four models and an intelligent, user-friendly system that automates CV analysis and provides personalised skill-building recommendations, contributing to students' career readiness and success in a competitive job market. This research aims to contribute to the growing literature around the use of AI in the career services industry.

# 4 LITERATURE REVIEW

**The Need for CV Classification**

In the current job market, characterised by intense competition, both job seekers and employers encounter significant obstacles. Job seekers are tasked with effectively showcasing their qualifications and competencies, while employers frequently find themselves overwhelmed by a vast number of applications for even a single position. The process of manually evaluating CVs can be time-consuming, inconsistent, and prone to human error (Bogen & Rieke, 2018). Consequently, there’s now an urgent for automated solutions to streamline the hiring process. These systems aim to match the right candidates with suitable opportunities more efficiently.

CV classification applies machine learning to address challenges faced in resume screening. It automates the categorisation process by analysing skills, experience, and job roles (Aggarwal & Zhai, 2012). It can assist recruiters in quickly discovering candidates that fit specific requirements, saving time spent on initial screening and enhancing total hiring efficiency (Raghavan et al., 2020). It can also aid in avoiding missed candidates due to fatigue.

**The Need for CV Optimisation**

Alongside the need for CV classification, there is a growing emphasis on CV optimisation. CV optimisation involves tailoring and enhancing resumes to maximise their impact, ensuring they effectively communicate a candidate's skills, experience, and suitability for a specific job role. As automated systems, such as Applicant Tracking Systems (ATS), have become an integral part of most hiring processes, optimising CVs to pass through these systems has become increasingly important.

Applicant Tracking Systems often use keyword matching to screen resumes, which means that a well-optimised CV needs to include the right keywords to make it through the initial screening process (Indeed [Indeed Editorial Team], 2024). By leveraging feature extraction techniques such as Term Frequency-Inverse Document Frequency (TF-IDF), job seekers can better understand which terms are relevant and should be emphasised in their resumes to align with the job descriptions. Optimising a CV this way increases makes it more likely to make it to the next stage of recruitment.

In the future Natural Language Processing tools, used together with Large Language Models, may be able to provide suggestions on improving phrasing, highlighting achievements, and structuring information effectively. Before this idea can be brought to life, more work needs to be completed on grasping fundamental text classification and producing reliable and fair datasets. Conjunctively, as discovered by Zinjad et al. (2024), more work needs to be completed regarding mitigating the phenomenon of AI hallucinating.

From an ethical perspective, while CV optimisation provides significant advantages for job seekers, it also raises questions about equity and fairness in the hiring process. Candidates who are more knowledgeable about optimisation techniques or have access to specialised tools may have an advantage over those who do not, potentially exacerbating inequalities in the job market (Bogen & Rieke, 2018). Ensuring that all job seekers have access to resources and guidance for optimising their CVs is crucial to promoting fairness and equal opportunity in the hiring process.

**The Role of Transparent Machine Learning in Ensuring Equity for All Job Seekers**

As discussed by Wang (2024), Machine Learning so far has mainly focused on equality; however, with its increasing impact on societal issues, it is becoming more and more important that we build algorithms focused on equity, accounting as fairness with societal context and historical injustices. To ensure equity in the recruiting process, machine learning is key, but transparency is also necessary. By making decision-making procedures transparent and accountable, responsible machine learning systems can lessen human biases and enable stakeholders to assess and have faith in the models' results.

Machine learning models can unintentionally reflect the biases in the training data (Barocas & Selbst, 2016), which may lead to discriminatory practices against certain applicants. Transparency in machine learning involves clearly communicating how models are trained, what data is used, and how decisions are made. This helps identify potential biases and ensure that corrective measures are taken to promote fairness.

To address these issues, it’s essential to develop fair and transparent algorithms that are trained on diverse and representative datasets. Techniques such as bias detection, algorithmic auditing, and providing clear explanations of model outputs can help identify and correct discrepancies, ensuring that machine learning systems are equitable in their decision-making processes (Raghavan et al., 2020).

Machine learning can also help promote equity by offering open-source applications to all job seekers. These would be used to guide on how they can improve their CVs chances against the ATS of large companies. Text classification and natural language processing (NLP) techniques can be used to analyse resumes and provide individualised feedback, helping job seekers improve their CVs based on specific job requirements. This democratises access to career development resources, giving candidates from diverse backgrounds a fair opportunity to present themselves effectively (Aggarwal & Zhai, 2012).

**Introduction to Text Classification Algorithms and NLP**

Text classification is a key component of CV classification and optimisation. It relies on machine learning algorithms to categorise resumes based on their content. Some of the most commonly used algorithms for text classification include Naive Bayes, Support Vector Machines (SVM), Decision Trees, and K-Nearest Neighbors (KNN). Each of these algorithms has unique strengths and weaknesses when applied to text data.

* **Naive Bayes**: This probabilistic classifier is based on Bayes' theorem, assuming independence between features. Despite the strong independence assumption, Naive Bayes performs well in text classification tasks, especially when dealing with high-dimensional data (McCallum & Nigam, 1998).
* **Support Vector Machines (SVM)**: SVM is a powerful supervised learning algorithm that works well for high-dimensional text data. It is effective for binary classification tasks (Cortes & Vapnik, 1995). As Aggarwal and Zhai (2012) mentioned, SVM classifiers work by "determining the optimal boundaries between classes".
* **Decision Trees**: Decision Trees are popular for their easy interpretability and ability to handle both numerical and categorical data (Quinlan, 1986). In a HR setting, this would be useful as it would allow hiring managers to view how the models decided on a candidate. However, they can become increasingly complex as the amount of data increases.
* **K-Nearest Neighbors (KNN)**: KNN is a non-parametric, instance-based learning algorithm that classifies resumes by finding the 'k' most similar instances in the training data based on a similarity metric such as cosine similarity. It is simple to implement and effective for smaller datasets, but it can become computationally expensive as the dataset grows (Cover & Hart, 1967).

Natural language processing (NLP) is crucial for extracting meaningful features from resumes. Several techniques are commonly used for similar purposes, including **Bag of Words (BoW)**, **Term Frequency-Inverse Document Frequency (TF-IDF)**, and **Word Embeddings**. BoW and TF-IDF convert text into numerical representations that machine learning models can use, while word embeddings capture semantic relationships between words, allowing for a deeper understanding of the content (Mikolov et al., 2013).

By using these algorithms and feature extraction techniques, CV classification systems can analyse resumes efficiently and categorise them accurately. However, the development and deployment of these models must be carefully managed to ensure that they do not inadvertently introduce bias or unfair practices. Continuous monitoring, transparency, and the use of diverse datasets are essential to maintain equity in the hiring process.

**Gaps in Literature Regarding a Comparison of Classification Models for CV Classification**

Despite the growing interest in CV classification, there are notable gaps in the literature regarding a comprehensive comparison of the different traditional classification models, such as Naive Bayes, Support Vector Machines (SVM), Decision Trees, and K-Nearest Neighbors (KNN), in the context of CV classification. This section discusses the existing gaps in comparing these models across three critical dimensions: accuracy, efficiency, and scalability, and their overall use in a production setting.

**Accuracy**: Accuracy is crucial when evaluating classification models, especially in a real-world application like CV classification, where incorrect classifications could lead to overlooked qualified candidates. Current literature provides fragmented insights into the accuracy of different models, but no comprehensive study systematically compares Naive Bayes, SVM, Decision Trees, and KNN for CV classification. Each model has strengths and weaknesses in different scenarios. For example, SVM is known for its high accuracy in text classification tasks, but its effectiveness depends on the careful tuning of hyperparameters (Cortes & Vapnik, 1995). Decision Trees, on the other hand, are more interpretable but tend to overfit when faced with high-dimensional data (Quinlan, 1986), which can affect their accuracy in CV classification. A systematic comparison of these models on the same dataset would provide valuable insights into their relative performance in the context of CV classification.

**Efficiency**: Efficiency is another critical area where gaps exist in the literature. Efficiency refers to the time and computational resources required to train and apply these models. In a production setting, especially when dealing with a high volume of CVs, the model's efficiency becomes a key factor. Naive Bayes is typically efficient because of its simplicity, while SVM can be computationally expensive, mainly when the dataset is large (McCallum & Nigam, 1998; Cortes & Vapnik, 1995). KNN is an instance-based learner, which means that it can become computationally expensive during the prediction phase, especially for large datasets (Cover & Hart, 1967). A detailed comparison of these models in terms of training and prediction time would help determine which model is best suited for a production environment where efficiency is crucial.

**Scalability**: Scalability refers to the ability of a model to handle increasing volumes of data without a significant drop in performance. In the context of CV classification, scalability is important because organisations may receive thousands of applications for a single position. When applied to CV classification, the literature needs a detailed comparison of the scalability of Naive Bayes, SVM, Decision Trees, and KNN. Decision Trees, for example, can become vast and complex as the dataset grows, which makes them harder to manage and interpret (Quinlan, 1986). SVMs can also struggle with scalability due to their reliance on support vectors, which may increase as the dataset grows (Cortes & Vapnik, 1995). KNN, being a memory-based algorithm, faces significant scalability issues as all training data must be stored and compared during classification. A comparative analysis of these models' scalability in handling large CV datasets would benefit anyone aiming to implement these models in real-world hiring processes or the wider career services industry.

**Use in a Production Setting**: The use of these models in a production setting involves not only accuracy, efficiency, and scalability but also considerations such as interpretability and ease of integration. Decision Trees are often preferred in production settings for their interpretability (Quinlan, 1986), which allows hiring managers to understand why a particular CV was classified in a certain way. However, SVMs, while accurate, are often seen as black-box models, which makes them less transparent (Rudin, 2019). Due to its simplicity, Naive Bayes can be easily integrated into production systems, but its strong independence assumption may limit its applicability for complex CV data (McCallum & Nigam, 1998). KNN, while simple, may be impractical for production due to its inefficiency with large datasets. A thorough investigation into how each of these models performs in a production setting for CV classification would help bridge the current gap in the literature.

**Conclusion**

In conclusion, CV classification and optimisation are crucial in addressing the challenges of today's job market, where job seekers need to present themselves effectively, and employers must efficiently screen many applications. Machine learning models, particularly responsible ones, provide an opportunity to streamline the hiring process. Effectively, minimising human biases, and ensuring that the right candidates are identified for the right roles.

While machine learning models such as Naive Bayes, SVM, Decision Trees, and KNN have demonstrated their potential for CV classification, there remain significant gaps in comparing these models comprehensively across critical dimensions in the context of CV classification. This project will contribute to the current knowledge base by systematically comparing these models in the context of CV classification. By analysing their strengths, weaknesses, and practical use cases, this research will provide valuable insights for both researchers and practitioners, guiding the implementation of effective, fair, and scalable CV classification systems for real-world hiring processes.

# 5 RESEARCH METHODS

This project will employ a combination of **Experimental Research** and **Simulation and Application Development** to address the research question effectively. This approach ensures a comprehensive evaluation of machine learning algorithms and a practical demonstration through a user-centric web application.

## 5.1 Research Methods and Tools

### 5.1.1 Experimental Research

**Purpose**: The purpose of the experimental research is to evaluate and compare different machine learning algorithms. Specifically, this research will focus on KNN, SVM, Naive Bayes, and Decision Trees for CV classification.

**Reason for Selection**: Experimental research aims to empirically determine which machine learning models best classify CVs. By experimenting with various algorithms, this research aims to successfully evaluate accuracy, efficiency, scalability, and overall performance.

**Deployment**:

* **Model Training and Evaluation**: The models will be implemented using **Scikit-Learn**. The dataset will be split into training and test sets, with **cross-validation** performed to assess each model’s baseline performance.
* **Hyperparameter Tuning**: Hyperparameter tuning will be performed using **Grid Search**. This method ensures that each algorithm achieves its best possible performance by exhaustively searching the parameter space.
* **Scalability Testing**: To assess scalability, each model will be tested across several metrics. Training time, prediction time, and memory usage will be evaluated on different subsets of the dataset—ranging from 20% to 100%. Tools such as the **time** library and **memory\_profiler** will be used, and the results will be visualised using **Matplotlib** for clear comparative analysis.

### 5.1.2 Simulation and Application Development

**Purpose**: The purpose is to develop a web-based application simulating real-world scenarios. In this application, students upload their CVs, which the machine learning model then analyses to classify and conduct a skill gap analysis.

**Reason for Selection**: Simulation and application development bridge the gap between theoretical research and practical usage. By developing a web application, the project demonstrates how machine learning can be used in a real-life scenario to support students' career development. Integrating the selected models into a functional system also validates their usefulness and feasibility.

**Deployment**:

* **Web Application Back-end**: **Flask** will be used to build the back-end for the application, enabling smooth integration of the machine learning models.
* **Front-end Development**: A simple **HTML, CSS, and JavaScript** interface will allow students to upload their CVs, making the tool accessible and easy to use.
* **Feature Implementation**: Once users upload their CVs, the application will classify them into a job category. This categorisation is powered by the best-performing machine learning model. Based on the job category, a skill gap analysis will be performed, providing tailored recommendations for career improvement.

**Simulation of Real-World Use**: The application will simulate a real-world environment where users can interact with the system, giving a clear picture of how machine learning-based CV analysis could assist students in improving their employability.

### 5.1.3 Summary

* **Experimental research** will be used to evaluate and compare the performance of machine learning algorithms for CV classification and skill gap analysis. The experiments will include hyperparameter tuning and scalability testing to determine the best model for the task.
* **Simulation and Application Development** involves building a web-based application using Flask. This application integrates the trained models to allow students to upload and analyse their CVs in a simulated real-world environment. This ensures practical applicability and proof-of-concept for automated CV analysis.

## 5.2 Standard Practices

### 5.2.1 Software Development Approach

**Agile Methodology**: The project will follow an Agile approach with iterative development cycles and regular reviews.

**Justification**: Agile is a flexible methodology that allows continuous refinement of features and feedback-driven improvements. Iterative cycles ensure that the models and features are tested and adjusted regularly, making Agile suitable for a research-focused project. The methodology promotes continuous attention to technical excellence and good design (Beck et al., 2001), which matches perfectly with this project.

### 5.2.2 Technology Stack

**Machine Learning**: **Scikit-Learn** for implementing the models (KNN, SVM, Naive Bayes, and Decision Trees).

**NLP**: **spaCy** for pre-processing and extracting relevant information from CVs.

**Web Development**: **Flask** for back-end development and simple **JavaScript**, **CSS**, and **HTML** for front-end implementation.

**Data Storage**: **SQLite** as the database solution to store job categories, skills, and user data.

## 5.3 Project Plan

The project plan is structured into timeboxes of two weeks with defined tasks and deliverables for each stage:

|  |  |  |
| --- | --- | --- |
| **Week** | **Tasks** | **Deliverables** |
| Project Proposal | - Conduct literature review to refine understanding of CV classification and skill gap analysis. | - Comprehensive literature review summary.  - Identified datasets and tools for the project. |
| 1-2 | - Collect and pre-process data (e.g., clean and tokenise Kaggle Resume Dataset). | - Pre-processed dataset ready for use.  - Data pre-processing scripts. |
| - Set up project environment and dependencies. | - Functional development environment (Python, Flask, etc.). |
| 3-4 | - Implement and test initial machine learning models (KNN, SVM, Naive Bayes, Decision Tree). | - Trained and tested models with default parameters.  - Initial performance metrics. |
| - Perform initial performance evaluation using cross-validation. | - Cross-validation results for all models.  - Identified strengths and weaknesses of each model. |
| - Develop a basic Flask web application for uploading CVs and displaying results. | - Functional prototype of the web application for primary CV classification. |
| 5-6 | - Conduct hyperparameter tuning using Grid Search for each model. | - Optimised hyperparameters for each model.  - Improved performance metrics. |
| - Conduct scalability testing to assess training time, prediction time, and memory usage. | - Scalability test results for each model.  - Visualisations of training and prediction times. |
| - Expand the web application to include skill extraction and gap analysis functionality. | - Enhanced web application with skill extraction and gap analysis. |
| 7-8 | - Compare all models based on accuracy, scalability, and efficiency. | - Comprehensive comparison report for all machine learning models.  - Selection of best model. |
| - Integrate the best-performing model into the web application. | - Finalised classification and skill gap analysis module in the web application. |
| 9-10 | - Finalise the web application and prepare documentation. | - Fully functional web application.  - User manual and technical documentation. |
| - Write up results, analysis, and discussion for the dissertation. | - Completed dissertation document with results and analysis. |
| - Submit the completed project. | - Submission of the dissertation and project deliverables. |

## 5.4 Conclusion

This project combines Experimental Research and Simulation and Application Development. By integrating these approaches with standard software development practices, it aims to fully address the research question and deliver a practical solution.

# 6 TECHNICAL ASPECTS

The project will develop a web-based application that uses machine learning to classify CVs and provide skill gap analysis. It will also develop a comparative analysis of the four algorithms used in CV classification as a basis for future research. The following technical aspects outline the artefact, including the hardware and software used, data sources, and implementation details.

## 6.1 Artefact Overview

**Analysis of the Four Machine Learning Models:**

* The first artefact is an analysis of the four machine learning models: K-Nearest Neighbors (KNN), Support Vector Machine (SVM), Naive Bayes, and Decision Tree.
* This analysis will evaluate each model based on performance metrics such as accuracy, scalability, training time, prediction time, and memory usage.
* This analysis aims to determine the most suitable model for CV classification and skill gap analysis. It will also help provide insight into each model's strengths and weaknesses and their practical applicability to the problem at hand.

**Web Application for Real-World Simulation:**

* The second artefact is a web-based application built using Flask. This application will simulate a real-world scenario where university students can upload their CVs for analysis.
* The application will classify the CV into an appropriate job category. It will then conduct a skill gap analysis, providing personalised recommendations to improve the CV and enhance employability.
* The broader job categories will be split into the top 10 common jobs within each category, allowing users to explore a more holistic view of possible career paths.
* This web application serves as a proof-of-concept, demonstrating the practical feasibility of using machine learning to provide career support.

Together, these artefacts will provide both a deep theoretical understanding of the different machine learning models and a practical demonstration of how the models can be applied to improve CV quality and support career development.

## 6.2 Hardware Requirements

* **Development Machine**: The development will be conducted on a desktop computer with the following specifications:
  + **Processor**: AMD Ryzen 5 5600x
  + **RAM**: 32 GB DDR4
  + **Storage**: 2 TB M.2 SSD
  + **Operating System**: Windows 11
  + **Graphics Card**: Nvidia GTX 3070ti
* **Deployment Server**: The application will be hosted on the development machine to avoid server costs.

## 6.3 Software Stack

* **Back-end: Flask**
  + **Justification:** Flask is an incredibly flexible and lightweight Python web framework. Its flexibility makes it well-suited for developing web applications that require integration with machine learning models. Flask allows seamless integration of Scikit-Learn models and is highly customisable. It is ideal for creating a proof-of-concept system with minimal setup time (Copperwaite & Leifer, 2015).
* **Front-end: HTML, CSS, and JavaScript**
  + **Justification:** HTML, CSS, and JavaScript are standard technologies for front-end development. The base languages are an easy way to create simple user interfaces for web applications. Using these technologies keeps the front-end development simple, allowing more focus on implementing machine learning functionalities.
* **Database: SQLite**
  + **Justification:** SQLite is a lightweight and serverless SQL database that is ideal for small to moderate data storage needs. Given that this project is an academic proof-of-concept with limited data storage requirements, SQLite offers an easy-to-implement solution that eliminates the overhead of setting up a more complex database system.
* **Machine Learning and NLP: Scikit-Learn and spaCy**
  + Scikit-Learn:
    - **Justification:** Scikit-Learn is a comprehensive library for machine learning in Python that supports a wide range of models, pre-processing, and evaluation tools. It has proven to be one of the most efficient and well-maintained libraries for implementing classic machine learning models like KNN, SVM, Naive Bayes, and Decision Trees. Its consistency and flexibility make it ideal for experimentation (Pedregosa et al., 2011).
  + spaCy:
    - **Justification:** spaCy is an industrial-strength NLP library in Python that provides state-of-the-art models for tokenisation, named entity recognition (NER), and other NLP tasks. It is highly efficient and optimised for processing large volumes of text, making it suitable for parsing and pre-processing CV data. spaCy was selected due to its unique ideology of being for production use, meaning it is positioned well to help answer the research question (Honnibal & Montani, 2017).
  + TF-IDF Vectorizer:
    - **Justification:** TF-IDF (Term Frequency-Inverse Document Frequency) is a feature extraction method commonly used for text data. It helps identify the most significant words in a document relative to a corpus. TF-IDF balances text representation and efficiency for classic machine learning algorithms (Ramos, 2003).
* **IDE: Visual Studio Code (VSCode) with Jupyter Notebook Extension**
  + **Justification:** VSCode is a powerful and versatile code editor that supports multiple programming languages and offers extensions for different purposes. The Jupyter Notebook extension allows it to use the features of Jupyter Notebooks within the familiar IDE.
  + **Justification:** Jupyter Notebook is an open-source web application that facilitates creating and sharing documents containing live code, equations, visualisations, and narrative text. The code can be organised into cells, which can be individually run, and the output is stored as part of the document (Kluyver et al., 2016).
* **Version Control: Git**

## 6.4 Data Source

* The primary dataset for training and testing the machine learning models is the **Kaggle "Resume Dataset**". This dataset contains resumes in text format, categorised by job category. It will be pre-processed using **spaCy** and **TF-IDF**.
* **Data Format**: The dataset consists of text data in CSV format, with columns representing the resume content as a string and the following job categories:
  + HR, Designer, Information-Technology, Teacher, Advocate, Business-Development, Healthcare, Fitness, Agriculture, BPO, Sales, Consultant, Digital-Media, Automobile, Chef, Finance, Apparel, Engineering, Accountant, Construction, Public-Relations, Banking, Arts, Aviation
* **Acknowledgement**: My supervisor has approved the use of the Kaggle Resume Dataset, and appropriate ethical considerations will be addressed in the ethics application.
* The Link to the dataset is: https://www.kaggle.com/datasets/snehaanbhawal/resume-dataset

## 6.5 Software Versions

* **Python**: 3.9 or higher
* **Flask**: 2.0
* **Scikit-Learn**: 0.24
* **spaCy**: 3.0
* **SQLite**: 3.32
* **VSCode**: Latest version with Jupyter Notebook extension

## 6.6 Deployment and Access

The web application will be hosted on the development machine. Users can access the application using a secondary 2021 MacBook Air for user testing.

# 7 ACKNOWLEDGEMENTS

Not Applicable

# 8 OPTIONAL EXTRAS

## 8.1 RISK ANALYSIS

|  |  |  |
| --- | --- | --- |
| **Risk** | **Impact (1-5)** | **Mitigation** |
| Difficulty in Acquiring Necessary Machine Learning Libraries | 4 - Major delays in implementing and testing models. | Use alternative libraries or open-source versions; adjust project scope to work with available resources. |
| Inadequate Performance of Machine Learning Models | 3 - Poor classification accuracy could lead to subpar skill gap analysis and recommendations. | Perform additional hyperparameter tuning or switch to other machine learning models; provide analysis of model limitations and include improvements in future work. |
| Computational Limitations for Model Training | 3 - Limited computational resources could delay the training and optimisation of models. | Optimise models for computational efficiency (e.g., using dimensionality reduction); switch to simpler models if necessary; perform training on a cloud platform with higher computational power if budget permits. |
| Bugs in Flask Application | 2 - Bugs could impact the functionality and user experience of the application. | Test thoroughly during each stage of development; use debugging tools and logs; engage peers to help with bug identification through code reviews. |
| Time Constraints | 4 - Limited time may affect the completion of the project, particularly for the training and testing phases. | Set realistic goals and prioritise tasks; use an agile approach to make adjustments when needed; regularly review progress and reallocate time as necessary. |
| Poor Hyperparameter Tuning Results | 3 - Suboptimal tuning could lead to lower accuracy or efficiency. | Use Grid Search or Random Search for a wider parameter sweep; consult literature for recommended hyperparameters; discuss these challenges and areas of potential improvement in the report. |
| Version Control Issues | 2 - Mismanagement of code versions could lead to lost work or delays. | Use Git for version control with proper branching strategies; frequently commit changes; create backups to avoid code loss. |
| Lack of Supervisor Guidance | 3 - Lack of feedback might lead to unclear direction in specific parts of the research. | Schedule meetings well in advance; seek alternative guidance from peers or online academic forums; document and work on tasks independently to maintain progress until guidance is received. |

**References**

Aggarwal, C. C., & Zhai, C. (2012). *Mining text data*. Springer Science & Business Media.

Barocas, S., & Selbst, A. D. (2016). Big data’s disparate impact. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.2477899

Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., & Thomas, D. (2001). Manifesto for Agile software development. *Agile Manifesto*. https://agilemanifesto.org/

Bogen, M., & Rieke, A. (2018). Help wanted: an examination of hiring algorithms, equity, and bias. *Upturn*. https://apo.org.au/sites/default/files/resource-files/2018-12/apo-nid210071.pdf

Brown, P., & Hesketh, A. (2004). The Mismanagement of Talent: Employability and Jobs in the knowledge economy. *OUP Catalogue*. https://ideas.repec.org/b/oxp/obooks/9780199269549.html

Copperwaite, M., & Leifer, C. (2015). *Learning Flask Framework*. Packt Publishing Ltd.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*(3), 273–297. https://doi.org/10.1007/bf00994018

Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, *13*(1), 21–27. https://doi.org/10.1109/tit.1967.1053964

Honnibal, M., & Montani, I. (2017). spaCy 2: Natural language understanding with bloom embeddings, convolutional neural networks, and incremental parsing. *To Appear*. https://spacy.io/

Indeed [Indeed Editorial Team]. (2024, July 2). Get Your Resume Seen With ATS Keywords. *Indeed Career Guide*. Retrieved December 3, 2024, from https://www.indeed.com/career-advice/resumes-cover-letters/ats-resume-keywords

Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B. E., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J. B., Grout, J., Corlay, S., Ivanov, P., Avila, D., Abdalla, S., & Willing, C. (2016). Jupyter Notebooks – a publishing format for reproducible computational workflows. *International Conference on Electronic Publishing*, 87–90. https://doi.org/10.3233/978-1-61499-649-1-87

McCallum, A., & Nigam, K. (1998). A comparison of event models for naive bayes text classification. *National Conference on Artificial Intelligence*, 41–48. http://aaaipress.org/Papers/Workshops/1998/WS-98-05/WS98-05-007.pdf

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. In *arXiv* (arXiv:1301.3781v3). Retrieved December 3, 2024, from https://arxiv.org/abs/1301.3781

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, É. (2011). SciKit-Learn: Machine Learning in Python. *Journal of Machine Learning Research*. https://doi.org/10.5555/1953048.2078195

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, *1*(1), 81–106. https://doi.org/10.1007/bf00116251

Raghavan, M., Barocas, S., Kleinberg, J., & Levy, K. (2020). Mitigating bias in algorithmic hiring: evaluating claims and practices. *FAT\**, 469–481. https://doi.org/10.1145/3351095.3372828

Ramos, J. (2003). Using TF-IDF to determine word relevance in document queries. *Unpublished Manuscript*. https://citeseerx.ist.psu.edu/document?repid=rep1;type=pdf;doi=b3bf6373ff41a115197cb5b30e57830c16130c2c

Rudin, C. (2019). Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, *1*(5), 206–215. https://doi.org/10.1038/s42256-019-0048-x

Wang, A. (2024). *Operationalizing Responsible Machine Learning: From Equality Towards Equity* [PHD Dissertation, Princeton University]. <https://media.proquest.com/media/hms/PFT/2/Z2zjY?_s=Fiu4ykGDnJP7VMxCOhJLIQIB8%2F4%3D>

Zinjad, S. B., Bhattacharjee, A., Bhilegaonkar, A., & Liu, H. (2024). ResumeFlow: an LLM-facilitated pipeline for personalized resume generation and refinement. *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. https://doi.org/10.1145/3626772.3657680