Data Processing Pipeline for Eye-Tracking Analysis

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

[1]Hochschule Neu-Ulm, Germany
[2]University of Regensburg, Germany

Abstract

The overarching topic of this research project is academic misconduct in online assessments, aiming to understand students´ behavior and methods. To gain deeper insights, an eye-tracking experiment was conducted to capture when and how students engage in academic misconduct. Data from this experiment will reveal irregularities in cheating behavior. This paper presents a data engineering pipeline for the preparation of the future eye-tracking data analysis implemented in Python and a reasoning for the chosen order. Steps like Feature Selection, Data Preparation, Outlier Detection and Treatment, Filtering, Smoothing, and Normalization are included in this pipeline. We describe the data set, the setting and conduction of the experiment, and the data engineering pipeline. This article contributes to the current discussion of the preprocessing and analyse of eye tracking data.

Keywords

Data Pipeline, Data Preprocessing, Machine Learning

1. Introduction

Academic misconduct still persists as a challenge to the integrity of higher education, especially within the context of digital examinations. The importance of robust methodologies to detect and prevent such behaviors is further underscored. While much attention is understandably focused on the act of cheating itself, it's equally critical to recognize the significance of preparing data for subsequent analysis.

This paper aims to address this crucial intersection between academic misconduct and data analysis, specifically focusing on the preprocessing pipeline tailored for collected eye-tracking data during an experiment. With the rise of digital assessments, accelerated by the COVID-19 pandemic [1], there's an urgent need to explore cheating behaviors through novel avenues. However, before we can deliver insights, raw data must undergo several preprocessing steps to ensure its quality and usability.

This study presents an approach to prepare eye-tracking data for an upcoming analysis with machine learning models. While the overarching project endeavors to understand and mitigate academic misconduct among students, this specific endeavor delves into the process of data preparation and refinement—a critical prerequisite for meaningful analysis.

The challenge inherent in eye-tracking data adds an additional layer of complexity to the preprocessing task. The nature of eye-tracking data, which often includes noise, calibration errors, and variability between participants, underscores the necessity of rigorous preprocessing to ensure accurate and reliable analysis results. Therefore, addressing these challenges in preparing eye-tracking data is essential for effectively detecting patterns indicative of academic misconduct and advancing our understanding of cheating behaviors in digital examination settings.

The subsequent sections of this paper outline the various preprocessing steps undertaken to cleanse and enhance the eye-tracking dataset, laying the foundation for subsequent analyses. Chapter 2 provides a comprehensive review of relevant literature, emphasizing the significance of preprocessing. Chapter 3 delves into the experiment design and dataset characteristics, setting the stage for the preprocessing procedures detailed in Chapter 4. Finally, Chapter 5 offers a summary of findings and discusses potential avenues for future research, emphasizing the pivotal role of preprocessing in the broader landscape of academic misconduct detection.

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.

The project unfolds in distinct phases. Initially, a quantitative survey on basis of [2] was conducted to gain insights into students' cheating behaviors and the contextual factors influencing these behaviors [3]. 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.

35th GI-Workshop on Foundations of Databases (Grundlagen von Datenbanken), May 22-24, 2024, Herdecke, Germany.
jennifer.landes@hnu.de (J. Landes); sonja.köppl@hnu.de (S. Köppl); meike.klettke@ur.de (M. Klettke)
© 2024 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR-WS.org/Vol-3710/paper6.pdf

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073

The eye-tracking experiment involved the collection of data from 20 participants during the examination. This paper focuses specifically on the data preparation phase of the project, which is integral to the subsequent analysis and pattern detection. The preprocessing steps undertaken ensure that the collected data is refined and ready for analysis.

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 delve deeper into cheating behaviors. The preprocessing of collected data is a crucial preparatory step, paving the way for meaningful analysis and detection of patterns indicative of academic misconduct.

2. Related Work

The preprocessing literature discusses several steps like detection and treatment of noise, outliers or missing values, feature selection to reduce the dimension and the important step of normalizing. In case of sensing data like Eye tracking, the challenge of noise needs the use of filters in the preprocessing stage. These filters are able to eliminate instances in the dataset, which may lead to misclassification issues [4]. The identification and management of outliers are categorized in statistics-based, distance-based, and density-based. A statistics-based method by Huang et al. [5] assumes a statistical model for the dataset and outliers are detected using statistical tests. Other methods, such as by Buzzi-Ferraris and Manenti [6] also evaluate mean, variance, and outlier values. For large datasets, Angiulli and Pizzuti [7] introduced a distance-based outlier detection algorithm, HilOut, to identify the top outliers in a dataset. HilOut computes the weight of a point as the sum of distances to its k-nearest neighbors, identifying outliers as points with the highest weight. An approach by Ghoting et al. [8], presents the RBRP algorithm for mining distance-based outliers in high-dimensional datasets. Handling missing values stands out as a critical challenge during data preprocessing. Zhang et al. [9] proposed the NIIA imputation approach, an iterative scheme imputing missing data using information within incomplete instances. Luengo et al. [10] addressed the missing value problem through various imputation methods, focusing on a classification task and demonstrating improved accuracy with specific imputation methods. Lobato et al. [11] presented a solution by combining evolutionary computation techniques, specifically genetic algorithms (GA), for data imputation. Their multi-objective GA, named MOGAImp, is designed for mixed-attribute datasets. The data normalization is important for classifiers, neural networks and SVMs and comprises methods such as Min-Max normalization, Z-score normalization, and unit length scaling. These techniques are essential for ensuring optimal classifier performance, especially when dealing with large differences between feature values [4]. Discretization methods are discussed by Dougherty et al. [12], include equal size and equal frequency methods. These methods are relevant for handling continuous attributes in the preprocessing phase. Additionally, the significance of feature selection in preprocessing cannot be overstated. It is a crucial step for example for the k-NN procedure, involves identifying and eliminating irrelevant features by reducing data dimensionality. The selected literature gives an insight in the current preprocessing steps and their selected strategies, which are important for the upcoming choice of preprocessing steps and their detailed processing.

3. Experiment and Dataset

The following sections provide a detailed description of the experimental design, the question types, and the cheating methods allowed during the test.

3.1. Eye-Tracking Technology

With eye-tracking it is possible to record, measure and analyze a person´s eye movements to get insights into the visual focus and gaze patterns. It is used in psychology, user experience, and marketing. The resulting data provides an understanding of visual interaction with stimuli, which can be explored with analytic techniques. By employing tools such as heatmaps and saliency maps, the data can be visualized. During the recording, data, like fixations, saccades or their durations are captured in a structured format for analysis [13].

3.2. Experimental Design

The experiments have been conducted in the Eye-Tracking Laboratory at Hochschule Neu-Ulm on two data collection sessions on May 9th and June 15th, 2023 with 20 bachelor students from Prof. Sonja Köppl's lecture in Industrial Engineering ranging from 1st to 5th semester. In the laboratory are two external Tobii eye-tracking devices employed. The students are organized into groups of five, while one participant serves as a supervisor. This arrangement is implemented to simulate a real examination scenario. During the experiment, participants choose from three methods of cheating. First, a cheat sheet containing information relevant to the test questions, second, a mobile phone and third, the collaboration with the neighbor during the test. After each section of the test, the participants mark, which cheating method they have used. Also, they have the option to mark that

[h]

Features	Description
GazeLeftx, GazeLefty, GazeRightx, GazeRighty	The X and Y coordinates of the left and right eye.
PupilLeft, PupilRight	The pupil size of the left and right eye.
DistanceLeft, DistanceRight	The distance of the left and right eye.
CameraLeftX and Y, CameraRightX and Y	The camera coordinates of the left and right eye.
ValidityLeft, ValidityRight	The validity of the gaze data from the left and right eye.
Gaze X, Gaze Y	The X and Y coordinates of the participant's gaze point on the screen.
Interpolated Gaze X and Y, Interpolated Distance	Interpolated X and Y coordinates and distance of the gaze point.
Gaze Velocity, Gaze Acceleration	The velocity and accelaeration of the gaze point.
Fixation Duration	Time interval from beginning to the end of a fixation. Longer fixation durations indicate important or intriguing points.
Saccade Duration	Saccades occur between fixations and are rapid eye movements between points in the visual field. Saccade duration is the time it takes to move from one point to another.
Fixation X, Fixation Y	The X and Y coordinates of the participant's fixation point on the screen.

Table 1
Excerpt of Features of Eye-Tracking Dataset, description by [13], [14]

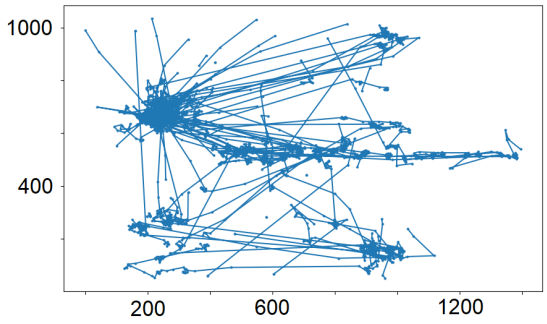

Figure 1: Gaze Plot of Task 5 by one participant

they have not cheated. The participants complete a test of 20 minutes of five distinct question types, from 2.5 to 4 minutes per question. The time was limited per task controlled by the software iMotions.

- **Definition Task:** A definition of a given term.
- **Transfer Task:**: Apply knowledge to solve a problem in a unfamiliar context.
- **Multiple Choice Task:** Select the correct answer from a set of options.
- **Single Choice Task:** Choose a single correct option from a list.
- **Programming Task:** Apply programming skills to solve practical problems.

Figure 1 provides an exemplary gaze plot for programming task of one participant. The coordinates of Gaze X are depicted on the X-axis and the coordinates of Gaze Y on the Y-axis.

3.3. Dataset and Feature Selection

Feature selection (FS) is an important step, especially for k-NN, SVMs and neural network training. FS identifies irrelevant and redundant features and reduces the dimensionality of the data to enhance efficiency. Features are generally categorized into relevant, irrelevant and redundant. An exemplary selection algorithm generates proposed feature subsets to find an optimal subset or an evolutionary algorithm that assesses the quality of the proposed feature subset by providing a 'measure of goodness' to the selection algorithm [15].

The recording of eye movements is conducted using the iMotions software. The original output dataset includes several features, the selected features are shown in Table 1. The selected features are chosen with a focus on their significance for analysis. This selection is an iterative process. As the analysis progresses, there may be adjustments made to the set of features. The data set from 20 participants is divided into each task, which results in total in a data size of 100.

4. Preprocessing Pipeline

By following a sequence of preprocessing steps, the data will be progressively enhanced and prepared for analysis of eye-tracking patterns. The rationale behind this order is to eliminate errors and noise upfront, then progressively refine the data to ensure accurate and meaningful results.

Data Cleaning is the initial step to ensure that the dataset is free of obvious errors, missing values, or inconsistencies. This step is crucial as it lays the foundation for reliable analysis by removing any erroneous or incomplete data points. The subsequent step involves outlier detection, aimed at identifying and handling outliers

early in the process. Outliers, which represent unusual eye movement data points, can significantly impact the analysis if not properly addressed. By identifying and addressing outliers early on, the preprocessing pipeline ensures the integrity of the dataset and enhances the accuracy of subsequent analyses. Following outlier detection, a low-pass filter is employed to reduce high-frequency noise or rapid fluctuations in the eye-tracking data. This step is essential for smoothing out erratic variations in the data, thereby improving its overall quality and coherence. Additionally, data smoothing is applied to further reduce noise, particularly in cases where minor fluctuations in eye-tracking measurements may obscure underlying patterns. Smoother data enhances the visibility of meaningful patterns and facilitates more accurate analysis. Towards the end of the preprocessing pipeline, normalization is performed to scale the data to a common range. Normalization is crucial for ensuring that all features are on a level playing field, facilitating easier comparison between different features or datasets. This step enhances the effectiveness of subsequent analyses by standardizing the data and mitigating the impact of varying scales.

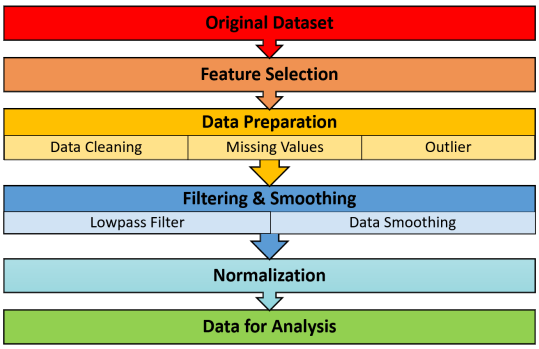

Figure 2: Pipeline for Data Preprocessing

Furthermore, it's important to note that this preprocessing pipeline represents an initial draft and serves as a starting point for experimentation and refinement. In future research, various alternative methods for handling missing values, outliers, and other preprocessing tasks will be explored. By experimenting with different approaches, such as various imputation techniques for missing values and outlier detection methods, the impact of these preprocessing variations on machine learning analyses with classifiers will be investigated. This iterative approach aims to refine and optimize the preprocessing pipeline for more robust and accurate analysis of eye-tracking data. The steps are visualized in Figure 2 [16].

4.1. Data Cleaning

In the data cleaning process, several steps are followed to ensure the dataset's integrity. The initial step standardizes column names by removing any leading or trailing whitespaces. The values of several columns with details of technical specification are deleted or converted to a numeric format, with non-numeric values being coerced into NaN (Not-a-Number) values. Rows containing NaN values are eliminated from the dataset. Data is filtered based on the $ValidityLeft$ and $ValidityRight$ columns, retaining only the rows where data for both eyes are valid (both columns have a value of 1). Furthermore, filters are applied to the $FixationDuration$ and $SaccadeDuration$ columns, preserving rows where these durations fall within predefined minimum and maximum thresholds. Any duplicate columns present in the DataFrame are eliminated. [17].

4.2. Missing Values

The issue of missing feature values is a critical challenge during data preprocessing. Classifiers such as neural networks and k-NN necessitate careful handling of incomplete information [9]. Numerous methods have been developed to handle missing data, like filling with the most frequently occurring value in the dataset, with the values of the same class, imputing with the mean value of the feature, developing regression or classification models to predict missing values based on other known features or considering unknown elements as complete new values for features containing missing values [4].

To identify and treat missing values in this dataset, a report revealed a high number of missing values in the 'Duration' column, which resulted 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) had missing values ranging from 32% to 49%. Therefore, an imputation by the mean was done for the further analysis. Third, the fixation data columns (with 55% to 56% missing values) require a closer examination of the cause of missing values. Therefore, they will not be removed, as they may only appear on single events.

```
# Treatment of Missing Values
# Duration
df = df.drop(columns=['Duration'])
# Gaze-Data
gaze_columns = ['Gaze X', 'Gaze Y', 'Interpolated
    Gaze X', 'Interpolated Gaze Y', 'Interpolated
    Distance', 'Gaze Velocity', 'Gaze Acceleration']
df[gaze_columns] = df[gaze_columns].fillna(df[
    gaze_columns].mean())
```

4.3. Outlier

Outliers also impact dataset integrity. With an outlier detection it is possible to identify and treat data points that deviate significantly from the majority of observations. These outliers can arise due to various factors, including measurement errors, participant distractions, or genuine deviations in gaze behavior. The methods for identifying outliers are categorized into statistics-based, distance-based, and density-based methods [18].

For this data set, the method used for identifying outliers is the "Z-score" of a data point, 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. $z = \frac{X-\mu}{\sigma}$

- Z - Z-score.
- X - individual data point.
- μ - mean of the data.
- σ - standard deviation of the data.

```
# Identification of outliers using Z-scores
z_scores = np.abs((df[col] - df[col].mean()) / df[col
    ].std())
outliers_mask = z_scores > zscore_threshold
df[col][outliers_mask] = np.nan
```

In the current data set, a set of gaze target columns ($columns\_to\_check$) is created for outlier detection, which include $Gaze\_X$, $Gaze\_Y$, $Gaze\_Velocity$, and $Gaze\_Acceleration$. The Z-score is determined for each data point in the columns and therefore, the means (μ) and standard deviations (σ) are calculated and a threshold value ($zscore\_threshold$) is established for identifying outliers. The values in each column are converted into floating-point numbers (float) using pd.to_numeric(). In cases where this conversion fails, these values are substituted with NaN (Not-a-Number). Any identified outliers are replaced with NaN values, indicating their removal from the dataset. Figure 3 depicts a visualisation of the outlier detection in the case of Gaze acceleration by one student completing the task type "Multiple Choice" [19].

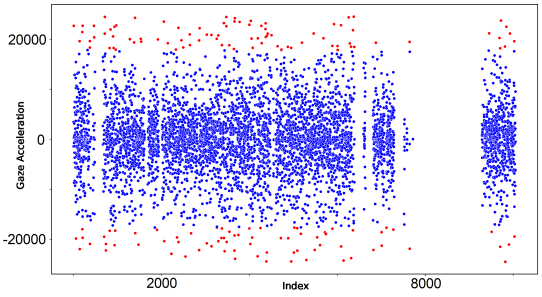

Figure 3: Outlier Detection for Gaze Acceleration

4.4. Low-pass Filter

Eye-tracking data is affected by noise, which can impact the accuracy of results. Filtering with a low-pass filter provides a method to reduce noise and enhance fixation stability. In the case of eye-tracking data, a low-pass filter attenuates high-frequency components while preserving low-frequency components to identify fixations and reduce noise. A commonly used low-pass filter is the Butterworth filter, an IIR (infinite impulse response) filter used in signal processing [20].

$$H(\Omega) = \frac{1}{1 + \left(\frac{\Omega}{\Omega_c}\right)^{2n}}$$

- $H(\Omega)$ - filter function in Laplace domain.
- Ω - frequency in the Laplace domain.
- Ω_c - cut-off frequency of the filter, the attenuation is 3 dB below the maximum level in the passband.
- n - filter order, which determines the steepness of the filter's roll-off.

The Butterworth low-pass filter works on discretizing the filter equation and applying convolution to the gaze data. The filtering process is performed in Python using SciPy. The coordinates of gaze points for the left and right eyes are filtered separately to reduce unwanted rapid eye movements and stabilize fixations. The low-pass Butterworth filter is created by using the signal.butter function. Metrics such as variance reduction and fixation stability are calculated and compared with the original unfiltered data [20], [21].

```
filter_type = 'lowpass'  # Low-pass filter
cutoff_freq = 2.0        # Cutoff frequency (Hz)
sampling_freq = 30.0     # Sampling frequency (Hz)
# Calculate normalized filter parameters
nyquist_freq = 0.5 * sampling_freq
normal_cutoff = cutoff_freq / nyquist_freq
# Create a low-pass filter
b, a = signal.butter(4, normal_cutoff, btype='low')
```

4.5. Smoothing

The smoothing is done with the Fourier Transformation to manipulate signals in the frequency domain. Fourier Transformation decomposes the original signal into sinusoidal components, each characterized by a specific frequency ω. It is defined by the following formula for continuous signals:

$$F(\omega) = \int f(t) \cdot e^{-i\omega t}\, dt$$

- $F(\omega)$ - frequency-domain representation.
- $f(t)$ - time-domain signal.
- ω - angular frequency (2π times the frequency).

When applied to eye-tracking data, it converts temporal gaze coordinates into the frequency domain by selecting the cutoff frequency. Unwanted frequency components are identified by their frequency characteristics and reduced in influence, preserving fixation-related components. The filtered data is then transformed back to the time domain for further analysis. This method is chosen for eye-tracking data due to its ability to analyze both periodic and non-periodic signals, making it suitable for data containing a mix of fixations and noise. The $Gaze\_X$ and $Gaze\_Y$ columns are cleaned by removing date entries and rows with NaN values. Fourier transformation is applied to both columns to smooth the data, reducing high-frequency noise [22], [21].

```
# Smoothing using Fourier transformation
        x = df['Gaze X'].to_numpy()
        y = df['Gaze Y'].to_numpy()
        x_smoothed = ifft(fft(x))
        y_smoothed = ifft(fft(y))
        df['Gaze X Smoothed'] = x_smoothed.real
        df['Gaze Y Smoothed'] = y_smoothed.real
```

4.6. Normalization

Disparities between feature values need careful treatment to ensure that all attribute values become appropriate. Data normalization is crucial for various classifiers, including neural networks, SVMs, k-NN algorithms, and fuzzy classifiers. The primary normalization methods for addressing this issue are Min–Max normalization or feature scaling in [0, 1] or [a, b], Z-score normalization or standardization or Unit length scaling. By normalizing eye-tracking data, different stimulus presentation durations are accounted for, allowing gaze data to exist within a uniform coordinate system [4].

Here, the normalization is employed by the *MinMaxScaler* from scikit-learn. Min-Max scaling was chosen for normalizing eye-tracking data due to its simplicity, interpretability, robustness to outliers, and preservation of data distribution. The resulting normalized gaze coordinates are represented as values between 0 and 1. For each gaze coordinate (X or Y) in the dataset: X_normalized = (X - X_min) / (X_max - X_min), where:

- X - original gaze coordinate.
- X_min - minimum value of coordinate.
- X_max - maximum value of coordinate.
- X_normalized - normalized gaze coordinate.

```
# Normalize the selected numeric columns
    scaler = MinMaxScaler()
    data[numeric_columns] = scaler.fit_transform(data
      [numeric_columns])
```

5. Conclusion

In this paper, a data preparation process is designed and described using Python in preparation for the upcoming analysis of eye tracking data. The process consists of several steps. Subsequently, a feature selection is performed to identify the most relevant ones for analysis. The data cleaning phase comprises data preparation to handle any missing values and to ensure data quality. One of the characteristics of eye-tracking data is its inherent noise and the presence of outliers. To address these issues, an outlier detection and treatment step is conducted, which helps in mitigating the effects of extreme data points. Additionally, filtering and smoothing techniques are applied, such as lowpass filtering and data smoothing, to enhance the interpretability of the gaze data. As eye-tracking data often contains high-frequency fluctuations, effective filtering helps extract meaningful insights. Normalization is an essential step to ensure that the data is on a consistent scale for comparisons. The resulting clean and processed dataset are then ready for the upcoming in-depth analysis.

5.1. Future Research

The next phase of the research is to evaluate and optimize the preprocessing steps applied to the eye-tracking data. That means, to experiment with different sequences of these steps to ensure that the resulting cleansed data is primed for analysis. This iterative process will require further investigation and refinement, with the goal of achieving optimal data quality. In this case, it is also aimed to generalize the results of the ordering of the steps of a data preprocessing pipeline also suitable for other eye-tracking data to make them ready for analysis. Te first observations reveal, that the filtering and smoothing process need to be refined, so that no data will be lost. Furthermore, the handling of missing data is still a detailed process, so that different strategies will be applied to different features.

Looking ahead to the future analyses, the choice of analytical techniques and the research questions at hand will reveal about patterns and predictions of cheating behaviour. Subsequently, the focus will now shift towards the analysis phase, where K-Means and classifiers like random forest or SVM will be employed and compared.

The study is limited by the current survey size, the data set comprises sensor data from 20 participants with a split for each task, in total 100. In the next time, more data will be collected through the conduction of upcoming experiments. Furthermore, the pipeline will be tested on other data sets, so that an review on a generalization will be possible.

References

[1] S. Janke, S. C. Rudert, A. Petersen, T. M. Fritz, M. Daumiller, Cheating in the wake of COVID-19: How dangerous is ad-hoc online testing for academic integrity?, Computers and Education Open 2 (2021) 100055. URL: https://www.sciencedirect.com/science/article/pii/S2666557321000264. doi:10.1016/j.caeo.2021.100055.

[2] L. Hillebrecht, Einflussfaktoren des Studienerfolgs im Vollzeit-Studium, in: Studienerfolg von berufsbegleitend Studierenden, Springer Fachmedien Wiesbaden, Wiesbaden, 2019, pp. 77–124. URL: http://link.springer.com/10.1007/978-3-658-26164-1_3. doi:10.1007/978-3-658-26164-1_3.

[3] Jennifer Landes, 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.

[4] S.-A. N. Alexandropoulos, S. B. Kotsiantis, M. N. Vrahatis, Data preprocessing in predictive data mining, The Knowledge Engineering Review 34 (2019) e1. URL: https://www.cambridge.org/core/product/identifier/S026988891800036X/type/journal_article. doi:10.1017/S026988891800036X.

[5] Huang, Lin, Chen, Fan, Review of outlier detection, Application Research of Computers 8 (2006).

[6] G. Ferraris, F. Manenti, Outlier detection in large data sets, Computers & Chemical Engineering 35 (2011) 388–390. doi:10.1016/j.compchemeng.2010.11.004.

[7] F. Angiulli, C. Pizzuti, Outlier Mining in Large High-Dimensional Data Sets, Knowledge and Data Engineering, IEEE Transactions on 17 (2005) 203–215. doi:10.1109/TKDE.2005.31.

[8] A. Ghoting, S. Parthasarathy, M. Otey, Fast mining of distance-based outliers in high-dimensional datasets, Data Mining and Knowledge Discovery 16 (2008) 349–364. doi:10.1007/s10618-008-0093-2.

[9] S. Zhang, Z. Jin, X. Zhu, Missing data imputation by utilizing information within incomplete instances, Journal of Systems and Software 84 (2011) 452–459. URL: https://linkinghub.elsevier.com/retrieve/pii/S0164121210003092. doi:10.1016/j.jss.2010.11.887.

[10] J. Luengo, S. García, F. Herrera, On the choice of the best imputation methods for missing values considering three groups of classification methods, Knowledge and Information Systems 32 (2012) 77–108. URL: https://doi.org/10.1007/s10115-011-0424-2. doi:10.1007/s10115-011-0424-2.

[11] F. Lobato, C. Sales, I. Araujo, V. Tadaiesky, L. Dias, L. Ramos, A. Santana, Multi-objective genetic algorithm for missing data imputation, Pattern Recognition Letters 68 (2015) 126–131. URL: https://www.sciencedirect.com/science/article/pii/S0167865515002883. doi:10.1016/j.patrec.2015.08.023.

[12] J. Dougherty, R. Kohavi, M. Sahami, Supervised and Unsupervised Discretization of Continuous Features, in: A. Prieditis, S. Russell (Eds.), Machine Learning Proceedings 1995, Morgan Kaufmann, San Francisco (CA), 1995, pp. 194–202. URL: https://www.sciencedirect.com/science/article/pii/B9781558603776500323. doi:10.1016/B978-1-55860-377-6.50032-3.

[13] J. Z. Lim, J. Mountstephens, J. Teo, Eye-Tracking Feature Extraction for Biometric Machine Learning, Frontiers in Neurorobotics 15 (2022). URL: https://www.frontiersin.org/articles/10.3389/fnbot.2021.796895.

[14] U. Ghose, A. A. Srinivasan, W. P. Boyce, H. Xu, E. S. Chng, PyTrack: An end-to-end analysis toolkit for eye tracking, Behavior research methods 52 (2020) 2588–2603. doi:10.3758/s13428-020-01392-6.

[15] J. Hua, Z. Xiong, J. Lowey, E. Suh, E. R. Dougherty, Optimal number of features as a function of sample size for various classification rules, Bioinformatics (Oxford, England) 21 (2005) 1509–1515. doi:10.1093/bioinformatics/bti171.

[16] A. Famili, W.-M. Shen, R. Weber, E. Simoudis, Data Preprocessing and Intelligent Data Analysis, Intelligent Data Analysis 1 (1997) 3–23. doi:10.3233/IDA-1997-1102.

[17] J. Brownlee, Data Preparation for Machine Learning: Data Cleaning, Feature Selection, and Data Transforms in Python, Machine Learning Mastery, 2020.

[18] S. Chen, W. Wang, H. van Zuylen, A comparison of outlier detection algorithms for ITS data, Expert Systems with Applications 37 (2010) 1169–1178. URL: https://www.sciencedirect.com/science/article/pii/S0957417409005843. doi:10.1016/j.eswa.2009.06.008.

[19] R. Domingues, M. Filippone, P. Michiardi, J. Zouaoui, A comparative evaluation of outlier detection algorithms: Experiments and analyses, Pattern Recognition 74 (2018) 406–421. URL: https://www.sciencedirect.com/science/article/pii/S0031320317303916. doi:10.1016/j.patcog.2017.09.037.

[20] S. Butterworth, others, On the theory of filter amplifiers, Wireless Engineer 7 (1930) 536–541.

[21] J. G. Proakis, D. G. Manolakis, Digital Signal Processing (3rd Ed.): Principles, Algorithms, and Ap-

plications, Prentice-Hall, Inc, USA, 1996.

[22] J. Makhoul, A fast cosine transform in one and two dimensions, IEEE Transactions on Acoustics, Speech, and Signal Processing 28 (1980) 27–34. URL: http://ieeexplore.ieee.org/document/1163351/. doi:10.1109/TASSP.1980.1163351.