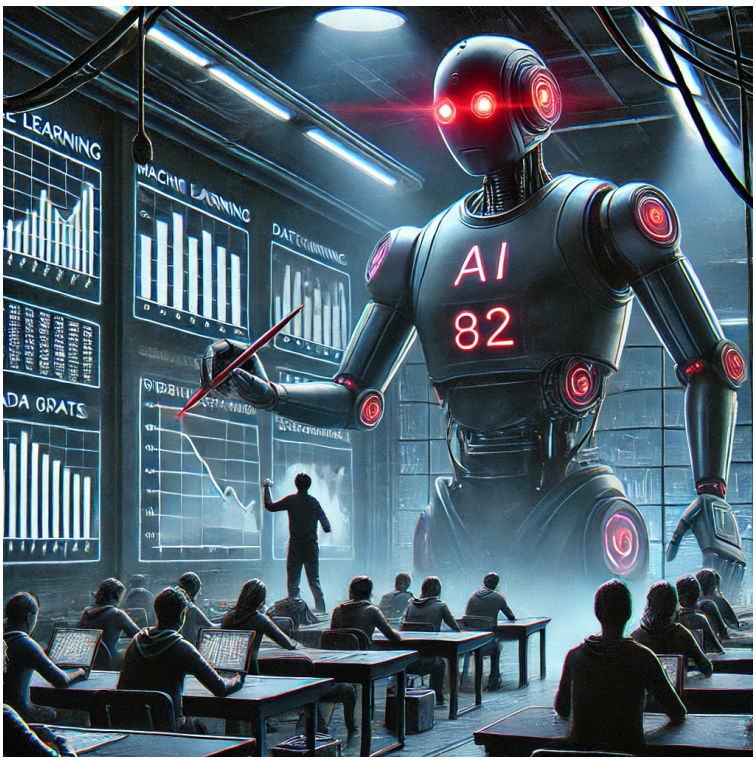


Automatic Scoring of Plots in Higher Data Science Education

Exploring the Potential and Challenges of Machine Learning in Scoring Student-generated Plots

Master's thesis in Computer Science and Engineering

Albin Larsson, Alex Wolf

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2025

Master's thesis 2025

Automatic Scoring of Plots in Higher Data Science Education

Exploring the Potential and Challenges of Machine Learning in Scoring Student-generated Plots

Albin Larsson, Alex Wolf

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2025

Automatic Scoring of Plots in Higher Data Science Education
Exploring the Potential and Challenges of Machine Learning in Scoring Student-generated Plots
Albin Larsson, Alex Wolf

Supervisor: Matti Karppa, Computer Science and Engineering
Examiner: Peter Damaschke, Computer Science and Engineering

Master's Thesis 2025
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: AI-generated image of how a dystopian version of automated scoring of plots could look.

Typeset in LaTeX
Gothenburg, Sweden 2025

Automatic Scoring of Plots in Higher Data Science Education
Exploring the Potential and Challenges of Machine Learning in Scoring Student-generated Plots
Albin Larsson, Alex Wolf
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

This thesis explores the potential and challenges of using machine learning to score student-submitted plots in data science education, an area largely untouched in existing literature. As the number of students in data science courses grows, the demand for teacher assistant hours increases. This thesis investigates how machine learning could alleviate the work of teaching assistants by automating the scoring of student plots, thus saving time and costs.

A pipeline was developed to convert each plot into a machine learning compatible format using student submissions and reference solutions written in Python. Based on this format, relevant plot features were extracted and compared for similarity against reference solutions. The extracted features are based on the pre-defined scoring rubrics. These features were then used to train four machine learning models to identify the most accurate model for this task.

The results show the potential of the random forest regressor model and the broader feasibility of saving time and costs with fewer teaching assistant hours and automatic scoring of plots. But challenges are evident from the human-labelled dataset. Challenges such as unconscious bias, erroneous scores and inconsistent interpretations of rubrics prove difficult for the model to predict. The findings highlight a trade-off between staying consistent with human scoring and focusing on objective, automated evaluation, raising important questions for future development.

Keywords: Charts, Data Transformation, Education, Ensemble, Machine Learning, Pipeline, Plots, Similarity

Acknowledgements

We would like to thank our supervisor, Matti Karppa, for his patience, encouragement and unwavering support during this project. The countless outside-of-business-hours emails have sped up the process of solving problems and resolved a good number of headaches (and caused some as well).

Albin Larsson Alex Wolf, Gothenburg, 2025-05-20

Contents

Acronyms

BERT	Bidirectional Encoder Representations from Transformers
CLIP	Contrastive Language-Image Pretraining
CNN	Convolutional Neural Network
LLM	Large Language Model
LPIPS	Learned Perceptual Image Patch Similarity
ML	Machine Learning
NLP	Natural Language Processing
PlotQA	Plot Question and Answer
RF	Random Forest
RFR	Random Forest Regression
SBERT	Sentence-BERT
SeSS	Semantic Similarity Score
SMOTE	Synthetic Minority Over-sampling Technique
SSIM	Structural Similarity Index Measure
SVM	Support Vector Machine
SVR	Support Vector Regression
TA	Teaching Assistant
UFS	Universal Structured Format
UKÄ	Universitetskanslerämbetet
ViT	Vision Transformers
VLM	Vision Language Model

List of Figures

List of Tables

1

Introduction

In introductory data science courses, assignments cover various tasks, including discussions, statistical analyses, data cleaning, and data visualisation. As courses grow in size, automatic scoring methods are increasingly used to provide scalable and consistent evaluation. Scoring here refers to assigning a discrete number that reflects how well a student's submission meets specified criteria. Some tasks, particularly those with structured outputs, can be evaluated automatically using various techniques. One common approach is unit testing, where students' outputs or code structures are compared against a reference solution provided by instructors. This reduces the need for manual evaluation by teachers and teaching assistants (TAs), especially for tasks with well-defined, deterministic outputs [1]. Unit tests can also be designed to enforce structural requirements and coding conventions by rejecting solutions based on predefined rules. However, tasks within assignments that require outputs akin to plots and graphs (referred to as *plots* onwards) are predominantly scored manually. Manual scoring disrupts workflows, consumes more time, and leaves less time for providing constructive feedback [2]. A machine learning tool is particularly effective in situations where monotonic and clearly stated rules apply. When scoring plots, TAs assess visualisations using specified rubrics. Such a tool becomes particularly valuable as enrollment in introductory data science courses continues to rise [3].

Manual scoring uses rubrics to maintain neutrality and ensure consistent scoring. These rubrics consist of certain traits or criteria that student submissions must satisfy [4]. For of plot scoring, rubrics typically address visual elements and overall correctness. Such criteria include the presence of axis labels, titles, legends, and whether the visualisation adequately answers the stated question. However, certain rubrics can be vague and require human interpretation, for example, rubrics like *The axes of figures are labelled correctly, including units*[1]. All the rubrics put together determine the score of the plot task, which is a subpart of a larger assignment.

The course targeted by this project, scoring primarily emphasises visual elements rather than textual details such as axis labels and legends [3]. Thus, the development will prioritise machine learning (ML) models and evaluation metrics that focus mainly on visual features, although support for evaluating textual components will be included as a secondary feature. The presence of textual elements will be evaluated using rule-based and shallow semantic analysis.

[1] Rubrics from the course that the supervisor of this work teaches.

1.1 Aim

This thesis developed machine learning (ML) models to automatically score student-produced plots in higher data science education. The project investigated how ML can help reduce the need for additional teaching assistant hours as student enrollment increases.

A system was implemented that transformed students' plot-generating code into machine learning compatible features based on predefined rubrics. These features captured both visual aspects, such as similarity to reference plots, and textual components, including axis labels, legends, and titles, with an emphasis on visual evaluation.

The trained models were evaluated against human assigned scores to assess their accuracy and alignment with the rubric based grading criteria. The results demonstrated that ML based scoring can replicate many aspects of human grading, suggesting that such models could meaningfully reduce grading workload and improve scalability in large data science courses.

1.2 Limitations

The tool was developed using data and guidance from the project supervisor, who teaches a relevant data science course. As a result, it was specifically tailored to the course's needs, which introduced several limitations.

First, the system only supports plots created with Matplotlib, as this is the library used in the targeted course. While support for other libraries such as Plotly or Seaborn is technically possible, it was not included in this project.

Second, the data processing workflow was designed around the plot types found in the available dataset. Since student submissions only included a limited set of plot types, generalizing the system to handle additional types proved impractical. The work therefore focused on supporting line plots, histograms, bar plots, and scatterplots.

Third, textual elements such as axis labels and titles were compared to reference solutions using a pre-trained language model, SBERT [5]. The suitability of SBERT for this task was not formally evaluated; it served only as a practical method for measuring textual similarity.

Fourth, this work focuses only on textual and visual rubric elements. While some questions include detailed, question-specific requirements, these are not addressed here and are considered beyond the scope of this project.

Finally, open-ended tasks that required students to interpret or write about their plots were excluded. These tasks emphasize written or semantic interpretation rather than visual correctness, which falls outside the project's focus on evaluating the graphical elements of the plots.

2

Background

2.1 Terminology

This thesis references literature that often uses different terms to describe similar concepts. For consistency, the term *plot* is used when referring to data displayed visually, typically on a Cartesian plane or along categorical axes. Although terms such as *plot*, *graph*, and *chart* appear interchangeably in the literature, the term *plot* is used throughout this work.

The term *figure* refers to the canvas or container that can hold one or more plots. A figure can contain multiple individual plots (also known as subplots) and additional visual elements, such as titles, legends, and axis labels. Therefore, a *plot* specifically refers to the visual representation of data, while a *figure* refers to the overall visual layout containing and organising these plots.

Additionally, the literature uses terms such as *grading*, *scoring*, and *assessment* interchangeably, although they have distinct meanings. Grading typically refers to assigning a final mark or grade to a piece of school work, often summarising overall performance [6]. Scoring refers to the number of points awarded for correct answers in a test or task [7]. Assessment refers more broadly to the evaluation of learning outcomes and student performance over time [8]. Since this thesis aims to predict student scores on plot tasks within assignments, the term scoring will be used throughout.

2.2 Related Work

Automatic scoring of plots intersects with several areas: automated assessment frameworks, the development and usage of rubrics, techniques for extracting and analysing plots, chart understanding using neural networks, and commercial auto-grading tools. This section provides an overview of relevant work in these areas, highlighting existing approaches, their limitations, and how they influence the methodology.

The broader context of automatic grading is examined first, including efforts in grade prediction and code evaluation using ML. This provides a foundation for understanding where automated plot grading fits within the wider scope of educational tools. Afterwards, rubrics are presented, which serve as structured criteria for scoring

and are essential for developing explainable and interpretable models. Next, prior work on automatic grading tools for plots is explored, highlighting both the potential and limitations of current approaches. Building on this, datasets from the field of chart mining and chart understanding are reviewed, which inform how visual plot elements can be detected, parsed, and interpreted. Similarity metrics ranging from pixel-based to semantic comparisons are also covered, focusing on their applicability in evaluating how closely a student's plot matches a reference. The following subsections discuss embedding techniques and neural network architectures, including convolutional neural networks (CNNs) and large language models (LLMs), which underpin many of the systems used for visual and textual analysis. Lastly, we survey commercial autograding tools and position our work within this space, identifying the need for more robust and semantically aware solutions for evaluating student-generated plots.

2.2.1 Automatic Grading

There are several papers regarding the prediction of student course grades in higher education using various ML models [9]–[11]. Predictive models are commonly used to detect at-risk students early, allowing timely academic interventions before students fail key courses. These models can also estimate dropouts and recommend suitable courses to students [9]. However, dataset imbalance is a key challenge in grade prediction, as noted by Bujang et al. [9]. Since student grade distributions often follow a normal distribution, ML models struggle to accurately predict underrepresented categories such as perfect scores or failing grades. To address this issue, Bujang et al. [9] employed SMOTE[1]. SMOTE generates synthetic samples by interpolating between an existing minority class data point and one of its randomly selected k-nearest neighbours in feature space. Their main finding is that Random Forest (RF) outperforms other classifiers, including Naïve Bayes, k-Nearest Neighbours (kNN), Support Vector Machines (SVM), Logistic Regression, and J48. Before applying SMOTE, RF achieved an accuracy of 57.1% in predicting the "not passed" class. After applying SMOTE, this accuracy increased significantly to 98.5% [9]. This finding aligns with Badal and Sungkur [10], who also identified RF as the most accurate model for grade prediction. However, the effectiveness of SMOTE in Bujang et al. [9] is limited by data quality. If the training set contains noisy or mislabeled cases, SMOTE may replicate that noise, expand class overlap, and push the model toward overfitting [12]. The issue of data imbalance is also present in the context of automatic scoring in this thesis. While SMOTE could potentially be useful for balancing the dataset, its effectiveness will depend on the quality of the data used for training. If the dataset contains errors or inconsistencies, SMOTE could amplify these issues, resulting in equal or worse model performance.

Automatic grading has also been explored extensively in programming courses. Messer et al. [1] summarised the field of instant feedback on student-produced code, identifying four key aspects of code quality that can be graded: correctness, maintainability, readability, and documentation. Each of these elements contributes to writing good code, and providing instant feedback helps students improve their

[1]Synthetic Minority Over-sampling Technique

skills more efficiently. The study categorised feedback approaches into static analysis, dynamic analysis, and ML-based methods. Static analysis evaluates code without executing it, focusing on properties such as syntax, structure, and maintainability. In contrast, dynamic analysis assesses the behaviour of code during execution, typically through unit tests and runtime checks. Static and dynamic analysis are commonly used for evaluating unit tests, semantic meaning, and code structure. However, Messer et al. [1] highlighted that ML approaches have primarily been applied to assessing correctness, with significantly less attention given to maintainability, readability, and documentation.

An automatic tool to assess scientific line plots in physics and chemistry was created by Vitale et al. [13]. Their study aimed to help students with varying degrees of plotting skills to improve their ability to create plots. By implementing a model that identified common pitfalls of plotting, such as misinterpretation of the question, feedback could be given immediately. The main drawback of their study was that the tool only supported line plots. This thesis aims to support more types of plot, making it more practical in a production environment.

Another automatic scoring tool for plots has been developed by Yang [14] that has been used in higher education. The tool converts the underlying plot objects in code to a single unified format, such that comparisons between student and instructor plots can be made based on rubrics. The unified format created by Yang [14] allowed for easier comparisons of the structure and contents of the code, such as what methods are used and the size of the plot. Furthermore, the format also allows for the inclusion of data points, enabling further analysis of the plotted data. The motivation for Yang's [14] tool is to help students get better at creating plots by providing feedback. However, the effectiveness of the tool in supporting the scoring or grading work of teaching assistants (TAs) remains somewhat unclear, which is the focus of this work.

2.2.2 Rubrics

The scoring of plots requires clearly defined rubrics to evaluate their quality and effectiveness. Rubrics are used to evaluate students' solutions in producing an answer or solution to a stated question. Questions in this context can be a coding problem, writing an essay or carrying out a project. Rubrics contain three distinct parts: evaluative criteria, quality definitions and scoring strategy [15]. The evaluative criteria focus on defining the criteria that determine whether a solution is accepted or rejected. The quality definitions address different levels of quality based on the evaluative criteria. Lastly, the scoring system can either be holistic or analytical, where the former aggregates the scores from the evaluative criteria and gives a final quality score, while the latter considers each evaluative criterion in isolation, where feedback can be given [15].

Specific rubrics for plots have been explored by Angra & Gardner [16], who present aspects for consideration when assessing plots. Two examples include *does the plot have a descriptive title* or *does the plot have appropriate x-label units*. The details and usage of rubrics can vary depending on the field. In biology education, where Angra & Gardner had their basis, the focus is on ensuring that graphs effectively

communicate experimental results and align with research questions [16]. Meanwhile, areas like data visualisation may emphasise aesthetic principles such as clarity and colour usage.

Furthermore, Angra & Gardner defined three categories that constitute a plot: plot mechanisms, communication and plot choice. Plot mechanisms focus on the fundamental components of a plot, including titles, axis labels, scaling, and the use of colours to differentiate data series. These elements ensure that the plot is properly formatted and easy to interpret. Communication focuses on how effectively the plot conveys information. This includes clarity, aesthetics, and ensuring that the plot highlights the key message or trend in the data. A well-communicated plot should be visually clear, free from unnecessary distractions, and structured in a way that makes its insights easily understandable to the viewer. Plot choice relates to selecting the most appropriate type of graph based on the nature of the data and its intended purpose. It involves considering how the data should be represented, whether the chosen visualisation aligns with the research question, and ensuring that the graph effectively supports the intended analysis [16].

Lastly, each rubric within the three categories (plot mechanisms, communication and plot choice) is assessed using three levels of quality: present/appropriate, needs improvement, and unsatisfactory, each assigned a specific weight. The plot mechanisms category is weighted lower than the communication category because evaluating communication requires a more nuanced judgement of how effectively the plot conveys information rather than just checking for the presence of required elements. Each plot is assessed separately for plot mechanisms, communication, and plot choice, generating three category-specific scores. These scores are then aggregated into a final score, allowing students to receive targeted feedback on which aspects of their graphing skills need improvement [16].

While well-defined rubrics provide a structured method for evaluating the quality of student-generated plots, applying these rubrics at scale depends on having access to the plots in a structured and interpretable format. This challenge has given rise to research in plot mining, which focuses on extracting and analysing plots from documents such as scientific articles to make their visual content accessible for further processing.

2.2.3 Plot Mining

Davila et al. [17] studied the extraction and information mining from plots. Leveraging plots within scientific articles and applying ML techniques, they processed a large number of research papers and automatically identified and extracted the plots. The motivation for this research area is accessibility, aiming to help the visually impaired since the information conveyed in the figure can be converted into accessible formats such as audio [18]. Two distinct approaches to extraction were investigated [17], raster-based and vector-based. Raster-based extraction uses deep neural networks to identify the location of the figure, classify the plots or graphs in the figure, and then extract them. Surrounding text of the figures is referred to as candidate captions, which are labels and titles. Meanwhile, vector-based extraction exploits

popular formats such as PDF for their function of drawing operations for elements in documents [17].

2.2.4 Datasets

Using the work of Davila et al. [17] among others in plot mining, there has been work to create large datasets. For instance, the PlotQA-dataset have elements such as X/Y-axis labels, and legends, annotated with bounding boxes [19]. Another synthetic data set, built by Davila et al. [20], has been used in the ICDAR competition[2]. A smaller part of the dataset contains material from the United States PubMedCentral [21] open archives. The above two datasets have, in total, over 400,000 data points of plots with annotated plot elements. Annotations are essentially bounding boxes that are represented by coordinates in a two-dimensional Cartesian plane. Ten different types of plots are included, for example, line plots, bar plots, and scatter plots [22].

The datasets are mainly used to train object-recognition models to learn a plot's different elements. For example, the ICDAR competitions let teams globally compete against each other in developing and evaluating their models to extract relevant information from unseen plots [20]. The aim of these kinds of competitions and developments of object-recognition is accessibility for the visually impaired [18]. However, these datasets are not suited for training ML-models for automatic scoring, as they lack labelled data related to scores or grades. While such datasets help models detect visual components, they do not help in understanding their meaning or purpose.

2.2.5 Embeddings

While datasets provide the foundation for training models to recognise visual elements, they do not capture the semantic or contextual meaning of these elements for ML models. To help ML models interpret and reason about images and visual content, embeddings are commonly used. Embeddings represent data, such as words or images, in continuous vector spaces, enabling models to capture semantic relationships and structural similarities. By mapping high-dimensional data into a lower-dimensional space, embeddings make it easier for ML models to process and generalise patterns. They serve as a bridge between raw data and meaningful representations, facilitating tasks such as classification, retrieval, and pattern recognition [23], [24].

Various methods exist for generating embeddings, each with its approach to capturing relevant features. Some methods rely on statistical relationships within data, extracting patterns based on co-occurrence and distributional properties [24]. Others dynamically adjust representations based on context, allowing embeddings to reflect nuanced variations in meaning [25], [26].

Embeddings are widely used across domains. In natural language processing (NLP), they help represent words and sentences in a way that preserves semantic similarity, improving applications such as sentiment analysis, translation, and information

[2]Competition on Harvesting Raw Tables from Infographics

retrieval. In image processing, embeddings serve to represent visual features, aiding in object recognition and related tasks [27], [28].

Over time, advancements in embedding techniques have improved their adaptability and expressiveness. Earlier approaches generated fixed representations, while more recent methods adjust dynamically based on surrounding information. This evolution has significantly enhanced the ability of ML models to understand and process complex data structures efficiently.

2.2.6 Neural Networks

Neural networks play a central role in enabling machine learning models to interpret visual elements in plots. Leveraging embeddings as internal representations of data allows them to extract meaningful patterns. Convolutional neural networks (CNNs) work by learning hierarchical patterns in visual data, starting from simple edge detection in the lower layers and progressing to more complex structures, such as textures and shapes, in the deeper layers [29]. This ability to extract abstract visual features makes them ideal for tasks involving perceptual similarity.

In recent years, CNNs have been combined with perceptual similarity metrics, such as Learned Perceptual Image Patch Similarity (LPIPS), to assess the semantic differences between images in a manner that aligns more closely with human perception [30]. LPIPS leverages the powerful representations learned by pre-trained CNNs, comparing deep feature representations rather than relying solely on pixel-wise differences. By harnessing both local and global image features through various CNN layers, LPIPS uses the network's activations to detect subtle differences between images. The pre-trained CNNs are trained on large datasets on classification tasks and varying kinds of datasets [30].

However, not all evaluation tasks rely on spatial relationships alone. Some require an understanding of sequential and contextual dependencies, particularly when working with natural language processing (NLP). Unlike images, where meaning is derived from structural composition, text must be processed based on linguistic context and relationships between words. This motivates the use of large language models (LLMs), which are specifically designed for such tasks. LLMs leverage transformer architectures to process and generate human-like text by capturing complex contextual relationships. Unlike traditional sequence-based models, transformers utilise self-attention mechanisms, allowing them to consider dependencies between words regardless of their position in a sentence [25]. This enables LLMs to perform a range of natural language understanding tasks, including semantic analysis, reasoning, and text classification.

While most transformer-based models operate at the token level, like Bidirectional Encoder Representations from Transformers (BERT) [26], capturing relationships between individual words, some tasks benefit from sentence-level representations. Models like Sentence-BERT (SBERT) adapt the BERT architecture to produce fixed-size embeddings for entire sentences [5]. This not only enables efficient semantic comparisons across sentences but also significantly speeds up inference, especially in

scenarios involving large-scale similarity search.

The influential transformer architecture that underpins many LLMs has also been adapted for image processing. While images are not inherently sequential like language, the method of splitting an image into patches and sequentially processing these patch embeddings with self-attention has proved successful, as demonstrated by Dosovitskiy et al. [27]. However, vision transformers (ViTs) generally require larger datasets and higher-resolution images during training, which can pose challenges when applied in niche domains [31].

Beyond standalone ViTs, hybrid models that combine vision and language modalities have emerged, with Contrastive Language-Image Pretraining (CLIP) [32] being one of the most influential. CLIP extended the transformer-based approach by jointly training image and text encoders in a contrastive learning setup, aligning image embeddings with their corresponding textual descriptions. This enabled zero-shot classification and robust cross-modal understanding without task-specific fine-tuning. While CLIP performed well on diverse datasets, its reliance on large-scale pretraining means it struggled with highly specialised domains.

While deep learning models have gained prominence due to their ability to learn complex representations from large datasets, classical ML models remain essential tools, particularly when data is limited or interpretability is important [33]. These models are well-established, efficient to train, and often serve as strong baselines or components in hybrid systems.

2.2.7 Classic Machine Learning Models

Among the classical ML models, linear regression, support vector machines, and random forests have emerged as particularly influential due to their strong empirical performance, theoretical grounding, and widespread use across domains [34]. These models can be applied to classification tasks, where the goal is to predict categorical outcomes, or regression tasks, where the goal is to predict continuous values.

Ensemble methods are powerful ML techniques that combine the predictions of multiple classifiers or regressors to improve overall model performance. The foundational idea is that while individual models may make errors, a diverse collection of models can compensate for one another's weaknesses. As Dietterich [35] explained, ensemble methods typically involve training multiple models and aggregating their outputs through mechanisms like majority voting or weighted averaging. The same logic applies to random forests, where multiple decision trees are grown and each makes a prediction that contributes to the final prediction. These models will be central to the implementation of machine learning models that predict student scores.

2.2.8 Similarity Metrics

To assess the performance of both advanced ML and classical models in classification, regression tasks, and feature extraction, similarity metrics are commonly used. Understanding how different model architectures capture visual patterns provides a

foundation for evaluating similarity, which is essential in tasks such as comparing student-generated plots to reference plots. Although there is considerable research on image similarity and various methods for determining it [36], the specific case of plot similarity and how it should be properly assessed remains largely unexplored in the literature. In the broader field, Fan et al. [36] presented three camps in the field of image similarity: pixel, structure and semantic similarity.

Pixel similarity deals with classical metrics such as mean squared error (MSE). Even though pixel similarity is quick and easy to calculate, there are significant drawbacks to the metrics. The two main drawbacks are that small differences, unnoticeable to the human eye, are regarded as a large difference and that pixel similarity does not pick up the overarching semantics of the image. Structural similarity builds upon pixel similarity, but instead of individual pixels, the metric divides images into regions and tries to find similar regions. Metrics such as the structural similarity index measure (SSIM) are considered a metric for structural similarity [37]. The same lack of semantic drawback as pixel similarity metrics can be seen with SSIM, but the benefit is that it is not as sensitive to small pixel differences.

Semantic similarity measures how similar two images are at a conceptual level, beyond just pixel-level differences. Examples of such metrics are LPIPS [30] and ViTScore [38], which utilise pre-trained neural network embeddings to assess both global structure and local details in images. Meanwhile, another semantic similarity metric, CLIPScore [39], measures similarity by comparing text and image alignment rather than directly learning perceptual features. Some metrics excel at capturing high-level semantic relationships, while others focus more on fine-grained textures and features. Which metric should be in use is highly dependent on the context. A more recent approach, Semantic Similarity Score (SeSS), goes further by converting images into scene graphs, which represent objects and their relationships. SeSS and LPIPS are considered particularly good at mimicking human perception of image similarity [30], [36].

LPIPS compares the deep feature representations of images extracted from pre-trained CNNs such as AlexNet, VGG, or SqueezeNet. The backbone CNN's weights are frozen, and a new learnable layer is added on top, which is trained on human perceptual judgment data. By optimising this layer, LPIPS learns to predict perceptual similarity based on high-level visual features, which align better with human perception. The original LPIPS authors found AlexNet to perform best, making it the preferred choice for this work [30] to assess visual similarity. Additionally, LPIPS offers a publicly available and documented implementation, making it more accessible for practical applications compared to SeSS, which currently lacks a public implementation.

2.2.9 Chart Understanding

While similarity metrics provide valuable tools for comparing visual elements at different levels, such as pixel-level details or semantic features, they are limited in their ability to capture the full contextual meaning of a plot. This limitation becomes especially clear in tasks that require an understanding of the plot's purpose and content.

Building on advancements in plot extraction from scientific articles, the field of automatic chart understanding has seen significant progress with the rise of vision transformers (ViTs) and large language models (LLMs) [40]. Leveraging large corpora of plots with corresponding captions such as ChartQA [41] or PlotQA [19], models like OpenAI's general-purpose multi-modal CLIP from 2018 has been adapted to connect plots with the corresponding caption given in the dataset [32]. These models, known as vision-language models (VLMs), can process both images and text, making them useful for tasks requiring reasoning over visual data. This field is driven by enhancing plot accessibility and reducing the workload of professionals such as data analysts, thereby improving efficiency in data-driven decision-making [42]. There are models catered to specific tasks within chart understanding, such as Chart Summarisation, Chart Question Answering and Chart Captioning [42].

One of the most recent models addressing such open-ended tasks is ChartGemma [43], which was trained exclusively on plots and their corresponding descriptions, making it specialised for chart understanding rather than general image recognition. ChartGemma aimed to be as general as possible, such that it can be applied to varying kinds of fields and tasks. Compared to closed-ended tasks such as Chart Question Answering (CQA), where there is a predefined answer, open-ended tasks require VLMs to identify key elements, such as data points, titles, and labels and then generate a natural language description that encapsulates the plot's content [43]. For instance, a query might ask, *What trend is visible in this plot* or *What type of plot is depicted, and what do the labels indicate*. The integrated LLM processes the image alongside the query to produce an output that not only describes these elements but also provides context and reasoning.

To develop the model to be as general as possible, the developers of ChartGemma trained the model on actual images rather than on the underlying tabular data, which was the norm for previous chart understanding models [43]. This training approach significantly improved the model's ability to capture both high-level and low-level features of plot images. High-level features include trends, overall shapes, and patterns, while low-level features cover details such as colours, ticks, and peaks/lows. Being trained on images allowed the model to better interpret the nuanced visual cues that are often ambiguous when represented solely in tabular form [43].

2.2.10 Commercial Autograder Tools

Finally, there exist commercial auto-grader tools such as nbgrader [44]. Nbgrader does not support scoring of plots, but they suggest another tool called Plotchecker [45]. Plotchecker programmatically evaluates plots based on test cases. However, Plotchecker only checks for method calls and cannot score images of plots. As a consequence of only programmatically evaluating plots, it cannot semantically analyse and score them against a reference plot.

From previous research, it suggests that the field of automatic scoring of plots is largely untouched apart from the works of Yang [14] and Vitale et al. [13]. However, neither study provides a clear assessment of how automatic scoring tools can reduce the teaching assistant's (TAs) workload. This gap defines the primary objective:

to develop a data transformation pipeline that prepares data for machine learning models and thereafter, predicts student-produced plots.

The models classify student plots into three kinds of scores: *not passed, needs improvement, passed* or, in another form, score 0, 1 and 2, respectively. This allows TAs to focus their attention on the most challenging or ambiguous cases. By automating routine assessments, the system aims to streamline the scoring process while maintaining human oversight. The overarching goal is to evaluate how effectively such a tool can support TAs and how well it agrees with TAs in scoring student-generated plots.

3

Methods

This chapter describes the method for creating a pipeline and evaluating the automatic scoring of plots. First, data is presented, which consists of Python code submissions from an introductory data science course. Next, the introduction of the universal format for representing plot objects in both student submissions and reference solutions. Afterwards, an explanation of the rubrics, distinguishing between basic and complex categories, is presented and shows how they are used to score student-produced plots. After that, we present the models used for tasks such as perceptual similarity measurement, along with the pipelines that convert code into data and metrics. These data and metrics will be used for the training of machine learning models that predict student scores. Finally, metrics, risk assessment, and ethical considerations are discussed.

3.1 Data

First, the dataset used in this work is described, including its structure, preprocessing steps, and overall characteristics. The dataset consists of student submissions from an introductory data science course, along with corresponding reference solutions. Below is an outline of the process used to extract relevant code snippets and the steps taken to preprocess the data.

3.1.1 Python Code

The data used in this work consists of student solutions for plotting tasks, stored as Python (.py) files. Although students originally submitted their work as Jupyter notebooks (.ipynb), the dataset used in this project includes the exported Python script version. It should be noted that the original Jupyter notebook submissions were not part of the dataset. These Python files contained code for data loading, transformation, and plotting. All submissions were anonymised to ensure no personally identifiable information was accessible. The data includes submissions from the course instances in autumn 2024 (HT24) and spring 2025 (VT25). The data was collected by this work's supervisor, who informed students that their code would be used in a research project if they consented. A total of 2,107 student submissions were collected, with 136 from HT24 and 1,971 from VT25.

Reference solution code was also included and provided as Jupyter Notebooks (.ipynb)

for each assignment. These notebooks were used by the TAs to score the submissions and included detailed explanations, sample implementations, and expected output for the plot tasks. They served as a basis for assessing the correctness of student submissions.

A summary of the data filtering process is presented in Table 3.1. As a limitation (see Section 1.2) in this project, open-ended plotting tasks were excluded. However, a potential pipeline to support open-ended tasks can be seen in Section 3.7.2.

Table 3.1: Overview of data availability and filtering

Description	**Number of Submissions**
Original total submissions	2107
Excluded open-ended assignments	139
Final usable submissions	**1968**

3.1.2 Student Scores

In addition to the student code and reference solutions, access to a CSV file containing the scores assigned to student submissions by teaching staff was provided. Each row in the CSV corresponded to a specific question within a student's submission and included the assignment name, the submission ID, the question number, and the labelled scores. This meant that a single student ID could appear multiple times in the file, once for each question. However, the scores were aggregated at the question level, meaning detailed evaluations of individual plot components, such as titles, axis labels, legends, or plot types were not available. Still, the overall scores served as a valuable reference point for comparing the general performance of manual scoring versus automated scoring.

The two course instances used a score scale of 0, 1 and 2. However, some questions in VT25 instead used a scale with 0 and 1. A score of 0 reflected a plot that failed to sufficiently answer the question or lacked the most critical elements, such as titles, labels, and tick marks. A score of 1 meant certain plot aspects were incorrect, missing, or misleading. For instance, missing labels or legends when showing multiple data series, whereas a score of 2 indicated a perfect plot with no objections [46].

3.1.3 Preprocessing of Data

Preprocessing steps were applied to the data to prepare it for downstream applications. First, the reference Jupyter Notebooks were transformed into regular Python scripts. Second, the TA-assigned scores were max-normalised across the assignments and questions. These two preprocessing steps are described.

To align the reference notebooks with the per-assignment and question structure of the student submissions, individual code snippets were extracted based on question-specific tags included within each reference solution. Each snippet was tagged in the reference notebook, indicating the specific question it addressed. An example of the reference solutions and the tags can be observed below in Listing 3.1.

```
import pandas as pd
import matplotlib.pyplot as plt

data = pd.read_csv("example_dataset.csv")

# Codegrade Tag Question1

summary_stats = data.describe().round(2)

# Codegrade Tag Question2

categories = data['category'].unique()
for category in categories:
    subset = data[data['category'] == category]
    plt.scatter(subset['var_x'], subset['var_y'], label=
       category, alpha=0.7)

plt.xlabel("Variable X")
plt.ylabel("Variable Y")
plt.legend(title="Category")
plt.title("Scatter plot grouped by category")
plt.show()

# Codegrade Tag Question3

another_dataset = pd.read_csv("df.csv")
head = another_dataset.head(5)
```

Listing 3.1: Example structure of reference solution with tags

Taking advantage of the tags present in the reference solutions allowed for the dynamic extraction of relevant code snippets, specifically the tags involving plotting. For instance, if question 2 (a plotting task) were to be extracted from Listing 3.1, all the code until (exclusive) the tag `# Codegrade Tag Question2` was extracted, as code following this tag related to question 3. This approach included all necessary data transformations relevant to the plotting task.

To achieve consistency in the data, each student's score was normalised by dividing the labelled score by the maximum possible rubric score for the corresponding assignment and question (i.e. max normalisation). This normalisation resulted in continuous scores ranging from 0 to 1. Normalisation allowed comparisons to be made between submissions in different course instances since score scales were different and within assignments. Additionally, normalisation also made model training more robust. Below in Figure 3.1, the distribution of the normalised scores can be observed.

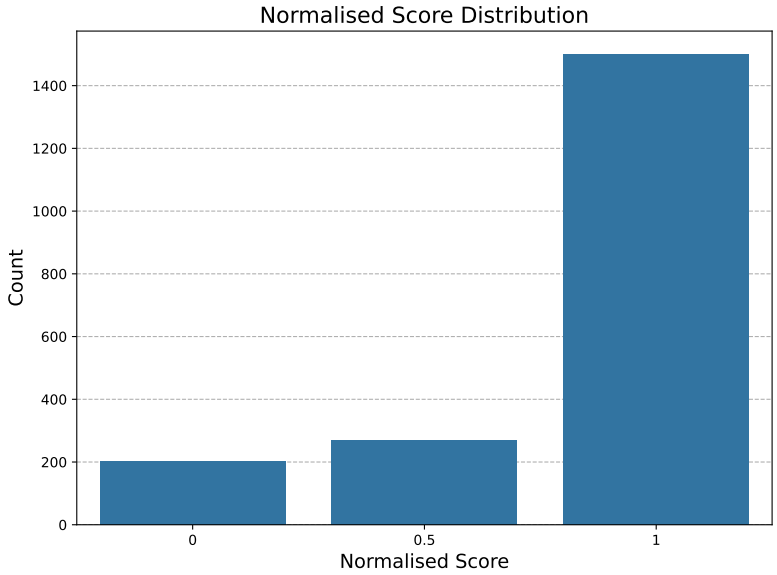


Figure 3.1: Distribution of normalised scores

Normalising the scores across both course instances also highlighted the underlying class imbalance in the dataset. As shown in Figure 3.1, a large proportion of submissions, 1499, received a normalised score of 1.0, indicating that most student plots were fully correct and aligned with the expectations defined in the rubrics. A total of 268 submissions received a normalised score of 0.5, while 201 submissions received a score of 0.0.

3.2 Universal Structured Format

Following Yang [14], plot objects were extracted into a structured format for both student submissions and reference solutions, enabling uniform analysis and facilitating structural transformations when necessary. JSON was chosen for this universal structured format (USF) due to its flexibility with Python-based scripts and its potential for integration into other applications, such as a web-based graphical interface.

A JSON structure was defined, tailored specifically for this work. The structure captured all relevant plot elements required for accurate analysis, including textual elements (e.g., titles, labels), data points, and visual components. It was designed

to handle single or multiple subplots within a single plot, reflected by assignment requirements. To reliably extract these elements, the built-in methods provided by the Matplotlib API were leveraged (e.g., `get_title()`, `get_xlabel()`, `get_lines()`, etc.). Table 3.2 summarises the key fields included in the USF. An example of the JSON structure is available in Appendix A.

Table 3.2: Overview of fields in the Universal Structured Format (USF)

Field Name	**Data Type**	**Description**
`title`	string	Title of the subplot.
`xlabel`	string	Label of the x-axis.
`ylabel`	string	Label of the y-axis.
`legend`	boolean	Boolean indicating whether a legend is present.
`type`	string or null	Detected plot type (e.g., line, bar, scatter).
`lines`	list of dicts	Data for each line plot, including x and y values.
`scatters`	list of dicts	Data for each scatter plot, including x and y values.
`bars`	dict	Dictionary with bar plot data: `bars` (list of bar coordinates).
`ticks`	dict	Dictionary containing major tick positions and labels for both x and y axes.
`num_classes`	int	Number of detected class labels in the legend (if any).

A challenge with unifying plots into a single unified structure stems from the fact that students could write code in varying ways while producing semantically identical outputs. Students could use different colours, subplot order and plot sizes. For example, the order of subplots or data series within a single plot differed between a student submission and the reference solution. Since the method involved comparing textual elements (e.g., titles, labels) and visual data representations, a requirement of a standardised structure and mapping of subplots was needed to ensure meaningful, element-wise comparisons for both the textual elements and visual data representations.

The USF enabled the direct comparison of textual elements, but this did not apply to the representation of visual data. Reconstruction of plots from the extracted data stored in the USF was done to accurately compare the visual representation between student submissions and reference solutions. The next two sections present methods for handling subplot mapping in figures with multiple subplots and the reconstruction of plots based on the data stored in the USF.

3.3 Subplot Mapping

In tasks where multiple subplots were present within a single figure, the number of possible subplot orderings was $n!$, where n is the number of subplots. This factorial growth introduced difficulties when comparing student solutions to the reference solution for both textual and visual similarity. Therefore, it was necessary to ensure that each student subplot was compared to the corresponding reference subplot.

The process mapped each subplot from a student's figure to the best corresponding subplot in the reference figure by computing numerical feature representations of each subplot (e.g., distributional summaries of plotted data) and comparing them using Euclidean distance and correlation. In addition to data-based features, the mapping process also incorporated label similarity (e.g., axis labels) and accounted for the possibility of flipped orientations.

Flipped orientations occurred in scatterplots, where students may reverse the plotted relationship (e.g., plotting Y vs. X instead of X vs. Y). To handle this, each student subplot was evaluated in both its original and flipped versions when the plot type was identified as scatter. For each orientation, a cost was computed based on the distance between quantile summaries, differences in range ratios, and when textual labels were present, the string similarity between axis labels. If swapping the axis labels significantly increased the similarity to the reference plot, the flipped version was selected.

The orientation (original or flipped) that yielded the lowest overall cost was selected for every possible pair of student and reference subplots. Once all pairwise costs had been computed, the optimal mapping between student and reference subplots was determined using the Hungarian algorithm [47]. This ensured a globally optimal mapping that accounts for ordering and orientation differences.

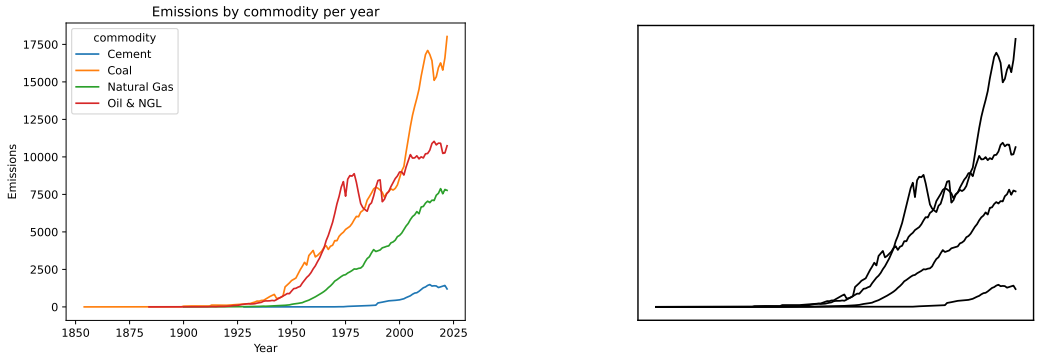
The final output of this step was a mapping that specified which student subplot corresponded to which reference subplot, including whether a flipped orientation was selected. This mapping ensured consistency in subsequent stages in downstream comparisons.

3.4 Standardised Plot Reconstruction

While the USF and subplot mapping allowed for rigorous comparisons of textual elements such as axis/tick labels and title, the visual elements needed to be transformed to a standardised format for downstream visual comparisons. Differences in styling, such as colour schemes, figure sizes, line-width and such, could still significantly affect direct visual comparisons. To address this, a standardised plot reconstruction step was required. In this step, each universal structured format (USF) JSON file was processed by a Python script that reconstructed the plots using a standardised, clean, minimalistic styling. These reconstructed plots retained only the essential visual data elements, omitting elements such as labels, titles, legends, and colour diversity. The motivation for this reconstruction was to suppress differences that

could distort visual similarity metrics. By reconstructing each plot, both student and reference, with a consistent, minimalistic style, enabled comparisons solely on the underlying structure of the data, rather than stylistic variations.

Early experimentation showed the necessity of this step. Directly comparing student-generated plots against reference plots proved unreliable because differences in colour palettes, figure sizes, and layout often caused large deviations in visual similarity scores, even when the underlying data was accurately represented. These inconsistencies highlighted that it is not accurate and rigorous to use the original plots for comparison and that a controlled visual standardisation process was needed. Figure 3.2 illustrates this by showing an original student plot alongside its standardised reconstruction.



(a) Original plot

(b) Reconstructed version

Figure 3.2: Original plot (left) and reconstructed version (right)

There were two considerations when reconstructing plots from the USF. The first consideration involved categorical scatterplots with grouped data. In these plots, each data series often corresponds to a category along the X-axis. However, students may plot these categories in a different order than the reference plot. To ensure consistent alignment, the reconstruction process always plotted categorical features in ascending order, based on class frequency. This guaranteed that the series appears in the same left-to-right order in both student and reference plots, regardless of the student's original plotting order.

The second consideration regarded the fact that students may plot relationships in a flipped orientation (e.g., Y vs. X instead of X vs. Y). While this is semantically the same in many cases, such decisions may conflict with the task instructions, especially if students were explicitly told to place a specific variable on the X-axis. The detection and handling of such flipped orientations are addressed in the subplot mapping step in Section 3.3.

The two above considerations were designed to remove unintended discrepancies between student and reference plots. Without such standardisation, visual similarity comparisons could be misleading and/or unfair. By enforcing consistent styling,

series ordering, and simplified structure, the reconstruction process ensured that the resulting plots are suitable for reliable similarity assessment.

3.5 Rubrics

To increase modularity and ease of implementation, the rubrics were categorised into two types: basic and complex. Splitting them into two distinct groups enabled separate development and implementation since they used different scoring logic.

3.5.1 Basic Rubrics

Basic rubrics are those that could be checked for presence in the universal structured format (USF). These rubrics served as the foundation of a properly structured plot, ensuring that students understood how to convey necessary information for the correct interpretation of the plotted data. In Table 3.3, the basic rubrics in this work can be observed.

Table 3.3: Overview of evaluated plot features

Feature	**Description**
Title presence	The plot includes a title.
Correct plot type	The correct plot type is used.
Legend presence	The plot includes a legend.
X-axis label presence	The x-axis is labelled.
Y-axis label presence	The y-axis is labelled.
X-ticks presence	The x-axis contains ticks.
Y-ticks presence	The y-axis contains ticks.
X-axis unit	The x-axis label includes an appropriate unit.
Y-axis unit	The y-axis label includes an appropriate unit.

The basic rubrics listed above did not necessarily apply to all questions within an assignment. Different questions in the same assignment may have required a specific subset of rubrics. For example, when plotting a histogram, including a y-axis label is often redundant since histograms typically represent the frequency of x-axis values. To easily adapt basic rubrics for each assignment and question, their configuration was structured in a JSON file. Each field in the JSON corresponded to the requirement of each basic rubric, i.e *True* for required and *False* for not required. A field indicating the maximum earnable score was also included to reflect the real rubrics. An example of this JSON structure for the basic rubrics is shown below in Figure 3.3.

The listed rubrics in Figure 3.3 are based on the real rubrics, and those can be seen below in Table 3.4 with the corresponding scoring rules. These rubrics are just an example of how they can be structured and how the scoring rules can be formulated.

```
{
    "required_plot_type": "histogram",
    "require_title": true,
    "require_legend": false,
    "require_xlabel": true,
    "require_ylabel": true,
    "require_x_ticks": true,
    "require_y_ticks": true,
    "require_xunit": true,
    "require_yunit": false,
    "max_score": 2
}
```

Figure 3.3: JSON structure of assignment 2 question 5

Table 3.4: Rubrics for assignment 2, question 5

Rubrics
1. The plot is of the correct type (histogram)
2. Axes are clearly labelled, with price on the x-axis and count on the y-axis, including the unit on the x-axis (MSEK)
3. Ticks are given in MSEK at 0.25 MSEK intervals
4. There is a descriptive title
5. Sturges' rule ($k = 1 + \lceil \log_2 n \rceil$) is used for selecting the number of bins

Scoring Rules:

- **2 points:** All given criteria have been met.
- **1 point:** The plot *looks correct* (with respect to the model solution), but some criteria are missing.
- **0 points:** The plot is incorrect or fails to meet key requirements.

As mentioned in section 1.2, some rubrics fell outside the scope of this project. One such example is to enforce specific shapes or bin sizes in plots as seen in Table 3.4 point 5. Enforcing detailed requirements, such as Sturges' rule, would go beyond the intended functionality. Instead, the evaluation of such aspects relied on visual comparison between the student's plot and the reference solution to assess how well the student met the specified rubrics.

Complex rubrics required the use of similarity metrics or machine learning techniques for proper scoring. These rubrics evaluated elements beyond simple presence checks and involved deeper analysis. The complex rubrics are shown in Table 3.5.

Table 3.5: Complex rubrics

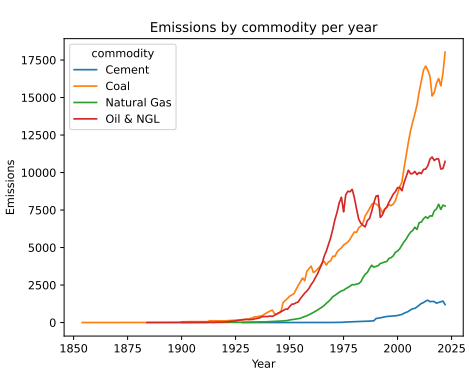
Criterion	**Description**
Data representation	The plot is similar to the reference solution
Title descriptiveness	The title is sufficiently descriptive.
X-axis label descriptiveness	The x-axis label is sufficiently descriptive.
Y-axis label descriptiveness	The y-axis label is sufficiently descriptive.

These complex rubrics required similarity metrics to evaluate them, or even using natural language processing models such as SBERT [5]. Due to the aggregated data, it was not possible to define reliable pass/fail thresholds for each complex rubric. While the similarity metrics ranged from 0 to 1, making it theoretically possible to establish such thresholds, the lack of labelled examples for each rubric inhibited accurate determination of where these thresholds should be drawn.

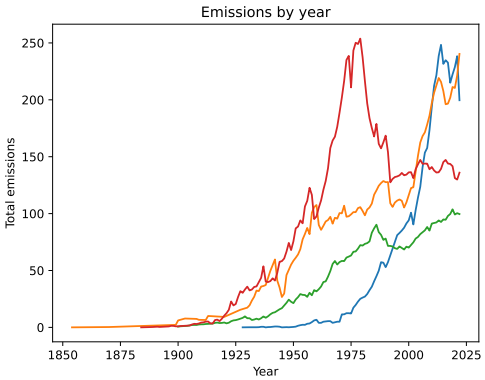
3.6 Metrics and Models

Metrics and models are used in the data transformation pipeline to assess the similarity between student and reference solutions. Basic rubrics, such as checking axis labels or function calls, could be extracted directly from students' Python code. However, assessing more complex aspects, such as the perceptual similarity between a student's plot and a reference plot, required a deeper approach. Simple rule-based methods struggled with subtle visual differences, making neural-network-based metrics such as Learned Perceptual Image Patch Similarity (LPIPS) a more suitable choice.

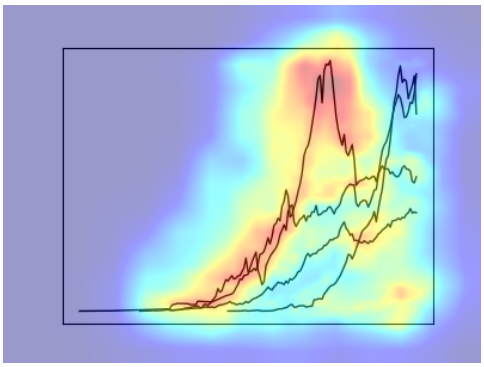
LPIPS performed significantly better than traditional metrics like Mean Squared Error (MSE) and Structural Similarity Index (SSIM), which rely on pixel-wise analysis [30]. MSE calculates numerical differences between corresponding pixel values, while SSIM incorporates luminance, contrast, and structural information but remains limited to the pixel level [37]. In comparison, LPIPS captures semantic and perceptual differences critical for human judgment, providing a more accurate measure of image similarity. The LPIPS score ranges between 0 and 1, but perfect similarity is indicated with 0 and most dissimilar with 1.



(a) Reference solution



(b) Student solution



(c) Heatmap overlay after using LPIPS

Figure 3.4: LPIPS overlay (=0.1779) between reference and student solution

Above in Figure 3.4, a reference solution, a student's plot, and the corresponding LPIPS-based overlay are shown. The student's plot deviates significantly from the reference, particularly in the red and blue line segments. These differences are caught by the model and highlighted in the overlay, where areas of high perceptual difference are prominently marked. It should be noted that the overlay comparison is based on the reconstructed plots, not the original images. This provides insight into what the model *sees* as different, thereby adding interpretability for specific submissions in the automated scoring process.

In addition to LPIPS, newer architectures trained on more recent datasets were also evaluated for perceptual similarity. The goal was to identify models capable of understanding and comparing plots in a meaningful way. Since embedding-based approaches appeared most promising, DISTS [48], CLIP [49], and the vision encoder from ChartGemma [43], a model specifically trained on plot images, were explored. However, the most consistent metric was LPIPS, making it the chosen visual similarity metric.

3.7 Data Pipelines

Building on the selected evaluation methods, the following data transformation pipeline prepares the student submissions for the downstream task of training machine learning models. The first pipeline presented compares student solutions to a predefined reference solution. After machine learning models were developed and trained, predictions were made on unseen data to evaluate their performance.

In contrast, the second pipeline is only a proposal designed for open-ended tasks, where no fixed reference exists. Instead, it evaluates solutions by comparing the embeddings of the visual representation with those of the textual description, ensuring alignment between the plotted data and its explanation. The second pipeline is left for future research to implement and evaluate, but a potential pipeline can be seen in Appendix B.

3.7.1 Tasks with a Reference Solution

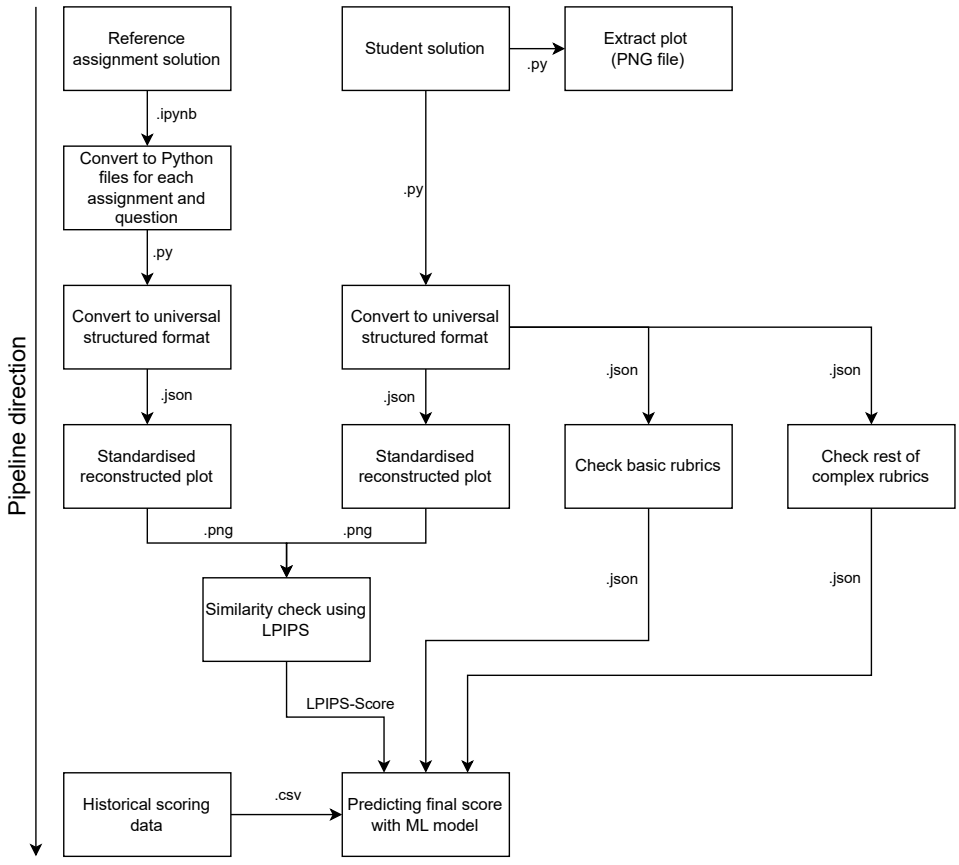


Figure 3.5: Scoring pipeline when reference solution is available

The following is a step-by-step explanation of the pipeline, from start to end, as visualised in Figure 3.5. It is important to first consider the input to understand how the pipeline operates. As mentioned in Section 3.1, the input consisted of student

Python files containing only the code used to generate plots. Reference solutions, however, were provided in Jupyter Notebooks. Before processing, these notebooks were converted to Python scripts and integrated into the same structure as student submissions. Both student and reference scripts were then executed to generate the corresponding plots and thereafter saved. While these plots were not used in later processing, storing them facilitates manual scoring when necessary.

Next, the data from the plots was extracted and stored in a universal structured format, where possible. This ensured consistency, making scoring easier since information can always be accessed using the same keys in the JSON documents. Yang [14] used a similar approach for scoring plots, but his format did not meet the needs of this project, so only the core idea was adopted in this work. Submissions could vary in structure, containing multiple subplots and different data arrangements, with axes potentially flipped, adding another challenge when comparing plots. While flipped plots could still be correct, they would be scored incorrectly if not adjusted.

After that, the plots were reconstructed to the standardised JSON format. This step reconstructed each plot in a consistent style, removing titles, labels, and other stylistic elements that could interfere with visual assessment. This ensured that only the structural and visual data aspects of the plots influence the similarity between the student and reference plot.

With the data standardised in the universal JSON format and the reconstructed plot as a foundation, scoring began with the basic rubrics. A configuration file defined which rubrics were assessed for each assignment and question, with a maximum score. The structured JSON format allowed easy checking for presence.

Afterwards, the complex rubrics could be applied, including the perceptual similarity (LPIPS) between the reference and student plots. In addition to visual similarity, textual elements such as the descriptiveness of the plot's title, x-label, and y-label were compared. To perform this comparison, a language model (SBERT) was used to embed the textual elements and compute their similarity to those of the reference plot. Finally, each of these evaluations, both perceptual and semantic, yielded a similarity score. These scores, along with other rubric-based assessments, were recorded in a structured JSON file that consolidated all rubric criteria. By systematically structuring the scoring process, the pipeline ensured fair and transparent scoring while reducing manual effort, ultimately supporting the work of TAs.

3.7.2 Proposed Solution for Open-ended Tasks

In the previous section, a pipeline for tasks with reference solutions was introduced. Initially, open-ended plotting tasks without a reference plot but with an accompanying description were intended to be included, as outlined in the data section. However, these tasks were excluded midway through the project. Although not part of the current implementation, an outline of a potential approach for evaluating open-ended student visualisations is provided. While untested due to data limitations, the proposed solution could serve as a foundation for future research. A visualisation of the pipeline can be found in Appendix B.

This approach builds on the proposed solution outlined in the previous section. The preprocessing and basic rubrics will be applied in the same manner as described earlier and are therefore not repeated here. However, the evaluation of complex rubrics requires a different approach. Unlike tasks with reference, open-ended solutions vary significantly in form and content, requiring a different method. The following section outlines a possible framework for handling these complexities.

For evaluating student-generated plots for open-ended tasks, an initial consideration was to use ChartGemma, a model trained on plot data that generates textual descriptions. The idea was to input student plots into ChartGemma [43] and then compare them with the students' descriptions using BERT for similarity assessment. While this approach worked well in some cases, frequent hallucinations made it unreliable for fair scoring.

To address the issue of hallucinations, embedding-based models were explored. OpenCLIP [49] was selected due to its ability to handle significantly longer text inputs than the original CLIP model, which is limited to 77 tokens. This made OpenCLIP better suited for evaluating complex rubrics in open-ended tasks, where detailed textual descriptions are essential for accurate assessment.

The evaluation method relies on generating embeddings for both the student's plot and their accompanying description. By comparing these embeddings using cosine similarity, the model assesses the degree of alignment between the visualisation and its explanation. Initial experiments were conducted using submissions from a previous iteration of the course, along with controlled tests involving random images such as a dog playing with a ball in a park, which provided early indications of the model's effectiveness.

In these tests, cosine similarity scores of 0.3 and above generally indicated a strong match between the description and the plot, while lower scores often corresponded to vague or incorrect descriptions. Despite cosine similarity theoretically ranging from -1 to 1, this range appeared to be a practical threshold for evaluation. Since CLIP-based models perform well in without being trained on the specific task, this approach eliminates the need for additional training, making it a scalable and adaptable solution for assessing open-ended student work.

While this approach shows promise, several limitations must be considered. The lack of open-ended task data for evaluation means its effectiveness remains unverified in practical settings. Additionally, the 0.3 cosine similarity threshold was based on preliminary observations and may not generalise across different datasets or tasks. It also remains unclear how well the method captures minor but crucial details in plots or whether training could further enhance its performance. Further research is needed to refine the threshold and validate the approach across a broader range of student-generated plots.

3.8 Predicting student scores

The prediction of student scores is the core focus of this thesis. Building on the data prepared by the pipeline, machine learning models were developed and evaluated to predict the scores of student submissions. This part of the thesis covers topics such as train, validation, and test splits, parameter selection, conversion to a regression problem, model selection based on previous research [50], and feature engineering.

3.8.1 Data processing

The dataset consisted of 1968 data points, which were split into training, validation, and test sets. First, the data were divided into a training set (85%) and a test set (15%). Then, the training set was further split into training (90%) and validation (10%) sets, yielding 1504 training samples and 168 validation samples. The test set remained locked and was only used for the final evaluation after selecting the best model and features based on validation performance. The final training, validation, and test splits contained 1504, 168, and 296 datapoints, respectively.

Due to the imbalanced distribution of the score classes in the dataset, stratification was used for both the train/test and train/validation splits. Stratification ensured that the class distribution remained consistent across all splits, helping to prevent bias toward the majority classes [51]. As a result, the same class imbalance was preserved across all splits, as shown in Table 3.6, where the number of data points and the proportion of each class can be observed for each data split.

Moreover, synthetic data generation methods were considered as a means to mitigate the class imbalance and improve model robustness. Initial experiments with SMOTE[1] did not yield performance improvements compared to training on the original dataset. Further discussion regarding SMOTE is provided in Section 5.4.

Table 3.6: Split sizes across training, validation, and test sets with class proportion

Score class	**Training size(%)**	**Validation size(%)**	**Test size(%)**
0	154 (10.24%)	17 (10.12%)	30 (10.14%)
1	205 (13.63%)	23 (13.69%)	40 (13.51%)
2	1145 (76.13%)	128 (76.19%)	226 (76.35%)
Total	**1504 (100%)**	**168 (100%)**	**296 (100%)**

Furthermore, not all student submissions were successfully executed in the data transformation pipeline. Most failures were due to errors in student code, while a minority were from pipeline limitations in handling rare edge cases. In total, 23 submissions were unsuccessful in the pipeline. Due to time constraints, not all types of edge cases were handled in the pipeline.

[1] Synthetic Minority Over-sampling Technique

3.8.2 Cross-Validation

To ensure robust model performance despite limited data and class imbalance, cross-validation was employed, comprising an outer and inner loop. The outer loop, using 10-fold cross-validation on a dataset of 1504 points, provided an unbiased estimate of generalisation performance. Meanwhile, the inner loop handled model hyperparameter tuning independently from the final evaluation [52].

3.8.3 Converting to a Regression Problem

Rather than directly predicting the discrete score classes (0, 1, and 2) through a classification approach, the problem was reformulated as a regression problem. The model was trained to predict a continuous normalised score (`norm_score`) instead of fixed classes. This strategy provided greater flexibility, allowing the model to better account for the inherent variability and subjectivity in human scoring, such as the different leniency of TAs. The predicted normalised score was then converted back into the mentioned discrete score classes using bins that correspond to each class.

Converting the classification task into a regression problem enabled greater configurability in how the predicted scores are mapped back to discrete classes. For example, binning thresholds such as 0.3 and 0.8 could be defined. Predicted scores less than or equal to 0.3 would be assigned a score of 0, scores between 0.3 and 0.8 would be assigned a score of 1, and scores greater than 0.8 would be assigned a score of 2. This approach allows for a more flexible decision boundary compared to fixed class predictions and makes it easier to adjust the classification behaviour if needed. After the model outputs a continuous score, this binning is applied as a post-processing step to convert the prediction into the appropriate discrete class. Converting back to discrete classes allows for model evaluation using accuracy, precision, and recall. Below in Figure 3.6, three predictions are made on the normalised score (`norm_score`). These predictions are then mapped back into discrete score classes.

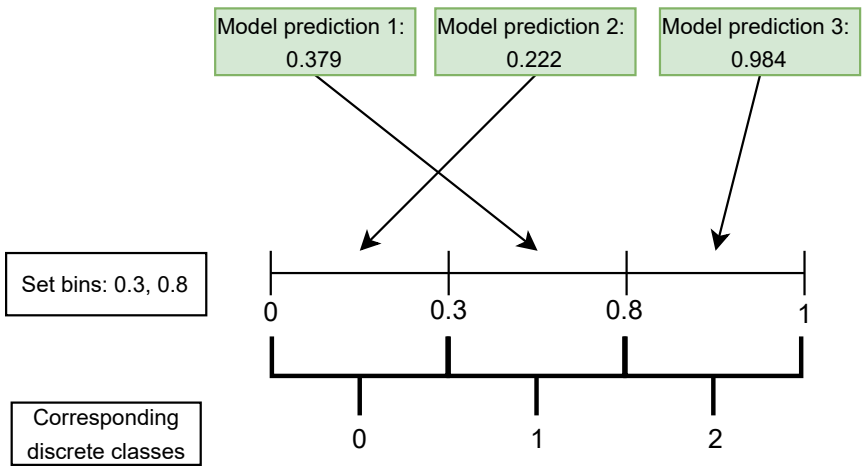


Figure 3.6: Example binning using bins 0.3 and 0.8

An important consideration is that both the binning thresholds and the model parameters influence the final performance. Optimising both at the same time would create a ”chicken and egg”-situation, where the optimal model depended on the bins, and the optimal bins depended on the model predictions. The solution to the ”chicken and egg” situation was to settle on model parameters first, then optimise the bins as described below.

Optimising bin thresholds based on model predictions introduced a certain degree of bias, but this approach was justified by the goal of reducing TA workload while preserving scoring fairness. Due to the imbalance in the dataset, specifically the lower frequency of submissions with scores of 0 or 1 compared to score 2, the optimisation strategy explicitly targeted two metrics: maximising recall for score 0 and maintaining high precision for score 2. In other words, this strategy could be formulated as a cost-minimisation problem, where certain misclassification *cost* more in terms of fairness.

On the one hand, if a true score 0 submission received a predicted score of 2, then the student is not likely to challenge that due to a generous score. On the other hand, if a true score 2 submission received a predicted score of 0, then it is more likely that a student will challenge that score with a manual review from TAs. However, this reasoning is based on domain intuition rather than empirical evidence. Furthermore, this strategy aimed to reduce unnecessary manual checks for the majority score class. In a real-world setting, it is more efficient to filter out the majority class (score 2), provided the model is confident that its score 2 predictions are correct, given the higher frequency of such submissions. To support this, bin thresholds were optimised individually for each model, ensuring that decision boundaries aligned with the strengths of each model. This approach led to fairer, more accurate, and practical scoring.

To find these bins, an objective function is defined below with a cost matrix as well as an example confusion matrix, where the product of the two matrices is performed. The process to find the best bins involved three steps: first, the predicted values of `norm_score` were put into bins. The second step calculated the cost of this `norm_score` allocation in the bins using the dot product of the cost matrix and the produced confusion matrix. The third step updated the bins. Then it went back to the first step, and the loop ended when no lower cost was found. The cost matrix and an example confusion matrix can be seen in Figure 3.7, and the optimisation formula is shown in Objective Function 3.1.

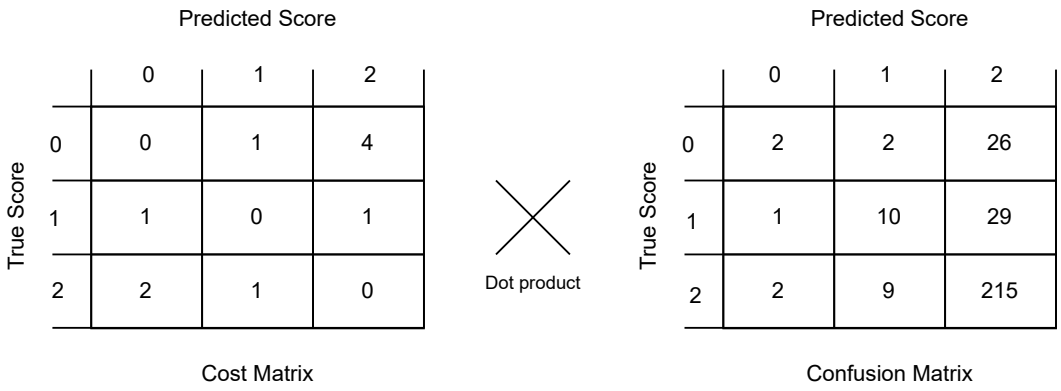


Figure 3.7: Cost calculation example

In this work, the cost matrix is asymmetric, as the cost of misclassifying a student who should have received score 0 but was predicted as score 2 is higher compared to the reverse (a score 2 being predicted as score 0), as earlier described. Conversely, there is no cost of correctly predicting the score, therefore, the diagonal in the cost matrix is 0.

Objective Function 3.1: Definition of the objective function combining recall for score 0 and precision for score 2

The weight matrix W is defined as:

$$W = \begin{pmatrix} 0 & 1 & 4 \\ 1 & 0 & 1 \\ 2 & 1 & 0 \end{pmatrix}$$

Where:

- W_{ij} represents the penalty for predicting class j when the true class is i.
- The matrix is asymmetric because the cost for predicting class 0 when the true class is 2 (i.e., W_{20}) is higher than the cost for predicting class 2 when the true class is 0 (i.e., W_{02}).

The objective of the regression is to minimise the total cost across all data points, represented by the following cost function:

$$C = \sum_{i=0}^{2} \sum_{j=0}^{2} W_{ij} \cdot P_{ij}$$

Where:

- C is the total cost function, which aims to be minimised.
- P_{ij} is the count of instances where the true class is i and the predicted class is j.

The bin optimisation described above was integrated into the evaluation process by using cross-validation to identify threshold values that best aligned with the model's characteristics. These bins ensured that each model was evaluated in a way that best

matched the practical goal of reducing manual grading workload without sacrificing scoring reliability. The defined cost matrix W is used in this work to find the best bins for each model.

3.8.4 Model Selection

Given that the task was a regression problem, several different models were considered for predicting scores on student submissions. Neural networks and other highly complex models, while powerful, were avoided due to their black-box nature and lack of interpretability. Since it is important in an educational setting to understand and justify how scores are assigned, the focus was placed on models that were, at least theoretically, interpretable. Following previous research [50], linear regression, support vector machine, random forest, and ensemble models were selected. As noted, the regression counterparts were used for the support vector machine and random forest, which are called support vector regression (SVR) and random forest regression (RFR), respectively.

These models allowed for some degree of transparency, for example, through decision paths in a random forest regressor that can be traced back through individual trees, or decision boundaries in the support vector regressor. Interpretability can be discussed both at a global level, where the overall behaviour of the model is understood, and at a local level, where individual predictions can be explained [53]. Global interpretability was addressed through model selection, while local interpretability was supported by saving the underlying data used for each prediction. For every submission, all relevant textual elements, subplot mappings, similarity scores, and LPIPS-based heatmaps were stored. If teaching staff or students felt that a score was incorrect, it would be possible to backtrack and inspect the values and decisions involved in the comparison process. This design ensures that each score prediction can be reviewed in detail. Thus, these four models were implemented, fine-tuned, and evaluated on the final test set. To ensure fair comparison and reproducibility, all models used a fixed random seed to share the same data splitting and cross-validation setup.

A GridSearch was conducted over a set of hyperparameters for each model to optimise performance. For linear regression, the grid included options for fitting the intercept, as seen in Table 3.7. For SVR, kernel type, regularisation and kernel coefficient, and class weighting were varied to explore different margin configurations and responses to class imbalance, as detailed in Table 3.8. For RFR, parameters such as tree depth, the number of estimators, splitting thresholds, and bootstrapping were tuned to manage model complexity and avoid overfitting, as shown in Table 3.9.

Based on the individual model performance after GridSearch, linear regression was excluded from the ensemble due to poor results (see Section 4.1). The final ensemble combined only SVR and RFR as base regressors. Its hyperparameter grid consisted of the hyperparameters from Tables 3.8 and 3.9.

Table 3.7: Description of parameters used in GridSearch for linear regression

Parameter	Description
`fit_intercept`	Whether to calculate the intercept.

Table 3.8: Description of parameters used in GridSearch for SVR

Parameter	Description
`kernel`	Chooses the function used to project data
`C`	Balances model complexity and training error
`gamma`	Controls influence of individual points

Table 3.9: Description of parameters used in GridSearch for RFR

Parameter	Description
`n_estimators`	Number of trees in the forest
`max_depth`	Maximum depth of each tree
`min_samples_split`	Minimum samples required to split a node
`min_samples_leaf`	Minimum samples required at a leaf node

3.8.5 Feature Selection and Engineering

Due to the high dimensionality of the dataset produced from the data transformation pipeline and the objective of identifying the most relevant features for predicting the student scores, mutual information was selected as the feature selection method. Mutual information quantifies the dependency between each feature and the target variable, effectively capturing both linear and non-linear relationships without assuming any specific model structure [54]. This characteristic makes it particularly well-suited for this context, where model-agnostic and interpretable feature rankings are critical to ensure generalisability across different predictive models.

Moreover, as a filter-based method, mutual information provides computational efficiency compared to wrapper-based approaches since it does not need to train a model per feature comparison, which is especially advantageous when dealing with many candidate features [55]. Using the `mutual_info_regression` function from the `sklearn` library, the most important features could be selected while preserving complex, non-linear dependencies between the features and the target variable (`norm_score`). In essence, the importance score can be seen as a weight assigned to the dependency between the features and the target variable. Since different models were used, the number of features selected varied between them. The number of features k from the most important features is determined through cross-validation.

While feature selection is important, feature engineering is also critical for developing a more accurate model capable of identifying patterns in the data that may not be obvious to humans. In this project, feature engineering was performed by averaging

applicable features, such as continuous similarity scores, to create more informative representations. For example, when a submission contained multiple subplots, a *composite* score was generated that summarised the similarity across all subplots instead of retaining n individual similarity scores. This aggregation simplified the feature space while still capturing the overall similarity between student and reference subplots. Features that indicated missing elements were also implemented.

3.9 Risk Assessment

This project involved several risks related to predicting student scores in an educational context. Key risks include the challenges of using predictions in production, generalisation, limitations in data quality and quantity, and the potential impact of regulatory frameworks on the future of automatic scoring tools.

This automatic scoring tool was developed in parallel with the data available throughout the project. As with any machine learning tool, there is always a risk in ensuring its usefulness, efficiency, and accuracy when exposed to data outside the original collection period. When deploying such tools, it is important to monitor and audit performance. There are many ways for students to write code and still achieve the same visual output. While it is inherently impossible to account for every possible edge case, which can impact accuracy and trust, achieving robust coverage of the most common coding patterns is both reasonable and feasible.

Accuracy is initially limited due to the small amount of available training and evaluation data. However, as more data is collected and assessed for quality, the model can be retrained and continuously improved to perform better on unseen data. The effectiveness of the tool on new data depends largely on the types of plots it contains and how closely those submissions match the structure of the original dataset. Since the tool was developed specifically for certain types of plots and assumes the use of Matplotlib, its ability to generalise is limited. Data drift is also a potential risk. For example, if the course were to change its assignment format, such as focusing more on open-ended tasks or using a different plotting library like Plotly, the tool would no longer provide the intended value in its current form.

Finally, this tool may be classified as *high risk* according to the upcoming EU AI Act [56]. This Act classifies certain AI/ML systems in education as high-risk because they can affect people's rights and future opportunities. Annex III, Section 3 of the Act includes systems used to evaluate students' work, alike this tool for scoring plots. These systems are considered high risk because mistakes or unfair assessments could harm students' learning or future chances [57]. Article 6 explains the rules for deciding which AI systems are high-risk. To make sure these systems are safe and fair, they must follow strict rules, like being transparent, having human oversight, and being carefully tested for risks [58]. While the Act is already in force, not all of its provisions are yet applicable, and relevant Commission implementation guidelines are still pending. Thus, compliance with the Act will be an important consideration for future iterations of this tool as AI/ML regulation continues to evolve.

3.10 Ethical Considerations

This section outlines the ethical principles guiding the use of student data, transparency of machine learning decisions, and the role of human oversight in automated assessment.

This project included work with anonymised student submissions consisting of plot-generating Python code. This work's supervisor, who is the examiner of the targeted course, allowed students to explicitly provide consent by including the variable `RESEARCH_CONSENT` with the assigned value *True* in their submission. Only submissions with this declaration were available in the development and testing of this tool. Furthermore, the project relied on the principle of Public Access to Information and Secrecy, which is fundamental to how Swedish institutions operate in the public sphere [59]. This principle also applies to universities, meaning submitted assignments are classified as public information. In this context, a submission refers to answers provided by students to questions issued by the examiner. Students originally submitted their submissions as Jupyter Notebooks (.ipynb) to the examiner, which were then converted into Python scripts (.py) for use in the dataset. As a result, the original Jupyter Notebook submissions were not accessible for this project.

The Swedish Higher Education Authority (Universitetskanslerämbetet, UKÄ) states that a student's submission becomes public information as soon as the university receives it [60, p.101]. UKÄ also highlighted that fully automatic grading of examinations is not permitted [60, p.120]. This project complies with that guideline, as it does not involve full examinations or fully automatic scoring/grading. This work focused on specific subtasks within larger assignments, where only certain elements, such as the correctness of generated plots, are evaluated using automation. Other components, including the students' written discussions and interpretations, are assessed manually. In addition, the final course grade is always determined by a human examiner, which is in line with UKÄ regulations.

Another important ethical aspect concerns transparency in the scoring of the plotting tasks. The goal was to implement a tool that, for each rubric, reports the elements detected in the student's plot. This ensures that any mistakes made by the tool can be reviewed by humans, preventing the system from functioning as a *black box*. Machine learning models must be explainable so that people can understand how decisions are made, identify potential errors, and take responsibility for automated outcomes. Transparency also includes clarity about the source of the data, how it is processed, how predictions are made, and which features are important when scoring. By making this information clear and accessible, users can better understand and trust the system. Finally, addressing the problem of bias is essential, as models trained on biased or inconsistent data can produce unfair results. Bias in this context means inconsistencies in scoring and how different TAs weigh different rubrics differently.

4

Results

The results for the models trained to predict the continuous feature `norm_score`, which was later mapped to discrete score classes of 0, 0.5, or 1, are presented here. The models include linear regression, support vector regression (SVR), random forest regression (RFR), and an ensemble model. During training, each model used the optimal bins that resulted in the lowest cost according to the Objective Function 3.1. These optimal bins were determined through cross-validation.

Results are shown for all models in order of increasing complexity, and include their optimal hyperparameters, binning strategies, and evaluation metrics on the test set. Evaluation metrics include both standard and weighted due to the multiclass nature of the problem. As a consequence of using the weighted metrics, recall and accuracy share the same value [61]. Finally, the selected features with importance scores are listed. As they are model-independent, they will remain constant. The entire list of features and their importance score can be seen in Appendix C.

4.1 Linear Regression Results

The first model, used as a simple dummy model, was a linear regression model trained on $k = 10$ features selected through cross-validation. While not a dummy model in the strict sense, it serves as a naive benchmark due to its simplicity and popularity in modelling tasks. Minor hyperparameter tuning was performed, as shown in Table 4.1. Using the best bins, it showed that the linear regression is deemed unusable in this context and is not more than a dummy model. The bins used were [0.45, 0.81], but using other bins would have led to the model predicting (almost entirely) score 0 or 1. It was therefore excluded from the ensemble.

The model was evaluated on the unseen test set following development on the validation set, achieving an accuracy of 65.88%. This high accuracy should not be taken as fact since the linear regression model predicts score 2 on almost all instances in the test set, as seen in the confusion matrix in Figure 4.1. This means it performs poorly at predicting true score 0, and does not predict a single true score 1 correctly.

Table 4.1: Best hyperparameter for Linear Regression

Parameter	**Value**
fit_intercept	True

Table 4.2: Selected features and importance scores for linear regression

Feature	Importance score
lpips_min	0.1277
composite_ylabel_sim	0.1217
max_score	0.0903
lpips_max	0.0875
average_lpips	0.0785
composite_title_sim	0.0734
lpips_mean	0.0713
y_unit_present	0.0666
composite_xlabel_sim	0.0605
title_present	0.0547

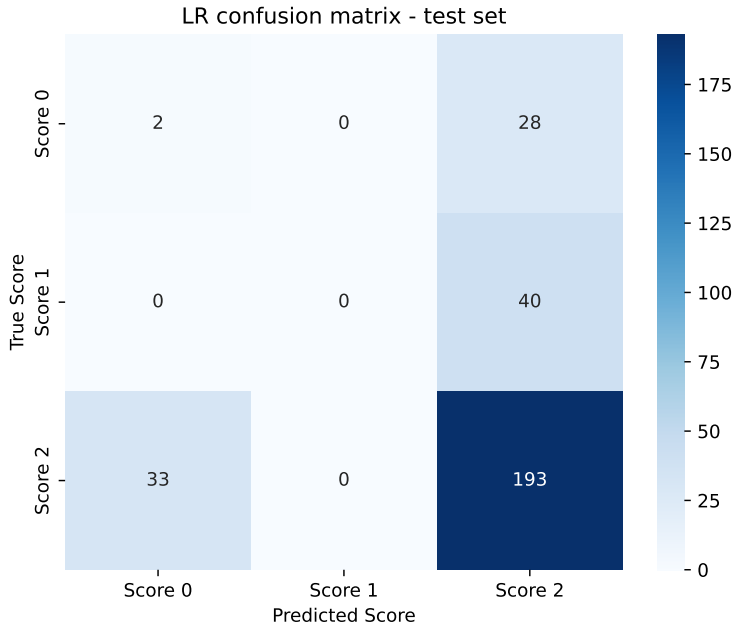


Figure 4.1: Confusion matrix for linear regression on test set

Table 4.3: Per-score and weighted metrics for linear regression on test set

Score	Precision	Recall	F1-Score	Support
Score 0	0.06	0.07	0.06	30
Score 1	0.00	0.00	0.00	40
Score 2	0.74	0.85	0.79	226
Weighted Avg	0.57	0.66	0.61	296
Accuracy		0.6588		

4.2 Support Vector Regression Results

The second model is the SVR model. It was trained using $k = 11$ and the optimal hyperparameters found were a regularisation strength $C = 0.15$ and the use of an RBF kernel with the Gamma (γ) parameter set to `scale`. The best bins with the lowest cost were [0.35, 0.86]. The hyperparameters are summarised in Table 4.4, and the selected features and their corresponding importance scores are shown in Table 4.5. Table 4.6 reports the accuracy and weighted precision, recall and F1-score. SVR achieved an accuracy of 77.36% on the test set.

Table 4.4: Optimal hyperparameters for SVR

Parameter	**Value**
Regularization strength (C)	0.15
Kernel	rbf
Gamma (γ)	scale

Table 4.5: Selected features and importance scores for SVR

Feature	**Importance Score**
`lpips_min`	0.1277
`composite_ylabel_sim`	0.1217
`max_score`	0.0903
`lpips_max`	0.0875
`average_lpips`	0.0785
`composite_title_sim`	0.0734
`lpips_mean`	0.0713
`y_unit_present`	0.0666
`composite_xlabel_sim`	0.0605
`title_present`	0.0547
`xlabel_present`	0.0470

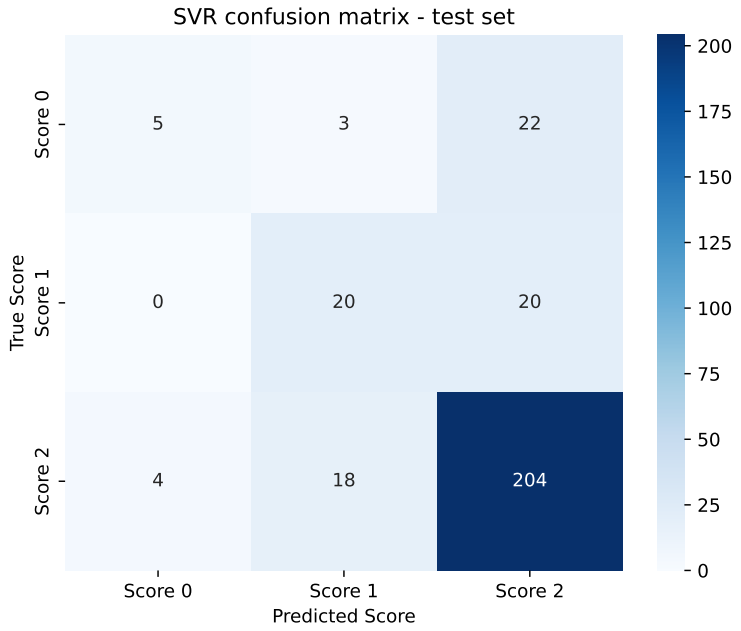


Figure 4.2: Confusion matrix for SVR on test set

The SVR outperformed linear regression, likely due to its ability to model non-linear relationships. As shown in Table 4.6, the SVR model correctly predicted the majority of the score 2 instances, captured a portion of the score 1 instances, and identified a small fraction of score 0 cases. While its overall performance was stronger than that of linear regression, the model's poor recall for score 0 limits its usefulness in reducing the scoring workload for TAs. Furthermore, the SVR incorrectly predicted many of the score 0 data points as score 2, as illustrated in the confusion matrix in Figure 4.2.

Table 4.6: Per-score and weighted metrics for SVR on test set

Score	**Precision**	**Recall**	**F1-Score**	**Support**
Score 0	0.56	0.17	0.26	30
Score 1	0.49	0.50	0.49	40
Score 2	0.83	0.90	0.86	226
Weighted Avg	0.76	0.77	0.75	296
Accuracy		0.7736		

4.3 Random Forest Regressor Results

The RFR served as the third model, and used $k = 12$ selected features for training. The optimal hyperparameters found were a maximum tree depth of 5, a minimum of 4 samples per leaf, a minimum of 2 samples per split, and 100 estimators and bootstrapping set to `True`. Table 4.7 showed the optimal hyperparameters, and Table 4.8 listed the selected features along with their corresponding importance scores. The best bins were [0.45, 0.76].

The model was evaluated on the unseen test set after development on the validation set, achieving a test set accuracy of 78.38%. The confusion matrix in Figure 4.3 shows the model's performance in predicting the three scores on the test set. The summary of classification metrics can be observed in Table 4.9.

Table 4.7: Optimal hyperparameters for RFR

Parameter	**Value**
Maximum depth	5
Minimum samples per leaf	2
Minimum samples per split	5
Number of estimators	200
Bootstrapping	False
Maximum features	sqrt

Table 4.8: Selected features and importance scores for RFR

Feature	**Importance Score**
`lpips_min`	0.1277
`composite_ylabel_sim`	0.1217
`max_score`	0.0903
`lpips_max`	0.0875
`average_lpips`	0.0785
`composite_title_sim`	0.0734
`lpips_mean`	0.0713
`y_unit_present`	0.0666
`composite_xlabel_sim`	0.0605
`title_present`	0.0547
`xlabel_present`	0.0470
`axis1_label_composite`	0.0465

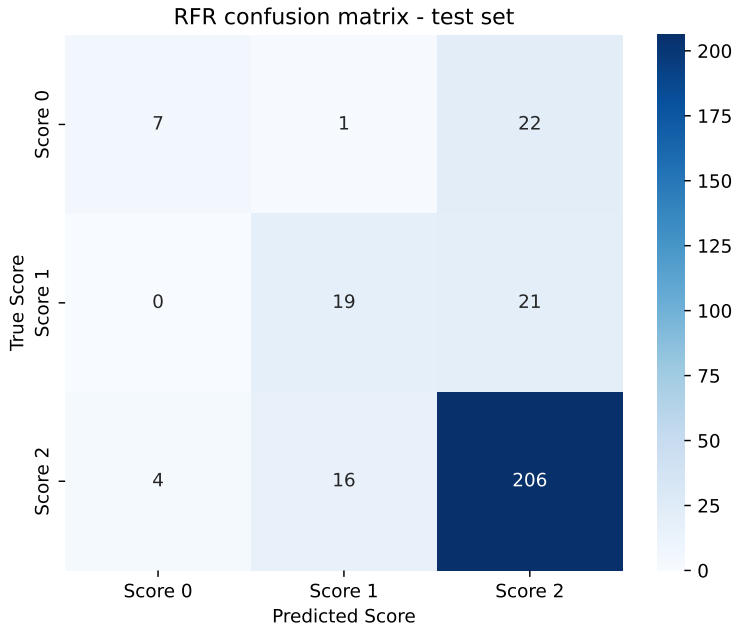


Figure 4.3: Confusion matrix for RFR on test set

For the RFR, a slight increase in performance could be observed. Analysing the per-score breakdown in Table 4.9 revealed, compared to SVR, a slightly higher recall on score 0 but at a cost of a bit lower score 2 precision.

Table 4.9: Per-score and weighted metrics for RFR on test set

Score	**Precision**	**Recall**	**F1-Score**	**Support**
Score 0	0.64	0.23	0.34	30
Score 1	0.53	0.47	0.50	40
Score 2	0.83	0.91	0.87	226
Weighted Avg	0.77	0.78	0.76	296
Accuracy		0.7838		

4.4 Ensemble Model Results

The ensemble model, combining SVR and RFR due to their acceptable performance, was trained using $k = 10$ selected features. Linear regression was left out due to its erroneous performance when evaluated on its own. The best hyperparameters can be seen in Table 4.10, while Table 4.11 presents the selected features and their corresponding importance scores. The best bins were [0.45,0.91].

The model was evaluated on the unseen test set after development on the validation set, achieving an accuracy of 66.89%. Figure 4.4 shows the confusion matrix on the test set. The summary of classification metrics can be observed in Table 4.12.

Table 4.10: Best hyperparameters for each model in the ensemble

Model	Parameter	Value
SVR	Regularization strength (C)	0.1
	Epsilon (ϵ)	0.01
	Kernel	linear
	Gamma (γ)	scale
RFR	Maximum depth	10
	Minimum samples per leaf	2
	Minimum samples per split	5
	Number of estimators	50
	Maximum features	sqrt

Table 4.11: Selected features and importance scores for the ensemble model

Feature	Importance Score
`lpips_min`	0.1277
`composite_ylabel_sim`	0.1217
`max_score`	0.0903
`lpips_max`	0.0875
`average_lpips`	0.0785
`composite_title_sim`	0.0734
`lpips_mean`	0.0713
`y_unit_present`	0.0666
`composite_xlabel_sim`	0.0605
`title_present`	0.0547

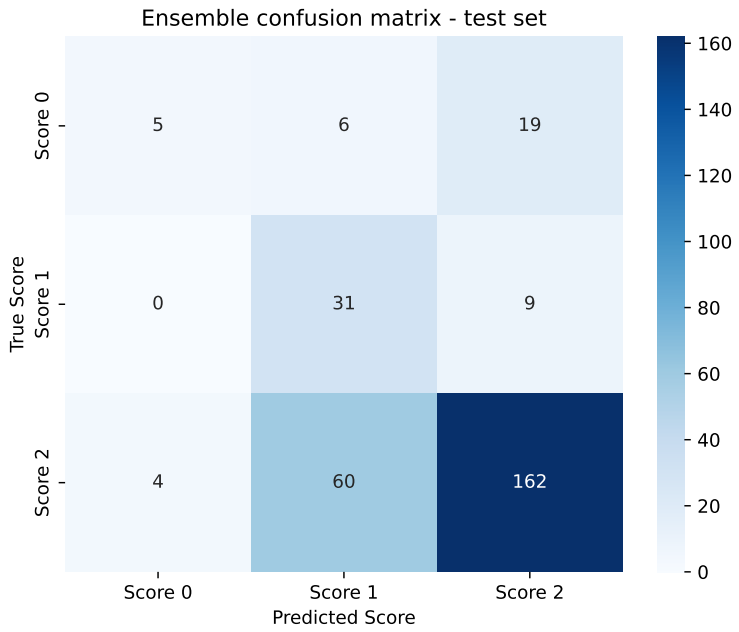


Figure 4.4: Confusion matrix for the ensemble model on test set

The confusion matrix in Figure 4.4 shows the performance of the ensemble model after converting its continuous regression outputs into discrete score classes. The model performs reasonably well on score 2, correctly classifying most instances in that category. However, its performance is noticeably weaker on score 0, with nearly two-thirds of the true score 0 samples misclassified as score 2. Additionally, while the model demonstrated good recall for score 1, its precision is low, indicating that many of the samples predicted as score 1 may not truly belong to that score class. These misclassification patterns are examined in greater detail in Section 5.1.

Table 4.12: Per-score and weighted metrics for ensemble on test set

Score	**Precision**	**Recall**	**F1-Score**	**Support**
Score 0	0.56	0.17	0.26	30
Score 1	0.32	0.78	0.45	40
Score 2	0.85	0.72	0.78	226
Weighted Avg	0.75	0.67	0.68	296
Accuracy		0.6689		

4.5 Result Summary

Table 4.13 provides a summary of the test set performance metrics for all evaluated models, including linear regression, SVR, RFR, and the ensemble model. The results show each model's classification accuracy after binning the predicted `norm_score`, weighted precision, weighted recall, and weighted F1-score. As a consequence of using the weighted metrics, recall and accuracy share the same value [61].

Table 4.13: Test set performance metrics and bins for all models. All metrics except accuracy are weighted

Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**	**Bins**
RFR	0.78	0.76	0.78	0.76	[0.45, 0.76]
SVR	0.77	0.76	0.77	0.75	[0.35, 0.86]
Ensemble model	0.67	0.75	0.67	0.68	[0.45, 0.91]
Linear regression	0.66	0.57	0.66	0.61	[0.45, 0.81]

The results showed that the RFR achieved the highest accuracy among the tested models, with 78%. It also reached a weighted F1-score of 76%, indicating a good balance between precision and recall. This suggests that the RFR did not significantly favour one score-class over another in the predictions. However, Table 4.13 does not provide a complete picture. Analysing per-score metrics for the RFR in Table 4.12 shows that score 0 had a lacklustre recall of 23% but acceptable precision at 64%. This means that when the model predicted score 0, it was often correct, but it failed to identify most of the true score 0 instances. In other words, the RFR model performed the best of the tested models in terms of the goal to lessen TA workload.

Meanwhile, the extreme case of linear regression predicted score 2 for almost all instances in the test set, as seen in the confusion matrix in Figure 4.1. This indicated that the dataset is not linearly separable and, therefore, the linear regression was ultimately excluded from the ensemble due to poor performance overall.

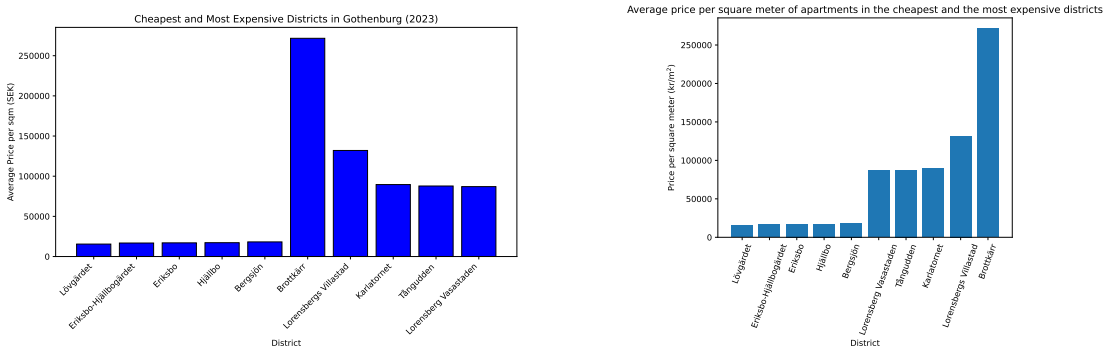
Although the RFR model did not perform as well as initially hoped, it still demonstrated clear potential. Its position as the best-performing model in this implementation can be explained by its robustness during training, and is supported by the literature [50]. However, the model was still affected by noise introduced through erroneous and/or inconsistent human labelling. The following misclassification analysis presents both model misclassifications and annotation misclassifications, highlighting the two sides of the coin when implementing automated scoring tools.

4.6 Misclassification Analysis

Examples of misclassified scores made by the top-performing model, the random forest regressor, are highlighted, along with the annotator's misclassifications identified in the labelled data. Human labellers introduce inconsistencies that could negatively impact model performance and reduce trust in automated scoring systems that rely on human-annotated data. It is important to note that all visual similarity was based on the reconstructed plot images.

4.6.1 Model Misclassifications

Although the weighted metrics suggest that the RFR model performs well overall, certain individual submissions are still misclassified. Below are examples that illustrate specific failure cases. The first example highlights the difficulty of supporting semantic similarity without allowing submissions that present the data in inadequate or misleading ways to receive high scores.



(a) Student solution

(b) Reference solution

Figure 4.5: Comparison between the student and reference solution for VT25 assignment 2 question 9

In Figure 4.5, the student and reference solutions are shown side by side. The

student submission received a human-labelled score of 0, while the RFR predicted a score of 2. One possible explanation for the human-assigned score is the lack of bar ordering in the student plot, which can make the data harder to interpret. As shown below in Table 4.14, the visual similarity received a score of 0.0, which is a perfect similarity. This outcome was influenced by the design of the preprocessing pipeline that reconstructed each plot in a standardised way before visual comparison.

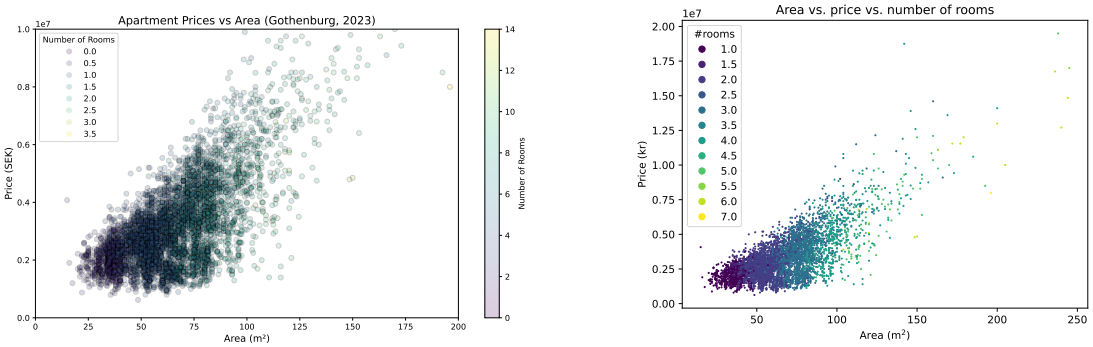
Table 4.14: Similarity scores between the student and reference solution for VT25 assignment 2 question 9

Similarity Score	**Value**
Title similarity	0.524
X-axis label similarity	1.000
Y-axis label similarity	0.584
Composite label similarity	0.703
LPIPS (visual similarity)	0.000

The pipeline's reconstruction process automatically reordered the bars in the student plot in ascending order, as required by the assignment. But the original plot did not order it in ascending order, which meant that the pipeline fixed the students' mistake. Although this standardisation was intended to support semantic similarity rather than exact visual matching, it unintentionally concealed important differences in structure and clarity. In this case, the automatic reordering likely contributed to the model assigning a higher score than the submission would have received if scored in its original, unprocessed form.

This example highlights a key challenge in designing systems that support semantic similarity. On one hand, the pipeline should avoid penalising submissions that are meaningfully similar despite minor visual differences. On the other hand, it should not obscure aspects that may be pedagogically important to humans. This creates a fundamental asymmetry between the model's processing of visual data and the way TAs interpret original student submissions.

In a second example, shown in Figure 4.6, a similar misalignment between human and model judgement can be observed. The student submission received a human labelled score of 0, but the RFR predicted a score of 2. Unlike the previous case involving bar ordering, this example involves a scatter plot with differences in axis labels, title, legend formatting, and data representation. The original student plot lacks several expected elements and is visually less clear than the reference solution.



(a) Student solution

(b) Reference solution

Figure 4.6: Comparison between the student and reference solution for VT25 assignment 2 question 6

The student used incorrect scaling, which hid parts of the dataset in the plot. Some data points in the dataset were also wrong, but these did not appear in the visualisation. The legend was inaccurate and does not reflect all the relevant categories in the data. The colour bar on the right was also incorrect, but this is not currently evaluated by the pipeline and therefore did not affect the score. Since the evaluation only checks for the presence of a legend, without verifying its accuracy, these issues were not detected. In addition, the pipeline reconstructed the plot to align it with the reference, which masked the effects of the incorrect scaling. While LPIPS did pick up on some of the incorrect visual data representation as seen in Table 4.15, the deviation in LPIPS and the textual elements was not large enough to result in a lower score. Consequently, the submission was automatically scored higher than it should have been.

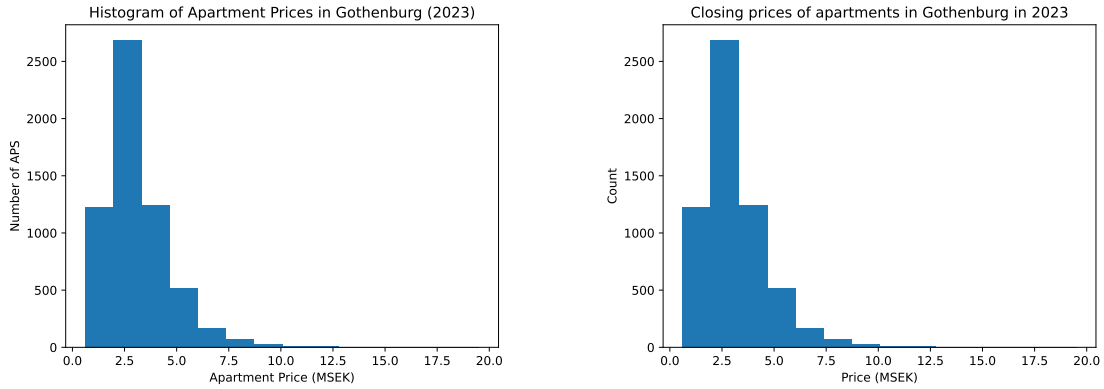
Table 4.15: Similarity scores between the student and reference solution for VT25 Assignment 2 Question 6

Similarity Score	**Value**
Title similarity	0.554
X-axis label similarity	0.691
Y-axis label similarity	0.639
Composite label similarity	0.628
LPIPS (visual similarity)	0.206

4.6.2 Annotator Misclassification

Below are two examples of student submissions that seem to have received erroneous or inconsistent labels from human annotators. The similarity scores were high, and the visual resemblance to the reference solution was strong, yet the assigned score by the human annotator was 0. The student solutions checked all the required rubrics and had high similarity scores on textual and visual elements.

The first student submission example received a human-assigned score of 0, while the RFR predicted a score of 2. The student submission is shown on the left, and the reference solution is shown on the right in Figure 4.7. The corresponding similarity scores are presented in Table 4.16.



(a) Student solution

(b) Reference solution

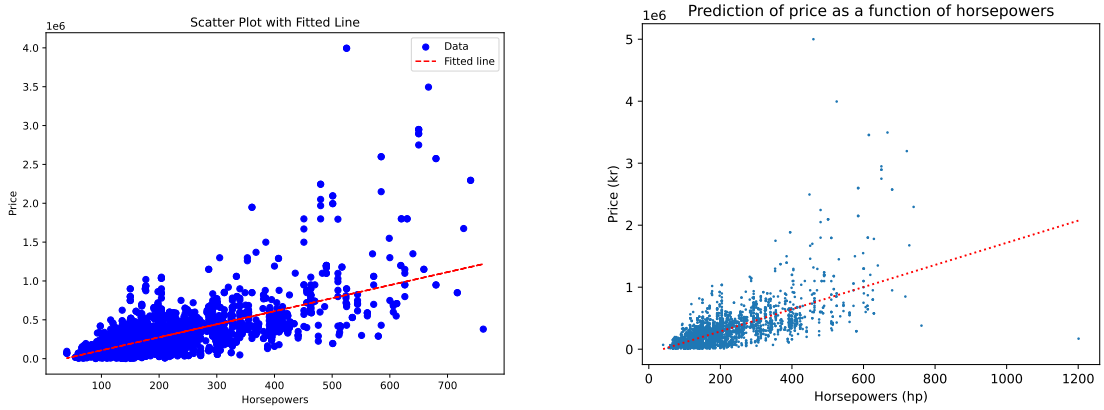
Figure 4.7: Comparison between the student and reference solution for VT25 assignment 2 question 5

The metrics for textual and visual similarity between the student and reference solutions are shown in Table 4.16. Among them, LPIPS yielded the highest similarity score, while the title, x-axis label, and y-axis label received lower scores of 0.742, 0.719, and 0.251, respectively. The textual elements generally received moderate to low similarity scores, with the y-axis label scoring particularly poorly. Although the abbreviation in the y-axis label may hinder interpretability, an assigned score of 0 by the TA appears overly harsh and may not be fully justified.

Table 4.16: Similarity scores between the student and reference solution for VT25 assignment 2 question 5

Similarity Score	**Value**
Title similarity	0.742
X-axis label similarity	0.719
Y-axis label similarity	0.251
Composite label similarity	0.571
LPIPS (visual similarity)	0.000

In the second example, the TAs assigned scores that were lower than the rubric suggested. By contrast to the previous examples, the following illustrates an example where a human-labelled score 2 should be lower.



(a) Student solution

(b) Reference solution

Figure 4.8: Comparison between the student and reference solution for VT25 assignment 2 question 6

In Figure 4.8, the student solution appears visually similar to the reference, primarily due to a change in scaling. Although the data points are incorrect, this is difficult to notice, and the altered scaling does not immediately lead to a poor score. However, the axis labels lack units, and the title is not descriptive. This is reflected in Table 4.17, where the similarity scores for the title are low, while visual similarity remains low as well. The reference solution lists all required rubric items, and a score of 2 should only be assigned when all of them are fulfilled. Since one of the rubric items requires both axes to include proper units, this criterion is not fulfilled. In addition, the title is not descriptive, and data representation is not correct, which further supports that a lower score would have been more appropriate.

Table 4.17: Similarity scores between the student and reference solution for VT25 assignment 4 question 5

Similarity Score	**Value**
Title similarity	0.073
X-axis label similarity	0.895
Y-axis label similarity	0.626
Composite label similarity	0.354
LPIPS (visual similarity)	0.299

5

Discussion

This chapter begins by discussing the results of predicting student scores using various machine learning (ML) models. It then explores the challenges of automatically scoring plots, including what the models are trying to score and how human scoring introduces noise and inconsistencies into the data. Potential solutions to mitigate these issues are proposed. Finally, we discuss dataset- and pipeline-related issues and how these impact the performance, reliability, and future applicability of the system.

5.1 Model Performance and Misclassification Patterns

A recurring pattern observed across all models in the confusion matrices is the difficulty in correctly identifying instances labelled with a score of 1. These are frequently misclassified as 2, raising questions about whether some of these examples may be mislabeled. The score of 1 appears to occupy a conceptual *grey area* between incorrect and correct solutions, making it inherently harder for the model to distinguish. This ambiguity likely contributes to the inconsistent performance in this class and highlights the need for clearer score definitions or more consistent labelling practices. To better accommodate this uncertainty and improve model flexibility, the task was reformulated as a regression problem. In addition, a potential fix for this issue is to make use of rule-based filters to *pre-screen* submissions that have a high chance of being a score 0 or score 2 case using the basic rubric criteria. If most requirements are fulfilled, it is likely a score 2. If many are missing, it is likely a score 0. Instances that fall in between, such as those missing one or two key elements, could then be treated as likely score 1 candidates, which may help reduce noise and improve class separation.

The limited dataset size and pronounced class imbalance were a major obstacle throughout the model development. For the random forest regressor, we constrained model complexity by limiting tree depth and enforcing a minimum number of samples per leaf. For the support vector regressor, L2 regularisation was applied via the C parameter to discourage overly complex decision boundaries. While these measures are intended to reduce overfitting, the combination of a small dataset and ambiguous class definitions, especially for score 1, makes generalisation inherently difficult. Under these constraints, some degree of overfitting appeared unavoidable.

The challenge to solve with the regression problem arose from the fact that multiple TAs are involved in scoring the submissions, introducing variability and noise into the labels. In contrast, if a single TA had scored all submissions, the dataset would have had a consistent bias and subjectivity as well as lower variance throughout. In such a case, an ML model could more easily detect and adapt to the consistent patterns present in the data. However, with multiple TAs, inconsistencies between them introduced label noise, making it more difficult for the model to learn stable and reliable patterns.

The results for recall and precision, especially for the 0 and 1 classes, appear suboptimal and fall short of being fully optimised. In Section 3.8.3, we highlight how binning strategies affect model comparability when predicting student scores. While weighted precision and recall appear nearly equal for most models (except linear regression), this balance may be misleading given the specific objectives of the task. Therefore, it was essential to consider per-class performance metrics when implementing the tool to reduce grading workloads.

To align with per-score performance priorities, the objective was to define bin thresholds that minimised the total classification cost. This was motivated by the desire to reduce the workload for TAs when evaluating student submissions. A high penalty for predicting class 2 on true class 0 submissions ensures that low-quality work is reliably identified, enabling TAs to focus their attention on a smaller number of cases.

From a cost perspective, the strategy of high recall of score 0 and precision for score 2 has different associated costs. For example, maximising recall on class 0 would ensure that most failing cases are flagged, allowing TAs to focus their attention on reviewing only these submissions. This reduces the manual burden (due to fewer submissions to look at) while minimising the risk of students receiving an undeserved zero. Conversely, the cost of mistakenly predicting a score 2 on a true class 0 submission is high, as it may go unnoticed and undermine scoring integrity. On the other hand, predicting a score 0 on a true score 2 has a lower cost, since affected students are more likely to contest their score and trigger a manual review by the teaching staff. This asymmetry in error costs suggests that recall on class 0 should be prioritised over overall accuracy in real-world deployment, where a true high accuracy on all score classes is unachievable.

There is, however, a subtle caveat. If the cost of misclassifying true class 0 submissions as score 2 becomes too dominant, for example, due to mislabeled data, the algorithm compensates by raising the threshold for assigning class 2. In doing so, it may prefer to classify uncertain cases as class 1, which carries a lower cost. This behaviour helps the model avoid the most critical errors, though it may also lead to some high-quality submissions being slightly undervalued.

Ultimately, the appropriate balance between precision and recall depends on whether the primary goal is to reduce manual scoring effort or to ensure strict scoring accuracy. Perfect performance across all classes (0, 1, and 2) is unlikely, so the decision to focus on precision, recall, or a combination of the two should be based on the specific use case. It is important to consider the acceptable trade-offs between scoring efficiency

and the risk of misclassifying student submissions when deciding how to apply the model in practice.

5.2 Human Errors in the Data

A closer look at the mistakes made by TAs, as discussed in Section 4.6.2, reveals that human labeling introduces significant noise into the dataset, stemming from both subjective judgments and inconsistencies. Analysing feature importance in Table 4.5 suggests that textual elements play as important a role as visual elements in the scoring process. The composite y-label similarity is among the top two features. This aligns with the observed behaviour of some TAs who appear to prioritise textual phrasing heavily. However, this emphasis can lead to unintuitive penalisation, as observed in Figure 4.7 where minor variations in title-wording and a debatable y-axis label resulted in a disproportionately lower score. The model learns to replicate this sensitivity, raising questions about whether the goal is to mimic TA behaviour or evaluate the submission's quality and correctness using *objective* metrics. This, in turn, invites a broader reflection on what aspects should define rubric criteria in such settings.

A further complication is that humans are naturally good at noticing small details in a plot and deciding whether they matter a lot or not at all. Humans are quick to tell whether a slight design issue affects the overall meaning of the plot or can be ignored. They make these kinds of judgment calls based on context and experience. However, ML models often treat all differences as equally important unless they are trained on a large and carefully labelled dataset. With only a few examples, it's hard for ML to learn which mistakes are critical and which are minor, making it difficult to replicate the flexibility and nuance of human scoring.

These nuances do not mean that humans are inherently good at scoring plots. The TAs who scored the provided dataset introduced noticeable inconsistencies. For the assignments in VT25, the scoring rubrics for plots were thoroughly revised in collaboration with our supervisor, as the previous versions were vague and left substantial room for interpretation. Even with clearer guidelines in place, human judgment remains rarely fully objective, as observed when reviewing the data. Differences in interpretation, unconscious bias, or simple oversight can lead to labelling inconsistencies that reduce the overall quality of the dataset. In some cases, plots were labelled as failed despite no visible issues, suggesting either annotation mistakes or hidden expectations not reflected in the rubric. Specifically, examining all test set class 0 data points that were mislabeled as class 2 by the RFR, 18 out of 22 instances were inconsistently labelled or outright had erroneous labels. In other words, we could not justify a score 0 when checking the corresponding rubrics. This level of noise hinders effective training and raises the question of how many more such inconsistencies might exist in the training data.

One potential way to improve the quality of the dataset would be to have multiple TAs score the same plot. If their evaluations align, the sample could be included in the dataset. Otherwise, it would be discarded to avoid introducing noise. While

having multiple TAs score the same plot could improve consistency, it is rarely feasible due to time and resource limitations. A more practical solution in real scoring settings could include structured calibration sessions held when students begin submitting their solutions. This can be complemented by regular inter-rater reliability checks on a small sample of plots and automated feedback tools that assist TAs in making more consistent judgments. These interventions require less overhead while still improving label quality.

5.3 Errors in the Students' Code

Errors in the dataset do not just come from the TAs, students also introduce issues that affect scoring. Some of these stem from the design of the pipeline itself, which executes student code directly. This means that unintended or careless code can have broader consequences. For example, students may accidentally overwrite parts of the dataset during execution, breaking the evaluation process and making it impossible to score their solution properly. While safeguards are in place to prevent such incidents, students are often creative in unexpected ways, so extra caution is still needed.

While not exclusive to this project, it is worth noting that the data transformation pipeline's reliance on direct code execution also leads to compatibility issues. It runs all imported libraries as-is, but since many libraries solve similar problems in different ways, the environment would need to support a wide range of them to ensure consistent execution, something that's neither practical nor secure. A common issue is the use of unsupported or unexpected libraries. Even if a submission would produce a correct and well-designed plot, it cannot be executed if it depends on libraries not available in the environment. This results in the pipeline failing to extract all the data necessary to make a correct prediction of the score.

This creates a discrepancy between human and automated evaluation. It also raises the question of whether such examples should be excluded from an already small dataset. On one hand, these submissions provide useful training data, signalling to the model that non-executable code should result in a failing score. On the other hand, they introduce label noise, as the model cannot assess what the TA saw, and the *correct* label may not reflect what the model has access to. Establishing clear coding standards and defining a set of supported libraries would reduce technical failures and ensure fair and consistent scoring. In the future, adopting a more standardised coding structure will help make automated evaluation smoother, more reliable, and more aligned with human judgment.

5.4 Structural Problems in the Data

As noted in Section 5.1, the ambiguity in score 1 likely stems from deeper structural issues in how the dataset was labelled. In this section, we expand on those issues by examining limitations in the dataset's design, including inconsistent scoring scales and imbalanced class distributions. These structural flaws reduce the interpretability of labels and the model's ability to learn meaningful patterns, ultimately limiting

performance.

The issue of the inconsistent scoring scales creates a problem for models to have a clear idea of the relationships between the input variables and the target variable. While most assignments use a 3-point scale (0, 1, 2), some assignments use a binary scale (0 or 1). These inconsistencies distort the weighting of different components and can cause unpredictable behaviour during model training, especially when predicting the final score. Aligning the scoring system across all rubrics would lead to more stable and interpretable predictions. Overall, these limitations highlight the need for better-designed annotation schemes, not just more data. High-quality, well-structured labels are equally important for building robust scoring models. However, improving annotation quality often requires additional effort from teaching staff, so any enhancements must balance label quality with the workload.

Another dataset-related challenge lies in the imbalanced distribution of class scores: instances labelled with 0 or 1 occur far less frequently than those labelled with 2, as seen in Figure 3.1. This skew in the data makes it particularly difficult for the model to learn meaningful patterns for the underrepresented classes, often resulting in biased predictions toward the dominant class. To address this issue, we experimented with SMOTE[1], a common strategy for balancing datasets by generating synthetic examples of minority classes. However, in our case, applying SMOTE did not lead to any noticeable improvement in model performance. This could be because several instances of the already small class 0 appear to be mislabeled as class 2. Such label noise not only reduces the effective size of classes 0 and 1 but also confuses the model during training. In this context, oversampling techniques like SMOTE may become ineffective or even counterproductive, as they risk generating synthetic data based on incorrect labels, further distorting the models. Additionally, we observed signs of overfitting when applying SMOTE. The model tended to memorise the synthetic examples rather than learning generalizable patterns, which led to poorer performance on unseen data. This further supports the idea that noisy and sparsely represented data can hinder the effectiveness of oversampling strategies. Ultimately, SMOTE not only failed to improve results but may have exaggerated the problem by reinforcing random patterns in the training data.

5.5 Balancing Human Judgment and Objectivity in Automated Scoring

Despite the challenges posed by the dataset, including its limited size, imbalances in class distribution, and inherent structural issues, the random forest regressor demonstrated a noteworthy performance. These outcomes highlight the potential for further improvement with a larger and more well-structured dataset with scores that are provided per rubric, which could enhance both the model's robustness and its overall predictive capabilities. The predictive capability of a tool like this is dependent on human-labelled data, and that raises an important question: Should the goal of an automated scoring model be to replicate human decisions or to aim

[1]Synthetic Minority Over-sampling Technique

for a more objective and consistent standard? Each direction has its trade-offs, and finding the right balance is critical for building a fair and reliable scoring system.

On the one hand, a model designed to mimic TA scoring would ensure consistency with historical assessments. Since students expect their solutions to be evaluated similarly to how a human TA would assess them, aligning the model with past scoring decisions provides continuity. This approach also means that any subjectivity, inconsistencies, or biases present in human scoring will be embedded into the model, potentially reinforcing errors rather than eliminating them. On the other hand, striving for full objectivity is theoretically ideal but practically challenging. The dataset itself originates from humans, meaning subjectivity is already embedded within it. Even if a model could be trained to follow strict evaluation criteria, certain aspects of scoring, especially those involving creativity, interpretation, or problem-solving approaches, may still require human judgment. As a result, a fully objective scoring model may not be entirely feasible.

The broader challenge in automatically scoring plots is figuring out what we are trying to score. Are we scoring the code that made the plot, the plot itself, or how well the plot explains the data? This matters a lot because it affects how to design these systems and what kind of ML model we can build. Right now, scoring policies often mix different things, for example, checking if a plot looks right, without thinking about whether it communicates the data well. That makes it hard to create ML models that can score clearly and consistently. If we do not clearly define what makes a plot *good*, the model does not know what to look for. As a consequence, scoring plots automatically is not just a technical task, it is also about deciding what is important in a plot and how to turn those values into something a computer can understand.

A potential compromise is a hybrid approach, which is training the model on TA assessments while implementing safeguards to minimise inconsistencies. The model could move toward a more consistent evaluation standard without completely discarding human judgment by filtering out highly subjective cases and relying on high-agreement scoring samples. Additionally, incorporating confidence estimation techniques could help flag uncertain cases for human review. The current implementation indeed makes mistakes, but leveraging the possibility that students can object to an automated decision to let humans review creates a hybrid system.

Ultimately, the decision depends on the priorities of the scoring system. If consistency with human scoring is essential, then mimicking TAs is reasonable. If fairness and accuracy are the main concerns, reducing reliance on subjective evaluations is preferable. In practice, a balanced approach that integrates both aspects is likely the most effective solution.

6

Conclusion

In this project, we developed machine learning models that automatically score plots in higher data science education and a pipeline to feed the models with applicable features. The pipeline transformed plots submitted by students into features that are machine learning compatible, which provided data about visual and textual similarity to the reference solution. This data was used to train and evaluate models found in related fields. By reformulating the classification problem of predicting score classes to a regression problem, we gained flexibility in how predictions could be mapped back to score classes.

Our key findings reveal both promise and limitations. The overall approach demonstrated that automated scoring of plots is feasible, even when working with a small and noisy dataset. Among the tested models, the random forest regressor model stood out by delivering the most robust performance, aligned to reduce teaching assistants' workload. It is not intended to replace humans in the scoring loop, but rather to support them by handling routine assessments, freeing up time to give, for example, individual feedback to students. One of the main challenges identified in this work is the noise and inconsistency introduced in the labelled dataset by the teaching assistants, which affects model training and evaluation. The system would benefit and become better from access to more data that is consistently and accurately labelled.

In conclusion, this project illustrates the feasibility of using machine learning to automatically score plots in higher data science education and sets the pathway for future systems that combine automation with human insight.

6.1 Future Research

Two main directions are outlined for future research. The first is to investigate more complex models while retaining the overall approach used in this project, such as neural networks. The second is to move beyond the data transformation-based pipeline and explore the potential of convolutional neural networks (CNNs) or visual-language models (VLMs).

This project adopted a two-step approach, transforming plot data and feeding features and similarity metrics into traditional machine learning models to predict scores. While this allowed for a certain degree of interpretability and insight into the models'

decisions, it also opens the door to exploring more advanced approaches. Leveraging complex models, such as neural networks, could offer increased robustness and capture more nuanced patterns in the data. However, the biggest challenge will be the interpretability and the *black box* nature of neural networks. Additionally, future research could investigate deeper semantic analysis of textual elements, including unit semantics. In this project, some textual elements were penalised as dissimilar even though they used alternative characters or formats that conveyed the same meaning.

The second potential route is the compatibility of visual-language models (VLMs) or convolutional neural networks (CNNs). The usage of VLMs or CNNs would severely simplify the data pipeline by using image-based models, or plot-specific VLMs such as ChartGemma[43]. More specifically, ChartGemma with prompt engineering within natural language processing could be an interesting route to investigate, due to being pre-trained on plot data as well as plot descriptions. This pre-training allows for image-prompting, essentially asking the VLM to identify trends, textual elements and overall correctness of a plot. A proposed pipeline can be seen in Appendix B. These image-based approaches would also make the scoring of plots independent of the programming language. Therefore, it could act as a double-edged sword in handling the two kinds of submissions, the ones with a reference solution available and the ones without a reference solution available.

Bibliography

[1] M. Messer, N. C. Brown, M. Kölling, and M. Shi, "Automated grading and feedback tools for programming education: A systematic review," *ACM Transactions on Computing Education*, vol. 24, no. 1, pp. 1–43, 2024.

[2] M. Hull, V. Pednekar, H. Murray, *et al.*, "Visgrader: Automatic grading of d3 visualizations," *IEEE Transactions on Visualization and Computer Graphics*, 2023.

[3] M. Karppa, *Visual element more prioritised than textual elements*, Personal communication, Mar. 2025.

[4] F. Z. Ricci, C. M. Medina, and M. Dogucu, "Designing and implementing an automated grading workflow for providing personalized feedback to open-ended data science assignments," *Technology Innovations in Statistics Education*, vol. 15, no. 1, 2024.

[5] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using Siamese BERT-networks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, K. Inui, J. Jiang, V. Ng, and X. Wan, Eds., Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 3982–3992. DOI: 10.18653/v1/D19-1410. [Online]. Available: https://aclanthology.org/D19-1410/.

[6] Oxford Learner's Dictionaries, *Grade*, Accessed: 2025-04-08, 2025. [Online]. Available: https://www.oxfordlearnersdictionaries.com/definition/english/grade_1?q=grade.

[7] Oxford Learner's Dictionaries, *Score*, Accessed: 2025-04-08, 2025. [Online]. Available: https://www.oxfordlearnersdictionaries.com/definition/english/score_1.

[8] Oxford Learner's Dictionaries, *Assessment*, Accessed: 2025-04-08, 2025. [Online]. Available: https://www.oxfordlearnersdictionaries.com/definition/english/assessment?q=assessment.

[9] S. D. A. Bujang, A. Selamat, R. Ibrahim, *et al.*, "Multiclass prediction model for student grade prediction using machine learning," *Ieee Access*, vol. 9, pp. 95 608–95 621, 2021.

[10] Y. T. Badal and R. K. Sungkur, "Predictive modelling and analytics of students' grades using machine learning algorithms," *Education and information technologies*, vol. 28, no. 3, pp. 3027–3057, 2023.

[11] M. Hooda, C. Rana, O. Dahiya, A. Rizwan, and M. S. Hossain, "Artificial intelligence for assessment and feedback to enhance student success in higher

education," *Mathematical Problems in Engineering*, vol. 2022, no. 1, p. 5 215 722, 2022.

[12] G. A. Pradipta, R. Wardoyo, A. Musdholifah, I. N. H. Sanjaya, and M. Ismail, "Smote for handling imbalanced data problem: A review," in *2021 sixth international conference on informatics and computing (ICIC)*, IEEE, 2021, pp. 1–8.

[13] J. M. Vitale, K. Lai, and M. C. Linn, "Taking advantage of automated assessment of student-constructed graphs in science," *Journal of Research in Science Teaching*, vol. 52, no. 10, pp. 1426–1450, 2015.

[14] Z. Yang, "Machine grading of charts based on formal specifications," M.S. thesis, Rice University, 2021.

[15] W. J. Popham, "What's wrong-and what's right-with rubrics," *Educational leadership*, vol. 55, pp. 72–75, 1997.

[16] A. Angra and S. M. Gardner, "The graph rubric: Development of a teaching, learning, and research tool," *CBE—Life Sciences Education*, vol. 17, no. 4, ar65, 2018.

[17] K. Davila, S. Setlur, D. Doermann, B. U. Kota, and V. Govindaraju, "Chart mining: A survey of methods for automated chart analysis," *IEEE transactions on pattern analysis and machine intelligence*, vol. 43, no. 11, pp. 3799–3819, 2020.

[18] K. Shahira, P. Joshi, and A. Lijiya, "Data extraction and question answering on chart images towards accessibility and data interpretation," *IEEE Open Journal of the Computer Society*, vol. 4, pp. 314–325, 2023.

[19] N. Methani, P. Ganguly, M. M. Khapra, and P. Kumar, "Plotqa: Reasoning over scientific plots," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2020, pp. 1527–1536.

[20] K. Davila, B. U. Kota, S. Setlur, *et al.*, "Icdar 2019 competition on harvesting raw tables from infographics (chart-infographics)," in *2019 International Conference on Document Analysis and Recognition (ICDAR)*, IEEE, 2019, pp. 1594–1599.

[21] National Center for Biotechnology Information, *Pubmed central (pmc)*, Accessed: 2024-11-27, 2024. [Online]. Available: `https://www.ncbi.nlm.nih.gov/pmc/`.

[22] J. Mathew *et al.*, "A survey on object detection from scientific plots," in *2021 5th International Conference on Computer, Communication and Signal Processing (ICCCSP)*, IEEE, 2021, pp. 94–99.

[23] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *Proceedings of Workshop at ICLR*, vol. 2013, Jan. 2013.

[24] J. Pennington, R. Socher, and C. Manning, "GloVe: Global vectors for word representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, A. Moschitti, B. Pang, and W. Daelemans, Eds., Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1532–1543. DOI: `10.3115/v1/D14-1162`. [Online]. Available: `https://aclanthology.org/D14-1162/`.

[25] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need," in *Advances in neural information processing systems*, vol. 30, 2017.

[26] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, J. Burstein, C. Doran, and T. Solorio, Eds., Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423. [Online]. Available: https://aclanthology.org/N19-1423/.

[27] A. Dosovitskiy, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[28] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[29] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. [Online]. Available: https://www.deeplearningbook.org/.

[30] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 586–595.

[31] O. Moutik, H. Sekkat, S. Tigani, *et al.*, "Convolutional neural networks or vision transformers: Who will win the race for action recognitions in visual data?" *Sensors*, vol. 23, no. 2, p. 734, 2023.

[32] A. Radford, J. W. Kim, C. Hallacy, *et al.*, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*, PMLR, 2021, pp. 8748–8763.

[33] C. Rudin, "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead," *Nature Machine Intelligence*, vol. 1, no. 5, pp. 206–215, 2019. DOI: 10.1038/s42256-019-0048-x.

[34] R. Caruana and A. Niculescu-Mizil, "An empirical comparison of supervised learning algorithms," in *Proceedings of the 23rd International Conference on Machine Learning*, ser. ICML '06, Pittsburgh, Pennsylvania, USA: Association for Computing Machinery, 2006, pp. 161–168, ISBN: 1595933832. DOI: 10.1145/1143844.1143865. [Online]. Available: https://doi.org/10.1145/1143844.1143865.

[35] T. G. Dietterich, "Ensemble methods in machine learning," in *Multiple Classifier Systems*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 1–15, ISBN: 978-3-540-45014-6.

[36] S. Fan, Z. Bao, C. Dong, H. Liang, X. Xu, and P. Zhang, "Semantic similarity score for measuring visual similarity at semantic level," *IEEE Internet of Things Journal*, 2024.

[37] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004. DOI: 10.1109/TIP.2003.819861.

[38] T. Zhu, B. Peng, J. Liang, *et al.*, "How to evaluate semantic communications for images with vitscore metric?" *IEEE Transactions on Cognitive Communications and Networking*, 2024.

[39] J. Hessel, A. Holtzman, M. Forbes, R. Le Bras, and Y. Choi, "CLIPScore: A reference-free evaluation metric for image captioning," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, M.-F. Moens, X. Huang, L. Specia, and S. W.-t. Yih, Eds., Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 7514–7528. DOI: 10.18653/v1/2021.emnlp-main.595. [Online]. Available: https://aclanthology.org/2021.emnlp-main.595/.

[40] M. Al-Shetairy, H. Hindy, D. Khattab, and M. Aref, "Transformers utilization in chart understanding: A review of recent advances future trends," Oct. 2024. DOI: 10.48550/arXiv.2410.13883.

[41] A. Masry, X. L. Do, J. Q. Tan, S. Joty, and E. Hoque, "ChartQA: A benchmark for question answering about charts with visual and logical reasoning," in *Findings of the Association for Computational Linguistics: ACL 2022*, S. Muresan, P. Nakov, and A. Villavicencio, Eds., Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 2263–2279. DOI: 10.18653/v1/2022.findings-acl.177. [Online]. Available: https://aclanthology.org/2022.findings-acl.177/.

[42] K.-H. Huang, H. Chan, Y. Fung, *et al.*, "From pixels to insights: A survey on automatic chart understanding in the era of large foundation models," *IEEE Transactions on Knowledge and Data Engineering*, vol. PP, pp. 1–20, Jan. 2024. DOI: 10.1109/TKDE.2024.3513320.

[43] A. Masry, M. Thakkar, A. Bajaj, A. Kartha, E. Hoque, and S. Joty, "ChartGemma: Visual instruction-tuning for chart reasoning in the wild," in *Proceedings of the 31st International Conference on Computational Linguistics: Industry Track*, O. Rambow, L. Wanner, M. Apidianaki, *et al.*, Eds., Abu Dhabi, UAE: Association for Computational Linguistics, Jan. 2025, pp. 625–643. [Online]. Available: https://aclanthology.org/2025.coling-industry.54/.

[44] D. S. Blank, D. Bourgin, A. Brown, *et al.*, "Nbgrader: A tool for creating and grading assignments in the jupyter notebook," *The Journal of Open Source Education*, vol. 2, no. 11, 2019.

[45] J. B. Hamrick, *Plotchecker*, https://github.com/charlespwd/project-title, 2015.

[46] M. Karppa, *Grading scale for plot tasks*, Personal communication, Jan. 2025.

[47] G. A. Mills-Tettey, A. Stentz, and M. B. Dias, "The dynamic hungarian algorithm for the assignment problem with changing costs," *Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-07-27*, 2007.

[48] K. Ding, K. Ma, S. Wang, and E. P. Simoncelli, "Image quality assessment: Unifying structure and texture similarity," *IEEE transactions on pattern analysis and machine intelligence*, vol. 44, no. 5, pp. 2567–2581, 2020.

[49] LAION, *Large openclip models*, Accessed: 2025-03-14, 2023. [Online]. Available: https://laion.ai/blog/large-openclip/.

[50] K. Fahd, S. Venkatraman, S. J. Miah, and K. Ahmed, "Application of machine learning in higher education to assess student academic performance,

at-risk, and attrition: A meta-analysis of literature," *Education and Information Technologies*, pp. 1–33, 2022.

[51] K. Fujiwara, "Knowledge distillation with resampling for imbalanced data classification: Enhancing predictive performance and explainability stability," *Results in Engineering*, vol. 24, p. 103 406, 2024.

[52] J. Wainer and G. Cawley, "Nested cross-validation when selecting classifiers is overzealous for most practical applications," *Expert Systems with Applications*, vol. 182, p. 115 222, 2021.

[53] W. J. Murdoch, C. Singh, K. Kumbier, R. Abbasi-Asl, and B. Yu, "Definitions, methods, and applications in interpretable machine learning," *Proceedings of the National Academy of Sciences*, vol. 116, no. 44, pp. 22 071–22 080, 2019.

[54] J. R. Vergara and P. A. Estévez, "A review of feature selection methods based on mutual information," *Neural computing and applications*, vol. 24, pp. 175–186, 2014.

[55] G. Chandrashekar and F. Sahin, "A survey on feature selection methods," *Computers Electrical Engineering*, vol. 40, no. 1, pp. 16–28, 2014, 40th-year commemorative issue, ISSN: 0045-7906. DOI: https://doi.org/10.1016/j.compeleceng.2013.11.024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0045790613003066.

[56] *Regulation (eu) 2024/1689 of the european parliament and of the council of 13 june 2024 laying down harmonised rules on artificial intelligence (artificial intelligence act)*, https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32024R1689, Official Journal of the European Union, L 168, 13 June 2024, 2024.

[57] European Union, *EU AI Act - Annex III*, https://artificialintelligenceact.eu/annex/3/, Accessed: 2025-02-11, 2024.

[58] European Union, *EU AI Act - Article 6*, https://artificialintelligenceact.eu/article/6/, Accessed: 2025-02-11, 2024.

[59] Government Offices of Sweden, *Public access to information and secrecy act*, Accessed: 2025-01-23, 2009. [Online]. Available: https://regeringen.se/informationsmaterial/2009/09/public-access-to-information-and-secrecy-act/.

[60] Universitetskanslersämbetet, *Rättssäker examination*, Fjärde upplagan, Diarienummer: 32-314-18, Universitetskanslersämbetet, Stockholm, Sverige, 2020. [Online]. Available: https://www.uka.se/download/18.16cf0f8c1849df46622152/1669103146069/Vagledning-2020-01-16-rattssaker-examination.pdf.

[61] S.-l. developers, *Sklearn.metrics.recall_score*, https://scikit-learn.org/stable/modules/generated/sklearn.metrics.recall_score.html, Accessed: 2025-04-22, 2024.

A

Example plot and corresponding universal structured format

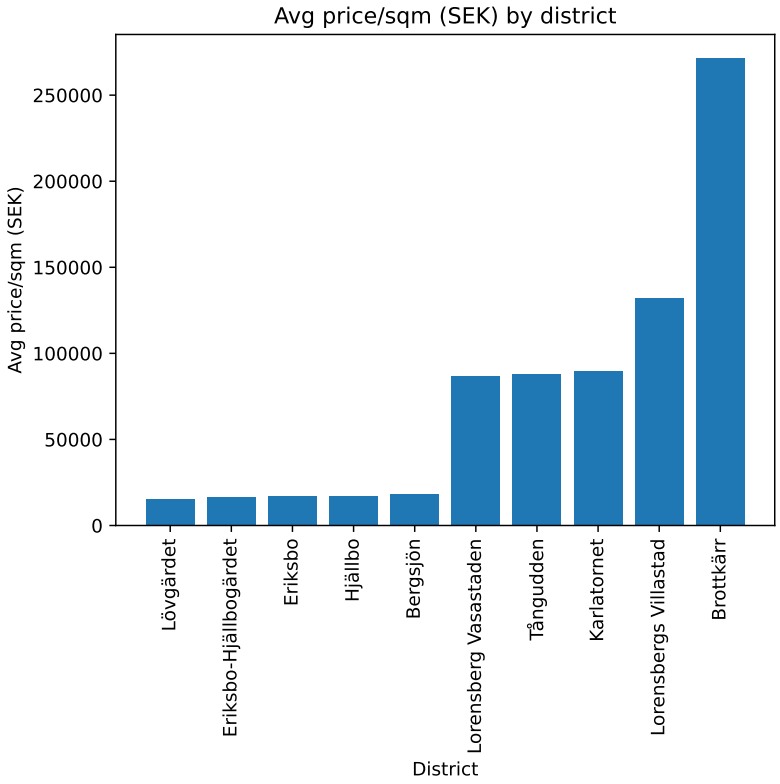


Figure A.1: Corresponding plot to the below universal structured format

```
[
    {
        "figure_number": 1,
        "axes": [
            {
                "title": "Average Price per sqm by District (Gothenburg 2023)",
                "xlabel": "District",
                "ylabel": "Average Price per sqm (SEK)",
                "legend": false,
```

```
"type": "bar",
"lines": [],
"scatters": [],
"bars": {
    "bars": [
        {
            "x": 0.0,
            "y": 15512.820512820512
        },
    {
        "x": 1.0,
        "y": 16830.985915492958
    },
    {
        "x": 2.0,
        "y": 17023.809523809527
    },
    {
        "x": 3.0,
        "y": 17228.37022132797
    },
    {
        "x": 4.0,
        "y": 18210.869471973638
    },
    {
        "x": 5.0,
        "y": 87028.76219825372
    },
    {
        "x": 6.0,
        "y": 87804.87804878049
    },
    {
        "x": 7.0,
        "y": 89583.33333333333
    },
    {
        "x": 8.0,
        "y": 132042.25352112675
    },
    {
        "x": 9.0,
        "y": 271666.6666666667
    }
],
```

```
    "stairs": null
},
"ticks": {
    "x_major": {
        "positions": [
            0,
            1,
            2,
            3,
            4,
            5,
            6,
            7,
            8,
            9
        ],
        "labels": [
            "L\u00f6vg\u00e4rdet",
            "Eriksbo-Hj\u00e4llbog\u00e4rdet",
            "Eriksbo",
            "Hj\u00e4llbo",
            "Bergsj\u00f6n",
            "Lorensberg Vasastaden",
            "T\u00e5ngudden",
            "Karlatornet",
            "Lorensbergs Villastad",
            "Brottk\u00e4rr"
        ]
    },
    "y_major": {
        "positions": [
            0.0,
            50000.0,
            100000.0,
            150000.0,
            200000.0,
            250000.0,
            300000.0
        ],
        "labels": [
            "0",
            "50000",
            "100000",
            "150000",
            "200000",
            "250000",
```

```
                        "300000"
                    ]
                }
            },
            "num_classes": 0
        }
    ]
}
]
```

Note that Unicode characters are displayed as escape codes instead of normal letters like “ö” or “ä”, which can make the output less readable. However, when the text is interpreted by a program or converted back into a regular string, the correct letters are displayed.

B

Scoring pipeline with no reference solution

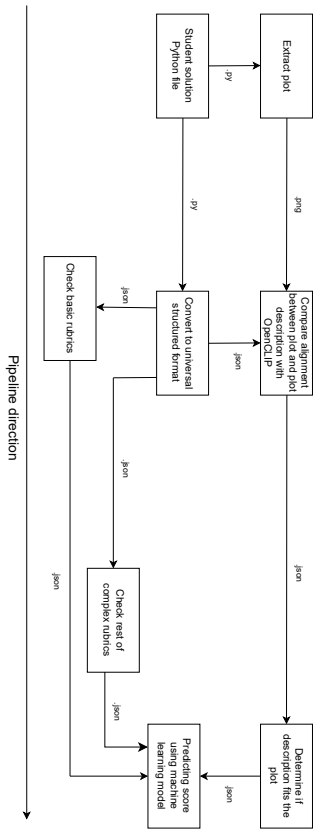


Figure B.1: Scoring pipeline when reference solution is not available

C

All feature importance scores

Table C.1: All features and importance scores

Feature	Importance Score
lpips_min	0.1277
composite_ylabel_sim	0.1217
max_score	0.0903
lpips_max	0.0875
average_lpips	0.0785
composite_title_sim	0.0734
lpips_mean	0.0713
y_unit_present	0.0666
composite_xlabel_sim	0.0605
title_present	0.0547
xlabel_present	0.0470
axis1_label_composite	0.0465
lpips_std	0.0447
ylabel_sim_missing	0.0279
x_unit_present	0.0271
y_ticks_present	0.0248
x_ticks_present	0.0221
composite_title_sim_missing	0.0212
legend_present	0.0164
composite_xlabel_sim_not_required	0.0117
num_classes	0.0097
num_classes_required	0.0070
composite_xlabel_sim_missing	0.0052
composite_ylabel_sim_not_required	0.0013
composite_title_sim_not_required	0.0000
ylabel_present	0.0000
flipped	0.0000