UNITED NATIONS ECONOMIC COMMISSION FOR EUROPE

CONFERENCE OF EUROPEAN STATISTICIANS

Expert meeting on Statistical Data Confidentiality

26–28 September 2023, Wiesbaden

Checking Data Outputs from Research Works: a Mixed Method with AI and Human Control

Titouan Rigaud, Remy Marquier, Eric Debonnel, Pengfei Liu (CASD)

remy.marquier@casd.eu, titouan.rigaud@casd.eu

Abstract

CASD is a public organization which can be likened to a research data center: it hosts a large amount of highly detailed data from various administrations (tax data, health data, industrial data, etc.). These data pertain both to firms and people and are protected by different legal confidentiality regulations: medical secrecy, statistical secrecy, business secrecy and so on.

The aim of CASD is to make such data available for research purposes, duly authorized by the data producers. CASD developed a specific remote access system which enables the researcher to visualize data, to interactively process any calculations for their research, with a wide range of scientific and data analysis software tools (SAS, R, Python...) and technologies (Spark, Pandas..) at their disposal. Scientific article editing software are also installed in the virtual environment.

At the end of these processes, final outputs for publication must be anonymized. The enrolment and training sessions, which users must complete before accessing the data, include rules for output anonymization. The CASD data platform enforces a strong security policy, which includes an output checking procedure: even if each researcher is accountable for their research work, especially when it comes to the compliance of the outputs retrieved from their virtual research environment with GDPR or various confidentiality regulations, additional safeguards are in place. In particular, most of the outputs are manually controlled by CASD staff according to the rules data producers have established for this purpose. However, some projects hosted by the CASD infrastructure can ask for "automatic outputs" where manual checking is not systematic. This requires the authorization of every data producer involved. Usually, some outputs are forbidden in order to prevent massive disclosure (size and number of outputs are limited, encrypted files are forbidden). Manual checks are conducted randomly for other outputs, by CASD staff or by producers directly.

In order to make the system more efficient, CASD has developed a new tool, based on learning models (DNN / Boosting / Random Forests). The model are trained using approximately 13k previous results of manual output checks: validated or refused, and the reason for elements rejected. A list of controls (the details of which are confidential), for metadata elements (elements of context) as well as for content of files, is implemented for each output. It is important to point out that the algorithm is able to open more than fifty file format types (csv, sas...) and to check their content, especially for detecting individual data patterns. Once all this has been carried out, the algorithm provides a "risk score": if the score is too high, the output is manually checked by a CASD employee. The final check result is used to train the model to continuously improve its accuracy. First training runs of the model are based on manual checks, then continuously increased with semi-automatic checks.

Many disclosure attempts result from misunderstandings of confidentiality rules, despite all researchers having received security training before they can gain access to any data through CASD. To avoid potential bias from the model, a few other automatic outputs, which present a lower risk score, are randomly selected and also

checked. The results are then fed into the boosting model. This also allows the tool to detect which additional variables would be useful for the model: this kind of model is not static and evolves over time.
This tool is regularly refined: the aim is not to control every automatic output but only those which seem to be at high risk. Tests are conducted to reach the best level of efficiency regarding the risks of disclosure. CASD also has a sanction system which ensures the legal and personal accountability of users. After investigation, all breaches are subject to actions in close collaboration with the data producers or competent authorities.

1 Context of the study

In today's era of data-driven research, access to confidential and sensitive data plays a vital role in advancing scientific knowledge [Kar and Dwivedi (2020)]. However, ensuring the privacy and security of such data poses significant challenges. The five safes framework is often used to analyze access to confidential data, as formalized by Desai et al. (2016). CASD addresses this concern by providing researchers with access to confidential data while upholding stringent confidentiality protocols. The CASD serves as a secure environment for researchers, safeguarding the confidentiality of the data they work with. By implementing statistical secrecy measures, CASD prevents users from directly retrieving or storing data on their personal machines. Instead, the center offers secure terminals that establish a protected connection to data centers and virtual machines. These terminals grant access to a virtual environment that is isolated from the internet, ensuring the integrity and privacy of the data. To ensure a robust security infrastructure, the terminals are equipped with multiple layers of protective measures. These include a smart card system and biometric controls, which enhance the authentication process and restrict unauthorized access. All these measures can be considered part of the safe data and safe settings. Safe people and safe projects are usually held by statistical secret committees or producers.

Upon completing their statistical work, researchers are required to aggregate the results to eliminate any remaining confidential information [Fefferman et al. (2005)]. Once this aggregation is achieved, they can request an export, which initiates a verification process for the documents they wish to retrieve from the secure workspace. If the exported files meet the stringent criteria for statistical secrecy, the researchers receive the approved documents externally, enabling them to use the data for potential publication purposes. Exports are a zipped file. They can contain a lot of files, from different types and formats. CASD does not allow its users to encrypt the exported data themselves, nor to protect it by password. These securities are applied after the export is controlled. In this paper, we want to bring innovation to the fifth safe : safe outputs. This safe aims to bring security to results coming from the secured working space and the outside world.

In the realm of confidential data access, exports serve as crucial entities for researchers to retrieve approved documents from secure environments. By analyzing past exports, which have undergone manual control, a comprehensive database of accepted and refused exports can be established. Because we try to detect the refused ones, from now on, we will consider **refused exports as positive examples**. Consequently, an **accepted export will be considered a negative example**. This database holds the potential to train a predictive model that can forecast the acceptance of a given export. However, it is important to acknowledge the challenges associated with training a model on exports, as they lack a straightforward numeric representation. To overcome this hurdle, the development of an application capable of transforming exports into a set of numerical features becomes essential. These features must encapsulate the entire export and exhibit correlation with acceptance or refusal. This preliminary step, known as feature engineering, forms the foundation of our proposed methodology.

Feature engineering aims to bridge the gap between the non-numeric nature of exports and the requirements of predictive modeling. By constructing a set of carefully selected numerical features, we can encode the pertinent characteristics of an export and capture its underlying patterns. These features should encompass various aspects, such as the structural attributes of the export document, metadata information, and any relevant

contextual cues. The challenge lies in identifying the most informative features that encapsulate the salient information necessary for export acceptance prediction.

In this article, we propose a methodological framework from engineering features to model deployment. We will be discussing how CASD implemented a semi-automated system to control exports. Future perspectives are also given, including continuous training and new aspects to analyze exports.

2 Generating features

2.1 Contextual variables

The first step in analyzing the export is opening it and listing all the files. This step allows finding contextual variables like the size, number of files, different extensions used inside the export. It is also possible to retrieve other information, like the date, the project the export is attached to, etc. We call these variables contextual ones, because they describe the export in itself. These variables can almost never be invoked as a criterion to accept or refuse an export on their own. However, they can be correlated to refusal. For example, in the case of a user trying to export a database, variables like the size and number of files can be useful. We expect the model to recognize this example is a case of data leaking, because the number of file is equal to 1, and the size is very heavy. The user could also try to cut the file in separate pieces, and make several exports. We then expect the model to be able to identify that behavior thanks to the frequency variables that measure the number of exports made in a certain period of time. These variables are useful to try to recognize repeating schemas and give alert of an abnormal situation, and eventually proceed to control.

2.2 Managing the content of files

A system is needed to open the files and analyze their content. This is the most challenging step, as there is no universal system to open files and retrieve their content in a standard format. This is not possible because the analysis applied to an image, document, plain text, dataset, or Microsoft PowerPoint presentation needs to be different. However, developing a file format-based control system is also not feasible due to the high cost. Previous analyses have shown that CASD users daily utilize over 90 file formats, with significant disparities in their usage levels. CASD offers various statistical software, including R, Stata, SPSS, SAS, and Excel (with XLstat). Each software employs its own file format. The complete list of software offered is available here[1]. As it is not possible to develop a comprehensive file-based control system while aiming for maximum coverage, a compromise is made by leveraging similarities between formats. The goal is to open the files and transform them into a neutral format for analysis.
As you can see on plot 1, categories were built to try to achieve this objective. Some categories are more difficult to control than others. For example, we have to set up optical character recognition (OCR) to detect text inside images [Smith (2007)]. We use regular expressions to detect patterns inside texts. We believe grouping the files in categories will allow further versions of the system to perform better. It is sometimes difficult to decide in which category a file should belong. We have decided to regroup the files that can contain images and text under the category : document. Our classification relies on technical concerns primarily.

[1] https://www.casd.eu/en/technologie/environnement-de-travail/

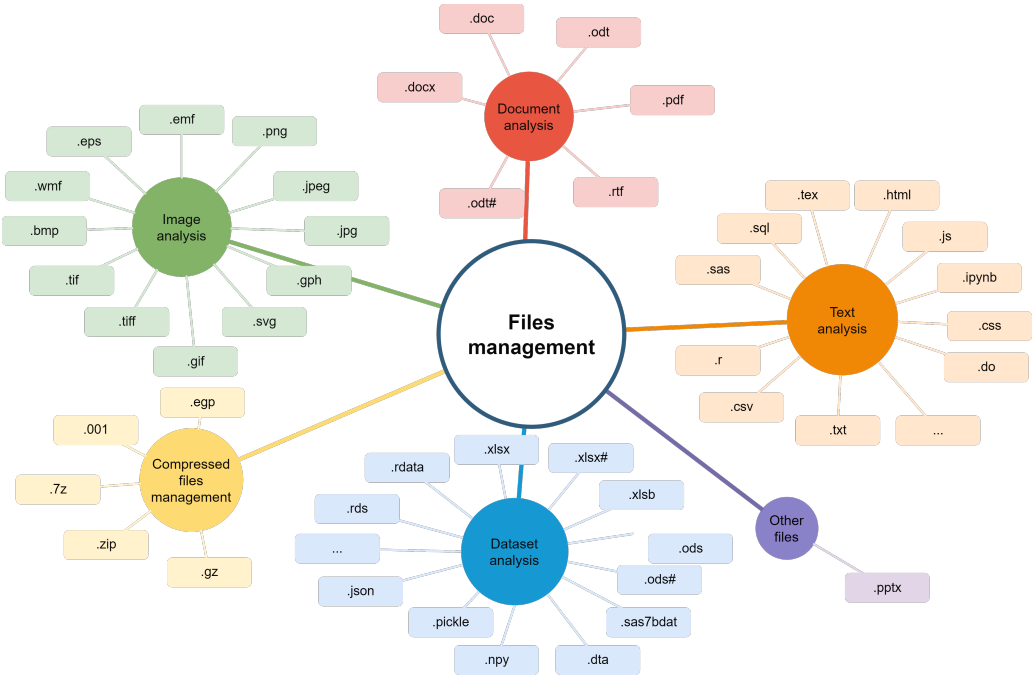

FIGURE 1. File management inside our system

Control algorithms are then developed for these standardized formats, enhancing responsiveness to new formats or their evolutions. The sole concern becomes the ability to open the file, rather than having to adapt the entire code. For example, the Parquet format is increasingly being used instead of compression formats offered by SAS or R. It performs better with Spark and occupies less space. To add it to the program, one simply needs to find a way to open the file and transform it into a Pandas DataFrame. The same control algorithm applied to formats like Microsoft Excel's XLSX can be applied to it. This approach requires developing fewer distinct control algorithms and delving deeper into their implementation.

The file-opening system allows access to the file content, enabling the search for specific information such as Siren and Siret numbers (firms ID). We use regular expressions to look for pieces of personal information left inside the files. Information like the number of lines, pages or even images size are used. It is also possible to search for column names or columns resembling headcounts to assess compliance with existing thresholds applicable to the datasets. The goal is to generate numerical variables that best describe the output and content of the files. It is important to note that the file format and the displayed extension may not always match. This is not necessarily intended to make the file uncheckable; some statistical programs export older Microsoft Excel files (.xls) that are actually XML files. However, attempting to open such a file with a Python library designed for Excel file opening would result in failure.

To address this issue, a system is needed to verify the consistency between the file content and its extension. By guessing the extension from the content of the file, we can compute a binary variable called consistency. This variable is a verification that the guessed extension is coherent with the extension given by the file. We can then use the guessed extension to overcome the issue described earlier. Using the opening system designed for the guessed extension rather than the original one often results in better opening percentage.

3 Training a model

3.1 Defining metrics

Doing a general analysis on our training dataset, we quickly noticed imbalanceness between the two classes to predict : there are far more accepted exports than refused ones. When dealing with imbalanced classes in machine learning datasets, it is crucial to carefully select appropriate evaluation metrics to assess model performance. Traditional metrics such as overall accuracy can be misleading in such scenarios, as they can be biased by the dominance of the majority class. Therefore, it is advisable to turn to more suitable metrics for measuring performance in a class imbalance context.
The work by Jeni et al. (2013) explores the evaluation of different metrics when dealing with heavily imbalanced datasets. The article concludes that most metrics show a degradation in evaluation capacity, except for AUC (Area Under the Curve), which maintains its performance level. AUC is a commonly used metric in classification tasks and represents the area under the Receiver Operating Characteristic (ROC) curve. This metric evaluates the model's ability to correctly rank positive and negative examples across various classification thresholds. AUC-ROC is robust against class imbalances as it takes into account the model's ability to establish a good ranking order between positive and negative examples, as demonstrated by Jeni.
In addition to AUC-ROC, two other useful metrics will be used: sensitivity and precision. Sensitivity (recall) measures the proportion of correctly identified positive examples among all actual positive examples. In our case, it will measure how many refused exports (positive) are found among all the exports that need to be refused. This metric is particularly important when the minority class is of specific interest or when minimizing false negatives is crucial. In our case, we are dealing with the detection of rare phenomena for alerting purposes. Precision assesses the proportion of correctly identified positive examples among all examples identified as positive by the model. This metric is important when the majority class can generate costly or undesirable false positives. It is also relevant for us, as performing a control action is costly. Therefore, a balance between precision and recall is sought, which is equivalent to finding a balance between false positives and false negatives. While Jeni does not directly use precision and recall in the study, they are combined into the AUC-ROC metric, which can provide a more comprehensive evaluation of model performance in a class imbalance context, considering both the correct classification of positive examples and the management of costly errors. The results obtained show that the precision-recall curve is less robust against imbalance compared to AUC, but can still be useful. Therefore, it will be used to confirm the results obtained by the previous metrics.

3.2 Training models in parallel

MLFlow is an open-source software component that enables recording and tracking of machine learning models in a database for comparison purposes [Zaharia et al. (2018)]. It fulfills our objective of comparing datasets, hyperparameters, and algorithms. Integration of MLFlow is done using a Python package manager. In the code, specific elements to be tracked, such as a model, a parameter, or a metric, need to be added. With each training performed, a new model is recorded. This model is associated with an experiment, which allows organizing models into categories. The application allows selecting the models to be compared, and also enables the selection of metrics and hyperparameters from the set previously recorded. With MLFlow, models can be compared in a much more detailed manner compared to using functions like GridSearchCV from the SKLearn package, for example. Another essential component of MLFlow is the management of model versions. It enables users to track and manage different versions of models developed over time. This facilitates collaboration, result replication, and decision-making using the best-performing models. Additionally, MLFlow provides features for model deployment, offering options to integrate models into applications or deploy them on production platforms. It is highly effective when combined with continuous deployment tools like ArgoCD.

This system allows evaluating the weight of features for each model with shap. For security reasons, the results of this part of the experiment will not be shown in this paper. However, we strongly advise the use of shap to evaluate the model, because it is an efficient way to determine if features are useful. It also gives important information on what the model uses to make its prediction. We can say that the total size of the export and number of exports made in one day (frequency-1day) do have an impact on the model and are interesting features to consider.

Argo Workflows is an open-source engine that enables task orchestration in parallel using Kubernetes technology [Dessalk et al. (2020)]. It allows defining steps and distributing the workload across containers. This technology facilitates container-based model training. In addition to ensuring that the failure of one model training does not affect others, it offers a significantly faster solution compared to sequential training. Multiple models can be trained in a matter of minutes. Argo Workflows fully leverages Kubernetes, benefiting from its scalability, resilience, and resource management capabilities.
One of its key advantages is the ability to orchestrate workflows based on containers. Workflows are defined using YAML files, describing tasks, dependencies, and relationships between different containers. This enables breaking down complex workflows into simpler steps for systematic and ordered execution. Argo Workflows also provides advanced features such as task parallelization, failure management, and error recovery. It allows controlling the flow of the production chain based on conditions and making pre-programmed decisions during execution.

3.3 Performance

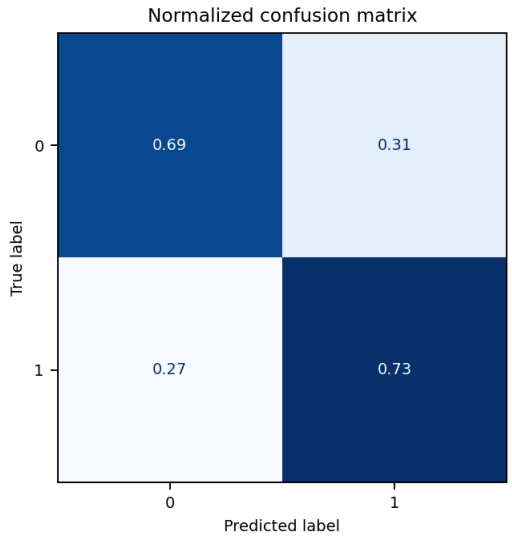

FIGURE 2. Confusion matrix of the model

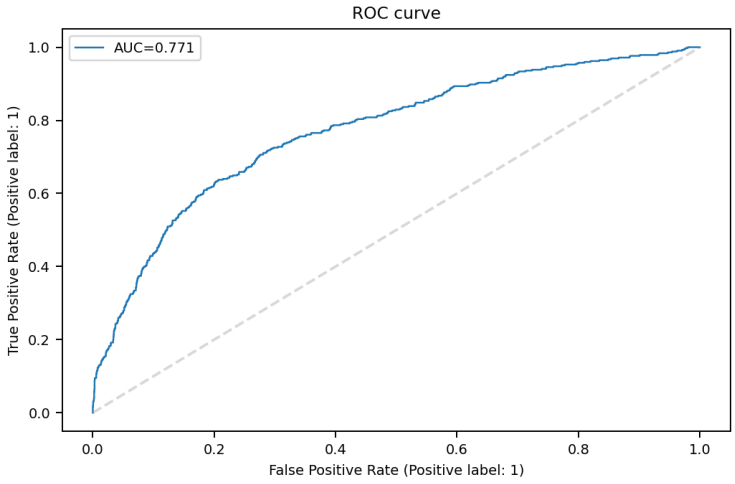

FIGURE 3. ROC curve of the model

We have compared different models. We present here the results for a XGBoost algorithm with a learning rate of 0.07 and 200 estimators (Chen and Guestrin (2016)). We believe for exports, this model is the most adapted to our dataset. It gave the best performance against the metrics presented in 3.1. A very important hyperparameter in our training phase was the maximum depth of estimators. We obtained satisfying results with low values, between 2 and 5.
The confusion matrix on the left represents the percentage of correctly predicted export among the accepted

ones (on the top) and the refused ones (on the bottom). For example : 73% of the refused exports were classified correctly the model, and 27% of them were accepted even if they should not have. Our analysis have shown that the performance level shown by the system is significantly better than random predictions. For example, with a rate of 15% of exports being refused, we can interpret the confusion matrix presented under in the following way : It would require to control 37% of total exports in total to find 73% of the refused exports with this method. However, that also means by controlling one export out of three, 27% of the exports, that would need to be refused, remain undetected.
On the right, we can see how many refused exports were detected (Y-axis), and how many accepted exports needed to be controlled to obtain this level of performance (X-axis). The roc curve is an interesting metric to evaluate a model, because it gives the capacity to calculate the percentage of refused exports caught for each level of global control. A higher false positive rate also allows catching more of the exports that actually need to be refused.

4 Using the model in a semi-automated system

4.1 Structure of the system

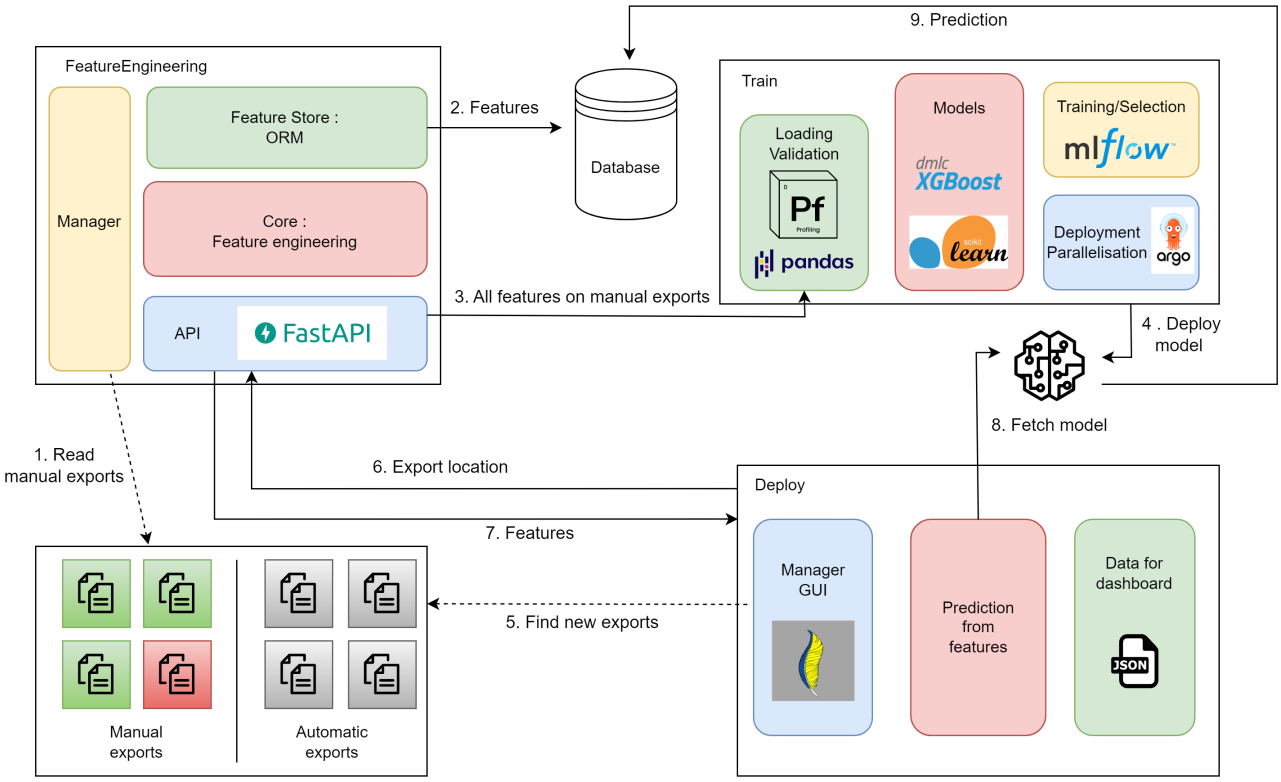

FIGURE 4. The complete process, from training the model on manual exports to using it on automatic ones

The Feature engineering application serves the purpose of providing a series of features for a given export, facilitating automated processing without human intervention. To enable seamless communication during export control, an API-based approach is adopted, ensuring efficient data exchange between applications. In

pursuit of high responsibility and reliability, we rely on the fastAPI framework, known for its robustness and performance. Furthermore, to ensure constant availability of features, we employ a database to store and retrieve them as needed. To execute the computations outlined in section 2.2, the export undergoes a decompression process, where it is extracted to a temporary folder. Subsequently, contextual variables are computed to capture additional information associated with the export. Finally, each file within the export is analyzed based on its detected type, allowing for comprehensive feature extraction.
The generation of a training dataset from a vast collection of manual exports requires significant computational resources. To address this challenge, an algorithm leveraging parallelization and database tracking mechanisms has been implemented. This approach ensures efficient progress monitoring and utilization of available computing power. Upon evaluating the results, we observed that the incorporation of parallelization techniques in export analysis expedited the generation of our training dataset by a remarkable factor of seven. We effectively distribute the computational workload across multiple computing resources, accelerating the feature extraction process. Features are then stored in our database, to make sure it can be cleaned and refactored as a dataset. (1-2)

Once a dataset was generated, we are able to train a model by sending the features to our second application (3). It is built for training many models parallelly, as explained in section 3.2. Using the capacities of both MLflow for comparison, and the Argo system for parallelization, we obtain the optimal model and are able to deploy it on the production structure.

Our system is now ready to analyze automatic exports. Therefore, once the batch algorithm finds new exports (5), the 3rd application contacts the API to obtain features and retrieve them (6-7). The application fetches the deployed model (8) to obtain a prediction on the automatic export (9). We now have all the analysis features, and the prediction in our export's database.

4.2 Perspective of continuous training

In the domain of export analysis and predictive modeling, continuous training offers a promising avenue for improving the performance of models trained on manual exports [Shahin et al. (2017)]. While manual exports provide a valuable training dataset, the availability of automatic exports presents an opportunity to further enhance model accuracy and adaptability. Continuous training involves incorporating these automatic exports into the existing model training pipeline, allowing for regular updates and adjustments based on real-time data. In this section, we explore the potential of continuous training as a future perspective in export analysis and discuss its implications for model improvement.
The integration of automatic exports into the training process brings several advantages. First and foremost, it enables the model to learn from a more comprehensive and diverse range of data. Manual exports, while valuable, may be limited in terms of quantity and coverage. By continuously incorporating automatic exports, the model becomes exposed to a larger volume of samples, capturing a broader representation of the export space. This exposure helps the model adapt to evolving trends, changing patterns, and emerging characteristics in the exported data.
Continuous training also addresses the challenge of model drift, which refers to the degradation of model performance over time due to changes in the underlying data distribution and behavior changes from researchers methods [Nayak et al. (2021)]. As the environment and export processes evolve, the model trained solely on manual exports may become less accurate or fail to capture emerging patterns. By incorporating automatic exports on an ongoing basis, the model can adapt to these changes, mitigating the impact of model drift and maintaining its predictive capabilities.

However, continuous training poses certain challenges that need to be carefully addressed. The quality and reliability of automatic exports must be monitored to ensure their suitability for training. Data preprocessing steps, such as outlier detection, data cleaning, and validation, are essential to filter out erroneous or unreliable automatic exports that may introduce noise into the training process. Furthermore, mechanisms for identifying

and handling concept drift in the automatic exports must be established to maintain model stability and performance. It is much more difficult to perform the preprocessing steps on automatic exports than on manual ones. Labelling the data will be done by manual control to make sure not to introduce bias.
From a technical standpoint, continuous training requires a robust infrastructure capable of seamlessly integrating automatic exports into the existing training pipeline. Automated data collection, preprocessing, and model retraining processes need to be implemented to enable regular updates and ensure the model's responsiveness to changing data patterns. As described, this is a "semi-automatic" output checks model, therefore the CASD staff will manually check exports identified as problematic by our model (to address the false positive issue) in addition to a sample of exports not detected (false negative issue).

5 Conclusion

In this study, we have developed a complete method to treat semi-automatically the exports of CASD. The first step involves obtaining the necessary numerical data. This step is crucial, as the quality and representativeness of the data will directly impact the performance of subsequent models. We need to transform an export, which is a compressed folder containing files, into numerical variables. This is done by using a unified file opener, capable of reading the content of over 90 file formats to analyze them. The next step is to define appropriate models and corresponding evaluation metrics. Depending on the specific problem, different types of models can be considered, such as neural networks, decision trees, boosting methods, etc. Additionally, it is important to choose appropriate metrics to evaluate model performance, taking into account class imbalance or other problem-specific considerations. Once the models are defined, we proceed with their training and comparison. This step involves splitting the dataset into training and test sets, adjusting model parameters, and evaluating their performance using the selected metrics. Model comparison allows us to select the optimal model that performs best on the test data.

We have presented the result for the best XGBoost model. It seems that in its current state, the model is obviously too weak to deal with outputs on its own. Even when controlling 1 export out of three, the roc-auc curve obtained tells us it would only be able to catch 70% of the exports that we would indeed refuse if we controlled all of them. However, we have high hopes for the future thanks to ideas that would improve significantly performance. We propose two solutions to improve our system and make the model capable of handling exports with more precision.
The first solution is to train the model on automatic exports data, after their labellisation by manual control (with a sample of exports). We plan to add another step to our methodology: continuous training. This step will be crucial to adapt to changes in production data, such as the emergence of new modalities or different behaviors. Detecting the Data Shifting phenomenon and regularly updating the models will be necessary to maintain optimal performance. This could be achieved by training new models or updating existing models to make them more robust and suitable for new data. This solution rely on the idea that training the model on a larger and more accurate dataset of exports will improve precision.
The second solution is to train the model at a different scale. We consider the data available at the level of an export to be quite limited (13k examples). The solution could be to train a model file by file. We have data available on more than 350k files, and their acceptance or not. That could make a huge difference in performance by multiplying learning examples by 25. This idea relies on our capacity to do feature engineering on a file instead of an export, which we have proven doable in this study. We cannot confirm, yet, that these variables are useful for prediction at a file by file scale.

References

Chen, T. and C. Guestrin (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794.

Desai, T., F. Ritchie, R. Welpton, et al. (2016). Five safes: designing data access for research. *Economics Working Paper Series 1601*, 28.

Dessalk, Y. D., N. Nikolov, M. Matskin, A. Soylu, and D. Roman (2020). Scalable execution of big data workflows using software containers. In *Proceedings of the 12th International Conference on Management of Digital EcoSystems*, pp. 76–83.

Fefferman, N. H., E. A. O'Neil, and E. N. Naumova (2005). Confidentiality and confidence: is data aggregation a means to achieve both? *Journal of public health policy 26*, 430–449.

Jeni, L. A., J. F. Cohn, and F. De La Torre (2013). Facing imbalanced data–recommendations for the use of performance metrics. In *2013 Humaine Association Conference on Affective Computing and Intelligent Interaction*, pp. 245–251.

Kar, A. K. and Y. K. Dwivedi (2020). Theory building with big data-driven research–moving away from the âwhatâ towards the âwhyâ. *International Journal of Information Management 54*, 102205.

Nayak, P. A., P. Sriganesh, K. Rakshitha, M. Manoj Kumar, P. B S, and S. H R (2021). Concept drift and model decay detection using machine learning algorithm. In *2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, pp. 1–8.

Shahin, M., M. A. Babar, and L. Zhu (2017). Continuous integration, delivery and deployment: a systematic review on approaches, tools, challenges and practices. *IEEE access 5*, 3909–3943.

Smith, R. (2007). An overview of the tesseract ocr engine. In *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, Volume 2, pp. 629–633.

Zaharia, M., A. Chen, A. Davidson, A. Ghodsi, S. A. Hong, A. Konwinski, S. Murching, T. Nykodym, P. Ogilvie, M. Parkhe, et al. (2018). Accelerating the machine learning lifecycle with mlflow. *IEEE Data Eng. Bull. 41*(4), 39–45.