Comparison of Classifiers for Eye-Tracking Data

Jennifer Landes [1], Sonja Köppl [1], and Meike Klettke [2]

Abstract: This paper delves into the initial stages of data analysis, focusing on the classification of eye-tracking data. Six machine learning algorithms, namely XGBoost, Random Forest, Naive Bayes, Logistic Regression, Gradient Boosting Machines, and Neural Networks, were employed to predict cheating behavior based on a dataset comprising records from 25 students. Their performance was evaluated using metrics such as accuracy, precision, recall, F1 score, confusion matrix, and feature importance. Results indicate that Random Forest and its optimized version exhibit balanced performance, making them promising candidates for cheating prediction. The overarching research project investigates academic misconduct in the realm of online assessments, seeking to comprehend the behaviors and methodologies involved. An eye-tracking experiment was conducted to gain deeper insights into the timing and mannerisms of students engaging in academic misconduct.

Keywords: Eye Tracking, Data Preprocessing, Data Analysis, Machine Learning, Random Forest, Classification, Academic Cheating

1 Introduction

Analyzing eye tracking data presents both challenges and opportunities. There are issues such as data instability and outliers to deal with. This requires solid preprocessing strategies to clean up the data and select the right features. By understanding these challenges, we can build effective classifiers that can pick up subtle cues of student behavior. The cornerstone of this investigation is an eye-tracking experiment designed to explore various aspects of student behavior while answering questions. The focus of this paper is on the subsequent processing and analysis of the eye-tracking data, which is approached through classification techniques. In this paper, several classifiers are compared for their ability to predict student cheating behavior. Gaze data has the challenge of unstable and inconsistent behavior with a high number of outliers. Therefore, to train a good classifier, data preparation and good feature selection are important steps. The trained classifier will decide whether the whole gaze data of a participant belongs to a cheating group (1) or a non-cheating group (0).

To shed light on the current state of research in this area, we first present relevant studies that address different aspects of the application of machine learning to eye-tracking data. These studies range from the classification of eye-tracking data in different application contexts to the analysis of gaze patterns in the medical field. Chapter 2 describes the

[1] Hochschule Neu-Ulm, Business Informatics, Neu-Ulm, Germany,
jennifer.landes@hnu.de, https://orcid.org/0009-0003-1914-598X;
sonja.koeppl@hnu.de, https://orcid.org/0000-0002-9806-9318

[2] Universität Regensburg, Data Engineering, Germany,
meike.klettke@ur.de, https://orcid.org/0000-0003-0551-8389

doi:10.18420/inf2024_126

experimental procedure and the data preparation steps for the analysis. Finally, in chapter 3, the analysis is performed with several classifiers. Starting from model building, prediction and visualization of the results with a confusion matrix, the own investigation is presented with a focus on accuracy and performance. In the last chapter a short summary and future outlook are given.

1.1 Prior Work and Motivation

This study is part of the broader project ii.oo (Digitales Kompetenzorientiertes Prüfen implementieren), which aims to address academic misconduct among students in digital examination settings. The primary objective is to delve into the various factors influencing cheating behaviors and to identify the methods students employ. In light of the ongoing academic cheating concerns by the COVID-19 pandemic [Ja21], exploring cheating behaviors within the context of online exams remains pertinent. The project unfolds in distinct phases. Initially, a quantitative survey was conducted to gain insights into students' cheating behaviors and the contextual factors influencing these behaviors [La23]. The survey encompassed various tasks and cheating scenarios, probing students' preferences and motivations. Drawing from the survey findings, an eye-tracking experiment was designed to gain deeper insights into cheating patterns. This experiment aimed to capture instances of cheating during question answering. The subsequent analysis of the eye-tracking data forms the crux of this study, leveraging machine learning techniques. So, this project is structured to first understand the landscape of academic misconduct through a quantitative survey, followed by the design and implementation of an eye-tracking experiment to explore cheating behavior. The preprocessing and analysis of collected data are further steps, paving the way for detection of patterns indicative of academic misconduct.

1.2 Related Work

To investigate the research in the area of analysing eye tracking data with Machine Learning tools, the selected papers present similar applications scenarios of the utilization of classification models on eye tracking data. Furthermore, the selected studies give information and challenges of used preprocessing steps and the setting of the classification models. To conduct the literature review, we systematically searched scientific databases like Google Scholar, IEEE Xplore, and ACM Digital Library using specific keywords related to eye tracking and machine learning. We selected studies based on criteria such as relevance, publication date, and methodological quality, then extracted and analyzed information on application scenarios, classification models, preprocessing steps, and challenges.

[Yi21] and [Yi18] demonstrate the effectiveness of Convolutional Neural Networks (CNNs) in classifying Eye Tracking Data in Visual Information Processing Tasks. They prepare the data by defining Areas of Interest (AOIs) and employing feature engineering techniques.

The study addresses the absence of a standardized process for transforming eye tracking data for deep learning models. Results show CNN models achieving over 80% accuracy, with fewer convolutional layers offering satisfactory results at lower GPU costs. In a study by [AMa20], Machine Learning algorithms are utilized to analyze raw eye-tracking data in the medical field. Two datasets are used: one labeled by machine learning algorithms and another by manual identification. Algorithms identify fixation, saccades, and smooth pursuits, achieving 98% accuracy for standard datasets using Random Forest and Decision Tree classifiers. However, manual labeling may affect performance. [Ze18] employ a random forest classifier, IRF, to detect eye movement events without manual parameter adjustment. This approach provides stable event detection across varying noise levels and data sampling frequencies, emphasizing the shift towards algorithm-determined feature combinations and thresholds. In another study, [Kl21] explore Support Vector Machines (SVM) achieving 76.1% accuracy in combination with electrooculography. They highlight the potential of ML combined with eye tracking to address challenges such as synchronization between head movement and handling noisy data with multiple IoT devices.

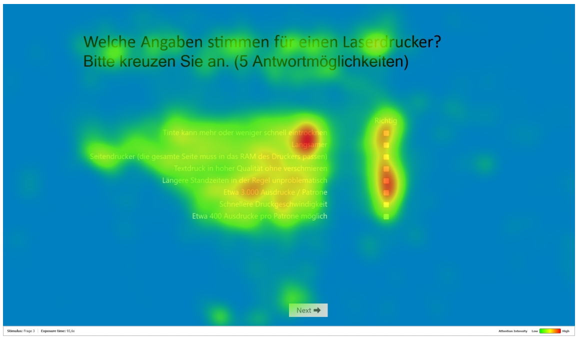
In contrast to these studies, the data set of our experiment includes saccades and fixations, so there is no need to detect them. It is important to note from the literature review that the data preprocessing step of standardization and a well-prepared feature selection are essential for good data analysis. Furthermore, the classifier Random Forest seems to achieve the highest accuracy results, so this classifier will be chosen for optimization in a later step of the analysis. Despite the different approaches and successes in previous research, a benchmarking approach to eye-tracking analysis is still crucial. Benchmarking allows for a standardized comparison of different methods under consistent conditions, facilitating the identification of the most effective techniques. This approach not only highlights the strengths and weaknesses of each method, but also drives improvements in the analysis of eye-tracking data by setting performance standards and promoting best practices in the field.

2 Experiment and Data Preparation

2.1 Experiment

Our eye-tracking experiment, conducted at the Neu-Ulm University of Applied Sciences with bachelor students of industrial engineering from the 1st to the 5th semester, collected a total of 125 datasets from 25 students. These datasets were divided into five different tasks to allow a comprehensive analysis. Each experiment lasted 15-20 minutes, during which students completed tasks including a definition task, multiple choice, single choice, an open-ended question, and a coding task. Students were asked to report their cheating methods, choosing from options such as an analog cheat sheet, a cell phone, or consulting another student.

To identify cheating behavior in the dataset, a control group consisting of datasets without cheating behavior was introduced. These datasets serve as a comparison group to analyze



Welche Angaben stimmen für einen Laserdrucker?
Bitte kreuzen Sie an. (5 Antwortmöglichkeiten)

	Richtig
Tinte kann mehr oder weniger schnell eintrocknen	▨
Langsamer	▨
Seitendrucker (die gesamte Seite muss in das RAM des Druckers passen)	▨
Textdruck in hoher Qualität ohne verschmieren	▨
Längere Standzeiten in der Regel unproblematisch	▨
Etwa 3.000 Ausdrucke / Patrone	▨
Schnellere Druckgeschwindigkeit	▨
Etwa 400 Ausdrucke pro Patrone möglich	▨

Stimulus: Frage 3 | Exposure time: 93,6s

Fig. 1: Heatmap of Task Type 3 - Single Choice of one participant

patterns and differences between the cheating and non-cheating groups. The lab setup included two sections with four workstations, each equipped with an external Tobii 120 eye-tracking device to monitor participants. Students were organized into groups of five, with one participant acting as a supervisor to simulate real exam scenarios. This setup facilitated the study of cheating behavior in a controlled environment.

2.2 Dataset

The experiment used sensor data exported to CSV using iMotions 5.1. After export, the data underwent standardization and column alignment to reconcile output differences between devices. Information unrelated to eye-tracking analysis, such as PC CPU values, StimType, and Eventsource, was omitted from further processing. Table 1 lists the selected features essential for understanding gaze behavior and eye movement dynamics, while system-level metrics and experimental metadata were excluded as they do not contribute to the classification task.

2.3 Data Preprocessing

In this section, the preprocessing steps to prepare the eye-tracking data for analysis with Python, Pandas and scikit-learn are presented. At the beginning, all numeric columns in the dataset undergo a **standardization**. Therefore, the StandardScaler from scikit-learn library was used. The data is transformed to achieve a mean of 0 and a standard deviation of 1. The process is done by an iteration through the CSV files and by identifying numeric columns, and standardizing their values.

The second step is to identify and treat **missing values**. Missing values can occur in gaze data when a student looks out of the recorded area. A report provides insight into the missing values. Observations show a high number of missing values in the Duration column,

Features	**Description**
GazeLeftx, GazeLefty	The X and Y coordinates of the left eye.
GazeRightx, GazeRighty	The X and Y coordinates of the right eye.
PupilLeft, PupilRight	The pupil size of the left and right eyes.
DistanceLeft, DistanceRight	The distance of the left and right eyes.
CameraLeftX/Y, CameraRightX/Y	The camera coordinates of the left and right eyes.
ValidityLeft, ValidityRight	The validity of the gaze data from left and right eyes.
Gaze X, Gaze Y	The X and Y coordinates of the participant's gaze point.
Interpolated Gaze X and Y	Interpolated X and Y coordinates of the gaze point.
Interpolated Distance	The interpolated distance of the gaze point.
Gaze Velocity	The velocity of the gaze point.
Gaze Acceleration	The acceleration of the gaze point.
Fixation Index	The number of the fixation in a specific order.
Fixation X, Fixation Y	The X and Y coordinates of the participant's fixation point.

Tab. 1: Features of Eye Tracking Dataset

resulting in the removal of this column. Furthermore, the gaze-related columns (Gaze X, Gaze Y, Interpolated Gaze X, Interpolated Gaze Y, Interpolated Distance, Gaze Velocity, Gaze Acceleration) show missing values ranging from 32% to 49%. Therefore, a mean imputation is performed. Third, the fixation data columns (with 55% to 56% missing values) require a closer look at the cause of the missing values, they are not removed because they may only occur in single events. These single events may be important in detecting cheating.

The next preprocessing step involves the identification and treatment of **outliers**. Outlier detection allows to identify and handle data points that deviate significantly from the majority of observations. These outliers can be caused by a variety of factors, including measurement error, participant distraction, or true deviations in gaze behavior. One criterion for identifying outliers is a data point's Z-score, which measures the number of standard deviations by which the data point deviates from the mean. A high Z-score suggests that the data point may be a potential outlier. $z = \frac{X-\mu}{\sigma}$

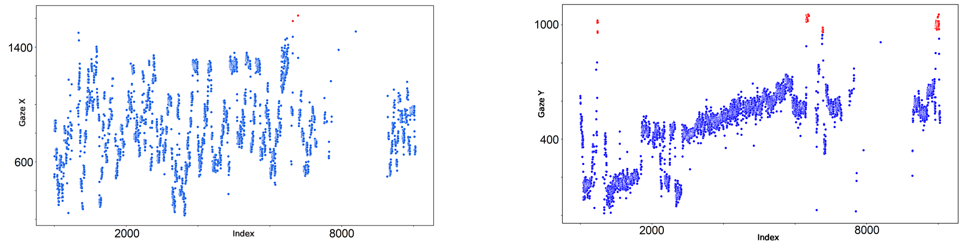


Fig. 2: Outlier Detection for Gaze X and Gaze Y

The Z-scores are calculated for each column and a threshold of 3 is used to determine if data points are outliers. The chosen strategy, based on the Observation Report, is to

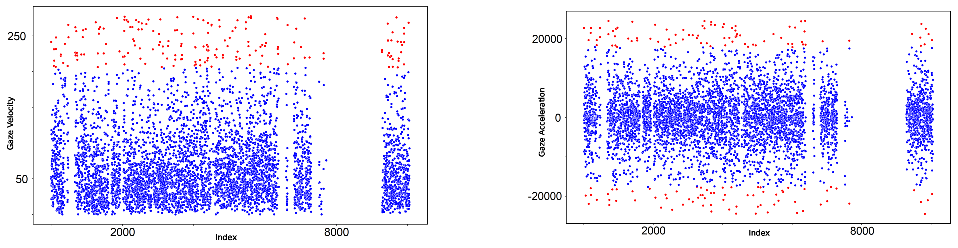


Fig. 3: Outlier Detection for Gaze Velocity and Gaze Acceleration

remove outliers; alternative methods such as replacing with the mean or Winsorizing are not applicable. A separate column of outliers is stored for further analysis. The features with the highest average number of outliers are EValidityLeft (341.6), DistanceRight (246.6), DistanceLeft (234.5), PupilLeft (219.6), GazeLeftx (217.4), GazeLefty (188.5), PupilRight (177.1), and GazeRightx (170.0). Other values include CameraLeftX (277.7) and CameraLeftY (276.8).

Lastly, no **duplicates** were detected, obviating the need for further treatment, because of the automatic output method with the eye tracking software.

3 Analysis

In this part, the several classifiers were chosen for the analysis phase. The classifiers—XGBoost, Random Forest, Naive Bayes, Logistic Regression, Gradient Boosting Machines, and Neural Networks—were selected and compared in their results of accuracy, confusion matrix, feature importance, f1-score, recall and precision.

3.1 Selection of Classifiers

The following classifiers were selected based on their suitability for analyzing eye-tracking data. Ensemble techniques such as Random Forest and Gradient Boosting Machines are ideal for capturing non-linear relationships and are robust to noise. Simple models such as Logistic Regression and Naive Bayes provide interpretability and serve as baselines for comparison, while Neural Networks capture complex patterns. These classifiers have proven effective at handling the complexity of eye-tracking data, as supported by existing research. We did not initially include deep learning methods due to data constraints, but their potential for future consideration is acknowledged.

Logistic Regression (LR) is a statistical method for binary classification tasks. It models the relationship between a dependent variable and one independent variables by estimating probabilities using the logistic function, which maps inputs to the range [0,1] [PM17].

Gradient Boosting Machines (GBMs) are ensemble learning methods and build a strong predictive model by combining multiple decision trees. GBMs optimize a loss function by iterattively fitting new models to the residual errors of the previous models [LKA15; Up21].

XGBoost is an optimization of gradient boosting and provides parallel processing and regularization techniques to prevent overfitting. It employs a robust splitting criterion and incorporates additional regularization terms to enhance model performance [EK21].

Neural Networks are inspired by the structure and function of biological neural networks. They consist of interconnected layers of artificial neurons that transform input data through nonlinear activation functions, allowing them to learn complex patterns [Yi18].

Naïve Bayes is a probabilistic classifier based on Bayes' theorem with the assumption of conditional independence between features given the class. Despite its simplicity, Naïve Bayes often performs well in practice, especially with high-dimensional data, making it a popular choice for text classification and other similar tasks [LKA15].

Random Forest: Decision trees start with a basic question and iteratively branch into subsequent questions to arrive at a final decision. These questions form decision nodes, splitting data, with the final decision represented by leaf nodes. The trees seek optimal splits based on metrics like Gini impurity or information gain. Ensemble learning methods, like bagging and boosting, aggregate predictions from multiple classifiers [PKP19].

3.2 Model Building

Classification is performed on the eye-tracking data to distinguish between students who did not cheat (non-cheating group) and those who did cheat (cheating group). Eye-tracking data is loaded from CSV files for both groups. The features include GazeLeftx, GazeLefty, GazeRightx, GazeRighty, PupilLeft, PupilRight, Gaze Velocity, Gaze Acceleration. The data is then combined with the corresponding target labels (0 for non-cheating, 1 for cheating). Then, eye tracking features are extracted for classification. The data is then split into training and test sets (80-20 split) using scikit-learn. The classifiers are trained on the training data (X_train and Y_train) using the fit method. Each classifier is trained with standard hyperparameters, and the resulting accuracies of the models are shown in Figure 5, the classifier Random Forest achieved the highest accuracy of 81.01%.

Optimization of Random Forest

Random forest was chosen for optimization because of its robust performance on large datasets and many features. It combines the results of multiple decision trees to produce more stable and accurate results than individual trees. Key hyperparameters include node size, number of trees, and features sampled. Each tree is built using a bootstrap sample from the training set, with feature bagging adding diversity. Classification uses majority voting for predictions. Out-of-bag samples are used for cross-validation [Br01; Pa18]. Random

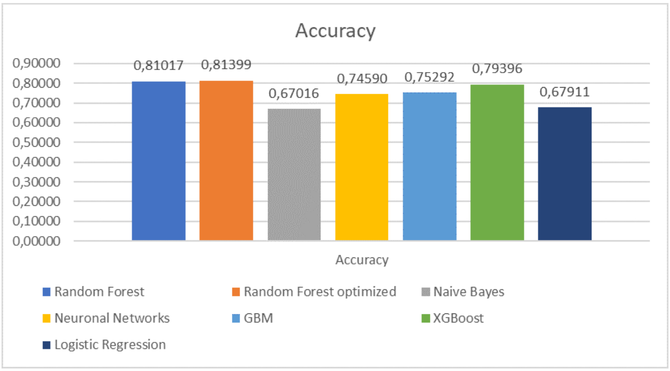


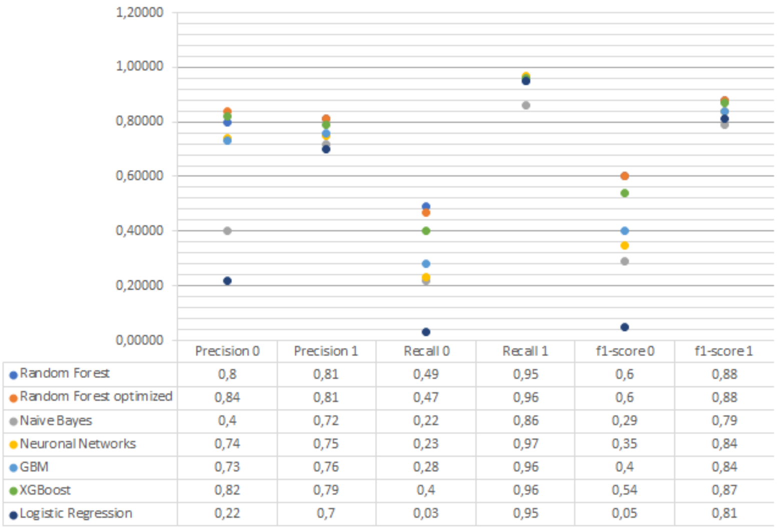
Fig. 4: Accuracies of the Selected Classifiers

Forest is optimized using **GridSearch**, initialized with a random state for reproducibility, and ’gini’ criterion for tree splitting [SD19; Sp19]. The best hyperparameters—Maximum Depth: None, Maximum Features: log2, Minimum Samples Leaf: 2, Minimum Samples Split: 2, and Number of Estimators: 100—achieve an accuracy of 81.4%.

```
param_grid = {
    'n_estimators': [10, 20, 50, 100],
    'max_depth': [None, 1, 2, 4],
    'min_samples_split': [1,2],
    'min_samples_leaf': [1,2],
    'max_features': [None, 'sqrt', 'log2']
}
grid_search = GridSearchCV(estimator=rf_classifier, param_grid=param_grid,
    cv=3, scoring='accuracy', n_jobs=-1)
grid_search.fit(X_train, y_train)
```

3.3 Comparison of Results

Both Random Forest and Optimized Random Forest show high overall accuracy (81.02% and 81.40%, respectively). Optimized Random Forest shows improved accuracy for class 0 (non-target) compared to Random Forest, while maintaining high accuracy for class 1 (target). Both models show significantly high recall for class 1, indicating their effectiveness in correctly identifying target instances. Naive Bayes achieves the lowest accuracy among the classifiers, but shows relatively high precision for class 1. Neural Networks, GBM, and XGBoost perform comparably, with accuracies ranging from about 74.59% to 79.40%. Logistic regression has the lowest accuracy and precision for class 0 among all classifiers, indicating its limitations in correctly identifying non-target instances.



	Precision 0	Precision 1	Recall 0	Recall 1	f1-score 0	f1-score 1
● Random Forest	0,8	0,81	0,49	0,95	0,6	0,88
● Random Forest optimized	0,84	0,81	0,47	0,96	0,6	0,88
● Naive Bayes	0,4	0,72	0,22	0,86	0,29	0,79
● Neuronal Networks	0,74	0,75	0,23	0,97	0,35	0,84
● GBM	0,73	0,76	0,28	0,96	0,4	0,84
● XGBoost	0,82	0,79	0,4	0,96	0,54	0,87
● Logistic Regression	0,22	0,7	0,03	0,95	0,05	0,81

Fig. 5: Scoring of Selected Models

Models using ensemble techniques (Random Forest, GBM, XGBoost) generally outperform Naive Bayes and Logistic Regression in accuracy and precision. Neural Networks and XGBoost show high recall for class 1, indicating their effectiveness in capturing target instances. Precision for class 0 varies considerably across classifiers, with some models performing better at identifying non-target instances than others [TK18]. While Random Forest and its optimized variant excel in balanced performance across precision, recall, and accuracy, Neural Networks and XGBoost also show competitive performance, particularly in detecting target instances. Conversely, Naive Bayes and Logistic Regression struggle to accurately identify non-target instances, suggesting room for improvement or alternative model selection. Figure 6 provides a visual comparison of these measures.

Predictions are made using the trained classifiers on a set of test data. The predictions, along with the actual label (0 for non-cheating, 1 for cheating), are printed for each file, and the results are stored in a list. The performance of the classifiers is evaluated by a **confusion matrix** (see Figure 7) to provide insight into the model's ability to correctly predict true positives, true negatives, false positives, and false negatives [SD19].

For the purposes of this analysis, we'll focus on the **minimization** of false positives (**Type I errors**) and false negatives (**Type II errors**). The matrix compares actual target scores with those predicted by the machine learning model. False negatives occur when students who did not cheat are correctly predicted as such. False positives (Type I errors) occur when non-cheating students are incorrectly predicted as cheaters. True negatives (type II errors) occur when cheaters are mispredicted as non-cheaters. True positives occur when cheaters are correctly identified. The confusion_matrix function from scikit-learn is used and visualized using seaborn's heatmap and Matplotlib. The x-axis represents the predicted labels, while the y-axis represents the true labels.

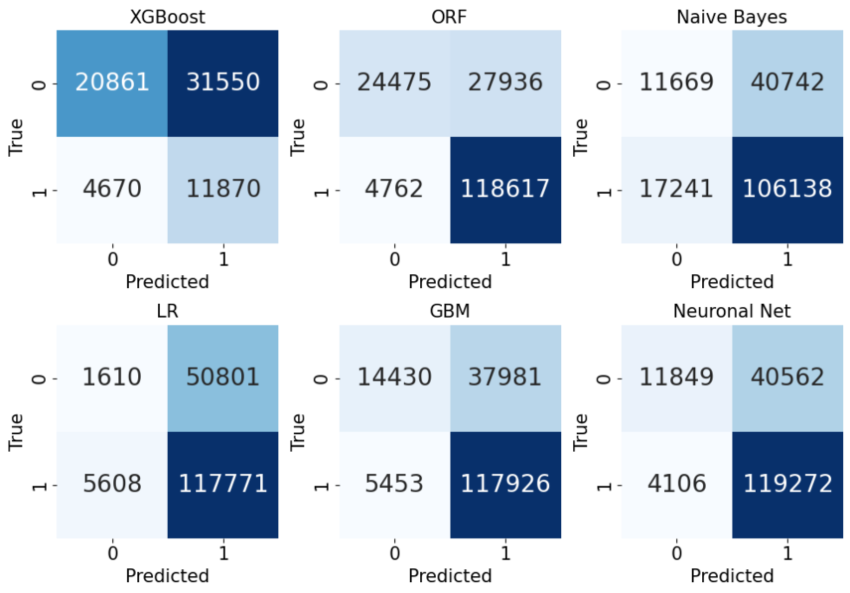
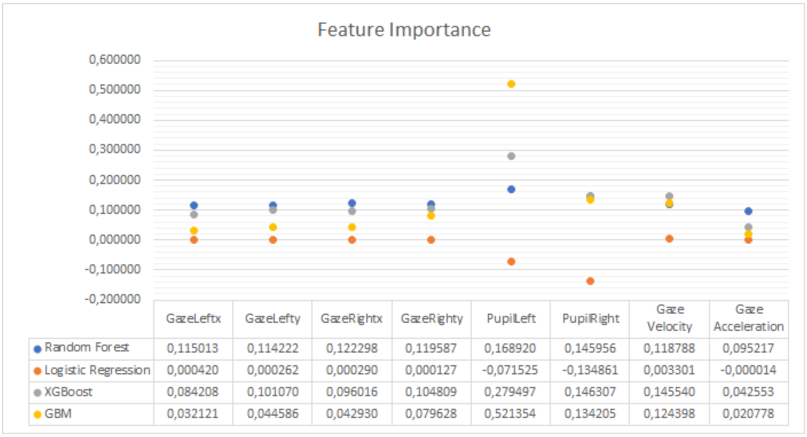


Fig. 6: Confusion Matrices of Classifiers

The classifier with the lowest number of false positives is Optimized Random Forest, followed by XGBoost. Naive Bayes and Neural Networks have higher false positive rates, while Logistic Regression and GBM have the highest false positive rates. The classifier with the lowest number of false negatives is also Optimized Random Forest, followed by GBM. Logistic regression has the highest number of false negatives, followed by Naive Bayes and Neural Networks.

The **feature importance** provides insight into which features play a significant role in the model's decision process [Jo02]. This information is crucial for identifying the key factors from the set of GazeLeftx, GazeLefty, GazeRightx, GazeRighty, GazePupilLeft, GazePupilRight, GazeVelocity, GazeAcceleration that influence the classification of eye-tracking data into non-cheating and cheating categories. The features of Random Forest, Logistic Regression, XGBoost, GBM are compared and visualized in Figure 8. This shows the varying degrees of influence that different features have on the classifiers' decision processes. Across all classifiers, PupilLeft consistently emerges as the most influential feature, particularly in the XGBoost and GBM models. Gaze Velocity also shows significant importance in all models. Interestingly, while Logistic Regression assigns minimal importance to most features, XGBoost and GBM emphasize features related to gaze coordinates.



	GazeLeftx	GazeLefty	GazeRightx	GazeRighty	PupilLeft	PupilRight	Gaze Velocity	Gaze Acceleration
● Random Forest	0,115013	0,114222	0,122298	0,119587	0,168920	0,145956	0,118788	0,095217
● Logistic Regression	0,000420	0,000262	0,000290	0,000127	-0,071525	-0,134861	0,003301	-0,000014
● XGBoost	0,084208	0,101070	0,096016	0,104809	0,279497	0,146307	0,145540	0,042553
● GBM	0,032121	0,044586	0,042930	0,079628	0,521354	0,134205	0,124398	0,020778

Fig. 7: Feature Importance of Classifiers

4 Discussion

In this section, we reflect on lessons learned from the analysis of eye-tracking data and future directions for this research. One critical lesson is the need for detailed annotation within the eye-tracking data. Accurate annotation is essential for identifying and analyzing specific instances of deceptive behavior. This finding highlights a limitation of the current study, where the granularity of the data annotation was insufficient for nuanced analysis of cheating points. Consequently, future experiments will incorporate more precise annotations to allow for finer distinctions in behavioral analysis.

The current approach to data analysis is relatively coarse, classifying entire data sets as either cheating or non-cheating. While this provides a high-level overview, it lacks the resolution needed to understand the intricacies of cheating behavior. Moving forward, implementing more granular classifications can improve the detection and understanding of specific cheating strategies. This involves not only refining data annotation, but also using advanced analytical techniques such as multidimensional time series analysis. Such methods can reveal temporal patterns and correlations within the data that may indicate cheating behavior. All program code was executed on the same computing infrastructure, ensuring consistency and reliability of results. This standardization is critical for reproducibility and for making accurate comparisons between different classifiers.

The choice of classification as the primary method of analysis is justified by the nature of the problem at hand. Classification algorithms are well suited to distinguish between different categories of behavior - in this case, cheating and non-cheating. Initial results highlighted the superior performance of the Random Forest classifier in terms of accuracy, which led to its selection for further optimization. Random Forest's robustness to complex, high-dimensional data makes it particularly suited to eye-tracking analysis, where the data often exhibits intricate patterns.

5 Conclusion and Outlook

In this paper, a classification analysis of eye-tracking data was conducted. Through data preparation by standardization, missing value treatment, outlier detection and feature selection, we have successfully prepared the data for classification. The classifiers XGBoost, Random Forest, Naive Bayes, Logistic Regression, Gradient Boosting Machines and Neural Networks were selected for the analysis phase and trained with their default settings. For each classifier, the comparison includes accuracy, precision, recall, f1 score, confusion matrix and feature importance. Based on the results, Random Forest was selected for optimization using GridSearch and initialized with a random state for reproducibility, using the 'gini' criterion for decision tree splitting.An accuracy of 81.4% was obtained, demonstrating the predictive potential of this classifier to discriminate between cheating and non-cheating behavior based on eye-tracking features. In this context, one feature to highlight across all classifiers is PupilLeft, which emerges as the most influential in predicting cheating. Performance is also evaluated using a confusion matrix, which provides insight into the model's ability to predict true positives, true negatives, false positives, and false negatives. The focus is on minimizing false positives and false negatives. The optimized Random Forest and XGBoost classifiers have the lowest false positive and false negative rates. Logistic Regression and Naive Bayes have higher false positive rates, while Logistic Regression has the highest false negative rate.

The results indicate that Random Forest and its optimized version show balanced performance in terms of precision, recall, and accuracy. Other models such as Neural Networks and XGBoost also show competitive performance, especially in detecting target instances. In contrast, Naive Bayes and Logistic Regression show limitations in accurately identifying non-target instances. This suggests that models using ensemble techniques tend to outperform simpler models such as Naive Bayes and logistic regression. The nature of eye-tracking data, which is often characterized by complex patterns and dependencies/correlations between features, benefits from the ensemble approach of Random Forest, which helps to capture the complexity of the data. [AMa20; Ze18]

While the analysis has yielded promising results, it is important to acknowledge certain limitations and areas for future research. Challenges such as sample size, data quality, feature engineering, and model interpretability remain relevant in this area. In addition, further investigations could explore more optimizations and advanced machine learning algorithms. Thus, in the coming period, different preprocessing techniques will be explored to evaluate their impact on classifier performance metrics.

References

[AMa20] Akshay, S. et al.: Machine Learning Algorithm to Identify Eye Movement using Metrics Raw Eye Tracking Data. In: 2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT). IEEE, Tirunelveli, India, pp. 949–955, 2020, isbn: 978-1-72815-821-1, doi: 10.1109/ICSSIT48917.2020.9214290, url: https://ieeexplore.ieee.org/document/9214290/, visited on: 01/19/2024.

[Br01] Breiman, L.: Random Forests. en, Machine Learning 45 (1), pp. 5–32, 2001, issn: 1573-0565, doi: 10.1023/A:1010933404324, url: https://doi.org/10.1023/A:1010933404324, visited on: 02/03/2024.

[EK21] Ergul Aydin, Z.; Kamisli Ozturk, Z.: Performance Analysis of XGBoost Classifier with Missing Data. 2021.

[Ja21] Janke, S.; Rudert, S. C.; Petersen, Ä.; Fritz, T. M.; Daumiller, M.: Cheating in the wake of COVID-19: How dangerous is ad-hoc online testing for academic integrity? Computers and Education Open 2, p. 100055, 2021, issn: 2666-5573, doi: 10.1016/j.caeo.2021.100055, url: https://www.sciencedirect.com/science/article/pii/S2666557321000264, visited on: 03/11/2024.

[Jo02] Joshi, M.: On evaluating performance of classifiers for r are c lasses. I n: 2 002 IEEE International Conference on Data Mining, 2002. Proceedings. Pp. 641–644, 2002, doi: 10 . 110 /ICDM . 2002 . 1184018, url: https://ieeexplore . ieee . org /abstract /document/1184018, visited on: 03/11/2024.

[Kl21] Klaib, A. F. et al.: Eye tracking algorithms, techniques, tools, and applications with an emphasis on machine learning and Internet of Things technologies. en, Expert Systems with Applications 166, p. 114037, 2021, issn: 09574174, doi: 10.1016/j.eswa.2020.114037, url: https://linkinghub.elsevier.com/retrieve/pii/S0957417420308071, visited on: 11/30/2023.

[La23] Landes, J.: Influence Factors on Academic Integrity revealed by Machine Learning Methods. 2023, url: https://gvdb23.informatik.uni-stuttgart.de/wp-content/uploads/2023/06/GvDB2023_Landes.pdf, visited on: 05/19/2024.

[LKA15] L., M. G.; Karthi, M.; Anu, M.: A statistical comparison of logistic regression and different bayes classification methods for machine learning. ARPN Journal of Engineering and Applied Sciences 10, pp. 5947–5953, 2015, visited on: 01/19/2024.

[Pa18] Paul, A.; Mukherjee, D. P.; Das, P.; Gangopadhyay, A.; Chintha, A. R.; Kundu, S.: Improved Random Forest for Classification. IEEE Transactions on Image Processing 27 (8), Conference Name: IEEE Transactions on Image Processing, pp. 4012–4024, 2018, issn: 1941-0042, doi: 10.1109/TIP.2018.2834830, url: https://ieeexplore.ieee. org/abstract/document/8357563, visited on: 01/23/2024.

[PKP19] Parmar, A.; Katariya, R.; Patel, V.: A Review on Random Forest: An Ensemble Classifier. In (Hemanth, J.; Fernando, X.; Lafata, P.; Baig, Z., eds.): International Conference on Intelligent Data Communication Technologies and Internet of Things (ICICI) 2018. Lecture Notes on Data Engineering and Communications Technologies, Springer International Publishing, Cham, pp. 758–763, 2019, isbn: 978-3-030-03146-6, doi: 10.1007/978-3-030-03146-6_86, visited on: 04/27/2024.

[PM17] Pranckevičius, T.; Marcinkevičius, V.: Comparison of Naive Bayes, Random Forest, Decision Tree, Support Vector Machines, and Logistic Regression Classifiers for Text Reviews Classification. In: Baltic Journal of Modern Computing 5 (2), 2017, issn: 22558950, doi: 10. 22364 /bjmc.2017.5.2.05, url: http://www.bjmc.lu.lv/fileadmin/user_upload/lu_portal/projekti/bjmc/Contents/5_2_05_Pranckevicius.pdf, visited on: 03/11/2024.

[SD19] Shekar, B. H.; Dagnew, G.: Grid Search-Based Hyperparameter Tuning and Classification of Microarray Cancer Data. In: 2019 Second International Conference on Advanced Computational and Communication Paradigms (ICACCP). Pp. 1–8, 2019, doi: 10.1109/ICACCP.2019.8882943, url: https://ieeexplore.ieee.org/abstract/document/8882943, visited on: 03/11/2024.

[Sp19] Speiser, J. L.; Miller, M. E.; Tooze, J.; Ip, E.: A comparison of random forest variable selection methods for classification prediction modeling. Expert Systems with Applications 134, pp. 93–101, 2019, issn: 0957-4174, doi: 10.1016/j.eswa.2019.05.028, url: https://www.sciencedirect.com/science/article/pii/S0957417419303574, visited on: 01/23/2024.

[TK18] Thanh Noi, P.; Kappas, M.: Comparison of Random Forest, k-Nearest Neighbor, and Support Vector Machine Classifiers for L and C over C lassification Us ing Sentinel-2 Imagery. en, Sensors 18 (1), Number: 1 Publisher: Multidisciplinary Digital Publishing Institute, p. 18, 2018, issn: 1424-8220, doi: 10.3390/s18010018, url: https://www.mdpi.com/1424-8220/18/1/18, visited on: 01/23/2024.

[Up21] Upadhyay, D. et al.: Gradient Boosting Feature Selection With Machine Learning Classifiers for Intrusion Detection on Power Grids. IEEE Transactions on Network and Service Management 18 (1), Conference Name: IEEE Transactions on Network and Service Management, pp. 1104–1116, 2021, issn: 1932-4537, doi: 10.1109/TNSM.2020.3032618,rl; https://ieeexplore.ieee.org/abstract/document/9236652, visited on: 03/11/2024.

[Yi18] Yin, Y.; Juan, C.; Chakraborty, J.; McGuire, M. P.: Classification of Eye Tracking Data Using a Convolutional Neural Network. In: 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA). IEEE, Orlando, FL, pp. 530–535, 2018, isbn: 978-1-5386-6805-4, doi: 10.1109/ICMLA.2018.00085, url: https://ieeexplore.ieee.org/document/8614110/, visited on: 01/19/2024.

[Yi21] Yin, Y. et al.: Classification of Eye Tracking Data in Visual Information Processing Tasks Using Convolutional Neural Networks and Feature Engineering. en, SN Computer Science 2 (2), p. 59, 2021, issn: 2662-995X, 2661-8907, doi: 10 . 1007 / s42979-020-00444-0, url: http://link.springer.com/10.1007/s42979-020-00444-0, visited on: 01/19/2024.

[Ze18] Zemblys, R. et al.: Using machine learning to detect events in eye-tracking data. en, Behavior Research Methods 50 (1), pp. 160–181, 2018, issn: 1554-3528, doi: 10 . 3758 / s13428 - 017 - 0860 - 3, url: https://doi.org/10.3758/s13428-017-0860-3, visited on: 01/19/2024.