Use of Machine Learning (Text Recognition, Natural Language Processing, and Large Language Models) for Hand-Written Answer Sheet Evaluation

By

Brian Mutugi
151174

Master of Science in Data Science and Analytics
June, 2024

Use of Machine Learning (Text Recognition, Natural Language Processing, and Large Language Models) for Hand-Written Answer Sheet Evaluation

By

Brian Mutugi

151174

Submitted in Partial Fulfilment of the Requirements for the Degree of Master of Science in Data Science and Analytics

Institute of Mathematical Sciences

Strathmore UNIVERSITY

Nairobi, Kenya

June 2024

Declaration and Approval

Declaration

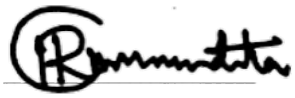
I declare that this work has not been previously submitted and approved for the award of a degree by this or any other University. To the best of my knowledge and belief, the thesis contains no material previously published or written by another person except where due reference is made in the thesis itself.

Student's Name: Brian Mutugi

Sign:  Date: 01/12/2023

Approval

The thesis of Brian Mutugi was reviewed and approved for examination by the following:

Sign:  Date: 18/01/2024

Dr. Rogers Ondiba Ochenge,

Institute of Mathematical Sciences,
Strathmore University

Dr. Kennedy Senagi,
Institute of Mathematical Sciences,
Strathmore University

Abstract

The realm of machine learning, encompassing text recognition, natural language processing and large language models, presents a transformative potential for the education sector, particularly in the evaluation of hand-written tests.

This dissertation explored the use of these technologies in hand-written tests, acknowledging their prevalence and addressing inherent challenges encountered when evaluating the tests.

The significant time required for evaluation often leads to delayed results and academic calendars, while the physical and mental strain on the evaluators, coupled with varying levels of skill and knowledge can lead to inconsistencies and inaccuracies in scoring. To address these challenges, this research explored the development of a machine learning approach capable of automatically extracting questions and student responses from images/pictures of exam papers and answer sheets. The approach then assessed student responses with the corresponding exam questions using pre-trained large language models.

This research adopted the CRISP-DM (Cross-Industry Standard Process for Data Mining) framework —business understanding, data understanding, data preparation, modelling, evaluation, and deployment, to streamline the development and comparison of machine learning models.
The result of was a machine learning model designed to process photos of question papers and answer sheets. It extracted text questions and answers, seamlessly facilitating the interaction between users and the technology. The textual content could then be analyzed by a pre-trained large language model, which performed the assessment and provided feedback.

Enhancing the efficiency of assessments and elevating the accuracy and objectivity of feedback provided to learners, this approach promised to significantly reduce the time and effort involved in the evaluation process; thereby overcoming the limitations of current practices in hand-written test evaluation.

Acknowledgements

I extend my deepest gratitude to all educators in my journey - lecturers, including my supervisors, teachers, mentors, parents, and all who impart knowledge with the hope of fostering betterment, enlightenment, and growth. Their invaluable role and the profound impact of their work have not only shaped my professional knowledge and personal development,
but also left an indelible mark on the lives of countless learners.

I am also indebted to Strathmore University for affording me the invaluable opportunity to further my skills and knowledge in the Data Science field. The experience and learning I have gained under their auspices have been indispensable to my academic and professional growth.

My profound gratitude goes to my family, their faith in my abilities has been a constant source of motivation.

Furthermore, I wish to express my sincere thanks to my friends and colleagues. Their encouragement and valuable inputs have significantly contributed to my journey. The journey has indeed been enriching, and I am deeply appreciative of their role in facilitating my growth and development.

I am truly thankful to everyone who played a part in this endeavor for their guidance, encouragement, and support.

Contents

List of Figures

List of Abbreviations

AI	Artificial Intelligence
NLP	Natural Language Processing
ML	Machine Learning
LLM	Large Language Model
GPT	Generative Pre-trained Transformer
JSON	JavaScript Object Notation
OCR	Optical Character Recognition
CSV	Comma-separated Values (CSV) File
CRISP-DM	Cross-Industry Standard Process for Data Mining framework
SVM	Support Vector Machine

Chapter 1

Introduction

1.1 Background to the Study

Machine learning is a subfield of artificial intelligence that gives computers the ability to learn without explicitly being programmed. It has significantly impacted various sectors, notably education. Machine learning's ability to analyze large datasets and make predictions or decisions without explicit programming is revolutionizing educational practices, (Jordan & Mitchell, 2015).

In education, machine learning has been instrumental in developing personalized learning experiences, automating grading systems, and facilitating adaptive learning technologies (Baker & Smith, 2019).

Despite the revolution, hand-written tests remain a cornerstone in educational assessment., valued for their flexibility and effectiveness in evaluating student understanding (Thompson & Lee, 2018). These tests, however, pose significant challenges, including the substantial time and effort required for manual evaluation, the potential for human error, and inconsistencies in scoring due to subjective judgements (Robinson, 2020).

Traditional methods for evaluating hand-written tests largely involve manual assessment, which is time-consuming and prone to evaluator fatigue and bias. The limitations of these methods include significant time commitments, strain on evaluators, and potential inconsistencies and inaccuracies in scoring, which can impact the overall quality of education (Johnson, 2021).

There is a growing need for more efficient,accurate and consistent methods to evaluate hand-written tests. Such improvements can enhance educational outcomes and reduce the administrative burden on educators (Williams, 2019).Machine learning emerges as a promising solution, with the potential to automate and standardize the evaluation process while minimizing human errors (Davis & Patel, 2020).

Key machine learning technologies relevant to this research included text recognition, natural language processing, and large language models. Text recognition allows for the conversion of handwritten text into digital formats, natural language processing enables the understanding and interpretation of human language, and large language models can process and evaluate complex text data (Nguyen & Brown, 2022).
These technologies have shown potential in various applications, from automated customer service to content analysis, suggesting their applicability in the educational context for evaluating hand-written tests (Lee & Kim, 2018).

This research aimed to explore machine learning solutions to address the identified gaps and challenges in current hand-written test evaluation methods. By automating the evaluation process, the study seeked to enhance efficiency, accuracy, and consistency in scoring.

The potential benefits of this research included reduced evaluation time, minimized evaluator strain, and standardized assessment procedures, contributing to the overall improvement of educational assessment methods.

1.2 Problem Statement

The primary problem addressed in this research was the inefficient and often inconsistent evaluation of hand-written answer sheets in educational settings.

Educators and examiners face a significant challenge in manual grading of these tests, which not only demands considerable time and effort but also introduces a risk of subjective bias and errors in scoring.

This inefficiency in the evaluation process can lead to delayed feedback for students too, affecting their learning and academic progression.

The lack of standardized, objective, and efficient method for assessing hand-written tests pose a critical issue in the current educational assessment landscape.

1.3 Research Objectives and Questions

This research aimed to achieve the following objectives:

1. Review existing literature on the application of machine learning (ML) in educational assessments,
2. Investigate and benchmark various machine learning techniques for extracting textual content from images of hand-written tests,
3. Select and configure the optimal machine learning model based on the comparative analysis,
4. Develop and implement a user-friendly online interface for deploying the chosen machine learning solution, time permitting.

To achieve the objectives, the focus shall be on addressing the following questions:

1. What is the current state of research on the application of machine learning (ML) in educational assessments?
2. Which ML techniques excel at processing hand-written exam images, and how do they compare?
3. Which is the most applicable and optimal ML approach for processing hand-written exam images, and how is it configured?
4. How can a user-friendly interface for an ML-based educational assessment tool be developed within time constraints?

1.4 Scope and Limitations

This research primarily focused on developing a machine learning model capable of extracting questions from images of question papers and student answers from images of student answers sheets, then integration of large language models for evaluation.
The initial scope was confined to exams that are theoretical in nature, involving minimal arithmetic calculations and diagrams. Such focus enabled a more targeted approach to text recognition and natural language processing, which are central to the design.
The deployment concentrated on the development of a core system, prioritizing functionality that can be implemented on a standard computing platform. Development of a web interface was considered as a secondary objective, contingent on the availability of time and resources.
This prioritization ensured that the primary goal of creating a robust and effective extraction and evaluation system was not compromised. Geographically, the research was focused on the Kenyan educational system, with adaptability being a nice-to-have feature of the system, but not a primary focus area.
The research was therefore limited in the following ways:

1. Its ability to scale to different types of exams, particularly those involving complex diagrams or extensive numerical data. The focus on theoretical exams may limit applicability to other forms of assessment,

2. Its applicability and effectiveness in settings beyond the Kenyan education system remained to be explored,

3. Performance was dependent on quality and clarity of the images. Poor image quality could significantly impact the accuracy of the question-and-answer extraction,

4. The development of a web interface was subject to time constraints and may not have been realized in the scope of the research. The accessibility of this system via mobile devices was not guaranteed,

5. While the system aimed to be adaptable, its initial development was not specifically addressing regional educational standards or languages other than English.

1.5 Research Relevance (Justification)

The usefulness of this research was in its potential to significantly enhance the efficiency, accuracy, and consistency of the evaluation process in educational assessments, particularly for hand-written answer sheets. This research addressed a critical need in the education sector, providing a solution that is beneficial to multiple stakeholders including educators, students, and educational institutions.

1. For educators, they are the primary beneficiaries of this research, burdened with the time-consuming task of manually evaluating and grading hand-written tests. By automating the extraction and evaluation of student answers, educators can redirect their efforts from the labour-and-time-intensive marking process to more pedagogically impactful activities such as curriculum development, student engagement and personalized instruction,

2. For students, they stood to benefit from this research through quicker feedback on their performance. This could enhance their learning experience by providing timely insights into their understanding of subjects, allowing for more immediate and effective interventions,

3. For educational institutions, such a system could lead to standardized and unbiased grading process, enhancing the reliability and credibility of the assessment results. This standardization is crucial for maintaining the integrity of educational qualifications and ensuring fairness in student evaluation,

4. For innovation in education technology, this showcased the practical application of machine learning in a critical area of education assessment. It paved way for further research and development in the use of advanced technologies for educational purposes, particularly in regions and institutions where such innovations are yet to be fully explored,

5. For future applications, the insights gained from this research have the potential to be adapted for a broader range of assessment types in the future. This scalability underscores the long-term relevance of the research in continually improving and modernizing educational assessment methods.

By addressing the inefficiencies in the current process of evaluating handwritten answer sheets, this research offered a substantial contribution to the field of education. Its utility extends beyond immediate academic settings, potentially influencing broader educational policies and practices by demonstrating the effective integration of machine learning technologies in assessment processes.

Chapter 2

Literature Review

The application of machine learning and natural language processing in educational settings is an emerging field of research with significant potential to revolutionize traditional methodologies. This literature review critically examines three pivotal studies that contribute to our understanding of how these technologies are being integrated into educational practices.

Each study provides unique insights into different aspects of machine learning and natural language processing applications, ranging from exam paper analysis to broader educational challenges, thereby laying a foundation for the proposed research on using these technologies for hand-written answer sheet evaluation.

2.1 Machine Learning in Education

(Das et al., 2019) explore the application of NLP in academic contexts, specifically focusing on the classification and analysis of exam paper contents. Their system not only aids in preparing students for exams by analyzing past papers but also assists educators in crafting effective exam questions. The utility of this system is further enhanced by its user friendly-interface, which allows the uploading and parsing of exam papers and employs sophisticated NLP techniques, like tf-idf, to analyze and categorize content.
In addition to the system's primary functionalities, Sharma et. al. also

discuss the potential customization and adaptability in different education contexts. The flexibility of their NLP-based system to cater to various subjects and academic levels demonstrates its broad applicability. This aspect is particularly relevant to the proposed research, as it shows the adaptability of NLP systems in handling diverse types of textual data, a feature essential for analyzing a wide range of hand-written answer sheets. The ability to customize and adapt to different educational materials aligns with the goal of creating a versatile system for evaluating hand-written tests.
The research by Sharma et al. is particularly relevant to this study as it showcases the practical applications of NLP in educational assessment, highlighting the potential of these technologies in automating and enhancing the analysis of written content, a core aspect of the proposed research.

(Kucak et al., 2018) provide a comprehensive overview of the various applications of machine learning in education. Their systematic approach categorized the uses of machine learning into grading, content retention, student performance prediction and testing. This research is crucial for understanding the multifaceted implications of machine learning in educational settings, from enhancing the objectivity of grading systems to personalizing learning experiences.
Further expanding on their findings, (Kucak et al., 2018) also emphasize the future potential of machine learning in educational settings, particularly in developing more personalized and adaptive learning experiences. This forward-looking perspective is crucial for the proposed research, as it highlights the evolving nature of machine learning applications in education.
The ability of machine learning to adapt and personalize educational experiences can be mirrored in the development of a more tailored and responsive system for evaluating hand-written answer sheets, thereby enhancing the educational assessment process.
The study's findings are particularly relevant to the current research, as they highlight the potential of machine learning in automating and standardizing assessment processes, a key objective in the development of a system for evaluating hand-written answer sheets.

(Urbina Najera & Calleja Mora, 2017) review offers a holistic perspective on the challenges in education and how data mining and machine learning can address them. The paper discusses the application of these technologies in various education contexts, demonstrating their capability to uncover in-

sights from large datasets and improve educational outcomes.
The significance of this work lies in its comprehensive overview of the data mining process and the practical application of machine learning in education. The research aligns closely with the objectives of the proposed study as it illustrates the potential of machine learning and data mining in enhancing educational assessment methods, especially in analyzing and interpreting complex data sets like hand-written answer sheets.
Moreover, the research highlights the importance of ethical considerations and data privacy in the application of data mining and machine learning in education. This aspect is very important, especially when dealing with student data and performance metrics. The ethical considerations discussed in the paper provide a framework that is crucial for the proposed research. Ensuring that the developed system for evaluating hand-written answer sheets adheres to data privacy and ethical standards, safeguarding student information while providing accurate and unbiased assessments.

2.2 Large Language Models in Education

The paper titled "Application of Large Language Models for Language Teaching and Assessment Technology" by Andrew Caines et. al. (Caines et al., 2023) offers an insightful exploration into the field of large language models such as PaLM and GPT-4, particularly focusing on their applications in education technology. The release of these models gas not only captivated social media and public attention, sparking both excitement and concern over their capabilities, but also brought unprecedented attention in the field of NLP. This surge in interest is particularly promising for the realm of education technology, as it paves the way for innovative teaching and assessment methods.
Caines and team delve into the multifaceted use of LLMs in AI-driven teaching and assessment, covering areas such as content creation, calibration, feedback and the associated risks and ethical concerns. The paper emphasizes the necessity of accurate prompts for effective text-generation.
Notably, the paper highlights GPT-4, released in March 2023 by OpenAI as the most prominent LLM at the time of the research. This models integration with education technologies such as Duolingo and Khan Academy, and its predecessor GPT-3's application in the ChatGPT chatbot, are prime

examples of this synergy.
The collaboration between OpenAI and Duolingo in developing Duolingo Max, a conversation simulation tool, is a case in point for the applications of these technologies.
Furthermore, the paper address the concept of 'human in the loop' for content generation to mitigate issues like hallucinations in LLM outputs. This approach is vital in calibrating assessment results. The importance of human oversight is reiterated throughout the paper, aligning with studies such as those by Jeon & Lee, who explored the use of ChatGPT for student assessment. An intriguing aspect discussed is the unexplored potential of LLMs in explaining assessment predictions, tapping into the field of explainable Artificial Intelligence (AI). The comprehensive coverage of these topics in the paper not only highlights the capabilities and potential uses of LLMs in education but also points to the critical role of human intervention in ensuring the effectiveness and ethical application of these technologies.

(Hsiao et al., 2023) delve into the rapidly evolving landscape of large language models (LLMs) such as ChatGPT, Bard, Bing Chatbot, and LLaMa, which have garnered significant attention since November 2022. Their study addresses the profound impact these models have had on academic settings, particularly in universities. The challenge they highlight is the difficulty in distinguishing content generated by LLMs from that written by humans, a distinction crucial in academic settings. This ambiguity raises concerns about potential misuse, such as ghostwriting, leading to unreliable assessment decisions. Hsiao et al.'s research is timely and relevant, as it underscores a growing challenge in educational assessment – ensuring the integrity and authenticity of student work in an era where AI-generated content is increasingly indistinguishable from human output.
In response to these challenges, (Hsiao et al., 2023) propose a framework based on the Design Science Research (DSR) approach to assist educators in redesigning writing assessments. This framework aims to enhance assessment methods to address the implications of LLMs in academic writing. The DSR approach involves recognizing and defining the problem, establishing specific objectives for a proposed solution, developing and demonstrating the solution, evaluating its effectiveness, and communicating the problem, solution, and practical value to relevant audiences. The significance of this study for the current research lies in its innovative approach to redesigning assessment methods in light of advanced AI technologies. It provides valuable insights

into adapting educational assessment frameworks to maintain their relevance and effectiveness in the rapidly evolving landscape of AI and machine learning.

In their groundbreaking work, (Gan et al., 2023) explore the rapidly evolving field of large language models (LLMs) in the context of education. Highlighting the advancements in big data, artificial intelligence, and Web 3.0, the authors underscore the growing significance of LLMs as powerful tools in understanding and generating natural language. These models, trained on extensive corpora, demonstrate remarkable capabilities in various applications, including natural language processing, machine translation, and dialogue systems. The paper presents an in-depth analysis of the potential applications of LLMs in education, addressing challenges like individual differences among students, allocation of teaching resources, and assessment of teaching effectiveness. The authors propose that incorporating LLMs into education can support personalized learning, intelligent tutoring, and adaptive assessment, thereby enhancing the quality and experience of education. (Gan et al., 2023) further discuss the challenges in traditional education models, such as low student engagement and unequal distribution of resources. LLMs, with their advanced processing and analytical capabilities, are posited as a solution to revolutionize these traditional models. By enabling personalized learning and intelligent tutoring, LLMs can meet the diverse needs of students more effectively. The paper highlights how the use of big data in education has led to the accumulation of extensive learning data, which, when analyzed by LLMs, can uncover learning patterns, evaluate outcomes, and offer personalized recommendations. The authors emphasize the uniqueness of large models, particularly their capacity to handle complex tasks and analyze large-scale data, making them highly applicable in educational settings for providing more nuanced and tailored educational services.
The research presented by (Gan et al., 2023) is highly relevant to the current study focusing on the use of machine learning for hand-written answer sheet evaluation. The insights into the capabilities of LLMs in processing and analyzing extensive data sets align closely with the objectives of the proposed research, which seeks to leverage similar technologies for evaluating educational assessments. The discussion on personalized and adaptive assessment tools provided by LLMs offers valuable perspectives for developing a system that can accurately interpret and analyze hand-written content, enhancing the efficiency and effectiveness of educational assessments. This

paper's exploration of EduLLMs and their potential in transforming traditional educational methods provides a foundation for understanding how similar technologies can be applied in the specific context of hand-written answer sheet evaluation.

In the article "ChatGPT has entered the classroom: how LLMs could transform education," (Extance, 2023) the impact of Large Language Models (LLMs) like ChatGPT in educational environments is explored. Beghetto's experiment at Arizona State University, where creativity-focused chatbots enhanced diverse thinking, exemplifies the creative application of these models in education (Extance, 2023). Despite their potential for personalized learning and time-saving, concerns about cheating and data privacy were raised, along with the models' "brittleness" or inaccuracies in certain contexts (Extance, 2023).
The implementation of LLMs in classrooms has elicited both concerns and acknowledgment of their potential as 'thought partners.' Despite the risk of misuse for cheating and privacy concerns, these models have been recognized for their ability to act as cost-effective, always-available educational assistants. Their innovative applications include enhancing creativity and offering personalized educational support, highlighting their transformative potential. However, balancing these opportunities with critical oversight is essential to ensure accuracy and ethical usage (Extance, 2023).
The article's insights are particularly relevant to the current research on using machine learning for hand-written answer sheet evaluation. The application of LLMs in text recognition and analysis aligns with the project's components. Addressing challenges similar to those highlighted in the article, such as ensuring accuracy and ethical considerations, is paramount in this research. The strengths and limitations of these models, as well as their potential in personalizing learning and assessment, provide a valuable framework for developing the proposed machine learning-based evaluation system (Extance, 2023).

2.3 ML, NLP and NLP in Assessment

(Gautam & Srinivasulu, 2023) emphasize the transformative role of machine learning (ML) and deep learning (DL) in modernizing the educational system, particularly in digital examinations. The study highlights the challenges in developing ethical and unbiased assessment algorithms, addressing assessment biases, and ensuring fairness for diverse student backgrounds. It underscores the need for research into algorithms that provide impartial evaluations and explores the ethical implications of AI in educational assessments. Additionally, the paper delves into concerns about data security and privacy in digital examinations, advocating for the integration of advanced encryption and privacy-preserving techniques.
The authors argue that while the integration of ML and DL into digital examinations offers enhanced efficiency and accessibility, it is not without its challenges. These challenges include developing ethical and unbiased assessment algorithms and addressing concerns about data security and privacy. The paper proposes a multi-faceted approach to tackle these challenges, combining fairness-aware assessment algorithms with robust data security measures. This approach aims to safeguard student data, prevent academic dishonesty, and ensure compliance with data protection regulations.
The findings and discussions in Gautam & Srinivasulu et al.'s (Gautam & Srinivasulu, 2023) paper are highly relevant to the current research on the use of machine learning for hand-written answer sheet evaluation. The focus on developing unbiased and ethical assessment algorithms aligns with the objectives of ensuring fairness and accuracy in evaluations. The exploration of data security and privacy considerations provides crucial insights into handling sensitive student data, a key aspect of the proposed research. The study's comprehensive approach to integrating ML and DL in educational assessments offers valuable guidance for implementing similar technologies in hand-written answer sheet evaluation systems.

The paper "Machine Learning for Prediction of Grades" (Lenæs & Myksvoll, 2023) discusses the initiative by BI Norwegian Business School to explore machine learning (ML) applications for predicting grades in an effort to reduce grading costs. The school aimed to cut costs significantly, considering options like reducing printing, altering exam elements, and incorporating digital home exams. The key focus was on whether an ML model could predict grades akin to a human grader. The project evolved to view ML not

as a replacement for human graders but as a supplemental tool for student training. Their best model achieved a 90 percent accuracy in categorizing submissions into two grade groups, indicating potential for both cost reduction and quality consistency improvement in the grading process.
BI Norwegian Business School's exploration of ML for grade prediction is situated within a broader context of cost-cutting and efficiency in the education sector. The project underscores the challenges and opportunities of integrating ML into academic assessments. While the initial goal was to automate grading, the direction shifted towards leveraging ML as a supportive tool for students, reflecting on the multifaceted potential of AI in education. The project's evolution from a cost-saving initiative to a student-centric learning tool exemplifies the dynamic nature of ML applications in academic environments.
This study's exploration of ML for grade prediction is relevant to your research on hand-written answer sheet evaluation. The approach of using ML as a supportive tool rather than a direct replacement for human assessment aligns with the objectives of enhancing educational processes through technology. The insights from BI's initiative can inform the development of ML models in your research, particularly in terms of ethical considerations, accuracy, and the balance between automated processes and human oversight.

In their innovative research, (Hameed & Sadiq, 2023) address the challenges of automating the assessment of subjective questions in exams, a task typically demanding manual grading due to its complexity. They propose an automatic short answer grading system utilizing semantic networks and a Support Vector Machine (SVM) to measure semantic similarity between student answers and correct answers. This method represents a significant advancement in educational technology, offering a solution that closely mimics human comprehension and judgment in grading. By constructing semantic networks to represent the relationship between words in texts and extracting features to train the SVM model, their system efficiently predicts the degree of semantic similarity, outperforming previous methods on the Mohler dataset.
The significance of this study lies in its approach to handling the nuances of short answer grading, a task that has traditionally posed significant challenges for automation. The paper demonstrates the feasibility of using advanced ML techniques, specifically semantic networks and SVMs, to accurately assess free-text responses. This is particularly relevant in the context

of e-learning and distance education, where the demand for automated assessment systems has grown. The authors' approach of combining semantic analysis with ML algorithms provides an efficient and effective solution for evaluating short answers, highlighting the potential of AI in revolutionizing traditional assessment methods.
Hameed and Sadiq's work is particularly relevant to the current research on the use of machine learning for hand-written answer sheet evaluation. Their method of semantic similarity measurement aligns with the objectives of accurately assessing subjective answers. The application of semantic networks and SVM in their research provides valuable insights into the development of sophisticated grading systems capable of understanding and evaluating complex textual responses, directly applicable to the goals of enhancing the assessment process in educational settings.

(Phan et al., 2023) present a novel approach in automated essay feedback (AEF) for Japanese language students. Their study explores a hybrid system that combines rule-based traits with data-driven models, utilizing the 6+1 writing-trait theory to provide both holistic and detailed feedback on student essays. The system is designed to identify areas of improvement in student writing and suggest corrective feedback, thereby enhancing the learning experience. This method diverges from traditional automated essay scoring (AES) systems by providing more meaningful and actionable feedback, rather than just a holistic score.
The implementation of this AEF system represents a significant advancement in educational technology, particularly in the context of language education. By integrating semantic networks and machine learning techniques, such as Support Vector Machines (SVM), the system effectively assesses short answers, offering detailed feedback on various aspects of writing, including structure, style, and readability. This comprehensive approach to feedback is crucial in distance education and e-learning environments, where personalized and detailed assessment is essential for student development.
The research by Phan, Hasegawa, and Gu is relevant to the current project on machine learning for hand-written answer sheet evaluation. Their approach to integrating semantic analysis and machine learning aligns with the objective of providing detailed and actionable feedback on student responses. The insights from their study can inform the development of a similar system for evaluating hand-written answers, focusing on providing comprehensive feedback that goes beyond mere scoring.

In their study, (Vanathi et al., 2023) ,Dr. B. Vanathi and colleagues explore an automated exam paper evaluation system designed to address the inefficiencies in manual grading. The system is particularly focused on subjective response evaluation, comprising the extraction of answers from handwritten sheets and assessing their similarity to provided answers. Utilizing optical character recognition (OCR) and natural language processing (NLP), the system aims to automate the evaluation of handwritten texts by extracting keywords and comparing them to a teacher-uploaded answer key.
The researchers delve into the challenges posed by manual evaluation, such as time consumption and potential bias. They emphasize the need for automated systems, especially in the context of distance education and e-learning, where exams are increasingly conducted electronically. The paper also discusses the use of string-based similarity and semantic analysis in evaluating answers, highlighting the innovative use of technology in educational assessment.

In related works, the paper references various studies, including Yusuf Perwej and Ashish Chaturvedi's work, (Perwej & Chaturvedi, 2011) on handwriting recognition using neural networks and Sakhapara et al.'s, (Sakhapara et al., 2019), machine learning approach for subjective response rating systems. These studies underline the evolving landscape of automated assessment methods, moving from simple pattern matching to more sophisticated semantic analyses and machine learning techniques.
The proposed system utilizes APIs for OCR and deep learning for handwriting recognition, demonstrating the integration of advanced technological solutions in educational assessment. The focus on subjective questions, often a challenge for automated systems, marks a significant advancement in the field. The researchers also consider the implications of automation on educational institutions, highlighting how such systems can streamline processes and reduce the manual workload for educators.

The work of Dr. B. Vanathi and colleagues, (Vanathi et al., 2023) is a valuable contribution to the field of educational technology. It showcases the potential of combining OCR and NLP in creating efficient and unbiased assessment systems for subjective responses. This study is relevant to the broader context of technological advancements in education, offering insights into how automation can enhance the accuracy and efficiency of academic

evaluations.

The study "AUTOMATED EXAM PAPER EVALUATION SYSTEM", (Sanuvala & Fatima, 2021) is closely related to the proposed research as it demonstrates the practical application of OCR and NLP in automating the grading process, particularly for subjective responses, which aligns with the objective of developing an efficient and unbiased system for evaluating hand-written answers. The insights from this study can inform the development of the system, especially in terms of integrating advanced technologies to accurately interpret and assess hand-written content.

2.4 Summary

The literature review has provided a comprehensive overview of the integration of machine learning and natural language processing in the field of education. The studies by Sharma et al., Kucak et al., Urbina Najera, Caines et al., Hsiao et al., Gan et al., and others collectively underscore the transformative potential of these technologies in revolutionizing educational assessment methods. These works highlight the capabilities of NLP and ML in automating the analysis and evaluation of written content, particularly in the context of hand-written answer sheets, thus offering significant improvements in terms of efficiency, accuracy, and objectivity.

The insights gained from these studies also emphasize the need for ethical considerations and data privacy, especially in handling sensitive student information. Overall, this body of research provides a solid foundation for the proposed study, demonstrating the feasibility and value of applying advanced technological solutions to enhance educational assessment processes. As the field of AI continues to evolve, its application in education offers promising avenues for further research and development, particularly in creating more personalized, adaptive, and fair assessment systems.

Chapter 3

Methodology

3.1 Introduction

The target setting was a system capable of processing photographs of question papers and answer sheets, and subsequently parsing them into a machine-readable format. By first extracting the textual content from these images using machine learning models and then transcribing this information into a standardized format, specifically JSON (JavaScript Object Notation). This structured data format is integral to the system's function, facilitating the seamless transfer of information to subsequent analytical processes.

Once the questions and answers are extracted and structured, the system pairs each answer to its corresponding question. This pairing is crucial as it lays the groundwork for the next phase of the system's operation—the automated evaluation. Here, a pre-trained large language model (LLM) steps in to assess the student's responses. It provides an objective evaluation, yielding output that includes the marks awarded, the questions posed, the answers given, and any relevant feedback, all encapsulated within the versatile JSON format.

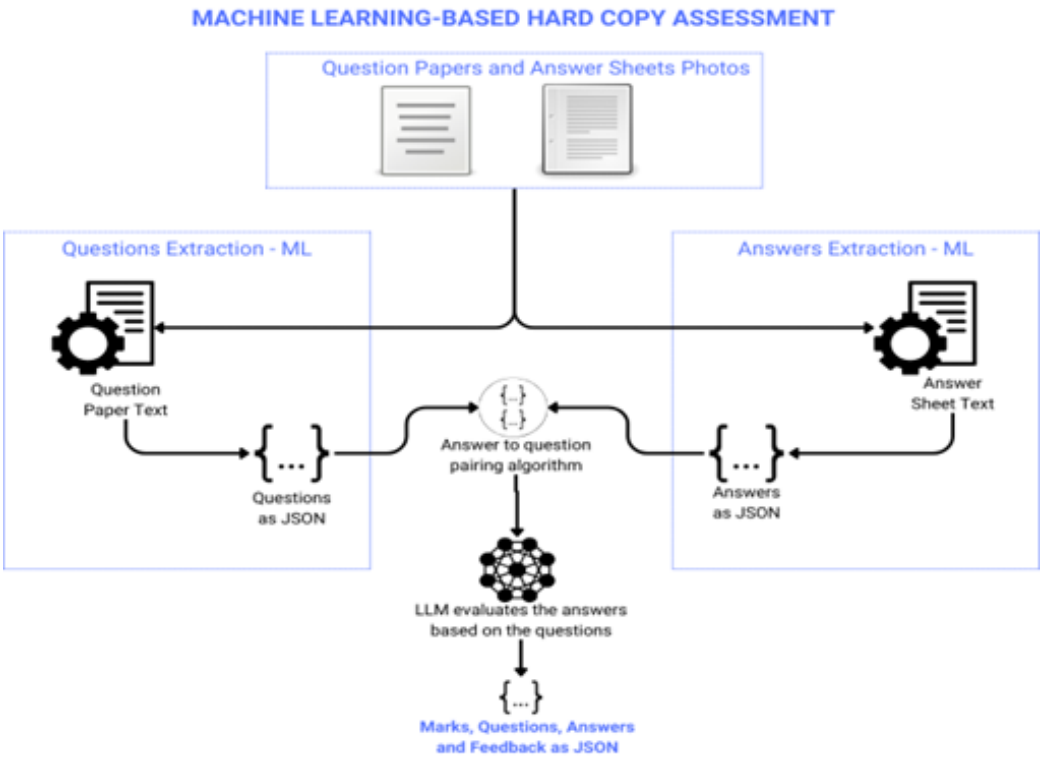


Figure 3.1: The high level system architecture

This chapter outlines the research methodology that was adopted to achieve this, utilizing the Cross-Industry Standard Process for Data Mining (CRISP-DM) framework. CRISP-DM, a widely recognized methodology in data science projects, provides a structured approach to solving complex problems through data mining and machine learning.

This framework was chosen for its flexibility, industry-agnostic applicability, and emphasis on iterative learning and improvement. By following the CRISP-DM steps, this research systematically addressed the challenges of evaluating hand-written tests using machine learning models, from understanding the requirements and data to technical setup of the solution.

3.2 Business Understanding

The prevalence of hand-written exams across educational institutions presents a significant challenge in terms of time and energy expended by educators in marking these exams. This traditional method of assessment, while ensuring that students engage deeply with the material, often leads to prolonged waiting periods for results.

Furthermore, the subjective nature of manual grading can introduce biases, affecting the fairness and reliability of outcomes. Compounding these issues is the lack of standardized evaluation criteria across many institutions, which places the burden of assessment entirely on educators. This not only strains resources but also results in the loss of valuable learning insights, as most question-and-answer centric assessments are not systematically collected or analyzed for long-term educational enhancement.

Amidst these challenges, the disruptive potential of Artificial Intelligence (AI) in the educational sector cannot be overstated. AI technologies offer promising solutions to streamline the assessment process, reduce evaluator burden, and provide timely and unbiased feedback to students. Moreover, the recent surge in AI capabilities presents untapped opportunities for enhancing the evaluation of hand-written assessments. Utilizing AI for educational assessments could revolutionize how educators approach testing and grading, making the process more efficient and objective.

In Kenya, the penetration of smartphones offers a unique opportunity to leverage technology in bridging the gap between traditional hand-written assessments and the possibilities afforded by AI. A tool that can seamlessly convert images of hand-written answer sheets into digital text for analysis the automate evaluations could significantly streamline the grading process. Such a solution would not only save time for educators but also ensure that students receive prompt and fair evaluations. Furthermore, the data collected through this digital transformation can provide valuable insights into learning outcomes and trends, facilitating a more informed approach to education.

This research aimed to explore the application of AI in the assessment of hand-written exams, focusing on developing a machine learning model capable of accurately extracting and evaluating text from images of exam papers

and answer sheets. By harnessing the recent advancements in AI and the widespread use of smartphones, this study seeks to address the challenges, offering a novel solution that enhances the efficiency, fairness, and insightfulness of educational assessments.

3.3 Data Understanding

In the quest to create a model that mirrors real-world conditions as closely as possible, the foundational dataset comprised smartphone camera photos of question papers and answer sheets. This initial dataset was ethically compiled from a variety of sources, including photographs of exams and answers I personally took during my academic tenure at Strathmore University, alongside datasets sourced from publicly available platforms such as Kaggle. To augment this initial dataset and ensure a robust foundation for machine learning, we also generated additional synthetic data, aiming to achieve a dataset size conducive to reaching reliable machine learning performance thresholds.

The initial collection consisted of 30 question papers and 100 answer sheets. Recognizing the need for a more extensive dataset to train our model effectively, an additional set of 30 question papers and 100 answer sheets was synthetically generated. This approach not only increased the volume of data available for training but also enhanced the diversity and complexity of the dataset, simulating a wide range of real-life scenarios more accurately.

The question papers, predominantly printed, presented a relatively straightforward challenge in terms of text extraction.The uniformity of the printed text and the generally high quality of these images facilitated easier preprocessing and analysis. Conversely, the answer sheets posed a more complex challenge. These documents varied significantly, with some being fully handwritten, while others combined printed questions with hand-written answers. This variety necessitated a more sophisticated approach to text extraction, as the model had to accurately interpret and differentiate between printed and cursive handwriting under varying conditions.

Furthermore, the quality of the images in the dataset varied widely, from clear and vivid to blurred and noisy.

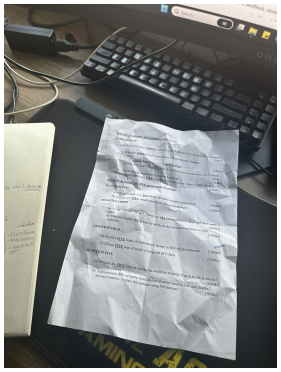


Figure 3.2: An example of a question paper photo of poor quality

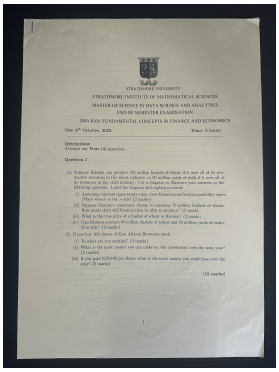


Figure 3.3: An example of a question paper photo of fairly good quality

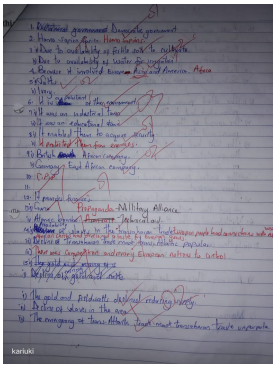


Figure 3.4: An example of an answer sheet photo of poor quality

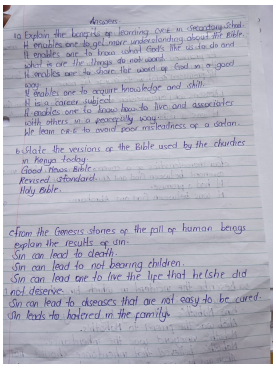


Figure 3.5: An example of an answer sheet photo of fairly good quality

This variance is a realistic representation of the challenges inherent in processing real-life data, where conditions are not always ideal. The diversity in image quality required the implementation of advanced preprocessing techniques to ensure that the text could be extracted accurately across all documents, regardless of their initial clarity.

The images contain the following sections which are useful in text extraction and evaluation:

1. **exam_question**: The specific area in the photo where the exam question is located, serving as a focal point for extracting the text of the question,

2. **exam_answer**: The section of the answer sheet where the student's answer is written, crucial for evaluating the response against the question,

3. **exam_question_section**: Identifies the broader section of the exam that contains a group of related questions, often themed or categorized together,

4. **exam_question_number**: The number assigned to a specific question within the paper, aiding in the organization and referencing of questions,

5. **exam_question_marks**: Indicates the marks allocated for a particular question, essential for assessing the weight of each question in the overall score,

6. **exam_institution**: The name of the educational institution conducting the exam, providing context regarding the source of the examination,

7. **exam_name**: The title or name of the exam, often reflecting the nature or purpose of the assessment,

8. **exam_instructions**: General instructions provided at the beginning of the exam, guiding how students should approach answering the questions,

9. **exam_unit_name**: The specific unit or subject name of the exam, clarifying the academic focus of the assessment,
10. **exam_level_name**: Denotes the academic level or class for which the exam is intended, such as "Grade 10" or "First Year University",
11. **exam_examined_name**: The name of the examinee, usually found on the answer sheet, personalizing the assessment,
12. **exam_examined_id**: The identification number or code assigned to the examinee, ensuring the anonymity and security of the assessment process,
13. **exam_answer_question_number**: The question number as it appears on the answer sheet, linking the answer back to the corresponding question on the question paper,
14. **exam_question_paper_header**: The header section of the question paper, which may include the institution's logo, exam name, date, and other relevant information,
15. **exam_question_section_marks**: The total marks allocated for a specific section of the exam, important for understanding the section's significance within the overall assessment,
16. **exam_question_section_instructions**: Specific instructions given for a section of the exam, directing how to approach the set of questions within that section,
17. **exam_question_section_name**: The name or title of a specific section within the exam, often indicating the theme or subject matter of the grouped questions,
18. **exam_question_paper_footer**: The footer section of the question paper, which may contain additional information such as page numbers, confidentiality notes, or institution details.

Through careful selection and augmentation of our dataset, we aimed to capture a wide spectrum of scenarios encountered in the evaluation of handwritten exams. This diverse and realistic dataset provided a solid foundation for developing a machine learning model capable of navigating the complexities of text extraction from various document types and conditions.

3.4 Data Preparation

3.4.1 Introduction

To achieve the required level of automated data extraction, two distinct approaches were explored:

1. **Traditional Classification Machine Learning Algorithms**: These algorithms are particularly advantageous for question extraction from question papers. Their benefits, such as being lightweight and straightforward to understand, make them a viable option for the initial stages of text extraction.

2. **Deep Learning CV Model Approach - Specifically YOLO 8**: This approach is adept at handling the dual task of extracting both questions from question papers and answers from answer sheets. YOLO (You Only Look Once) version 8's prowess in real-time object detection makes it a promising candidate for more intricate extraction needs.

The forthcoming sections will delve into the specifics of how the photographs were prepared and processed to optimize them for these two machine learning approaches. We will explore the preprocessing steps, feature engineering, and the fine-tuning required to ready the data for the tasks ahead, ensuring the system's efficacy in interpreting and analyzing hand-written text.

3.4.2 Preparation for Traditional Classification Machine Learning Algorithms

These are the steps undertaken in the preparation process to ready the data for the Classification machine learning models. This phase is critical as it involves tailoring the dataset for a binary classification task: determining whether a given text excerpt represents a question. The preparation steps are crafted to enhance the model's ability to discern questions from non-questions with precision.

3.4.2.1 OCR raw text extraction from photograph

This is the process of Optical Character Recognition (OCR) raw text extraction from a photograph, an essential step in converting image-based data

```
# Read the image
image_path = "images/IMG_1904.jpg"
image = cv2.imread(image_path)
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
ocr_text = pytesseract.image_to_string(gray)
```

Figure 3.6: Python code snippet - using pytesseract for ocr text from image

into a textual corpus for analysis.

The operation commences with the invocation of OpenCV's cv2.imread function, which reads the photograph from a specified path. Subsequent conversion of the image to grayscale is achieved using cv2.cvtColor, an operation that simplifies the image data and enhances the subsequent OCR process's accuracy.

Utilizing the pytesseract library, which interfaces with the Tesseract OCR engine, the grayscale image undergoes text extraction, outputting a digital corpus. The adoption of this method is crucial for transitioning from static images to a modifiable, analyzable format, setting the groundwork for automated data processing:

3.4.2.2 Raw text preparation

The extracted text corpus undergoes processing to identify and structure line breaks accurately, which is crucial for ensuring the seamless prediction of continuous question text.

A dedicated function is employed to parse through the text line by line, segregating and concatenating lines based on specific rules that signify the continuation of a question.

This process organizes the OCR output into a format that more effectively represents the structure and flow of the original document, setting the stage for the accurate identification of discrete data elements, such as individual questions and answers, in later stages of the machine learning pipeline.

Function to return text line by line from mixed up image text corpus

```
def prep_text(raw_text):
    # Split the raw text into lines
    lines = raw_text.split('\n')

    lines = [line for line in lines if line.strip()]


    # Initialize variables
    processed_lines = []
    current_line = ''
    next_line_starts_lower = False  # Flag to indicate if the next line starts with a lowercase letter

    for i, line in enumerate(lines):
        # Split by 'ks) ' and process each part
        if 'ks) ' in line:
            parts = line.split('ks) ')
            for part_index, part in enumerate(parts):
                # Append 'ks) ' to all but the last part
                processed_line = part + ('ks) ' if part_index < len(parts) - 1 else '')

                if processed_line.strip():
                    processed_lines.append((current_line + processed_line).strip())
                    current_line = ''
        else:
            # If the next line starts with a lowercase letter, join it with the current line
            if i + 1 < len(lines) and lines[i + 1][0].islower():
                current_line += line + ' '
            else:
                processed_lines.append(current_line + line)
                current_line = ''


    # Add the last line if it's not empty
    if current_line:
        processed_lines.append(current_line.strip())

    # Join processed lines with line breaks and return
    refined_text = '\n'.join(processed_lines)
    return refined_text
```

Figure 3.7: Python code snippet - raw text preparation function

3.4.2.3 Feature generation from extracted lines of text

Specific characteristics typical of exam questions, such as prompt words ("what," "how," "explain"), numbering patterns, and the presence of the word 'question' at the start, are identified. Additionally, the detection of total score or marks typically found at the end of questions is incorporated into the feature set.

The function **extract_attributes** systematically checks for these features within the text to construct a multidimensional feature space that machine learning models can utilize.

By analyzing the structure of the lines and identifying key textual features, the script prepares a dataset that encapsulates both the raw text and its associated features, along with a binary label indicating whether the line is a question. This dataset, comprising 1000 rows, is then saved as a comma-separated values (CSV) file, creating a structured, labeled dataset ready for training and testing the classification machine learning models. The CSV format offers the dual benefit of accessibility and ease of use, enabling seamless integration into the machine learning pipeline for model development.

```python
def extract_attributes(text):
    # List of prompt words
    prompt_words = ["what", "why", "when", "where", "explain", "describe",␣
↪"give", "how", "discuss", "write", "define","do"]

    # First word, ignoring leading spaces and converting to lowercase for␣
↪case-insensitive comparison
    first_word = text.strip().split()[0].lower() if text.strip() else ""
    single_word = int(len(text.split()) == 1) if text else 0

    # Check if the first word is a prompt word
    prompt_start = int(first_word in prompt_words)

    # Refined patterns for 'indexed_start'
    patterns = [
        r"^\(?[a-zA-Z]\)\s",    # Matches patterns like (a) , b)
        r"^\d+\.\s",            # Matches patterns like 1.
        r"^\(?[ivxlcdm]+\)\s",  # Matches patterns like (i) , ii)
    ]
    # Check the pattern against the first 1 to 5 characters of the text
    indexed_start = int(any(re.match(pattern, text.strip()[:7]) for pattern in␣
↪patterns))

    question_start = int(first_word == "question")
    last_four_chars = text[-4:] if len(text) >= 4 else text
```

```python
    ends_with_ks = int(last_four_chars == "ks) " or last_four_chars == "rks)" ␣
↪or last_four_chars == "mks)")
    lower_start = int(text.strip()[0].islower()) if text.strip() else 0
    upper_start = int(text.strip()[0].isupper()) if text.strip() else 0

    return first_word, indexed_start, question_start, last_four_chars,␣
↪ends_with_ks, lower_start, upper_start, prompt_start, single_word
```

Figure 3.8: Python code snippet - feature generation function

3.4.2.4 Tokenizing the text and converting boolean values to numerical

This includes tokenization, where the text is broken down into individual tokens or words, a process executed by the vectorizer. Concurrently, boolean values that indicate the presence or absence of certain features are converted into binary form, ensuring that all inputs to the model are numerical. These numerical representations are crucial as they enable the machine learning algorithms to apply mathematical operations to the data.

The function also horizontally stacks the vectorized text and numeric features to form a composite feature set. This combined feature array is then ready to be fed into the classification models, providing a rich, multi-dimensional representation of the original text that is optimized for predictive analysis. The harmonization of tokenized text with binary numerical features ensures a more nuanced dataset, facilitating the development of a model that can accurately discern and classify questions from the corpus.

```
def features_from_text(text):

    # Extract features using the provided function
    extracted_features = extract_attributes(text)

    # Creating a DataFrame with the extracted features
    features_df = pd.DataFrame([extracted_features], columns=['first_word',␣
↪'indexed_start', 'question_start', 'last_four_chars', 'ends_with_ks',␣
↪'lower_start', 'upper_start', 'prompt_start', 'single_word'])

    # Exclude the first feature ('question') which is a string and not used in␣
↪the numeric model
    numeric_features = features_df[['ends_with_ks', 'indexed_start',␣
↪'prompt_start', 'lower_start', 'upper_start']]
    # Vectorize the line text
    line_vectorized = vectorizer.transform([text])

    # Combine vectorized text with other features
    combined_features = hstack([line_vectorized, np.array(numeric_features).
↪reshape(1, -1)])

    return combined_features
```

Figure 3.9: Python code snippet - tokenization function

3.4.2.5 Correlation analysis

Employed to discern the strength and direction of relationships between various textual features and the likelihood of a text being classified as a question. By converting boolean feature columns to integers, this analysis quantifies the degree to which each characteristic correlates with the target variable 'is_question'.

```
# Convert boolean columns to integer for correlation analysis
bool_cols = ['indexed_start', 'question_start', 'ends_with_ks', 'lower_start',␣
 ↪'upper_start','prompt_start','single_word']
df[bool_cols] = df[bool_cols].astype(int)

# Compute correlation matrix
correlation_matrix = df.corr()

# Plotting the heatmap
plt.figure(figsize=(10, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title("Correlation Matrix")
plt.show()

# Recommendations for best features to predict 'is_question' based on␣
 ↪correlation
correlation_with_target = correlation_matrix['is_question'].
 ↪sort_values(ascending=False)
best_features = correlation_with_target.index[1:]  # Exclude 'is_question'␣
 ↪itself
best_features_recommendations = best_features[:3]  # Top 3 features
```

Figure 3.10: Python code snippet - correlation analysis

A heatmap visualization is then generated to provide an intuitive representation of these correlations, allowing for immediate visual discernment of the most predictive features:

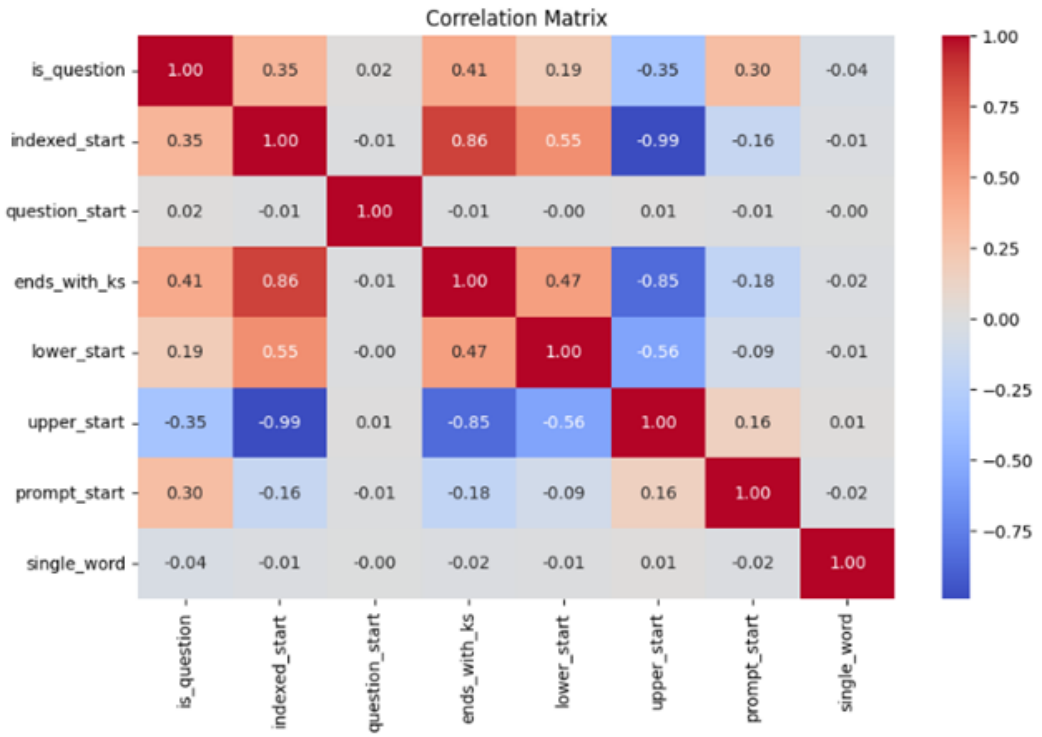


Figure 3.11: Features correlation matrix

3.4.2.6 Feature selection

Chi-squared statistical test

The Chi-Squared statistical test is a non-parametric method extensively utilized to evaluate the relationship between categorical variables. It aims to determine whether a significant association exists between the expected frequencies and the observed frequencies across one or more categories. Within the scope of feature selection for machine learning models, the Chi-Squared test is particularly invaluable for identifying features that may depend on the response variable, thereby being potentially predictive.
The Chi-Squared Test works as follows:

1. **Hypothesis Formulation**: The null hypothesis (H_0) posits no association between the feature and the target variable.The alternative hypothesis (H_1) suggests that such an association does exist,

2. **Contingency Table**: The analysis begins with constructing a con-

tingency table for each feature against the target variable. Each cell in this table represents the observed frequency (O) for the feature-category combination,

3. **Expected Frequencies**: The expected frequencies (E) for each cell are then calculated under the independence assumption. The formula for an expected frequency in a cell is given by:

$$E = \frac{\text{row total} \times \text{column total}}{\text{total observations}}$$

,

4. **Chi-Squared Statistic**: The chi-squared statistic (χ^2) is computed by summing the squared differences between the observed and expected frequencies, divided by the expected frequency for all categories:

$$\chi^2 = \sum \frac{(O - E)^2}{E}$$

This metric quantifies the deviation between the actual data and the hypothetical model of independence under the null hypothesis,

5. **Contingency Table**: The analysis begins with constructing a contingency table for each feature against the target variable. Each cell in this table represents the observed frequency (O) for the feature-category combination,

6. **P-value and Conclusion**: The calculated χ^2 statistic is then compared against the chi-squared distribution to ascertain the p-value. This p-value denotes the probability of observing such a result if the null hypothesis were true. A p-value below a predetermined significance level (commonly 0.05) leads to rejecting the null hypothesis, indicating a probable association between the feature and the response variable.

The chi-squared test is employed in feature selection to identify and retain features that have the strongest relationships with the target variable. Features that are independent of the target variable are less informative and may not contribute to the model's ability to make accurate predictions.

Chi-squared statistical test - application
The SelectKBest method, applying the (χ^2) function, filters the features, selecting those with the highest scores as demonstrated in the accompanying image. These scores are derived from the (χ^2) test, ranking each feature by its ability to predict whether a line of text is a question. This, combined with

```
# Preparing the dataset for feature selection
X = df.drop(['text', 'is_question', 'label', 'first_word', 'last_four_chars'],␣
 ↪axis=1)
y = df['is_question']

# Apply SelectKBest class to extract top features
bestfeatures = SelectKBest(score_func=chi2, k='all')
fit = bestfeatures.fit(X,y)
df_scores = pd.DataFrame(fit.scores_)
df_columns = pd.DataFrame(X.columns)

# Concatenating two dataframes for better visualization
featureScores = pd.concat([df_columns,df_scores],axis=1)
featureScores.columns = ['Feature','Score']  # Naming the dataframe columns
featureScores = featureScores.sort_values(by='Score', ascending=False)

featureScores.head()  # Display the top features based on score
```

Figure 3.12: Python code snippet - application of chi-squared

the list of features, provides a clear visualization of which characteristics hold the most significance. Features such as 'ends_with_ks', 'indexed_start', and 'prompt_start' emerge as top predictors, indicating their substantial contribution to the model's decision-making process.

This data-driven approach ensures that the final classification model is both efficient and effective, focusing on the most salient aspects of the data for question identification.

	Feature	Score
2	ends_with_ks	407.465901
0	indexed_start	310.692749
5	prompt_start	206.785000
3	lower_start	101.866475
4	upper_start	36.496747

Figure 3.13: Features chi-squared scores

3.4.3 Preparation for a YOLO8 Deep Learning CV Model Approach

These are the steps undertaken in the preparation process to ready the data for the Deep Learning (YOLO8) model.
Target class identification is done, followed by meticulous image annotation to define our training dataset. We then extract these annotations for YOLO8's use, set up the necessary computational environment, and ensure proper directory structure for efficient workflow.
The final touch is enhancing image quality, optimizing the data for the model's object detection tasks. This foundation is key for YOLO8 to accurately assess and interpret educational material.

3.4.3.1 Target Class Identification

The initial step in our YOLO8 deep learning approach was the identification of target classes within our image dataset. These classes represent distinct sections of the examination materials that are pivotal for text extraction and evaluation. The following categories were established:

1. **exam_question**: Area containing the question text,
2. **exam_answer**: Area where the student's answer is located,
3. **exam_question_section**:broader section encompassing a set of related questions,
4. **exam_question_number**: Identifier for each question in the exam,
5. **exam_question_marks**: Marks allocated to each question,
6. **exam_institution**: The educational institution's name,
7. **exam_name**: The title of the examination,
8. **exam_instructions**:Instructions for the exam,
9. **exam_unit_name**: The specific subject or unit name,
10. **exam_level_name**: Academic level or class designation,
11. **exam_examined_name**: The examinee's name,

12. **exam_examined_id**: Number associated with each answer,
13. **exam_answer_question_number**:Header of the question paper,
14. **exam_question_paper_header**:Header of the question paper,
15. **exam_question_section_marks**:Marks for each section,
16. **exam_question_section_instructions**:Instructions for sections,
17. **exam_question_section_name**: Name of the exam section,
18. **exam_question_paper_footer**: Footer information of the paper.

These classes were annotated in the images to guide the YOLO8 model in recognizing and categorizing the different sections accurately. By doing so, we ensured that the model could effectively learn from and evaluate the educational content within the provided images.

3.4.3.2 Image Annotation

For the annotation of our dataset images, the online platform CVAT AI was employed. CVAT AI is a powerful, web-based tool designed for the task of annotating visual data. The choice of this platform stems from its intuitive interface and the rich set of features it offers, which are well-suited for the detailed work required in annotating various image-based elements.

The annotation process on CVAT AI involves delineating each of the previously identified target classes within the images. Using the platform's tools, precise bounding boxes are drawn around the exam question texts, answer sections, and other defined elements. The process ensures that each relevant area is labeled with the corresponding class, such as 'exam_question_number' or 'exam_institution', to create a clear and accurate dataset for model training.

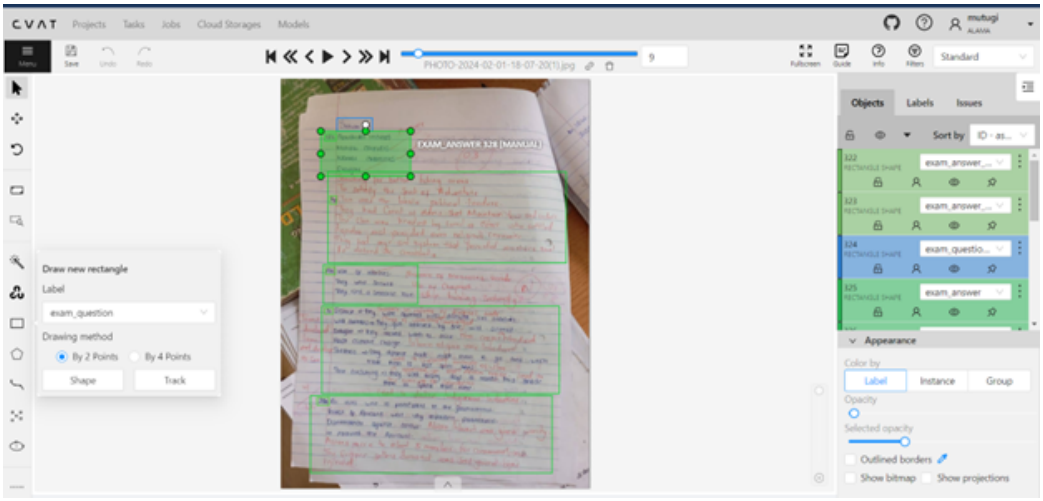


Figure 3.14: CVAT AI Annotation Interface

Upon completion, the annotated images constituted a training dataset that provided the YOLO8 model with the necessary ground truth for learning. This well-annotated dataset was essential for the subsequent training phase, ensuring the model was exposed to accurately labeled data that reflected the complexity and variability of real-world educational materials.

3.4.3.3 Environment Setup

The environment setup constituted a critical step in preparing for the application of the YOLO8 model to our image dataset. This section covers the establishment of the Python environment, which included the installation of necessary libraries and configuration of API services essential for the project's goals.

Libraries
A Python environment was created and configured with libraries essential for image processing, machine learning, and interaction with cloud services. The following imports signify the foundational tools incorporated into our Python scripts:

1. **os**:for directory and file manipulation,
2. **io**: for handling byte streams,
3. **datetime**:to timestamp our processes,
4. **json**: for parsing and outputting in JSON format,
5. **ultralytics.YOLO**: the package provided by Ultralytics for interacting with YOLO models,
6. **cv2 (OpenCV)**: a library for computer vision tasks,
7. **PIL (Python Imaging Library)**: for image file operations,
8. **pytesseract**:an OCR tool for text extraction,
9. **google.cloud.vision**:Google's Vision API for enhanced image analysis,
10. **openai**:for accessing OpenAI's APIs, including GPT services,

API Services
Two vital cloud services were set up to integrate external AI capabilities:

1. **Google Cloud Platform (GCP) for Vision API**:An account was created on GCP, enabling access to the Vision API. This service provides advanced image recognition capabilities that supplement our YOLO model's performance. Necessary APIs were enabled, and a service account was created to facilitate secure interactions with the Vision API from our Python environment. The **service_account** module was used to authenticate our environment with the provided API keys,

2. **OpenAI API Platform**: An account on the OpenAI API platform was also established, particularly to utilize the GPT service for the automatic evaluation component of the project. With the API keys obtained upon registration, our environment was configured to send requests and receive responses from the GPT service, which would provide the advanced language model's capabilities for analyzing and evaluating textual data extracted from images,

With these steps, the environment setup was completed, providing a robust and capable backend for carrying out the processes of image annotation, data extraction, and text analysis with the assistance of state-of-the-art machine learning and cloud computing services. This environment serves as the backbone for the execution of the subsequent data processing and analysis tasks in this project.

3.4.3.4 Directory Setup

The directory setup was an important process that established a structured storage system for managing the dataset and annotations used in training and evaluating the YOLO8 model. This section outlines the organization of directories and the configuration that was required to ensure smooth model operation and data handling.

Establishing Training and Evaluation Directories
A clear directory structure was essential for segregating the dataset into training and validation subsets, which allowed the model to learn effectively and generalize its capabilities to new, unseen data. The directory setup involved creating two primary folders within the project's workspace:

1. **images/train**: This directory houses the annotated images that will be used for training the model, enabling it to learn the patterns and features of the target classes,

2. **images/val**: This directory contains the images for validation, used to evaluate the model's performance and tune hyperparameters without biasing its learning process,

Saving Exported Annotations
The annotations, meticulously created using the CVAT AI tool, were exported and organized within the training and validation directories. Proper categorization and naming of these files are crucial for the subsequent training phase, ensuring that the model has access to correctly labeled and segmented data.

Configuration with config.yaml
To facilitate the model's interaction with the directory structure and data, a config.yaml file was set up. This configuration file, located at **/Users/leo/Desktop/Vault/segmentation/data/root**, specifies the paths to the training and validation directories, the number of classes, and the class names, structured as follows:

```yaml
path: /Users/leo/Desktop/Vault/segmentation/data/root
train: images/train
val: images/val

nc: 18
names:
    [
        'exam_question',
        'exam_answer',
        'exam_question_section',
        'exam_question_number',
        'exam_question_marks',
        'exam_institution',
        'exam_name',
        'exam_instructions',
        'exam_unit_name',
        'exam_level_name',
        'exam_examined_name',
        'exam_examined_id',
        'exam_answer_question_number',
        'exam_question_paper_header',
        'exam_question_section_marks',
        'exam_question_section_instructions',
        'exam_question_section_name',
        'exam_question_paper_footer',
    ]
```

Figure 3.15: Python code snippet - the environment's config.yaml file

3.4.3.5 Image Quality Enhancement

Image quality enhancement was a crucial preprocessing step that directly impacts the effectiveness of the YOLO8 model. The process involved a series of operations that refined the visual data to ensure optimal performance of the object detection system.

Orientation Correction
The first operation addressed the image orientation, which was corrected based on the EXIF data—a form of metadata that digital images contain. The code iterated through the EXIF tags, identified the orientation tag, and then rotated the image to the correct orientation if needed. This step was fundamental in maintaining the consistency of the dataset, as incorrect orientation could severely hinder the model's ability to learn and infer accurately.

Contrast Adjustment
Enhancing the image contrast was the next step, using the **ImageEnhance.Contrast** function from the PIL library. By adjusting the contrast ratio to a higher value, the distinction between the image elements became more pronounced, which was particularly beneficial for text recognition as it improved legibility.

Color Balance
Color balance was refined using the **ImageEnhance.Color** function, which could adjust the color intensity and balance of the image. A balanced color profile ensured that the model was not misled by color biases that may occur due to different lighting conditions during image capture.

Brightness Normalization
Brightness was adjusted through the **ImageEnhance.Brightness** function. By standardizing the brightness levels across images, the model was less likely to be affected by variations in illumination, allowing for a more consistent detection of features irrespective of the lighting conditions.

Sharpness Enhancement
Finally, the sharpness of the image was enhanced using the **ImageEnhance.Sharpness** function. Sharper images yield clearer edges and textures,

which assisted the model in delineating objects within the images more accurately.

```python
for img_path in images:
    image_extractions = []
    original_image = Image.open(img_path)

    # Correcting orientation based on EXIF data
    try:
        for orientation in ExifTags.TAGS.keys():
            if ExifTags.TAGS[orientation] == 'Orientation':
                break
        exif = dict(original_image._getexif().items())
        if exif[orientation] == 3:
            original_image = original_image.rotate(180, expand=True)
        elif exif[orientation] == 6:
            original_image = original_image.rotate(270, expand=True)
        elif exif[orientation] == 8:
            original_image = original_image.rotate(90, expand=True)
    except (AttributeError, KeyError, IndexError):
        # cases: image doesn't have getexif
        pass

    # Enhancing the image
    # Adjust contrast
    enhancer = ImageEnhance.Contrast(original_image)
    original_image = enhancer.enhance(1.5)  # Adjust the factor to taste

    # Adjust color balance
    enhancer = ImageEnhance.Color(original_image)
    original_image = enhancer.enhance(1.5)  # Adjust the factor to taste

    # Adjust brightness
    enhancer = ImageEnhance.Brightness(original_image)
    original_image = enhancer.enhance(1.2)  # Adjust the factor to taste

    # Adjust sharpness
    enhancer = ImageEnhance.Sharpness(original_image)
    original_image = enhancer.enhance(2.0)  # Adjust the factor to taste
```

Figure 3.16: Python code snippet - image quality enhancement process

This sequence of enhancements was designed to transform the raw images into a form that was more conducive to automated analysis. By systematically correcting orientation, and improving contrast, color balance, brightness, and sharpness, the images were rendered more suitable for the high-performance demands of YOLO8's object detection tasks. Each modified image, thus optimized, provided a strong foundation for the subsequent model phases.

3.5 Modelling

3.5.1 Introduction

Data modelling stands as a pivotal phase in data science research, where theoretical constructs are transformed into tangible predictive systems. This research utilized a stratified approach to data modelling, splitting the dataset into training and testing subsets. The research explored a spectrum of machine learning models to address different aspects of the exam-and-question from text extraction problem. The models were:

1. **Logistic Regression**:Provides baseline binary classification,
2. **Naive Bayes**: Efficient in text classification due to its simplicity,
3. **Support Vector Machine (SVM)**: Skilled at handling high-dimensional data,
4. **XGBoost**: Renowned for performance in predictive modeling with structured data,
5. **Decision Trees**: Offers clear, interpretable decision paths,
6. **Tensorflow & Keras**: Frameworks for deep learning, enabling complex pattern recognition,
7. **YOLO8 Deep Learning CV Model**: A deep learning model designed for real-time object detection in images,

3.5.2 Train-Test Split

In the construction of a reliable machine learning model, the division of data into training and testing sets is a fundamental step. For this study, the dataset predominantly adhered to an 80-20 train-test split. This partitioning strategy allocated 80% of the data for training, which allows the models to learn and understand the patterns and complexities inherent in the dataset. The remaining 20% is reserved for testing, serving as a separate, untouched dataset to evaluate the models' performance and generalize their predictive power to new, unseen data.

Such a split strikes a balance between having enough data to train on, thus enabling the model to learn effectively, while still retaining a substantial portion for unbiased evaluation. It is a standard practice that helps in mitigating overfitting, ensuring that the model's accuracy on the training data translates to unseen data.

3.5.3 Logistic Regression

Logistic regression is a statistical method for analyzing a dataset in which there are one or more independent variables that determine an outcome. The outcome is measured with a dichotomous variable (in which there are only two possible outcomes). In the context, logistic regression estimated the probability that a given input point belongs to a certain class(is_question) or not.

3.5.3.1 Formulation

1. **Linear Combination:** Logistic regression begins by calculating a linear combination of the input features x_i with a set of weights w_i, plus an intercept term b:

$$z = w_1x_1 + w_2x_2 + \cdots + w_nx_n + b \tag{3.1}$$

   Where:
   w_i: weight coefficient for feature i,
   x_i: feature i,
   b: intercept term.

2. **Logistic Function:** The result of this linear combination is then passed through a logistic function (also known as the sigmoid func-

tion), which transforms the output into a probability value between 0 and 1:

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{3.2}$$

The sigmoid function curve maps any real-valued number into a value between 0 and 1, making it suitable for probability estimation.

3. **Prediction and Threshold:** The probability obtained from the sigmoid function is then used to make a prediction. If the probability is greater than a certain threshold (commonly 0.5), the instance is classified as a question; otherwise, it is not.

$$\text{prediction} = \begin{cases} 1 & \text{if } \sigma(z) > 0.5 \\ 0 & \text{otherwise} \end{cases} \tag{3.3}$$

Where:
· 1 represents the "question" class,
· 0 represents the "not a question" class.

Logistic regression's suitability for predicting if text is a question stems from its interpretability, efficiency, solid performance on linearly separable classes, and ability to provide probability estimates. This makes it a robust choice for binary classification problems where understanding the impact of features on outcomes, quick model training, and the necessity for probabilistic interpretation are paramount.

3.5.3.2 Implementation

The dataset was partitioned into training and testing sets using selected features indicative of whether a text segment is a question. Features such as 'ends_with_ks', 'indexed_start', 'prompt_start', 'lower_start', and 'upper_start' were utilized. The target variable, 'is_question', was isolated as the dependent variable for modeling. Text features were combined with other extracted features through horizontal stacking (hstack) to create an aggregated feature set. This dataset was then divided into training (80%) and testing (20%) subsets, with a specific random state to ensure reproducibility of the split. A logistic regression model was then trained with an appropriate maximum number of iterations to ensure convergence. Post-training, the model's efficacy was evaluated on the test data, where it made predictions on unseen data. Subsequently, the predictions were assessed against the actual

labels, and a classification report was generated, detailing key performance metrics such as precision, recall, and F1-score for each class. This report was transformed into a Pandas DataFrame, enhancing the interpretability of the model's performance metrics and substantiating the logistic regression model's capacity to accurately classify text in a retrospective application:

```
# Update the code to use the selected features
# Split the dataset into training and testing sets using the selected features
X = df[['ends_with_ks', 'indexed_start', 'prompt_start', 'lower_start',␣
 ↪'upper_start']]
y = df['is_question']



# Combine the text features with other selected features
X_combined = hstack([X_text, X])

# Splitting the combined dataset
X_train, X_test, y_train, y_test = train_test_split(X_combined, y, test_size=0.
 ↪2, random_state=42)

# Train the logistic regression model
LRmodel = LogisticRegression(max_iter=1000)  # Increase max_iter if needed
LRmodel.fit(X_train, y_train)

# Evaluate the model
y_pred = LRmodel.predict(X_test)

# Generate classification report
report = classification_report(y_test, y_pred, output_dict=True)

# Convert report to pandas DataFrame
report_df = pd.DataFrame(report).transpose()
```

Figure 3.17: Python code snippet - logistic regression modelling

3.5.4 Naïve Bayes

Naive Bayes is a classification technique based on Bayes' Theorem with an assumption of independence among predictors. In a dataset where multiple features influence a binary outcome, Naive Bayes calculates the probability of each outcome and makes a prediction based on the highest probability. It assumes each feature contributes independently to the probability of the outcome, hence 'naive'. This method is particularly efficient in text classification problems where the features (words) are used to classify texts into two categories.

In the context of identifying whether a segment of text is a question, Naive Bayes determines the likelihood of the text being in one of two classes ('question' or 'not a question') based on the features extracted from the text. It uses the frequency of each word occurring in each class to compute the probability and then applies Bayes' Theorem to predict the class with the higher post-probability for the given input.

3.5.4.1 Formulation

Naive Bayes classifiers operate on Bayes' Theorem, which describes the probability of a feature, based on prior knowledge of conditions related to the feature. For a classification task like identifying if a text is a question, Naive Bayes calculates the posterior probability of each class (question or not a question) given a set of words.

The formula for Bayes' Theorem is:

1. **Bayes' Theorem:**

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)} \tag{3.4}$$

   Where:

   - $P(A|B)$ is the posterior probability of class A (such as 'is a question') given predictor B (such as a word in the text).
   - $P(B|A)$ is the likelihood, the probability of predictor given class.
   - $P(A)$ is the prior probability of class.

- $P(B)$ is the prior probability of predictor.

For text classification, the Naive Bayes classifier assumes that the effect of a word's presence (or absence) on the probability of a text being a question is independent of the presence (or absence) of any other word. It then uses the probabilities derived from the training data to estimate the probabilities for the new data and classifies each case based on the highest probability.

3.5.4.2 Implementation

The selected features such as 'ends_with_ks', 'indexed_start', 'prompt_start', 'lower_start', and 'upper_start' were collated from the dataset. These features, believed to be indicative of the likelihood that a segment of text is a question, were horizontally stacked with text features to form an integrated feature matrix.

This enriched feature set was divided into training and testing subsets, adhering to an 80-20 train-test split, with the split's reproducibility ensured by a consistent random state. The Multinomial Naive Bayes model, chosen for its applicability to discrete feature classification, was then trained with the training subset. It learned from the frequency of features' occurrence within each class to inform its predictions.

After the training process, the classifier's predictions were generated for the test set. The accuracy of these predictions was scrutinized by generating a classification report, contrasting the predicted and actual labels, and detailing the model's precision, recall, and F1-score across both classes. The report was subsequently formatted as a Pandas DataFrame, which provided a structured and interpretable presentation of the model's effectiveness.

```
# Other selected features
X_other = df[['ends_with_ks', 'indexed_start', 'prompt_start', 'lower_start',␣
 ↪'upper_start']]

# Combine the text features with other selected features
X_combined = hstack([X_text, X_other])

# Splitting the combined dataset
X_train, X_test, y_train, y_test = train_test_split(X_combined, y, test_size=0.
 ↪2, random_state=42)

# Train a Multinomial Naive Bayes model with the combined features
MNBmodel = MultinomialNB()
MNBmodel.fit(X_train, y_train)

# Evaluate the model
y_pred = MNBmodel.predict(X_test)

# Generate classification report
report = classification_report(y_test, y_pred, output_dict=True)

# Convert report to pandas DataFrame
report_df = pd.DataFrame(report).transpose()
```

Figure 3.18: Python code snippet - Naive Bayes modelling

3.5.5 Support Vector Machine (SVM)

Support Vector Machine (SVM) is a supervised machine learning algorithm mainly used for classification tasks. It finds the hyperplane that best separates different classes in a dataset by maximizing the margin between the closest points of each class. When applied to text, such as determining if a segment is a question, SVM classifies based on the placement relative to this hyperplane. The model's effectiveness is enhanced by kernel functions, allowing it to manage nonlinear data, making it a robust choice for text classification problems.

3.5.5.1 Formulation

The fundamental idea behind SVM is to find a hyperplane in an N-dimensional space (N — the number of features) that distinctly classifies the data points. To separate the two classes of data points, there are many possible hyperplanes that could be chosen, but the goal is to find the plane that has the maximum margin, i.e., the maximum distance between data points of both classes.

The formulation of SVM can be represented as:

$$\min_{w,b} \frac{1}{2}||w||^2 \tag{3.5}$$

Subject to:

$$y_i(w \cdot x_i + b) \geq 1, \quad \forall i \tag{3.6}$$

Where:

- w is the weight vector,
- b is the bias,
- y_i is the class label of the ith data point (x_i),
- The dot product $w \cdot x_i + b$ represents the decision function.

The SVM model then classifies texts by determining on which side of the hyperplane they fall, using the sign of the decision function. This capability makes SVM a powerful tool for text classification tasks like identifying questions within texts, leveraging its ability to handle complex feature spaces effectively.

3.5.5.2 Implementation

A selection of features from the dataset, namely 'ends_with_ks', 'indexed_start', 'prompt_start', 'lower_start', and 'upper_start', were first compiled. These features were believed to be significant predictors of whether a text segment constitutes a question. These selected features were then converted to a numpy array and combined with the vectorized text features, producing a comprehensive feature set for the SVM.

This combined dataset was subjected to an 80-20 split, partitioning it into distinct training and testing sets, with a specific random state ensuring consistency across experiments. An SVM model, using the SVC() class from the scikit-learn library, was trained on the training data to model the relationship between the features and the likelihood of text being a question.

After training, the SVM model's classification ability was assessed on the testing set. It made predictions which were then evaluated against the true labels to gauge the model's performance. A classification report was generated, outlining the precision, recall, and F1-score, offering a quantitative assessment of the model.

```
# Other selected features
X_other = df[['ends_with_ks', 'indexed_start', 'prompt_start', 'lower_start',␣
 ↪'upper_start']].to_numpy()

# Combine the text features with other selected features
X_combined = hstack([X_text, X_other])

# Splitting the combined dataset
X_train, X_test, y_train, y_test = train_test_split(X_combined, y, test_size=0.
 ↪2, random_state=42)

# Train an SVM model with the combined features
SVMmodel = SVC()
SVMmodel.fit(X_train, y_train)

# Evaluate the model
y_pred = SVMmodel.predict(X_test)

# Generate classification report
report = classification_report(y_test, y_pred, output_dict=True)

# Convert report to pandas DataFrame
report_df = pd.DataFrame(report).transpose()
```

Figure 3.19: Python code snippet - SVM modelling

3.5.6 XGBoost

XGBoost stands for Extreme Gradient Boosting, an advanced implementation of gradient boosting algorithms. It is known for its speed and performance, particularly in structured or tabular datasets. XGBoost works by building an ensemble of decision trees in a sequential manner, where each tree attempts to correct the errors of its predecessor. This boosting process continues until no significant improvements can be made.

In classifying whether text is a question or not, XGBoost looks at various features extracted from the text and learns to make decisions through multiple decision trees. It aggregates the results to make a final prediction. XGBoost is particularly valued for its ability to handle a variety of data types, its efficiency in large datasets, and its feature importance score, which provides insights into the predictive power of each feature.

3.5.6.1 Formulation

The core principle behind XGBoost involves sequentially adding predictors (trees), where each one corrects its predecessor.

The model is built in a stage-wise fashion as it optimizes the following objective function at each step:

$$\text{Obj}(\Theta) = \sum_i l(y_i, \hat{y}_i^{(t)}) + \sum_k \Omega(f_k) \tag{3.7}$$

Where:

- $l(y_i, \hat{y}_i^{(t)})$ is the loss function that measures the difference between the predicted $\hat{y}_i^{(t)}$ at step t and the actual y_i,
- $\Omega(f_k)$ is the regularization term that penalizes the complexity of the model, with f_k representing the kth tree.

XGBoost improves upon the standard gradient boosting method by introducing a more regularized model formalization to control over-fitting, which makes it efficient for a wide range of data science problems. In the context of classifying text as a question or not, XGBoost can leverage its robustness and scalability to handle large datasets with a multitude of features derived from

the text, effectively learning from the intricacies and nuances of language patterns.

3.5.6.2 Implementation

Crucial predictors such as 'ends_with_ks', 'indexed_start', 'prompt_start', 'lower_start', and 'upper_start' were converted to a numerical format and combined with vectorized text data. This composite of features was designed to enhance the model's ability to distinguish text as a question or not.

A stratified approach was taken to split the dataset, maintaining an 80-20 ratio for the training and testing sets, respectively. This ensured that the XGBoost model was trained on a varied and representative sample of the data, with the same distribution of classes as the full dataset.

Upon initiating the XGBoost classifier, it was configured with hyperparameters suitable for the classification task, notably opting out of the internal label encoder and specifying 'logloss' as the evaluation metric for its boosting rounds. The classifier was then fitted to the training data, learning from the patterns present in the feature set.

Post-training, the model's accuracy and predictive capabilities were evaluated on the test set. Predictions were made and compared to the actual labels, leading to the generation of a classification report.

```
# Other selected features
X_other = df[['ends_with_ks', 'indexed_start', 'prompt_start', 'lower_start',␣
 ↪'upper_start']].to_numpy()

# Combine the text features with other selected features
X_combined = hstack([X_text, X_other])

# Splitting the combined dataset
X_train, X_test, y_train, y_test = train_test_split(X_combined, y, test_size=0.
 ↪2, random_state=42)

# Train an XGBoost model with the combined features
XGBmodel = XGBClassifier(use_label_encoder=False, eval_metric='logloss')
XGBmodel.fit(X_train, y_train)

# Evaluate the model
y_pred = XGBmodel.predict(X_test)

# Generate classification report
report = classification_report(y_test, y_pred, output_dict=True)

# Convert report to pandas DataFrame
report_df = pd.DataFrame(report).transpose()
```

Figure 3.20: Python code snippet - XGBoost modelling

3.5.7 Decision Trees

Decision Trees are a non-parametric supervised learning method used for classification and regression. They model decisions and their possible consequences as a tree with branches representing decision paths and leaves representing outcomes.

In the task of classifying text as a question or not, a Decision Tree will evaluate the extracted features from the text and make binary splits at each node, leading to a decision that classifies the text. The simplicity of Decision Trees lies in their mimicry of human decision-making logic, making them transparent and easy to understand. They are particularly suitable for tasks where the decision process needs to be clear, as in the case of distinguishing questions within text data.

3.5.7.1 Formulation

The formulation of a Decision Tree involves selecting the features that result in the most homogeneous sub-nodes or leaves, where homogeneity is measured by the purity of the target variable within those nodes.

The purity of a node in a decision tree can be assessed using criteria such as Gini Impurity or Entropy, which are defined as follows:

- **Gini Impurity:** A measure of how often a randomly chosen element from the set would be incorrectly labeled if it was randomly labeled according to the distribution of labels in the subset. It is defined as:

$$\text{Gini}(D) = 1 - \sum_{i=1}^{C} p_i^2 \tag{3.8}$$

- **Entropy:** A measure of the amount of information disorder or uncertainty. It is defined as:

$$\text{Entropy}(D) = -\sum_{i=1}^{C} p_i \log_2 p_i \tag{3.9}$$

Where p_i represents the proportion of the samples that belong to class c for a particular node, and C is the number of classes.

The goal is to find the splits that reduce the impurity in the child nodes relative to the parent node. The process starts at the root of the tree and splits the data on the feature that results in the most significant reduction in impurity. This process is recursively repeated on each child node until the tree reaches a predetermined stopping criterion, such as a maximum depth or a minimum number of samples in a leaf.

In the context of classifying text as a question or not, a Decision Tree model examines the features extracted from the text—such as the presence of question marks, keyword density, or sentence structure—to decide how to split the data and classify each segment. Decision Trees are favored for their simplicity, interpretability, and the fact that they require little data preparation.

3.5.7.2 Implementation

An array of the selected features—'ends_with_ks', 'indexed_start', 'prompt_start', 'lower_start', and 'upper_start'—was prepared from the dataset. These features were deemed crucial in determining whether a text segment is a question. The array was then transformed into a dense format, conducive to the model's input requirements.

The features were merged with vectorized text data, and this combined feature set was split into training and testing sets, maintaining an 80-20 distribution to ensure enough data for training and an unbiased evaluation.

A Decision Tree model was initiated using the **DecisionTreeClassifier** from scikit-learn. The model was trained on the training set to develop a series of rules and decisions that could categorize the text effectively. After the training phase, the Decision Tree was employed to make predictions on the test set.

The accuracy and performance of the model were quantitatively assessed using a classification report, which included precision, recall, and F1-score metrics:

```
# Other selected features
X_other = df[['ends_with_ks', 'indexed_start', 'prompt_start', 'lower_start',␣
 ↪'upper_start']]

# Convert X_other to a dense array if it's not already
X_other_dense = X_other.to_numpy()

# Combine the text features with other selected features
# Convert the sparse matrix to a dense matrix
X_combined = hstack([X_text, X_other_dense]).toarray()

# Splitting the combined dataset
X_train, X_test, y_train, y_test = train_test_split(X_combined, y, test_size=0.
 ↪2, random_state=42)

# Train a Decision Tree model with the combined features
DTmodel = DecisionTreeClassifier()
DTmodel.fit(X_train, y_train)

# Evaluate the model
y_pred = DTmodel.predict(X_test)

# Generate classification report
report = classification_report(y_test, y_pred, output_dict=True)
```

Figure 3.21: Python code snippet - Decision Tree modelling

3.5.8 Tensorflow & Keras

TensorFlow and Keras are prominent libraries in the field of deep learning. TensorFlow provides a comprehensive ecosystem of tools and libraries for building and deploying machine learning models, while Keras offers a high-level neural networks API that can run on top of TensorFlow, simplifying many of the complexities associated with deep neural networks.

In the context of text classification, such as identifying whether a segment of text is a question, TensorFlow and Keras enable the creation and training of complex models that can handle the nuances of language. They support the implementation of intricate neural network architectures which can learn from vast amounts of text data and discern subtle patterns that might indicate the presence of a question. Keras, with its user-friendly interface, accelerates the development process by providing modular and composable components that facilitate the design of models tailored to the specific structure of the task at hand.

3.5.8.1 Formulation

TensorFlow and Keras are two of the most popular frameworks for deep learning, widely used for building and deploying complex models that involve extensive computational operations on data.

At its core, a deep learning model aims to approximate a function f that maps input data X to outputs Y, often through a series of transformations or layers. The architecture of a deep learning model is defined by its layers and how they are structured. This can be represented as:

$$Y = f(X; \Theta) \tag{3.10}$$

Where:

- X is the input data,
- Y is the output,
- Θ represents the parameters (weights and biases) of the model,
- f is the function modeled by the neural network.

In the context of TensorFlow & Keras, the definition and structuring of these layers are facilitated through high-level APIs that allow for the efficient and straightforward building of complex neural network architectures. Each layer in the network can perform different transformations on the data, with the overall network learning the optimal parameters Θ during the training process to accurately map X to Y.

3.5.8.2 Implementation

Key predictors from the dataset, like 'end_with_ks' and 'prompt_start', were first converted into a dense numpy array and combined with text features. The dataset was split into training and test sets, and features were standardized using StandardScaler for optimal neural network performance.

A sequential model was constructed with Keras, comprising dense layers with 'relu' activation and a final layer with 'sigmoid' activation for binary classification. A dropout layer was included to prevent overfitting. The model was compiled with the Adam optimizer and binary cross-entropy loss function, with accuracy as the chosen metric.

The model underwent training over multiple epochs, with its performance validated against the test set. The results were articulated through a classification report, providing a concise evaluation of the model's predictive accuracy. This approach demonstrated the practical application of TensorFlow and Keras in handling and analyzing text-based data for classification tasks:

```
# Other selected features
X_other = df[['ends_with_ks', 'indexed_start', 'prompt_start', 'lower_start',␣
 ↪'upper_start']].to_numpy()

# Combine the text features with other selected features
X_combined = hstack([X_text, X_other])

# Splitting the combined dataset
X_train, X_test, y_train, y_test = train_test_split(X_combined, y, test_size=0.
 ↪2, random_state=42)

# Standardize the features
scaler = StandardScaler(with_mean=False)  # Set with_mean=False for sparse␣
 ↪matrix handling
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Convert the sparse matrix to a dense matrix
X_train_dense = X_train_scaled.toarray()
X_test_dense = X_test_scaled.toarray()

# Deep Learning model
DLmodel = Sequential([
    Dense(128, activation='relu', input_shape=(X_train_dense.shape[1],)),
    Dropout(0.5),
    Dense(64, activation='relu'),
    Dropout(0.5),
    Dense(1, activation='sigmoid')
])
DLmodel.compile(optimizer='adam', loss='binary_crossentropy',␣
 ↪metrics=['accuracy'])

# Train the model
DLmodel.fit(X_train_dense, y_train, epochs=10, batch_size=32, verbose=1)

# Evaluate the model
y_pred = (DLmodel.predict(X_test_scaled) > 0.5).astype("int32")

# Classification report and confusion matrix
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy of the Deep Learning model: {accuracy * 100:.2f}%")

report = classification_report(y_test, y_pred, output_dict=True)
report_df = pd.DataFrame(report).transpose()
```

Figure 3.22: Python code snippet - Tensorflow and Keras deep learning modelling

3.5.9 YOLO8 Deep Learning CV Model

YOLO8, or 'You Only Look Once' version 8, is the latest iteration in the series of YOLO deep learning models for real-time object detection. With advancements in computer vision, YOLO8 stands out for its segmentation capabilities and is highly efficient in processing and analyzing visual data at high speeds.

For the task of segmenting educational materials, such as identifying different sections of a question paper or answer sheet in an image, YOLO8's deep learning architecture is fine-tuned to recognize and differentiate various textual and visual patterns. It achieves this through a convolutional neural network that can process the entire image in a single evaluation. This means it can detect multiple classes of objects simultaneously, making it exceptionally suitable for segmenting and classifying diverse components in educational documents. YOLO8's proficiency in handling complex image data makes it an ideal candidate for the automated analysis and segmentation of hand-written texts within educational assessments.

3.5.9.1 Formulation

YOLO8's architecture is designed to process an entire image in a single forward pass. This efficiency is achieved through a deep convolutional neural network (CNN) that divides the image into a grid and predicts bounding boxes and class probabilities for each grid cell simultaneously. The core of YOLO8's innovation lies in its ability to balance the trade-offs between precision and speed.

The YOLO model formulates the detection task as a single regression problem, directly predicting bounding box coordinates and class probabilities from full images. Each grid cell in the model is responsible for predicting bounding boxes if the center of a bounding box falls within it. The predictions made by the model include:

1. **Box coordinates:** (b_x, b_y, b_w, b_h), where (b_x, b_y) are the center coordinates of the box relative to the bounds of the grid cell, and (b_w, b_h) are the width and height of the box.

2. **Objectness score:** Reflects the confidence that a box contains an

object. This score helps in distinguishing between background and actual objects.

3. **Class probabilities:** Indicate the likelihood of each class being present within the box. This provides a distribution over the classes for each bounding box.

The network applies a sigmoid function to these outputs to normalize the predictions, ensuring they are within a suitable range for interpretation. The final step in the process involves applying non-maximum suppression (NMS) to remove overlapping bounding boxes. This is crucial for ensuring that each detected object is only counted once, thereby improving the precision of the detection.

3.5.9.2 Implementation

The configuration began with setting up the necessary directory structure and YAML files that specify the path to the training and validation datasets, along with the 18 different classes representing various segments of educational materials, such as 'exam_question' and 'exam_institution'.

The model was instantiated using the Ultralytics YOLO framework and was configured for training with the 'yolov8n.yaml' file, setting the parameters and hyperparameters necessary for learning. The training procedure was carried out over 600 epochs, indicating a comprehensive learning phase over a significant amount of iterations to ensure model robustness and accuracy.

For detection tasks, a separate script prepared a directory for storing results with time-stamped folders to organize outputs systematically. The pre-trained YOLO model weights were then loaded, and a set of images was processed to detect and segment the predefined classes within those images. Each image underwent detection, where the YOLO8 model utilized the trained weights to predict the locations and classes of the various educational elements present.

The predictions from the model included bounding boxes and class labels, which were evaluated for accuracy. This evaluation step is crucial as it informs the efficacy of the model in real-world scenarios, ensuring the model's

performance meets the expected standards for identifying and segmenting content within educational materials.

```
path: /Users/leo/Desktop/Vault/segmentation/data/root
train: images/train
val: images/val

nc: 18
names:
    [
        'exam_question',
        'exam_answer',
        'exam_question_section',
        'exam_question_number',
        'exam_question_marks',
        'exam_institution',
        'exam_name',
        'exam_instructions',
        'exam_unit_name',
        'exam_level_name',
        'exam_examined_name',
        'exam_examined_id',
        'exam_answer_question_number',
        'exam_question_paper_header',
        'exam_question_section_marks',
        'exam_question_section_instructions',
        'exam_question_section_name',
        'exam_question_paper_footer',
    ]
```

Figure 3.23: Python code snippet - YOLO8 configuration

```python
from ultralytics import YOLO

model = YOLO('yolov8n.yaml')

model.train(data='config.yaml', epochs=600)
```

Figure 3.24: Python code snippet - YOLO8 training

```python
# Create a base directory for all detections
base_dir = 'results'
timestamp = datetime.now().strftime('%Y-%m-%d_%H-%M-%S')
detection_dir = os.path.join(base_dir, timestamp)

# Ensure the base directory exists
os.makedirs(detection_dir, exist_ok=True)

#Set the model
model = YOLO('runs/detect/train10/weights/best.pt')

##Load the images
images = ['data/new/5.jpg','data/new/6.jpg','data/new/7.jpg','data/new/8.jpg','data/new/9.jpg']

##Detect sections of the image
detections = model(images)
```

Figure 3.25: Python code snippet - YOLO8 testing

3.6 Evaluation

3.6.1 Introduction

This critically examines the practical efficacy of the leading machine learning methodologies applied to the domain of educational material analysis.

In this stage, new, unseen images serve as the testing ground where the prowess of Logistic Regression, Naive Bayes, SVM, TensorFlow & Keras, and the YOLO8 Deep Learning CV Model is put to the test.

Each model's ability to accurately extract questions from these images is meticulously scrutinized. The focus is on assessing the models' performance and their transferability to real-world applications.

This assessment quantifies the accuracy and precision of each approach and gauges their robustness and reliability when confronted with the variability and unpredictability of real-life data. The outcome of this evaluation is pivotal in determining the feasibility and potential for deployment of these models in actual educational settings.

3.6.2 Logistic Regression

The model, having been trained on a dataset of labeled text, predicted labels for new, unseen data. The results demonstrated the model's capacity to identify questions with each prediction accompanied by a confidence level. Extracted segments identified as questions were collated, providing a clear view of the model's ability to parse and classify educational text. This application underlined the practicality of logistic regression in real-world educational scenarios, highlighting its potential to automate the extraction and classification of questions from academic texts.

Using the Linear Regression model to predict, then printing the results

```
text = []
all_questions = []

for line in refined.split('\n'):
    if not line.strip():  # Skip empty lines
        continue


    # Predict the label using Logistic Regression model
    label = LRmodel.predict(features_from_text(line))[0]

    # Store the text, label
    text.append({'text': line, 'label': label, 'confidence': 'N/A'})

    # Append to all_questions if it's a question
    if label == 1:
        all_questions.append(line)

# Output the results
print("\n------------OUTPUT----------------")
for t in text:
    print(t)

print("\n------------EXTRACTED QUESTIONS----------------")
for q in all_questions:
    print(q)
```

Figure 3.26: Python code snippet - Logistic Regression Evaluation

The output:

```
-----------OUTPUT---------------
{'text': 'DSA 8105: FUNDAMENTAL CONCEPTS IN a', 'label': 0, 'confidence': 'N/A'}
{'text': 'DATE: 6% September 2021', 'label': 0, 'confidence': 'N/A'}
{'text': 'Instructions', 'label': 0, 'confidence': 'N/A'}
{'text': 'Ibs This examination consists of FIVE questions.', 'label': 0,
'confidence': 'N/A'}
{'text': '2s Answer Question ONE (COMPULSORY) and any other Two ques e',


'label': 0, 'confidence': 'N/A'}
{'text': 'QUESTION ONE (20 MARKS)', 'label': 0, 'confidence': 'N/A'}
{'text': '(a) For each of the following scenarios, use a supply and demand
diagram to illustrate the effect of a given shock on the equilibrium price and
quantity u in the specified competitive market. Explain whether there is a shift
in the demand curve, the supply, or neither i) An unexpected temporary heat wave
hits the South Coast. Show the effect in the ice cream market in Mombasa. (3
Marks) ii) The government introduces a tax on ice cream which is paid by
producers. What is the effect in the ice cream market? (3 Marks) for crude oil
in country H has been estimated to be -0.06', 'label': 1, 'confidence': 'N/A'}
{'text': '(b) The price elasticity of demand', 'label': 1, 'confidence': 'N/A'}
{'text': '_0.45 in the long run. Why would the demand for crude oil be price in
the short-run and', 'label': 0, 'confidence': 'N/A'}
{'text': '(3 Marks) elastic in the long run than in the short run? Briefly
explain.', 'label': 0, 'confidence': 'N/A'}
{'text': '(c) Briefly explain why a supply shock leads to stagflation. (3 Marks)
n economic efficiency and', 'label': 1, 'confidence': 'N/A'}
{'text': '(d) In the context of financial markets, distinguish betwee
informational efficiency. (4 Marks)', 'label': 1, 'confidence': 'N/A'}
{'text': 'Page 1 of 3', 'label': 0, 'confidence': 'N/A'}

-----------EXTRACTED QUESTIONS---------------
(a) For each of the following scenarios, use a supply and demand diagram to
illustrate the effect of a given shock on the equilibrium price and quantity u
in the specified competitive market. Explain whether there is a shift in the
demand curve, the supply, or neither i) An unexpected temporary heat wave hits
the South Coast. Show the effect in the ice cream market in Mombasa. (3 Marks)
ii) The government introduces a tax on ice cream which is paid by producers.
What is the effect in the ice cream market? (3 Marks) for crude oil in country H
has been estimated to be -0.06
(b) The price elasticity of demand
(c) Briefly explain why a supply shock leads to stagflation. (3 Marks) n
economic efficiency and
(d) In the context of financial markets, distinguish betwee informational
efficiency. (4 Marks)
```

Figure 3.27: Logistic Regression Evaluation - Output

3.6.3 Naive Bayes

The model's algorithm, leveraging probability to discern patterns in text data, successfully identified segments with question-like characteristics, underpinned by a confidence metric. This metric was pivotal, serving as a threshold for classifying a segment as a question. The model's output, indicative of its predictive strength, provided an array of extracted questions, demonstrating the model's practical application:

```
text = []
all_questions = []

for line in refined.split('\n'):
    if not line.strip():  # Skip empty lines
        continue
    prediction = MNBmodel.predict(features_from_text(line))  # Predict the label
    prediction_proba = MNBmodel.predict_proba(features_from_text(line))  # Get␣
 ↪the probabilities

    # Get the index of the predicted label
    label_index = prediction[0]

    # Get the confidence for the predicted label
    confidence = prediction_proba[0][label_index]

    text.append({'text': line, 'label': label_index, 'confidence': confidence})

    # If the confidence is above a certain threshold and the label is 1,␣
 ↪consider it a question
    if confidence > 0.51 and label_index == 1:
        all_questions.append(line)


print("""

-----------OUTPUT---------------""")
for t in text:
    print(t)
print("""

------------EXTRACTED QUESTIONS---------------
""")
for q in all_questions:
    print(q)
```

Figure 3.28: Python code snippet - Naive Bayes Evaluation

The output:

```
----------OUTPUT----------
{'text': 'DSA 8105: FUNDAMENTAL CONCEPTS IN a', 'label': 1, 'confidence':
0.5924275096459716}
{'text': 'DATE: 6% September 2021', 'label': 0, 'confidence':
0.6890399773600088}
{'text': 'Instructions', 'label': 0, 'confidence': 0.5953457363517246}
{'text': 'Ibs This examination consists of FIVE questions.', 'label': 0,
'confidence': 0.800230707006505}
{'text': '2s Answer Question ONE (COMPULSORY) and any other Two ques e',
'label': 0, 'confidence': 0.7274745032787583}
{'text': 'QUESTION ONE (20 MARKS)', 'label': 1, 'confidence':
0.8162640271907494}
{'text': '(a) For each of the following scenarios, use a supply and demand
diagram to illustrate the effect of a given shock on the equilibrium price and
quantity u in the specified competitive market. Explain whether there is a shift
in the demand curve, the supply, or neither i) An unexpected temporary heat wave
hits the South Coast. Show the effect in the ice cream market in Mombasa. (3
Marks) ii) The government introduces a tax on ice cream which is paid by
producers. What is the effect in the ice cream market? (3 Marks) for crude oil
in country H has been estimated to be -0.06', 'label': 1, 'confidence':

0.9786562789335481}
{'text': '(b) The price elasticity of demand', 'label': 1, 'confidence':
0.99607081069921}
{'text': '_0.45 in the long run. Why would the demand for crude oil be price in
the short-run and', 'label': 0, 'confidence': 0.9395472156206535}
{'text': '(3 Marks) elastic in the long run than in the short run? Briefly
explain.', 'label': 1, 'confidence': 0.6816975865487692}
{'text': '(c) Briefly explain why a supply shock leads to stagflation. (3 Marks)
n economic efficiency and', 'label': 1, 'confidence': 0.9995940388597002}
{'text': '(d) In the context of financial markets, distinguish betwee
informational efficiency. (4 Marks)', 'label': 1, 'confidence':
0.9999972514734956}
{'text': 'Page 1 of 3', 'label': 0, 'confidence': 0.5124095484494093}


----------EXTRACTED QUESTIONS----------

DSA 8105: FUNDAMENTAL CONCEPTS IN a
QUESTION ONE (20 MARKS)
(a) For each of the following scenarios, use a supply and demand diagram to
illustrate the effect of a given shock on the equilibrium price and quantity u
in the specified competitive market. Explain whether there is a shift in the
demand curve, the supply, or neither i) An unexpected temporary heat wave hits
the South Coast. Show the effect in the ice cream market in Mombasa. (3 Marks)
ii) The government introduces a tax on ice cream which is paid by producers.
What is the effect in the ice cream market? (3 Marks) for crude oil in country H
has been estimated to be -0.06
(b) The price elasticity of demand
(3 Marks) elastic in the long run than in the short run? Briefly explain.
(c) Briefly explain why a supply shock leads to stagflation. (3 Marks) n
economic efficiency and
(d) In the context of financial markets, distinguish betwee informational
efficiency. (4 Marks)
```

Figure 3.29: Naive Bayes - Output

3.6.4 Support Vector Machine (SVM)

The evaluation involved parsing new image data into text and applying the SVM classifier to predict which segments constituted questions. Despite the lack of probability estimates inherent in SVM's design, a decision function was used as a confidence measure. The results showcased SVM's precision in extracting relevant text, aligning it with potential real-world educational applications. However, it is noteworthy that SVM's confidence scoring might require calibration for nuanced scenarios, highlighting the model's need for tailored thresholding to ensure accuracy in diverse contexts:

Using the SVM model to predict, then printing the results

```
text = []
all_questions = []

for line in refined.split('\n'):
    if not line.strip():  # Skip empty lines
        continue
    prediction = SVMmodel.predict(features_from_text(line))  # Predict the label

    # If you have trained your SVM with probability=True, you can uncomment the␣
↪next two lines
    # prediction_proba = SVMmodel.predict_proba([line])  # Get the probabilities
    # confidence = prediction_proba[0][1]  # Assuming 1 is the positive class

    # Otherwise, use the decision function as a proxy for confidence
    confidence_score = SVMmodel.decision_function(features_from_text(line))[0]
    # Assuming a binary classification, convert the decision function score to␣
↪confidence
    confidence = "N/A"
    text.append({'text': line, 'label': prediction[0], 'confidence':␣
↪confidence})

    # If the confidence score is above a certain threshold and the label is 1,␣
↪consider it a question
    # Adjust the threshold as necessary
    threshold = 0.0  # Define your threshold here
    if prediction[0] == 1:
        all_questions.append(line)

print("""

-----------OUTPUT----------------""")
for t in text:
    print(t)
print("""

-----------EXTRACTED QUESTIONS----------------
""")
for q in all_questions:
    print(q)
```

Figure 3.30: Python code snippet - SVM Evaluation

The output:

```
-----------OUTPUT---------------
{'text': 'DSA 8105: FUNDAMENTAL CONCEPTS IN a', 'label': 0, 'confidence': 'N/A'}
{'text': 'DATE: 6% September 2021', 'label': 0, 'confidence': 'N/A'}
{'text': 'Instructions', 'label': 0, 'confidence': 'N/A'}
{'text': 'Ibs This examination consists of FIVE questions.', 'label': 0,
'confidence': 'N/A'}
{'text': '2s Answer Question ONE (COMPULSORY) and any other Two ques e',
'label': 1, 'confidence': 'N/A'}
{'text': 'QUESTION ONE (20 MARKS)', 'label': 0, 'confidence': 'N/A'}
{'text': '(a) For each of the following scenarios, use a supply and demand
diagram to illustrate the effect of a given shock on the equilibrium price and
quantity u in the specified competitive market. Explain whether there is a shift
in the demand curve, the supply, or neither i) An unexpected temporary heat wave
hits the South Coast. Show the effect in the ice cream market in Mombasa. (3
Marks) ii) The government introduces a tax on ice cream which is paid by
producers. What is the effect in the ice cream market? (3 Marks) for crude oil
in country H has been estimated to be -0.06', 'label': 1, 'confidence': 'N/A'}

{'text': '(b) The price elasticity of demand', 'label': 1, 'confidence': 'N/A'}
{'text': '_0.45 in the long run. Why would the demand for crude oil be price in
the short-run and', 'label': 0, 'confidence': 'N/A'}
{'text': '(3 Marks) elastic in the long run than in the short run? Briefly
explain.', 'label': 0, 'confidence': 'N/A'}
{'text': '(c) Briefly explain why a supply shock leads to stagflation. (3 Marks)
n economic efficiency and', 'label': 1, 'confidence': 'N/A'}
{'text': '(d) In the context of financial markets, distinguish betwee
informational efficiency. (4 Marks)', 'label': 1, 'confidence': 'N/A'}
{'text': 'Page 1 of 3', 'label': 0, 'confidence': 'N/A'}


-----------EXTRACTED QUESTIONS---------------

2s Answer Question ONE (COMPULSORY) and any other Two ques e
(a) For each of the following scenarios, use a supply and demand diagram to
illustrate the effect of a given shock on the equilibrium price and quantity u
in the specified competitive market. Explain whether there is a shift in the
demand curve, the supply, or neither i) An unexpected temporary heat wave hits
the South Coast. Show the effect in the ice cream market in Mombasa. (3 Marks)
ii) The government introduces a tax on ice cream which is paid by producers.
What is the effect in the ice cream market? (3 Marks) for crude oil in country H
has been estimated to be -0.06
(b) The price elasticity of demand
(c) Briefly explain why a supply shock leads to stagflation. (3 Marks) n
economic efficiency and
(d) In the context of financial markets, distinguish betwee informational
efficiency. (4 Marks)
```

Figure 3.31: SVM Evaluation - Output

3.6.5 Tensorflow & Keras

The TensorFlow and Keras deep learning model underwent a critical evaluation phase, where it was tasked with processing previously unseen images and extracting question text. The model, employing a dense neural network architecture, predicted with high confidence which segments of text were questions. The output was a curated list of questions, revealing the model's adeptness at understanding and classifying educational content. This testing phase was vital in assessing the model's accuracy and its potential for real-world applicability in educational settings, demonstrating its strengths in pattern recognition and text classification:

Using the Deep Learning (Keras + TF) model to predict, then printing the results

```
text = []
all_questions = []

for line in refined.split('\n'):
    if not line.strip():  # Skip empty lines
        continue

    # Get the combined features as a dense array
    combined_features_dense = features_from_text(line).toarray()

    # Predict the label using the DL model
    prediction_proba = DLmodel.predict(combined_features_dense)
    prediction = (prediction_proba > 0.5).astype("int32")[0][0]  # Convert␣
↪probabilities to binary label

    # Get the probability of the positive class
    confidence = prediction_proba[0][0]
```

```
    # Append the result to the text list
    text.append({'text': line, 'label': prediction, 'confidence': confidence})

    # Add to all_questions if it's predicted as a question with high confidence
    if prediction == 1 and confidence > 0.5:
        all_questions.append(line)

# Output the results
print("\n-----------OUTPUT----------------")
for t in text:
    print(t)

print("\n-----------EXTRACTED QUESTIONS----------------")
for q in all_questions:
    print(q)
```

Figure 3.32: Python code snippet - Tensorflow & Keras Evaluation

The output:

```
-----------OUTPUT----------------
{'text': 'DSA 8105: FUNDAMENTAL CONCEPTS IN a', 'label': 0, 'confidence':
0.44610518}
{'text': 'DATE: 6% September 2021', 'label': 0, 'confidence': 0.44242573}
{'text': 'Instructions', 'label': 0, 'confidence': 0.46451402}
{'text': 'Ibs This examination consists of FIVE questions.', 'label': 0,
'confidence': 0.45010582}
{'text': '2s Answer Question ONE (COMPULSORY) and any other Two ques e',
'label': 0, 'confidence': 0.48759815}
{'text': 'QUESTION ONE (20 MARKS)', 'label': 0, 'confidence': 0.49411634}
{'text': '(a) For each of the following scenarios, use a supply and demand
diagram to illustrate the effect of a given shock on the equilibrium price and
quantity u in the specified competitive market. Explain whether there is a shift
in the demand curve, the supply, or neither i) An unexpected temporary heat wave
hits the South Coast. Show the effect in the ice cream market in Mombasa. (3
Marks) ii) The government introduces a tax on ice cream which is paid by

producers. What is the effect in the ice cream market? (3 Marks) for crude oil
in country H has been estimated to be -0.06', 'label': 1, 'confidence':
0.6098751}
{'text': '(b) The price elasticity of demand', 'label': 1, 'confidence':
0.6596327}
{'text': '_0.45 in the long run. Why would the demand for crude oil be price in
the short-run and', 'label': 0, 'confidence': 0.45082688}
{'text': '(3 Marks) elastic in the long run than in the short run? Briefly
explain.', 'label': 0, 'confidence': 0.48077786}
{'text': '(c) Briefly explain why a supply shock leads to stagflation. (3 Marks)
n economic efficiency and', 'label': 1, 'confidence': 0.6154952}
{'text': '(d) In the context of financial markets, distinguish betwee
informational efficiency. (4 Marks)', 'label': 1, 'confidence': 0.8849813}
{'text': 'Page 1 of 3', 'label': 0, 'confidence': 0.4325577}

-----------EXTRACTED QUESTIONS----------------
(a) For each of the following scenarios, use a supply and demand diagram to
illustrate the effect of a given shock on the equilibrium price and quantity u
in the specified competitive market. Explain whether there is a shift in the
demand curve, the supply, or neither i) An unexpected temporary heat wave hits
the South Coast. Show the effect in the ice cream market in Mombasa. (3 Marks)
ii) The government introduces a tax on ice cream which is paid by producers.
What is the effect in the ice cream market? (3 Marks) for crude oil in country H
has been estimated to be -0.06
(b) The price elasticity of demand
(c) Briefly explain why a supply shock leads to stagflation. (3 Marks) n
economic efficiency and
(d) In the context of financial markets, distinguish betwee informational
efficiency. (4 Marks)
```

Figure 3.33: Tensorflow & Keras Evaluation - Output

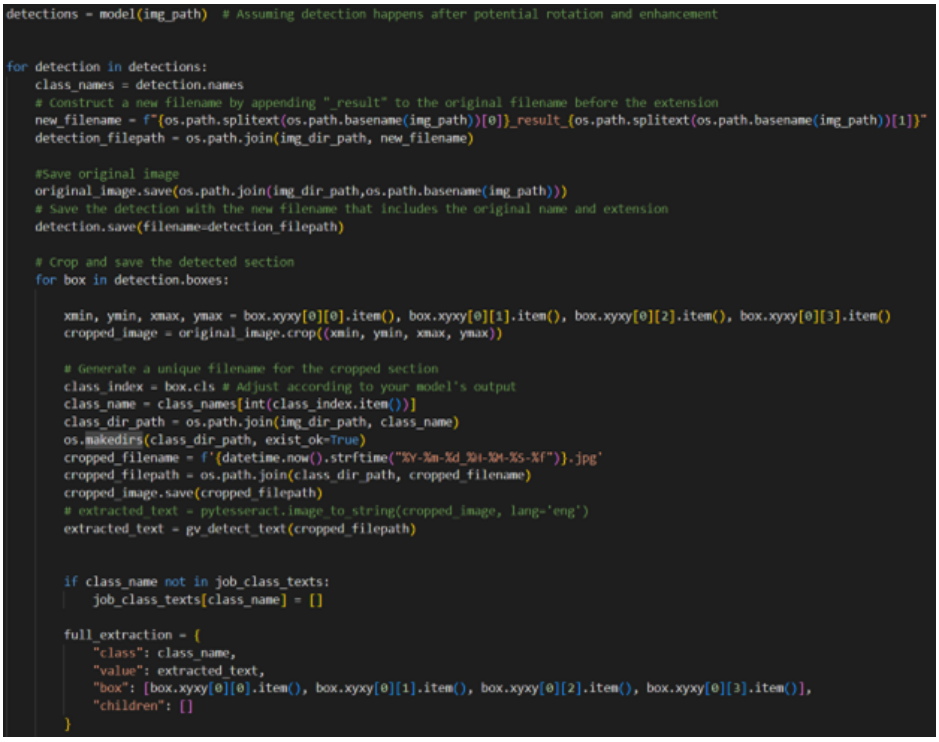
3.6.6 YOLO8 Deep Learning CV Model

The YOLO8 Deep Learning CV model analyzed an image of an examination paper. The model's objective was to identify and extract the image sections.

Utilizing YOLO8's advanced object detection capabilities, it localized and classified various text blocks accurately, as evidenced by bounding boxes around the question numbers.

This evaluation highlighted YOLO8's precision in detecting distinct text elements, confirming its viability for digitizing, and structuring educational content in a real-world educational setting.

The model's successful identification and extraction process demonstrates its potential to streamline the digital transformation of educational assessment methods.



```
detections = model(img_path)  # Assuming detection happens after potential rotation and enhancement


for detection in detections:
    class_names = detection.names
    # Construct a new filename by appending "_result" to the original filename before the extension
    new_filename = f"{os.path.splitext(os.path.basename(img_path))[0]}_result_{os.path.splitext(os.path.basename(img_path))[1]}"
    detection_filepath = os.path.join(img_dir_path, new_filename)

    #Save original image
    original_image.save(os.path.join(img_dir_path,os.path.basename(img_path)))
    # Save the detection with the new filename that includes the original name and extension
    detection.save(filename=detection_filepath)

    # Crop and save the detected section
    for box in detection.boxes:

        xmin, ymin, xmax, ymax = box.xyxy[0][0].item(), box.xyxy[0][1].item(), box.xyxy[0][2].item(), box.xyxy[0][3].item()
        cropped_image = original_image.crop((xmin, ymin, xmax, ymax))

        # Generate a unique filename for the cropped section
        class_index = box.cls # Adjust according to your model's output
        class_name = class_names[int(class_index.item())]
        class_dir_path = os.path.join(img_dir_path, class_name)
        os.makedirs(class_dir_path, exist_ok=True)
        cropped_filename = f'{datetime.now().strftime("%Y-%m-%d_%H-%M-%S-%f")}.jpg'
        cropped_filepath = os.path.join(class_dir_path, cropped_filename)
        cropped_image.save(cropped_filepath)
        # extracted_text = pytesseract.image_to_string(cropped_image, lang='eng')
        extracted_text = gv_detect_text(cropped_filepath)


        if class_name not in job_class_texts:
            job_class_texts[class_name] = []

        full_extraction = {
            "class": class_name,
            "value": extracted_text,
            "box": [box.xyxy[0][0].item(), box.xyxy[0][1].item(), box.xyxy[0][2].item(), box.xyxy[0][3].item()],
            "children": []
        }
```

Figure 3.34: Python code snippet - YOLO8 Evaluation

The output:

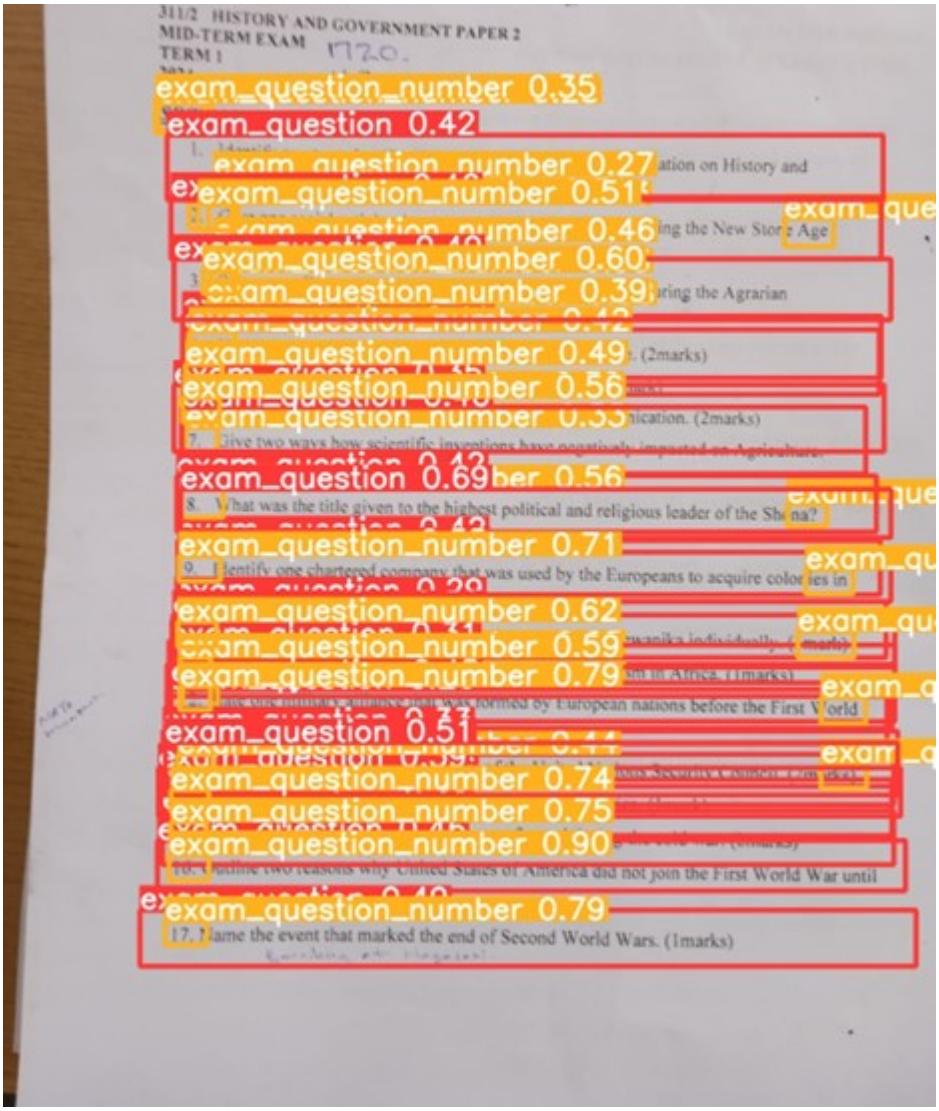


Figure 3.35: An example of a YOLO8-labelled exam image

Chapter 4

Deployment and Interface Development

4.0.1 Introduction

This chapter discusses the systematic process of developing a web interface that acts as a conduit between the end-user and the powerful capabilities of the YOLO8 model. It outlines the workflow from the initial user interaction to the final display of results, detailing each component's design and integration within the broader system architecture.
The narrative then progresses to the specifics of the deployment environment: from the architectural choices in database design, ensuring robust and efficient data management, to the aesthetics and usability considerations in interface design. The selection of a suitable web framework is considered pivotal and is discussed in relation to its ability to support dynamic interactions and seamless model integration. The chapter also explores the nuts and bolts of database and interface development, culminating in the deployment of the entire system on a cloud platform, poised to deliver scalable, accessible, and reliable service to users.

4.0.2 General Flow

The general flow begins with user registration, which sets up a personalized experience on the platform. Following registration, users proceed to log in, gaining access to the core functionalities of the system. The heart of the interface lies in the Question-and-Answer Extraction feature, which utilizes

the YOLO8 model to accurately identify and extract text from uploaded images. Post extraction, users have the option to modify the detected questions and answers, ensuring that the inputs to the subsequent automated evaluation are accurate and reflective of the user's intent. This end-to-end flow is crafted to ensure a seamless user experience, from initial engagement to receiving actionable insights from the automated evaluation.

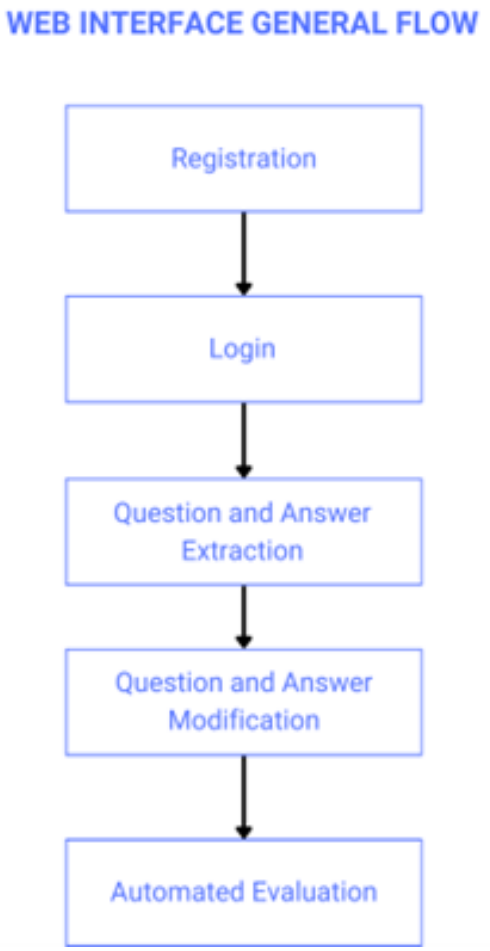


Figure 4.1: Web Interface General Flow

4.0.3 Interface Design

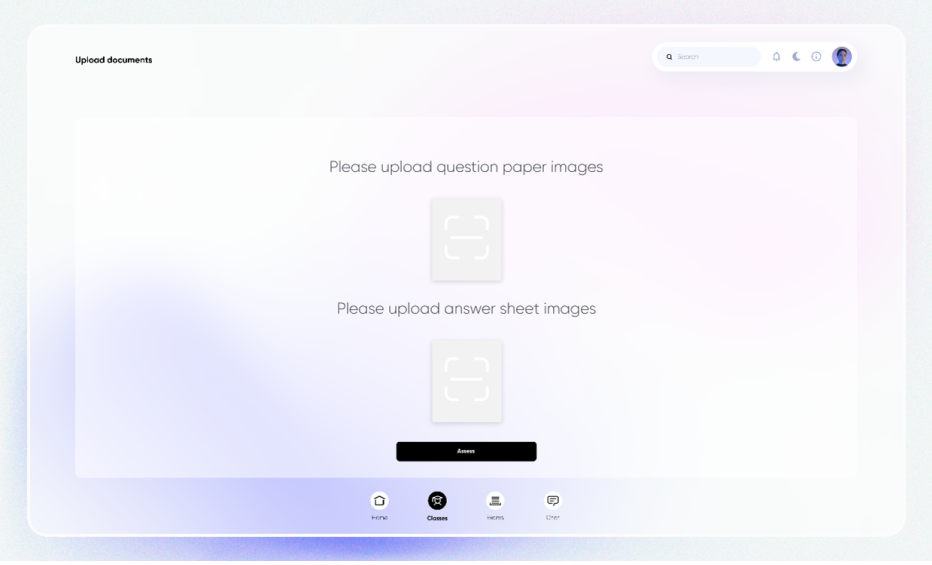


Figure 4.2: Exam Papers Uploading Screen

4.0.4 Database Design

4.0.5 Web Framework Selection

Python Django is selected as the web framework of choice. Django is chosen for its pragmatic design and its 'batteries-included' approach, providing a comprehensive suite of tools and features out of the box. This robust framework supports rapid development, enabling a clean and pragmatic design of the web interface. Its built-in components for handling common web development tasks—such as authentication, URL routing, and database schema migrations—allow for a focus on writing the app without needing to reinvent the wheel.

Furthermore, Django's emphasis on automation, along with its strong documentation and large supportive community, significantly streamlines the development process. The framework's compatibility with the YOLO8 model facilitates seamless integration, allowing for efficient back-end management

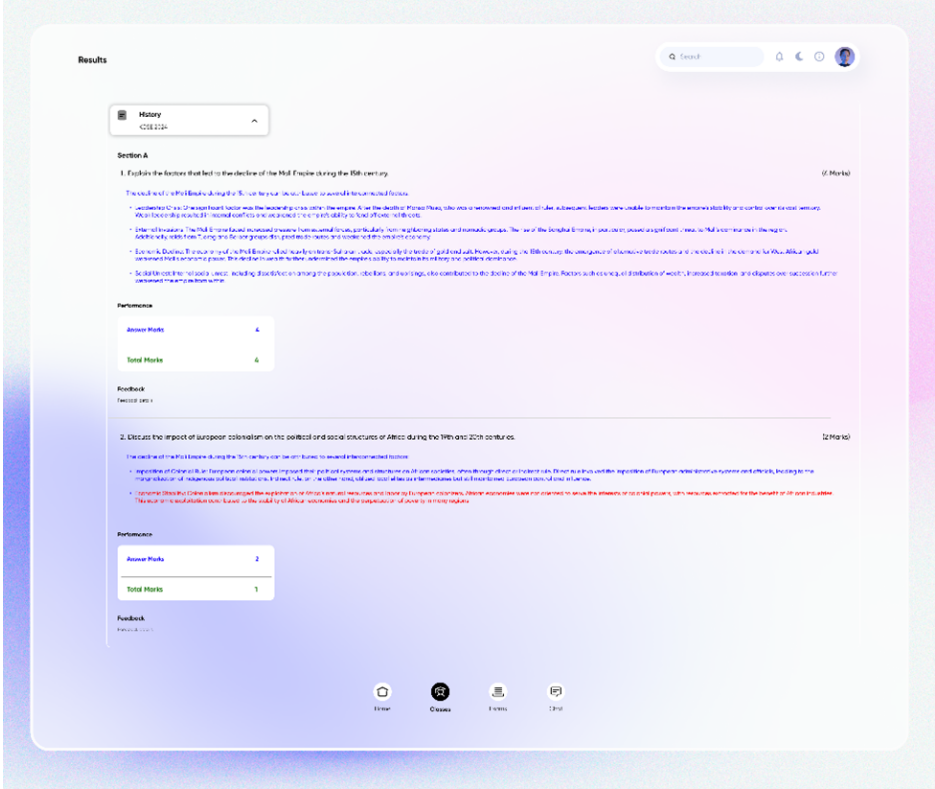


Figure 4.3: Exam Results Screen

and front-end presentation. Django's security features and scalability also align with the need for a reliable and secure user experience, particularly when handling sensitive educational data. By leveraging Django, the interface is poised to deliver a robust, secure, and user-friendly platform that effectively bridges the gap between the YOLO8 model's capabilities and the end-users' needs.

4.0.6 Database and Backend Development

4.0.7 Frontend and Logic Development

4.0.8 Cloud Hosting

Chapter 5

Implementation and Testing

5.1 Introduction

5.2 Authentication

5.3 Question and Answer Extraction

5.4 Automated Assessment

5.5 Future Enhancements

Post-research, the possibilities for enhancing the automated evaluation system are substantial and varied, aiming to augment its capabilities and broaden its applicability. One of the primary areas of future work will be the development of personalized feedback mechanisms. This feature can provide students with individualized insights into their performance, highlighting areas of strength and opportunities for improvement. Such personalized feedback can greatly aid in the learning process, offering tailored guidance that aligns with each student's unique learning trajectory.

Another significant enhancement could be the adaptation of the system for online exams. As educational institutions increasingly embrace digital platforms for assessments, the system can be evolved to seamlessly integrate with various online examination formats. This would involve ensuring compatibil-

ity with different types of digital answer submissions, such as typed responses or digital drawings, expanding the system's utility beyond traditional paper-based exams.

The ability to evaluate arithmetic and diagram-heavy documents is another prospective enhancement. This would entail developing sophisticated algorithms capable of interpreting and assessing numerical data and graphical representations, which are integral components of subjects like mathematics and science.

Chapter 6

Results and Discussion

6.1 Introduction

A thorough analysis is presented on the performance of the explored machine learning models—Logistic Regression, Naive Bayes, SVM, TensorFlow & Keras, and YOLO8 Deep Learning CV Model.

Employing a comprehensive approach, this chapter delves into the evaluation metrics, discussing the strengths and weaknesses of each model in the context of educational content analysis.

By integrating visual tools such as heatmaps and confusion matrices, the discussion offers a nuanced understanding of the models' classification capabilities. These instruments give the precision and recall of the predictions and provide insights into the models' behavior across various classes, shedding light on the practical implications of their deployment in real-world scenarios.

6.2 Model Performance

6.2.1 Logistic Regression

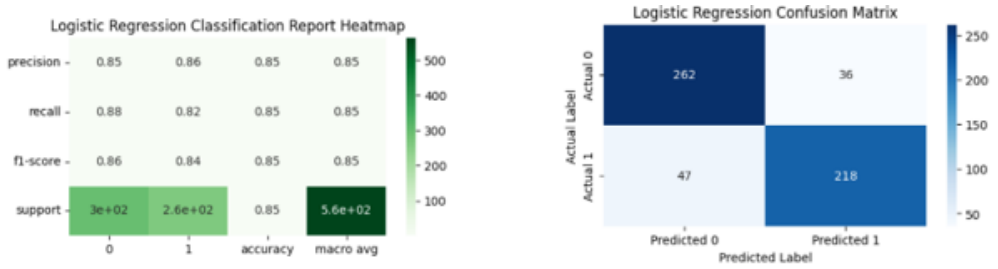


Figure 6.1: Performance Report - Logistic Regression

The heatmap illustrates high precision (0.85 for class 0, 0.86 for class 1), indicating a strong likelihood that the model's predictions are correct when it identifies a line of text as a question or not a question. Similarly, the recall is robust (0.88 for class 0, 0.82 for class 1), suggesting that the model is quite capable of identifying most relevant instances for both classes. The F1-scores, which balance precision and recall, are also commendable, particularly for class 0 (0.86), showing that the model has a harmonious balance of precision and recall. The support values indicate the number of instances for each class that were actually present in the dataset, providing context for the other metrics.

The confusion matrix further cements the model's performance by showing the number of true positive and true negative predictions. With 262 true negatives and 218 true positives, the model demonstrates a high rate of correct classifications for both classes. The relatively low number of false positives and false negatives (36 and 47, respectively) reinforces the model's accuracy. However, these incorrect predictions signal areas where the model may still improve. The discrepancies may arise from various factors, such as nuances in the way questions are phrased or presented in the text, which could provide valuable insights for refining the model further.

6.2.2 Naïve Bayes

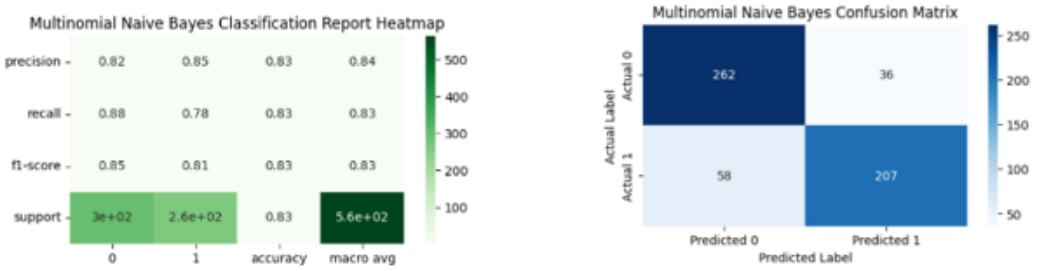


Figure 6.2: Performance Report - Naïve Bayes

The classification report heatmap reveals that precision is fairly strong, standing at 0.82 for class 0 and 0.85 for class 1. This indicates a high probability that the model's predictions are correct when it classifies a line of text as a question or not a question. The recall rates are 0.88 for class 0 and slightly lower at 0.78 for class 1, suggesting that while the model is good at identifying all relevant instances of non-questions, it's slightly less so for questions. The F1-score, which harmonizes the precision and recall, is solid across both classes, peaking at 0.85 for class 0, which shows a balanced performance for the negative class (non-questions).

The confusion matrix for the Multinomial Naive Bayes model complements the classification report by showing concrete prediction counts. It shows a substantial number of true negatives (262) and true positives (207), indicative of the model's strong performance in correctly classifying the classes. However, the model does make a notable number of false negative predictions (58), which suggests that while the model is reliable, there is room for improvement in identifying questions. The number of false positives (36) is relatively low, suggesting that when the model predicts a segment as a question, it is likely correct. These metrics together suggest that while the Multinomial Naive Bayes model is quite effective, particularly with non-questions, its ability to identify questions could benefit from further refinement, possibly by considering contextual dependencies between the features.

6.2.3 Support Vector Machine (SVM)

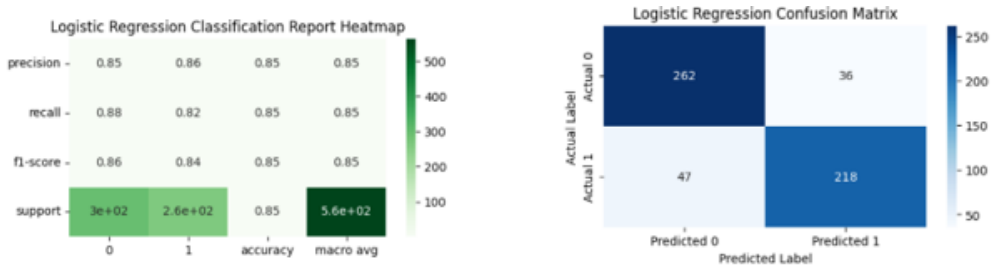


Figure 6.3: Performance Report - Support Vector Machine (SVM)

Precision values are commendable across both classes, with 0.85 for class 0 and 0.86 for class 1, suggesting the model is quite accurate in its predictions. Recall values are also strong for class 0 at 0.88, though slightly lower for class 1 at 0.82, indicating the model is more adept at correctly identifying non-questions than questions. The F1-scores, which are the harmonic means of precision and recall, show a balanced performance with 0.86 for class 0 and 0.84 for class 1, pointing towards a stable and reliable classification process for both classes.

From the confusion matrix, we see that the SVM model correctly identified a significant number of true negatives (262) and true positives (218), indicating a high success rate in classification. The false positives (36) and false negatives (47) are relatively low, which further supports the model's effectiveness. These figures suggest that SVM is proficient in classifying text segments, albeit with a margin for improvement, particularly in reducing the number of false negatives, which could further enhance its application in practical scenarios such as automated question extraction.

6.2.4 XGBoost

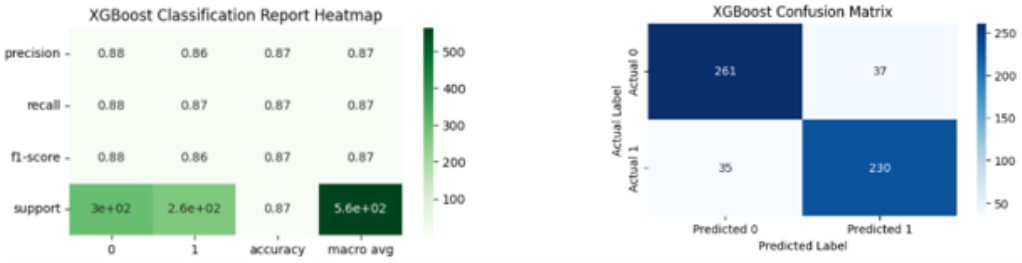


Figure 6.4: Performance Report - XGBoost

The precision metrics stand at 0.88 for class 0 and 0.86 for class 1, signifying that the model has a high rate of correct predictions when identifying lines of text as questions or not. The recall is equally impressive for both classes, at 0.88 for class 0 and 0.87 for class 1, indicating the model's strong capability to detect the majority of positive instances for both categories.

The F1-scores are consistently high at 0.88 for class 0 and 0.86 for class 1, reflecting a well-balanced precision and recall and suggesting that XGBoost is adept at dealing with both false positives and false negatives effectively. The support numbers show a substantial amount of data points for each class, reinforcing the reliability of these metrics.

In the confusion matrix, XGBoost demonstrated a substantial number of true negatives (261) and true positives (230), confirming its robustness in accurate classifications. The relatively small number of false negatives (35) and false positives (37) underscores the model's efficiency, although it also points to potential areas for refinement. These figures illustrate XGBoost's strong performance in text classification, confirming its utility in applications where identifying text-based questions accurately is crucial.

6.2.5 Decision Trees

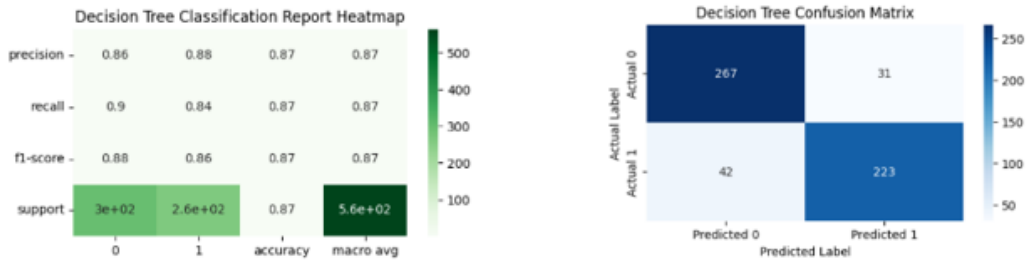


Figure 6.5: Performance Report - Decision Trees

The Decision Tree model's classification report and confusion matrix suggest a high level of performance. Precision is strong, with the model showing an 0.86 rate of correct predictions for class 0 and an even higher rate of 0.88 for class 1. This implies the model is reliably distinguishing between questions and non-questions. The recall rates are also notable, especially for class 0 at 0.9, indicating the model's proficiency in capturing actual question instances. The F1-score, which combines precision and recall into a single metric, stands at 0.88 for class 0 and 0.86 for class 1, highlighting the model's balanced performance.

The confusion matrix reinforces these findings with a high count of true negatives (267) and true positives (223), evidencing the model's effective classification capabilities. However, there are 42 false negatives and 31 false positives, which, while relatively low, suggest that there is room for improvement in the model's predictive accuracy.

6.2.6 Tensorflow & Keras

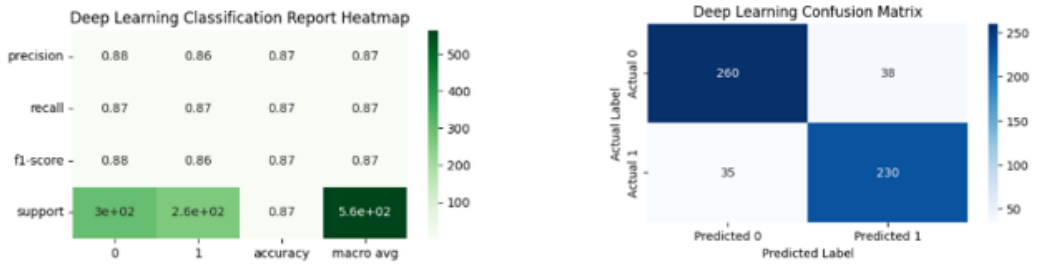


Figure 6.6: Performance Report - Tensorflow & Keras

The classification report heatmap and confusion matrix for the Deep Learning model utilizing TensorFlow & Keras show promising results. Precision is robust for both classes, scoring 0.88 for class 0 and 0.86 for class 1, indicating the model's strong predictive accuracy for identifying lines of text as questions or not. The recall scores are equally impressive, with 0.87 for class 0 and a slightly lower 0.87 for class 1, suggesting that the model is capable of correctly identifying most instances of both classes.

The F1-scores, which provide a balance between precision and recall, maintain a high level of 0.88 for class 0 and 0.86 for class 1, denoting a balanced classification performance. The support figures indicate a robust dataset with a substantial count for both classes, lending significant weight to these performance metrics.

Looking at the confusion matrix, the model has correctly classified a high number of true negatives (260) and true positives (230), which speaks to its effective classification capabilities. However, the model has also produced a few false negatives (35) and false positives (38), pinpointing areas where the model's performance could be further enhanced. Despite these misclassifications, the overall high performance of the TensorFlow & Keras model in this context illustrates its solid potential for real-world applications, particularly in the nuanced field of text classification within educational content.

6.2.7 YOLO8 Deep Learning CV Model

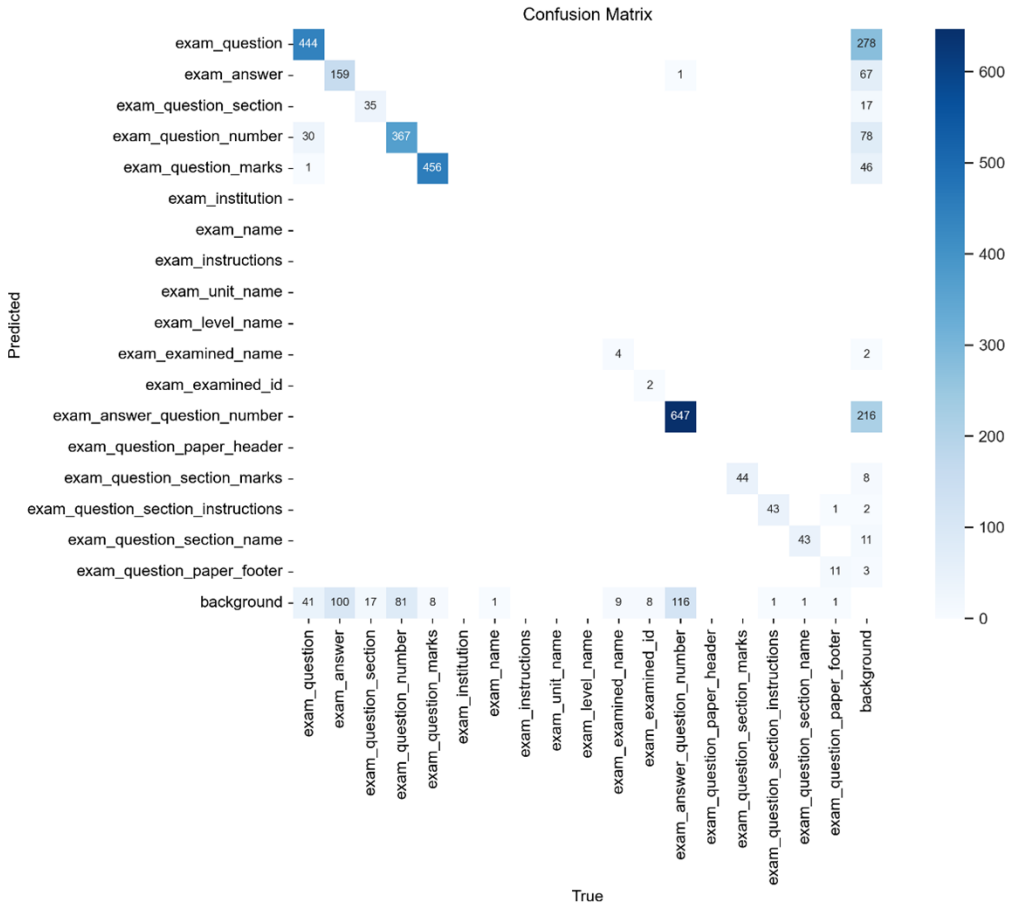


Figure 6.7: Confusion Matrix - YOLO8 Deep Learning CV Model

The YOLO8 Deep Learning CV Model's confusion matrix shows strong performance in accurately classifying different elements of educational material. The 'exam_question' class is predominantly correctly identified with 444 true positives. Notably, 'exam_answer' predictions are exceptionally accurate, with 159 instances correctly labeled, indicating the model's precise classification capabilities. For 'exam_question_number', the model correctly

identified 367 instances, signifying a reliable recognition ability for numerical identifiers within text.

Instances of confusion are minimal, suggesting that the model is proficient at distinguishing between various classes, such as questions, answers, and numerical identifiers. The minor misclassifications that do occur—evident in the small numbers away from the matrix's diagonal—highlight specific areas where the model may benefit from additional training data or refinement. However, these are relatively few, underscoring the model's overall robustness in classifying educational content for automated processing.

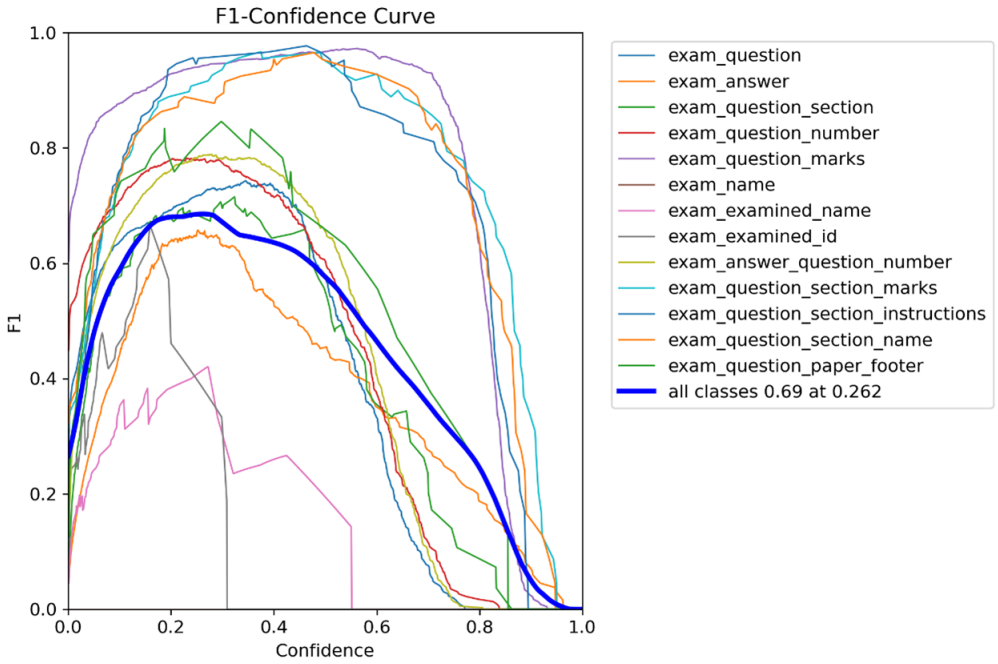


Figure 6.8: F1-Confidence curve - YOLO8 Deep Learning CV Model

The F1-Confidence curve depicts how the F1-score, balancing precision and

recall, varies with different confidence thresholds for classifying text segments. Each class, like 'exam_question', has a peak F1-score at an optimal confidence level, beyond which the score declines. The collective F1-score for all classes reaches its peak at 0.69 when the confidence threshold is 0.262, indicating the best trade-off between accuracy and coverage for the model's predictions across all categories.

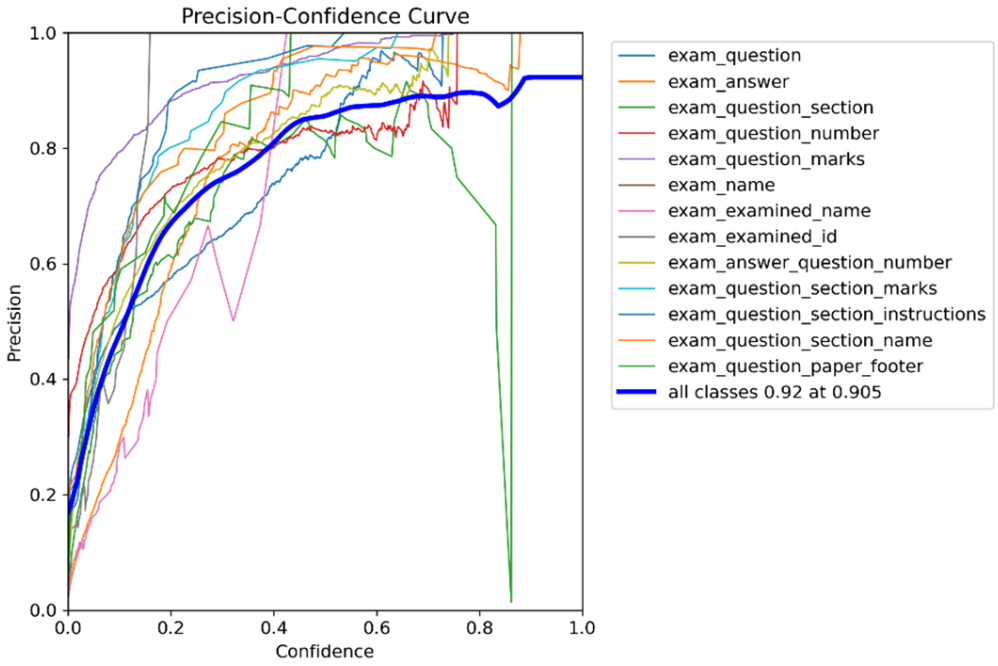


Figure 6.9: Precision-Confidence curve - YOLO8 Deep Learning CV Model

The Precision-Confidence curve illustrates the precision of the YOLO8 Deep Learning CV Model across various confidence levels for different classes. As the confidence threshold increases, the precision climbs, indicating a more accurate prediction of true positives within each category. The curve peaks for each class show the highest precision that the model can achieve at the

optimal confidence level.

The collective precision for all classes hits a maximum of 0.92 at a confidence threshold of 0.905, suggesting that at this point, the model's predictions are most accurate overall. Beyond this peak, a higher confidence threshold may not necessarily result in better precision, indicating the optimal balance between confidence in the model's predictions and its actual precision rate.

6.3 Model Selection

In the assessment of traditional classifiers, the XGBoost model emerges as the frontrunner, with precision rates of 0.88 for non-questions (class 0) and 0.86 for questions (class 1). It consistently demonstrates high recall for both classes (0.87) and F1-scores of 0.88 for non-questions and 0.86 for questions, underscoring its adeptness at accurate classification.

Trailing just behind, the SVM model shows precision of 0.85 for non-questions and 0.86 for questions, paired with recall rates of 0.88 for non-questions and a slightly lesser 0.82 for questions. Its F1-scores, at 0.86 for non-questions and 0.84 for questions, suggest it as a strong alternative, albeit with marginally less balanced performance than XGBoost.

The Decision Tree model excels in identifying non-questions with a recall of 0.9 but exhibits a modest decline in detecting questions, with a recall of 0.84. Its overall F1-scores are competitive, at 0.86 for non-questions and 0.84 for questions, marking it as a capable, if not optimal, classifier.

The TensorFlow & Keras deep learning model, while presenting impressive metrics—precision of 0.88 for non-questions and 0.86 for questions and recall and F1-scores both at 0.87—may not be the most resource-efficient choice given the comparable performance of less resource-intensive models.

Logistic Regression offers balanced performance across classes, with a precision of 0.85 for both non-questions and questions. However, it shines in the recall for non-questions (0.88) while being less effective for questions (0.82), resulting in F1-scores of 0.86 for non-questions and 0.84 for questions, indicating a solid yet outmatched contender to XGBoost.

The Multinomial Naive Bayes model, with the lowest precision (0.82 for non-questions, 0.85 for questions) and recall (0.88 for non-questions, 0.78 for questions), illustrates its constraints, particularly when dealing with the contextual nuances inherent in question detection tasks.

The YOLO8 Deep Learning CV Model showcases exceptional precision in detecting distinct textual classes within images, outstripping the traditional models with its spatial analysis capabilities. It demonstrates an aggregate

precision across all classes of 0.92 at a high confidence threshold of 0.905, coupled with a robust overall recall of 0.86 even at the zero-confidence threshold. This superior precision, along with the model's capacity for precise boundary box delineation, makes it exceptionally suitable for complex text-in-image classification tasks found in educational materials.

Due to its accurate classification and spatial recognition, the YOLO8 model stands out as the preferred choice. Its integration of boundary box detection not only amplifies its precision but also greatly aids in deployment, providing clear demarcations of classified segments for further processing. With these advanced capabilities, YOLO8 exceeds in both analytical performance and deployment readiness, marking it as the optimum model for real-world educational assessment applications.

6.4 Conclusion

The current state of research in applying machine learning (ML) to educational assessments demonstrates significant progress in automating and enhancing evaluation methods. Innovations span from processing handwritten answers using optical character recognition (OCR) and natural language processing (NLP), to employing semantic networks for grading short answers, and leveraging large language models (LLMs) for generating educational content and assessments. These advancements highlight ML's capacity to make educational assessments more efficient, objective, and personalized. Nonetheless, challenges remain in ensuring ethical application, data privacy, and maintaining human oversight. This evolving landscape suggests a promising future for educational practices, with AI and machine learning at the forefront of driving improvements in adaptability, inclusivity, and effectiveness.

In assessing the traditional machine learning algorithms for processing handwritten exam images, XGBoost leads with notable precision and F1-scores, making it highly effective for text classification. SVM follows closely, providing strong performance, albeit with a slightly less balanced output. The Decision Tree model excels in non-question identification but is less effective for questions, while TensorFlow & Keras demonstrate high metrics but at a greater resource cost. Logistic Regression and Multinomial Naive Bayes offer solid yet less competitive performances.

Remarkably, the YOLO8 Deep Learning CV Model stands out for text-in-image tasks, showcasing exceptional precision and recall, especially suited for complex handwritten content. This comparison highlights the specialized capabilities of YOLO8 for image-based classification, indicating its superiority in tasks requiring detailed analysis of handwritten exams, alongside the effective, albeit more general, performance of models like XGBoost and SVM.

The most applicable and optimal machine learning approach identified for processing handwritten exam images is the YOLO8 Deep Learning CV Model. This advanced model stands out due to its precision and ability to handle the variability inherent in handwritten text. It has been configured to recognize and extract textual content effectively by using bounding boxes, allowing for an accurate analysis of the exams. The YOLO8 model's architecture has been fine-tuned to process images with varying handwriting styles and

qualities, making it robust against common issues like poor image resolution or skewed text. This optimal configuration ensures that educational assessments can be conducted more efficiently, with greater accuracy and reduced time investment compared to traditional methods.

A user-friendly interface for an ML-based educational assessment tool can be developed by leveraging the Django web framework to create a streamlined web application. Django's modular nature allows for rapid development of the application's core components, such as user registration, login, and result display. The interface flow, designed for intuitive navigation, can seamlessly connect with the YOLO8 Model for processing handwritten exam images. Upon image upload and text extraction, the interface can then utilize the OpenAI API to evaluate the answers. This integration facilitates real-time analysis and feedback, offering a comprehensive solution that is both time-efficient in its development and effortless for end-users to operate.

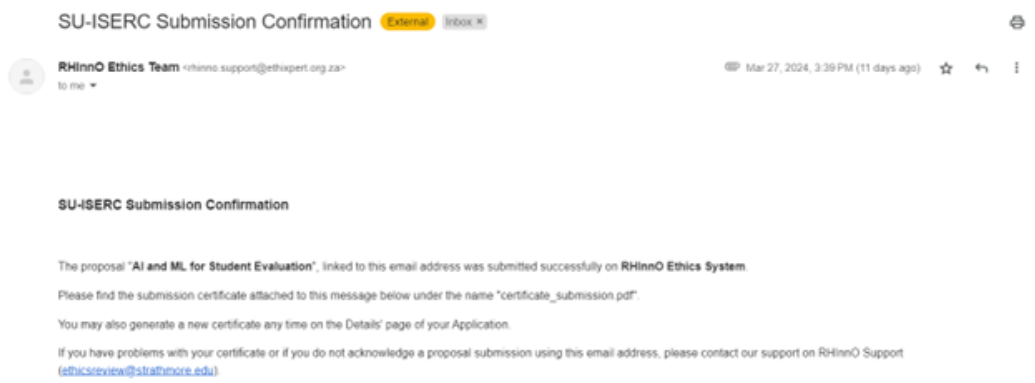
The research encapsulated in this dissertation paves the way for future work that could explore the integration of ML techniques across various educational frameworks, potentially influencing policy and pedagogical strategies globally. The advancements and insights garnered here serve as a testament to the transformative impact of machine learning on educational assessments, heralding a new era of data-driven educational practices.

As this study concludes, it also serves as a launching pad for future innovations—a testament to the untapped potential of ML in education, promising a landscape where continuous improvement is not just envisioned but realized.

Appendix A

Appendix

A.1 SU-ISERC Ethical Review Submission - Confirmation and Certificate


SU-ISERC Submission Confirmation External Inbox ×

RHInnO Ethics Team <rhinno.support@ethixpert.org.za>
to me

Mar 27, 2024, 3:39 PM (11 days ago)

SU-ISERC Submission Confirmation

The proposal "**AI and ML for Student Evaluation**", linked to this email address was submitted successfully on **RHInnO Ethics System**.

Please find the submission certificate attached to this message below under the name "certificate_submission.pdf".

You may also generate a new certificate any time on the Details' page of your Application.

If you have problems with your certificate or if you do not acknowledge a proposal submission using this email address, please contact our support on RHInnO Support (ethicsreview@strathmore.edu)

Figure A.1: SU-ISERC Ethical Review Submission - Confirmation

Completion of Online Research Ethics Review Submission

You have successfully submitted your application for ethics review "AI and ML for Student Evaluation"

Certificate awarded to: Mr Mutugi, Brian

Reference number: SU-ISERC2174/24

Date and Time: 2024-03-27 12:39:55

Figure A.2: SU-ISERC Ethical Review Submission - Certificate

A.2 Turnitin Plagiarism Report - 7% Similarity Score

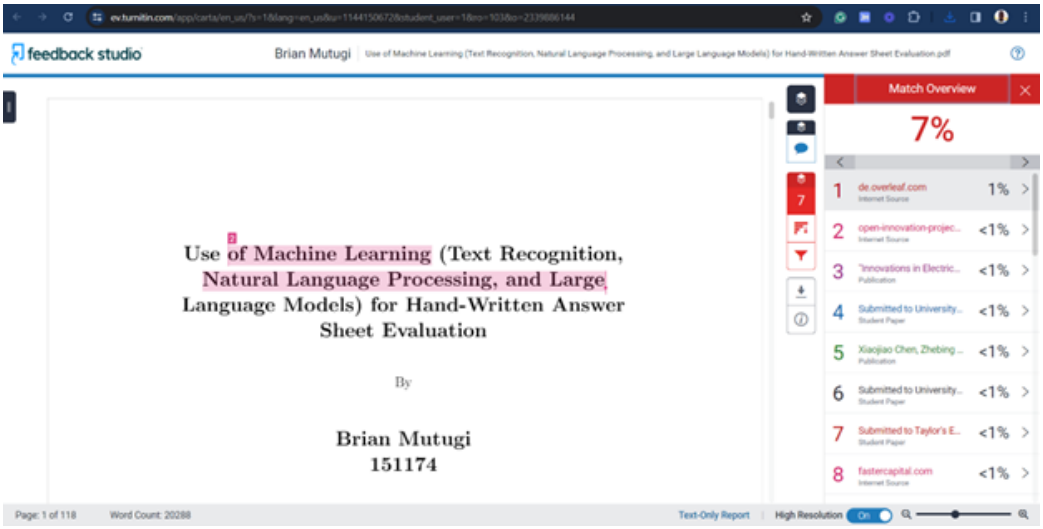


Figure A.3: Turnitin Plagiarism Report - 7% Similarity Score

References

Baker, R. S., & Smith, L. B. (2019). Technology trends in education: The impact of machine learning. *Journal of Educational Technology*, *20*(3), 34–47.

Caines, A., Benedetto, L., Taslimipoor, S., Davis, C., Gao, Y., Andersen, Ø., Yuan, Z., Elliott, M., Moore, R., Bryant, C., Rei, M., Yannakoudakis, H., Mullooly, A., Nicholls, D., & Buttery, P. (2023). On the application of large language models for language teaching and assessment technology [Empowering Education with LLMs – the Next-Gen Interface and Content Generation]. *arXiv preprint arXiv:2307.08393*. https://ar5iv.org/abs/2307.08393

Das, I., Sharma, B., Rautaray, S., & Pandey, M. (2019). An examination system automation using natural language processing, 1064–1069. https://doi.org/10.1109/ICCES45898.2019.9002048

Davis, A., & Patel, V. (2020). Machine learning in education: Applications and trends. *Artificial Intelligence Review*, *45*(2), 205–219.

Extance, A. (2023). Chatgpt has entered the classroom: How llms could transform education [This article explores the integration of Large Language Models like ChatGPT in education, highlighting their potential as creative and trustworthy learning partners despite existing challenges.]. *Nature*. https://www.nature.com/articles/d41586-023-03507-3

Gan, W., Qi, Z., Wu, J., & Lin, J. C.-W. (2023). Large language models in education: Vision and opportunities. *arXiv preprint arXiv:2311.13160*. https://arxiv.org/abs/2311.13160

Gautam, S., & Srinivasulu, A. (2023). Revolutionizing education: Harnessing machine learning and deep learning for digital examination transformation. *Journal of Research in Engineering and Computer Sciences*, *1*(5), 35–46.

Hameed, N., & Sadiq, A. (2023). Automatic short answer grading system based on semantic networks and support vector machine. *Iraqi Journal of Science*, 6025–6040. https://doi.org/10.24996/ijs.2023.64.11.44

Hsiao, Y.-P., Klijn, N., & Chiu, M.-S. (2023). Developing a framework to re-design writing assignment assessment for the era of large language models. *Learning: Research and Practice*, *9*(2), 148–158. https://doi.org/10.1080/23735082.2023.2257234

Johnson, P. (2021). Challenges of manual test evaluation: A review. *Journal of Educational Methods*, *37*(4), 211–225.

Jordan, M. I., & Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. *Science*, *349*(6245), 255–260.

Kucak, D., Juricic, V., & Dambic, G. (2018). Machine learning in education—a survey of current research trends. In B. Katalinic (Ed.), *Proceedings of the 29th daaam international symposium* (pp. 0406–0410). DAAAM International.

Lee, J., & Kim, D. (2018). Machine learning applications in education: Current trends and future prospects. *Educational Technology Research and Development*, *66*(5), 1317–1332.

Lenæs, A., & Myksvoll, K. (2023, May). Machine learning for prediction of grades [Executive Master of Management].

Nguyen, T., & Brown, S. (2022). Advances in text recognition and natural language processing. *Computer Science Review*, *50*, 100–115.

Perwej, Y., & Chaturvedi, A. (2011). Machine recognition of hand written characters using neural networks. *International Journal of Computer Applications*, *14*(2).

Phan, H., Hasegawa, S., & Gu, W. (2023). Implementation of automated feedback system for japanese essays in intermediate education. *IIAI Letters on Informatics and Interdisciplinary Research*, *3*, LIIR057. https://doi.org/10.52731/liir.v003.057

Robinson, M. (2020). Evaluating handwritten assessments: Challenges and opportunities. *Assessment in Education: Principles, Policy & Practice*, *27*(2), 190–207.

Sakhapara, A., Pawade, D., Chaudhari, B., Gada, R., Mishra, A., & Bhanushali, S. (2019, January). Subjective answer grader system based on machine learning: Proceedings of icscsp 2018, volume 2. https://doi.org/10.1007/978-981-13-3393-4_36

Sanuvala, G., & Fatima, S. (2021). A study of automated evaluation of student's examination paper using machine learning techniques, 1049–1054. https://doi.org/10.1109/ICCCIS51004.2021.9397227

Thompson, R., & Lee, C. (2018). Hand-written tests in the digital age: Relevance and implications. *Journal of Educational Research*, *111*(2), 213–228.

Urbina Najera, A. B., & Calleja Mora, J. d. l. (2017). Brief review of educational applications using data mining and machine learning. *Revista Electrónica de Investigación Educativa*, *19*(4), 84–96. https://doi.org/10.24320/redie.2017.19.4.1305

Vanathi, B., Ramya, R., Sashrutha, M., & Swetha, B. (2023). Smart paper evaluation system [Department of Computer Science and Engineering, Head of the Department - CSE. Project under the guidance of Dr. B. Vanathi, M.E., Ph.D. Team members include R. Ramya (142219104095), M. Sashrutha (142219104110), and B. Swetha (142219104134). The project aims to automate the evaluation process of exam papers using machine learning techniques to reduce time consumption and error rates associated with manual correction.].

Williams, J. (2019). Towards improved educational assessments: The role of technology. *Educational Technology & Society*, *22*(3), 58–68.