OpenL2D: A Benchmarking Framework for Learning to Defer in Human-AI Decision-Making

Jean V. Alves[1,*], Diogo Leitão[1], Sérgio Jesus[1], Marco O. P. Sampaio[1], Javier Liébana[1], Pedro Saleiro[1], Mário A. T. Figueiredo[2,3], and Pedro Bizarro[1]

[1]Feedzai
[2]Instituto Superior Técnico, ULisboa
[3]Instituto de Telecomunicações
[*]corresponding author(s): Jean V. Alves (jean.alves@feedzai.com)

ABSTRACT

Public dataset limitations have significantly hindered the development and benchmarking of *learning to defer* (L2D) algorithms, which aim to optimally combine human and AI capabilities in hybrid decision-making systems. The development of these systems is primarily hindered by the high cost of obtaining human predictions for training and evaluation, leading researchers to often consider simplistic simulated human behaviour in their benchmarks. To overcome this challenge, we introduce OpenL2D, a novel framework designed to generate synthetic expert decisions and testbed settings for L2D methods. OpenL2D facilitates the creation of synthetic experts with adjustable bias and feature dependence, simulates realistic human work capacity constraints, and provides diverse training and testing conditions. We employ OpenL2D on a public fraud detection dataset to generate the *financial fraud alert review dataset* (FiFAR), containing predictions from a team of 50 fraud analysts for 30K alerted instances. We benchmark L2D baselines under a diverse array of conditions, subject to expert capacity constraints, demonstrating the unique, real-world challenges posed by FiFAR relative to previous benchmarks.

Background & Summary

Recently, an increasing body of research has been dedicated to human-AI collaboration (HAIC), with several authors arguing that humans have complementary sets of strengths and weaknesses to those of AI[1,2]. Collaborative systems have demonstrated that humans are able to rectify model predictions in specific instances[1], and have shown that humans, in collaboration with an ML model, may achieve complementary performance - a higher performance than the expert or the model on their own[3].

The state-of-the-art approach to manage assignments in HAIC is *learning to defer* (L2D)[4–13]. These are algorithms that choose whether to assign an instance to a human expert or an ML model, aiming to take advantage of their complementary strengths. In the multi-expert setting, the algorithm must choose between $J+1$ decision makers: a human out of a team of J experts, or the ML Model. Training L2D algorithms requires large amounts of data on human decisions, with multi-expert methods demanding the existence of every expert's prediction for every single instance in training[5,12]. To the best of our knowledge, the only suitable public dataset for training multi-expert L2D is the NIH Clinical Center X-ray dataset[5,13,14], aimed at detecting airspace opacity. It features ground-truth labels adjudicated by a panel of 3 radiologists, sampled from a team of 22 experts. The main drawback of this dataset is the rare overlap in the images experts review. Hemmer et al.'s study finds only 4 expert pairs with substantial shared image predictions, limiting their experiments to 2-radiologist teams.

Due to the unavailability of large datasets containing human predictions, and the cost of obtaining large amounts of data reliably annotated by human experts, L2D research is frequently conducted by using synthetic human subjects. A common approach is to use *label noise* to produce arbitrarily accurate expert predictions. Various authors[4,8,11] use CIFAR-10[15] where they simulate experts with high accuracy on the first 5 classes, but random accuracy on the remaining 5. The main drawback of these synthetic experts is that their expertise is only dependent on an instance's class, meaning that the methods in these benchmarks are not being challenged to learn nuanced and varied types of expertise. In line with previous work on learning with noisy labels[16,17], we argue that *instance-dependent label noise* (IDN) is more realistic, as human error is more likely to be dependent on the difficulty of a given task, and, as such, should also be dependent on the instance's features. Finally, we observe that current L2D benchmarks and methods disregard the existence of limitations in human work capacity, despite these being a challenge in real-world assignment systems[1,18].

In order to promote thorough benchmarking in the L2D community, we present OpenL2D, an open-source framework that allows for the generation of synthetic expert predictions, as well as training and testing scenarios for L2D on any tabular dataset. OpenL2D allows a user to (i) Create highly customizable synthetic experts, with control over feature dependence, bias towards

a protected attribute, and average performance. (ii) Generate realistic human work capacity constraints, limiting the number of cases that can be deferred to humans, should that be supported by the methods being tested (iii) Generate a variety of training and testing conditions, subject to user defined capacity constraints. These allow for research into development and testing of L2D methods subject to real-world problems, such as changes in human availability and limited amounts of human prediction data. Finally, we use OpenL2D on a publicly available fraud detection dataset[19], generating the FiFAR dataset, comprised of a team of 50 synthetic fraud analysts with realistic decision-making properties. To allow for training of any L2D method, we provide every expert's prediction for every instance in the dataset. By simulating these expert's work capacity constraints, we also generate a set of training scenarios where each instance only has the prediction of a single expert, simulating more realistic data availability. We test two capacity-constraint aware L2D methods by training them in each of these scenarios, and testing them under a set of varied capacity constraints, demonstrating how L2D systems can be thoroughly evaluated under realistic conditions using FiFAR.

Methods

Synthetic Data Generation Framework - OpenL2D

We now present our framework, which is represented in Figure 1. OpenL2D takes as an input a tabular dataset for a binary classification task, represented on the center left of the diagram. Based on a set of user-defined properties, it creates an expert team by generating the expert prediction table, which is the set of predictions of every expert for every instance in the dataset, represented in the top right. This set of predictions can be used to develop and evaluate L2D methods that require every expert's prediction to exist for every instance. Optionally, the user may define a set of work capacity constraints, to be applied over batches of instances, represented in the bottom right. The capacity constraints limit the amount of predictions that can be made by a given expert over a specified number of cases. These can be used to simulate a training dataset with scarce human predictions (one expert prediction per instance), limited by expert work capacity. If an assignment system can be configured to assign under capacity restrictions, the user may also generate capacity constraints to be applied in testing.

Dataset

Our framework is designed to function on any tabular dataset for binary classification. To use the framework, the user must define the categorical and numerical features present in their dataset, as well as the column corresponding to the label. Should the classification task at hand entail fairness concerns, the user may also define which feature corresponds to the protected attribute. Finally, if the user wishes to simulate experts who have access to additional information, such as an ML classifier's score, the user may specify the column corresponding to it.

Synthetic Expert Generation

In order to generate our synthetic experts' predictions, we will resort to an *instance-dependent noise* (IDN) approach, similar to previous work[16,17] on learning with noisy labels. We generate the prediction $m_{j,i}$ of a synthetic expert j on an instance i by flipping said instance's label y_i with probability $\mathbb{P}(m_{i,j} \neq y_i | \boldsymbol{x}_i, y_i)$. We aim to accommodate scenarios with significant class imbalance (e.g. medicine, financial fraud), as well as cost-sensitive tasks, where the cost of misclassification for false positives (FP) and false negatives (FN) differs (e.g., in medicine, false alarms are generally considered less harmful than failing to diagnose a disease). To do so, we simulate the probability of error separately for type I and type II errors. This introduces a dependence on the label but allows for better individual control of FP and FN misclassifications.

In some real-world applications of human-AI collaboration[1,20], an ML model is trained to perform the same classification task as the human expert. In these collaborative systems, the model score predicted for a given instance may be shown to the expert, together with other relevant information. It has been shown that the expert's performance is significantly impacted by showing the ML model's score along with the rest of the information pertaining to an instance[1,20,21].

We define the expert's probabilities of error, for a given instance, as a function of a pre-processed version of its features $\bar{\boldsymbol{x}}_i$ and an optional model score $\bar{M}(\boldsymbol{x}_i)$, so that the scale of each feature's values does not impact their relative importance. The probabilities of error are given by

$$\begin{cases} \mathbb{P}(m_{i,j} = 1 | y_i = 0, \boldsymbol{x}_i, M) = \sigma\left(\beta_0 + \alpha \frac{\boldsymbol{w} \cdot \bar{\boldsymbol{x}}_i + w_M \bar{M}(\boldsymbol{x}_i)}{\sqrt{\boldsymbol{w} \cdot \boldsymbol{w} + w_M^2}}\right) \\ \mathbb{P}(m_{i,j} = 0 | y_i = 1, \boldsymbol{x}_i, M) = \sigma\left(\beta_1 - \alpha \frac{\boldsymbol{w} \cdot \bar{\boldsymbol{x}}_i + \boldsymbol{w}_M \bar{M}(\boldsymbol{x}_i)}{\sqrt{\boldsymbol{w} \cdot \boldsymbol{w} + w_M^2}}\right) \end{cases} \tag{1}$$

Where σ denotes a sigmoid function. Each expert's probabilities for type one and type two errors are parameterized by five parameters: $\beta_0, \beta_1, \alpha, \boldsymbol{w}$ and w_M. These parameters are sampled, for each individual expert, from user defined distributions, generating a group of experts with similar properties.

The weight vector $\mathbf{w}$ embodies a relation between the features and the probability of error. The weights are shared in both formulas, with opposite signs, ensuring that if a feature contributes to increase the false positive probability, it will, in turn,

decrease the false negative probability. We chose to enforce this restriction in order to produce a more intuitive decision-making process: if an expert commits more false positives in a given region of the feature space, we also expect them to commit less false negatives, assuming they will be biased towards classifying more instances as positive in said region. Each component w_i of this vector is independently sampled from a user-defined distribution. To introduce bias against a protected attribute (i.e. age or gender), the user may separately define the distribution of the weight vector component w_p, where p denotes the index of the protected attribute in the feature vector. To impose a dependence on the model score of an independent ML model, the user may also consider setting $w_M \neq 0$, by defining its distribution. We normalize the feature weights so that we can separately control, with the α parameter, the overall magnitude of the variation of the probability of error due to the instance's features.

The values of β_0 and β_1 are, respectively, related to the false positive rate (FPR) and false negative rate (FNR) of a given expert. The user can define the distributions of FPR and FNR within an expert group, or, alternatively, the distribution of the expected misclassification cost of the expert's predictions, by providing the costs of FP and FN errors. This allows for full control of the expert's average performance, based on commonly used metrics. Our framework then calculates the values of β_1 and β_0 that yield the sampled FNR and FPR, or the sampled expected misclassification cost.

Various groups of experts can be generated within a team, by setting the sampling distributions within each group. This allows users to create more varied behaviour within a team, for example, by creating a group of experts with a high dependence on the model score ($w_M \sim N(\mu \gg 1, \sigma)$), and a group with low dependence on the model score ($w_M \sim N(\mu \ll 1, \sigma)$). A more detailed description of the role and sampling process of each parameter follows.

Feature Dependence Weights Generation

Our framework provides two possible methods for generating the feature weights for a group of experts. The user may manually define the distribution for each feature i, so that $w_i \sim \mathcal{N}(\mu_i, \sigma_i)$. Alternatively, we allow the user to define the desired distribution to be shared across all features, by defining the parameters of a "Spike and Slab" distribution[22]. A spike and slab prior is a generative model in which a random variable u either attains some fixed value v, called the *spike*, or is drawn from another prior distribution p_{slab}, called the *slab*. In our case, we set $v = 0$. That is, u is either zero, or drawn from the slab density $N(\mu_w, \sigma_w)$, where μ_w, σ_w are defined by the user. To sample the values of w_i, we first sample a Bernoulli latent variable $Z \sim Ber(\theta)$ to select if w_i is sampled from the *spike* or the *slab*. If $Z = 0$, w_i attains the fixed value $v = 0$, if $Z = 1$, w_i is drawn from the slab density p_{slab}. As such, the spike and slab prior induces sparsity unless $\theta = 1$, allowing for the generation of experts whose probabilities of error are swayed by a varying number of features, by changing the value of θ. The distribution of $w_M \sim N(\mu_M, \sigma_M)$ can be defined separately to allow for control of the expert's model-dependency. Should there be a protected attribute, the user is also able to define μ_p, σ_p, in order to impose fairness/unfairness on the simulated experts' predictions. With these two options, users can easily define a variety of feature dependencies across their expert team.

Controlling Variability and Expert's consistency

While the weight vector controls the relative influence that each feature has on the probability of error, the parameter α, in turn, controls the global magnitude of this influence. For $\alpha = 0$, the probability of error would be identical for all instances within a class, which is equivalent to *class-dependent* label noise, as used in previous L2D testing[4,8,11]. In turn, for very large α, the probability would saturate at the extremes of the codomain of the sigmoid function, 0 or 1, resulting in a deterministic decision-making process. The value of α can therefore be seen as a measure of the consistency, or intra-rater reliability, of a decision maker, as larger values of α will result in experts who are more likely to produce the same decisions when faced with similar inputs. The value of α is sampled from the user defined distribution $\alpha \sim N(\mu_\alpha, \sigma_\alpha)$.

Controlling Expert Performance

In a binary classification task, two metrics often used to evaluate a classifier's performance are the false positive rate (FPR) and false negative rate (FNR). Our framework allows users to set the distributions from which each expert j's target performance metrics are sampled, $T_{\text{FPR}_j} \sim N(\mu_{\text{FPR}}, \sigma_{\text{FPR}})$ and $T_{\text{FNR}_j} \sim N(\mu_{\text{FNR}}, \sigma_{\text{FNR}})$. Alternatively, the user may wish to measure expert j's performance by using the expected misclassification cost $\mathbb{E}[C]_j$ resulting from their decisions. In this case, the user would define the distribution from which the target $\mathbb{E}[C]_j$ is sampled $T_{\mathbb{E}[C]_j} \sim N(\mu_C, \sigma_C)$. We now describe how the values of β_{1_j} and β_{0_j} are calculated in order to achieve the desired performance in both of these cases, starting by assuming that the user wishes to define $T_{\mathbb{E}[C]_j}$.

In a binary task, the misclassification cost for an instance is either 0, if the prediction is correct, or given by the cost of a false positive c_{FP} or a false negative c_{FN} mistake. Therefore, the expected misclassification cost per instance is given by

$$\mathbb{E}[C]_j = \mathbb{E}_{(\boldsymbol{x}_i, y_i)\sim D}\left[c_{FP}\mathbb{P}(m_{i,j} = 1 \wedge y_i = 0) + c_{FN}\mathbb{P}(m_{i,j} = 0 \wedge y_i = 1)\right] \tag{2}$$

By dividing the entire expression by the constant c_{FN}, we obtain

$$\mathbb{E}\left[\frac{C}{c_{FN}}\right]_j = \frac{1}{c_{FN}}\mathbb{E}_{(\boldsymbol{x}_i, y_i)\sim D}\left[\lambda\mathbb{P}(m_{i,j} = 1 \wedge y_i = 0) + \mathbb{P}(m_{i,j} = 0 \wedge y_i = 1)\right] \tag{3}$$

where $\lambda = c_{FP}/c_{FN}$. As division by a constant will not affect the ranking of different sets of assignments, λ represents the only degree of freedom in the cost structure of the classification task at hand. For simplicity, we assume that $c_{FN} = 1$ and $c_{FP} = \lambda$. We can now calculate the expected cost resulting from an expert's predictions as

$$\begin{aligned}\mathbb{E}[C]_j &= \mathbb{E}_{(\boldsymbol{x}_i,y_i)\sim D}\big[\lambda\mathbb{P}(m_{i,j}=1\wedge y_i=0)+\mathbb{P}(m_{i,j}=0\wedge y_i=1)\big]\\ &= \mathbb{E}_{(\boldsymbol{x}_i,y_i)\sim D}\big[\lambda\mathbb{P}(m_{i,j}=1|y_i=0)\mathbb{P}(y_i=0)+\mathbb{P}(m_{i,j}=0|y_i=1)\mathbb{P}(y_i=1)\big],\end{aligned} \tag{4}$$

where the value of $\mathbb{P}(m_{i,j}=1|y_i=0)$ and $\mathbb{P}(m_{i,j}=1|y_i=0)$ are known from Equations 1, and the values of $P(y_i=0)$ and $P(y_i=1)$ are empirically calculated based on the ground-truth labels. We can therefore approximate the expected misclassification cost by using a partition of the dataset with N instances

$$\mathbb{E}[C]_j \approx \frac{1}{N}\sum_{i=1}^{N}\big[\lambda\mathbb{P}(m_{i,j}=1|y_i=0)\mathbb{P}(y_i=0)+\mathbb{P}(m_{i,j}=0|y_i=1)\mathbb{P}(y_i=1)\big]. \tag{5}$$

Assuming that the values of $\boldsymbol{w}, \alpha$ and w_M are set, the value of $\mathbb{E}[C]_j$ will only be a function of β_0 and β_1. We must then be able to calculate values of β_0 and β_1 so that we obtain $\mathbb{E}[C]_j = T_{\mathbb{E}[C]_j}$. Let us simplify the above equation

$$\begin{aligned}T_{\mathbb{E}[C]_j} &= \lambda\mathbb{P}(y_i=0)\frac{1}{N}\sum_{i=1}^{N}\big[\mathbb{P}(m_{i,j}=1|y_i=0)\big]+\mathbb{P}(y_i=1)\frac{1}{N}\sum_{i=1}^{N}\big[\mathbb{P}(m_{i,j}=0|y_i=1)\big],\\ &= \lambda(1-\mathbb{P}(y_i=1))\,\mathrm{FPR}_j+\mathbb{P}(y_i=1)\,\mathrm{FNR}_j,\end{aligned} \tag{6}$$

where, assuming all other parameters have been sampled, $\mathrm{FPR}_j = \mathrm{FPR}_j(\beta_{0_j})$ and $\mathrm{FNR}_j = \mathrm{FNR}_j(\beta_{1_j})$. There is a degree of freedom in this equation: if we choose the value of either β_{0_j} or β_{1_j}, the other variable must attain a specific value so that the target $E[C]$ is achieved. Inverting the expression, we obtain:

$$\mathrm{FPR}_j = \frac{T_{\mathbb{E}[C]_j}}{\lambda(1-\mathbb{P}(y_i=1))}-\frac{\mathbb{P}(y_i=1)}{\lambda(1-\mathbb{P}(y_i=1))}\mathrm{FNR}_j, \tag{7}$$

Meaning that any pair of target values for FPR_j and FNR_j that obey the above equation will yield $T_{\mathbb{E}[C]_j}$. We allow the user to set $T_{\mathbb{E}[C]_j} \sim N(\mu_C, \sigma_C)$, and to also set an upper bound and lower bound for the costs. To sample our FNR and FPR in order to achieve $T_{\mathbb{E}[C]_j}$ we sample a random value of T_{FNR_j}, and calculate T_{FPR_j} according to the above expression, ensuring both values belong to the interval $(0,1)$. We can now calculate the value of β_{0_j} and β_{1_j} to obtain the desired T_{FNR_j} and T_{FPR_j}. From Equations 1, we know that

$$\mathrm{FPR}_j(\beta_{1_j}) = \frac{1}{N}\sum_i \sigma\left(\beta_{1_j}+\alpha\frac{\boldsymbol{w}.\boldsymbol{x}_i}{||\boldsymbol{w}||}\right). \tag{8}$$

Note that, for notational simplicity we set $w_M = 0$ but the result is similar when $w_M \neq 0$. It is then possible to show that the function $\mathrm{FPR}_j(\beta_{1_j})$ is monotonically increasing,

$$\frac{\partial \mathrm{FPR}_j}{\partial \beta_1} = \frac{1}{N}\sum_i \sigma\left(\beta_{1_j}+\alpha\frac{\boldsymbol{w}.\boldsymbol{x}_i}{||\boldsymbol{w}||}\right)\left(1-\sigma\left(\beta_1+\alpha\frac{\boldsymbol{w}.\boldsymbol{x}_i}{||\boldsymbol{w}||}\right)\right) > 0 \quad \text{for} \quad \beta\in\mathbb{R}\,. \tag{9}$$

Since the function is monotonically increasing,and in a bounded to the interval $(0,1)$, then, for any target false positive rate T_{FPR}, then the following equation has an unique solution:

$$\mathrm{FPR}_j(\beta_{1_j}) - T_{\mathrm{FPR}_j} = 0 \quad \text{for} \quad T_{\mathrm{FPR}} \in (0,1)\,. \tag{10}$$

A similar reasoning applies for β_{1_j} and T_{FNR_j}. Finally, we can control an expert's FPR and FNR by solving these equations for β_{1_j} and β_{0_j}. To solve these equations, a partition of the dataset is utilized to calculate the empirical approximations of expected values as well as FPR_j and FNR_j, meaning that deviations from the target performance metrics may occur in the rest of the dataset due to temporal distribution shifts. Due to the monotonous nature of the function, and the uniqueness of the solution, we solve it through a bisection method[23].

Feature Preprocessing

For our simulation of experts, the feature space is transformed as follows. Numeric features in $\mathbf{X}$ are transformed to quantile values, and shifted by -0.5, resulting in features with values in the $[-0.5, 0.5]$ interval. This ensures that the features impact the probability of error independently of their original scale. Categorical features are target-encoded, that is, encoded into non-negative integers by ascending order of target prevalence. These values are divided by the number of categories, so that they belong to the $[0,1]$ interval, and shifted so that they have zero mean.

Capacity Constraints

Since most real-world applications of human-AI collaboration are limited by work capacity constraints, L2D evaluation benchmarks should include them. Humans are not able to process instances at the same speed as ML models, being limited in the number of cases they may process on any given time period (e.g., work day). A real-world assignment system must then process instances taking into account the human limitations over a given "batch" of cases, corresponding to a pre-defined time period. The user may divide the dataset into several batches and, for each batch, define the maximum number of instances that can be processed by each expert. In any given dataset comprised of N instances, capacity constraints can be represented by a vector $\boldsymbol{b}$, where component b_i denotes which batch instance $i \in \{1,...,N\}$ belongs to, as well as a human capacity matrix H, where element $H_{b,j}$ is a non-negative integer denoting the number of instances expert j can process in batch b. To define the capacity constraints, the user must set the following parameters:

- *Batch Size:* a single value denoting the number of cases in each batch. Different batch sizes allow for testing assignment methods over long and short horizons. To allow for the generation of different batches with the same size, the user may set the seed for the distribution of cases throughout batches.
- *Deferral Rate:* a single value representing the fraction of each batch that can be deferred to the expert team.
- *Distribution:* a single value $\in \{`Homogeneous', `Variable'\}$. Should the distribution be homogeneous, every expert will have the same capacity. Otherwise, each expert's capacity is sampled from $N(\mu_d = \text{Deferral\_Rate} \times \text{Batch\_Size}/N_{experts}, \sigma_d \times \mu_d)$, with user defined σ_d. This distribution is chosen so that each expert's sampled capacity fluctuates around the value corresponding to an homogeneous distribution. We set the standard deviation as $\sigma_d \times \mu_d$ so that the user can define the capacity variability as a proportion of μ_d. The user may set the seed for this sampling, allowing for different configurations with the same variability.
- *Number of Experts:* a single value defined as the number of experts that are available in each batch. This allows for testing several different group configurations without generating new experts, or to generate scenarios where not all experts work in the same time periods. The user may set the seed for the sampling of the available experts.

Generating Training Scenarios

In most decision-making tasks where a team of humans is involved, due to the limited human work capacity, each instance is reviewed by a single human worker[1] or by a small fraction of a larger team[24]. If we were to consider the predictions of an expert team gathered over the course of their employment (i.e. keeping track of social worker's decision[1]), we would not have the necessary data to train most L2D methods. This means that the data necessary to train an L2D system would have to be purposefully collected for the development of said system, which would entail high costs. L2D research should then focus on developing algorithms assuming the existence of scarce human predictions, that is, a single expert's prediction per instance.

In order to generate such a training scenario, a user needs to have completed the steps described previously, generating the synthetic expert predictions as well as a set of capacity constraints to be applied these instances (see also Figure 1). Finally, we allow the user to distribute the cases in the training environment throughout the experts according to a given function (i.e. random assignment), while respecting their capacity constraints. This creates a training set for the development of assignment systems with limited expert predictions, a key barrier to real-world implementation of L2D algorithms.

FiFAR

We now describe how we used OpenL2D to create a team of 50 complex and varied synthetic fraud analysts on top of a public financial fraud detection dataset, generating the *Financial Fraud Alert Review* (FiFAR) dataset.

Dataset and Use Case

We choose to use the publicly available bank-account-fraud tabular dataset, introduced by Jesus et al.[19]. This dataset is sizeable (one million rows) and boasts other key properties that are relevant for our use case, and common across many high-stakes decision making applications, which we will highlight over the following section. Each instance represents a bank account opening application, in which the features contain information about the application and the applicant. Each instance has a label that denotes if the instance is a fraudulent application (1) or a legitimate application (0). The task of a decision maker

(automated or human) when reviewing an application is to either accept (predicted negative) or reject (predicted positive) an application.

This data was generated by applying state-of-the-art tabular data generation techniques on an anonymized, real-world bank account opening fraud detection dataset. Consequently, this dataset poses challenges common in real-world high-stakes applications. Firstly, there is a high class imbalance, with a fraud prevalence of 1.1% over the entire dataset. Secondly, this dataset poses a cost-sensitive problem, where the cost of a false positive (declining a legitimate event) must be weighed against the cost of a false negative (accepting a fraudulent event). Note also that a positive prediction (predicted fraud) results in a declined bank account application. As such, false positives in account opening fraud can significantly hinder a person's life (with no possibility to open a bank account or to access credit). ML models trained on this dataset without measures to preserve fairness tend to raise more false fraud alerts (false positive errors) for older clients (≥ 50 years), thus reducing their access to a bank account. Therefore, this task entails a relevant fairness concern and is appropriate for evaluating the fairness of not only ML models, but also assignment systems. Finally, the dataset contains changes in the data distribution over time, often referred to as concept drift[25], which can severely impact the predictive performance of ML Models.

Task

The optimization objective stated in the BAF dataset[19] is a Neyman-Pearson criterion: the aim is to maximize a fraud detection classifier's recall at 5% FPR. This establishes an implicit relationship between the costs of a false positive and a false negative error. In a cost sensitive task, our optimization goal is to obtain a set of predictions $\hat{y}$ that minimize the expected misclassification cost $\mathbb{E}[C]$. In the previous Section, we demonstrate that the expected misclassification cost can be calculated by having access to a single parameter λ. As such, all that remains is to establish a relationship between the Neyman-Pearson criterion and the value of λ.

According to Elkan[26], we can establish a relationship between the ideal threshold of a binary classifier and the misclassification costs. For a given instance, the ideal classification is the one that minimizes the expected loss. As such, the optimal prediction for any given instance $\boldsymbol{x}_i$ is 1 only if the expected cost of predicting 1 is less than, or equal to the expected cost of predicting 0, which is equivalent to

$$(1-p)c_{FP} \leq pc_{FN}, \tag{11}$$

where $p = P(y=1|\boldsymbol{x}_i)$, that is, the probability that $\boldsymbol{x}_i$ belongs to the positive class. An estimation of the value of p is given by our Alert Model, in the form of the model score output for a given instance, which is an estimate of the probability that x_i belongs to class 1. The decision threshold t is then given by equalizing both sides of the expression. As such, it leads us to the value for λ:

$$(1-t)c_{FP} = tc_{FN} \Leftrightarrow \lambda_t = \frac{t}{1-t} = 0.057 \tag{12}$$

As the optimal threshold t for our Alert Model was chosen such that the Neyman-Pearson criterion is met, we now may plug the value of t into this equation, obtaining the theoretical value λ_t for this classification task. This value will be used when generating our synthetic experts.

Alert Review Setup

There are several possible ways for models and humans to cooperate. In previous L2D research, it is a common assumption that any instance can be deferred to either the model or the expert team. However, in real-world settings, due to limitations in human work capacity and legal restrictions, it is common to use an ML model to screen instances, raising alerts that are then reviewed by human experts[1,27]. In an alert review setting, humans only predict on a fraction of the feature space, that is, the instances flagged by the "Alert Model". We will train a L2D system to work in tandem with the Alert Model, by deferring the alerts in an intelligent manner.

Alert Model

We train the Alert Model to predict the fraud label on the first three months of the dataset, validating its performance on the fourth month. We use the LightGBM[28] algorithm, due to its proven high performance on tabular data[29,30]. The model is trained by minimizing binary cross-entropy loss. The choice of hyperparameters is defined through Bayesian search[31] on an extensive grid, for 100 trials, with validation done on the 4th month, where the optimization objective is to maximize recall at 5% false positive rate in validation. In Table 1 we present the hyperparameter search space used, as well as the parameters of the selected model. This model yielded a recall of 57.9% in validation, for a threshold $t = 0.050969$, defined to obtain a 5% FPR in validation. In Figure 3 we present the ROC curve for the Alert Model, calculated in months 4-8.

Expert instantiation

In this section we detail the parameter selection for the generation of the team of 50 synthetic fraud analysts. The properties of our simulated team's decisions, as well as the reasoning behind the selection of the expert's parameters, are evaluated in detail in the "Technical Validation" Section, where we will assess the validity of the FiFAR dataset by analysing how our experts compare to previous literature on human decision making processes, cognitive-biases, and high-stakes decision making performance. We also compare the synthetic decision-making process to that of real-world fraud analysts.

Our team of synthetic experts aims to simulate real-world behavior, by ensuring that our experts exhibit properties that have been documented in high-stakes decision-making. To select the distribution of the α parameter, we conducted a series of experiments in order to ensure that the intra-rater consistency was similar to that of real-world experts, selecting $\mu_\alpha = 15, \sigma_\alpha = 3.5$. For more information, see subsection "Internal Consistency of Human Decision Making Processes".

We define the expert's cost distribution by assuming that the analyst's performance is better than simply rejecting all the alerted applications $E[C] = 0.0496$. We define $\mu_C = 0.035, \sigma = 0.005$, setting an upper bound for the expert cost at $Top_C = 0.046$ and the lower bound at $Bottom_C = 0.02$. We also impose bounds for the FPR and FNR of experts, ensuring that $FNR_j \in [0.01, 1]$ and $FPR_j \in [0.01, 1]$, in order to guarantee that the label flipping process produces both types of errors for all experts. For more information, see subsection "Expert Performance Distribution".

When generating the weight vector w, our objective is to create a team of experts with a reasonable level of inter-rater agreement, meaning that the decision-processes in an expert team should be relatively similar, but not identical. To generate a realistic decision-making process, we first determine the correlation ρ between the dataset's features and the ground-truth label. To quantify the correlation between the label and categorical features, we use Cramér's V, which is a measure of association between two variables, producing a value between 0 and 1. We use Pearson's correlation coefficient to quantify the correlation between the label and continuous numerical features. We then select the 15 features with the highest correlation with the label (either negative or positive), and set their respective $\mu = \rho, \sigma = 0.1$, making each feature's influence on the probability of error scale according to their correlation with the label, but introducing variations in the decision making processes of different experts. All other feature weights are sampled from N(0,0.05), in order to introduce further variation in the decision making processes, but guaranteeing that the experts are not too dissimilar. In this way, we create a set of relatively similar decision-making processes allowing for the possibility of some outliers (see subsections "Inter-Rater Agreement" and "Feature Dependence and Algorithmic Bias"). The values pertaining to the distribution of the model score weight w_M were manually chosen to produce a dependency on the model score similar to real-world fraud analysts (see subsection "Feature Dependence and Algorithmic Bias"). The distribution of the protected attribute's weight w_p was also manually defined, chosen so that the experts exhibit varying levels of bias against older customer's applications (see subsection "Unfairness Against Social Groups"). The values of μ and σ for each of the feature weight distributions used to instantiate our expert team are available in Table 2.

Note that a positive sign for the Pearson's correlation coefficient will result in the probability of a FP rising as the value of the numerical feature rises, while a negative sign will produce the opposite effect. This means that the generated experts will tend to produce false positives/negatives if a feature's value is positively/negatively correlated with the label. As the Crámer's V only produces positive values belonging to the [0,1] interval, we cannot draw conclusions on the direction of the correlation. However, taking into account that the categorical features are target encoded in the data pre-processing, all weights are made positive, so that the nominal values for which the fraud prevalence is higher produce a higher likelihood of a false positive.

Training Scenarios

Our dataset includes 25 constrained training scenarios with only one expert prediction per instance. Each of these simulates a system where the alerted applications are randomly deferred to an expert out of a team of 10 experts (randomly sampled from the 50 total experts), assuming this system is deployed from months 4-7. This simulates the human prediction data collected over the functioning of a system employing random assignment.

These scenarios were generated by setting the "Distribution" as homogeneous, the "Deferral Rate" to 1, and the the "Number of Experts" to 10. The variation between these scenarios are created by combining 5 different seeds for the expert sampling, generating 5 distinct teams, with 5 different seeds for the distribution of cases throughout batches, resulting in 25 total training scenarios.

Testing Scenarios

The Testing Scenarios are defined for each of the 5 teams. For each team, 5 different capacity constraints are imposed in testing: 1 homogeneous and 4 variable with $\sigma_d = 0.2$. The number of experts for these capacity constraints are sampled by using the same *n_experts_seed* values defined in training. This guarantees that the sampled expert teams are the same in testing as in training.

In a real-world scenario, the capacity of the classifier would be orders of magnitude larger than that of the human experts, as the cost of querying a model is much lower than that of querying a human expert. Furthermore, we could expect the classifier to flexibly stand in for absent humans, should an expert, for any reason, not be able to predict on a given batch, for example.

In our testing scenarios, however, we aim to measure the ability of L2D algorithms to intelligently assign, by identifying the most appropriate decision maker for a given instance. To better assess these capabilities, we choose to define rigid capacity constraints, demanding that each expert be deferred, on average, an equal number of instances, in order to ensure that the assignment system is able to make the most of each member of the human-AI team. For all capacity constraints, the deferral rate is set to $10/11$, in order to ensure that, on average, each decision maker (human or classifier) predicts on the same number of instances.

Data Records

The FiFAR dataset [DATA CITATION] has been made available for public download through [SELECT PLATFORM]. The dataset also contains the model object for the Alert Model, allowing for reproducible generation of the synthetic expert predictions.

Dataset Structure

The structure of the FiFAR dataset is schematically represented in Figure 2. The dataset's root directory, "Data Records" contains several sub-folders corresponding to separate components of our dataset.

alert_data

In the "alert_data" directory, the file "Base.csv" corresponds to the first version of the base variant of the BAF dataset, available here. In the sub-directory "processed_data" we include the file "BAF_alert_model_score.parquet", which contains the instances pertaining to months 4-8 of the original "Base.csv", accompanied by the score of the Alert Model, predicting the probability that each instance is fraudulent. Finally, the file "alerts.parquet" is comprised by the instances flagged by the alert model, corresponding to the instances that the simulated experts will predict on.

alert_model

This directory contains the selected alert model, stored as a binary file "best_model.pickle", corresponding to a *LGBMClassifier* object. This file is included in order to ensure reproducibility, as *LGBMClassifier* models can produce distinct classifiers if these are trained in different operating systems or with a varying number of processor cores, for example.

The file "config.json" contains the information regarding the set of trials in the hyperparameter search of the Alert Model. This file is structured as a list of dictionaries, containing the keys: "number" - which identifies the trial, "value" - corresponding to the recall at 5% FPR in validation, a set of keys named "params_{lgbm_param}" for each *LGBMClassifier* parameter whose value is sampled for each trial (see Table 1), and, finally, the key "state", which identifies if the trial was completed successfully. The file "model_properties.pickle" contains the values of a few key aspects of the model that may be needed for later data generation or tests, stored as a dictionary. These include the value of the model's FPR ("fpr") and FNR ("fnr") in months 4-8, the model's threshold ("threshold"), and the false positive rate disparity across age groups ("disparity").

synthetic_experts

This folder contains the data pertaining to the generated synthetic experts. The file "expert_predictions.parquet" contains every expert's prediction for every single alerted instance (these instances are contained in the file "alerts.parquet", in the "alert_data" folder). Each instance is identified by its *case_id*, and each expert is identified by their *expert_id*. The predictions of each expert are binary in nature, with each expert having predicted that the instance is fraudulent (1) or legitimate (0). In the file "expert_parameters.parquet", we store the value of each expert's set of parameters, which were sampled in the data generation process. These values are stored in a table with where indexes correspond to each expert, and the columns are given by each of the feature weights, as well as the model score's weight, and β_0, β_1, α. In the file "prob_of_error.parquet", we store a table where each row contains the value of the probability of error of every expert for a given instance, calculated according to Equations 1.

testbed

This folder is divided into two sub-directories. The "train" folder contains a sub-folder for each of the 25 training scenarios. Each training scenario directory, represented by a sub-containes the set of collected expert predictions in the file "train.parquet", stored as a table containing the alerted instances from months 4-7. This table contains two added columns: "assignment", which contains *expert_id*, which identifies which expert an instance was deferred to, and "decision", which contains the expert prediction for said instance. We also include the capacity constraints of said training scenario as defined by the vector b, stored in "batches.parquet" and capacity matrix H, stored in "capacities.parquet".

The directory "test" contains a set of 25 sub-directories pertaining to each of the testing scenarios. Each of the test scenario sub-directories contains the capacity constraints to be imposed in the deferral process, represented by the vector b, stored in "batches.parquet" and capacity matrix H, stored in "capacities.parquet".

l2d

This folder contains the models used for our L2D benchmark. The sub-directory "classifier h" contains the *LGBMClassifier* object "best_model.pickle" corresponding to the classifier used in our L2D experiments, as well as the file "model_properties.yaml", containing the values of several metrics (misclassification cost, FPR, FNR) regarding the classifier's performance in training, validation, and testing, as well as the value of the *init_score* used for the base estimator in training. (see Section "Technical Validation" for more information).

The sub-directory "expert_models" contains 2 folders: "OvA" and "DeCCaF", one for each L2D method tested in our benchmark. The "OvA" folder has 25 sub-directories, one for each training scenario in our Dataset. Within each sub-directory, there is one folder per expert in the training scenario's team, as the OvA method involves training one classifier per expert. These folders contain the *LGBMClassifier* object corresponding to said classifier, as well as the history of sampled hyperparameters during the training process, stored in the file "config.yaml". The "DeCCaF" folder also has a sub-directory for each training scenario in our Dataset. As the DeCCaF method trains one model for the entire expert team, these folders contain the *LGBMClassifier* object corresponding to the team model, as well as the history of sampled hyperparameters during the training process, stored in the file "config.yaml".

deferral

This directory contains the results of each of the deferral methods tested in our benchmark, stored in the following sub-folders."Only_Classifier" and "Full_Rejection" contain one sub-directory for each of the testing scenarios, each containing two files: "assignment.parquet" which contains the *expert_id* of the expert/classifier to which each instance was deferred, and "decision.parquet", which contains the set of decisions produced by the assignment system.

The "Random" directory contains 5 sub-directories: {"1","2","3","4","5"} one for each seed set for the random deferral process. Within each of these sub-directories, there are, again, 25 sub-folders containing "assignment.parquet" and "decision.parquet" for each of the testing scenarios.

The "OvA" and "DeCCaF" directories contains 25 sub-directories: one for each training scenario. Within each of these sub-directories, there are 5 sub-folders containing the decisions produced for each of the 5 testing scenarios that apply to the expert team considered in training. These sub-folders also contain "assignment.parquet" and "decision.parquet" for each of the testing scenarios.

Technical Validation

Realism of Synthetic Experts and Applicability in L2D

In this section we evaluate the properties of our synthetically generated experts, and how they compare to existing literature regarding real-world experts in high-stakes decision making tasks. We also conduct a set of experiments to assess how the decision making processes of real-world fraud analysts compare to our simulated fraud experts.

Internal Consistency of Human Decision Making Processes

When making a decision, experts follow an internal decision process based on the available information as well as their experience or training. However, even highly educated and trained individuals are still subject to flaws that are inherent to human decision making processes, one of these being inconsistency.

In a study involving seven experienced software professionals[32], these were tasked with estimating the effort needed to complete sixty software development tasks over a period of three months. Six of the sixty tasks were estimated twice, in order to test the consistency of the experts. The authors find that the mean difference of the effort estimates of the same task by the same estimator was as much as 71%, with the correlation between the corresponding estimates being 0.7. Expert decision making processes may also be affected by exogenous factors, such as the decision maker's mood, which are unrelated to the features of the instance on which the expert is predicting. A well known example of this phenomenon is commonly referred to as the "hungry judge effect", which states that judges are more inclined to be lenient after a meal, becoming more sever until the next break. It was shown[33] that the percentage of favorable rulings drops gradually from $\approx 65\%$ to nearly zero within each decision session and returns abruptly to $\approx 65\%$ after a break.

Evaluating the consistency of the decisions of a given expert, often referred to as "intra-rater consistency", involves using a measure of the likelihood that a decision maker will produce a different decision when faced with the same inputs. In our framework, the probability that an expert will commit a mistake is calculated according to Equation 1. If an expert's probabilities of error are consistently ≈ 0.5, the resulting decision process would be entirely random. Sampling the expert decisions twice, with the same inputs, would lead to wildly different sets of predictions, meaning that the expert would have low intra-rater consistency.

To allow for control of the consistency of decision makers, we introduced the α parameter (see section "Methods"), which controls how heavily the probability of error is swayed by the instance's features. To measure the internal consistency of

our synthetic experts over the entire dataset, we calculate their probabilities of error for each instance, and sample 10 sets of decisions for each expert, where the ground-truth is flipped with the calculated probability of error. We then calculate the value of Cohen's κ, a common measure of agreement between two sets of predictions, for each possible pair of decisions, taking the average value to be our measure of the intra-rater agreement. If the probabilities of error are closer to the edges of the interval $(0,1)$, then we expect the resulting sets of decisions to be much more similar between them. Note however, that the relationship between the intra-rater consistency and α is also impacted by different values of β_0 and β_1. Should the values of β_0, β_1 be 0 or 1, $\alpha = 0$ would also result in a deterministic process, as such, an increase within in the value of α will not always result in higher intra-rater reliability, particularly for lower values, but we expect to see an increase in intra-rater reliability as the value of α rises. In Figure 7 we demonstrate that there is indeed a correlation between the value of α and the intra-rater reliability of a given expert.

In a high-stakes scenario, experts are highly trained, and therefore expected to exhibit substancial to perfect intra-rater agreement (Cohen's κ of 0.6 to 1). Note, however, that the level of intra-rater agreement is not only a function of the expertise of the rater, but also of the difficulty of the classification task. In a study involving a series of cardiological diagnostic tests[34], intra-rater agreement of cardiologists was found to be consistently above $\kappa = 0.8$ for some medical tests. However, detecting some conditions was much harder even for experienced doctors, resulting in a lower intra-rater reliability $\kappa \in [0.4, 0.85]$. This was especially noticeable when the disease prevalence was very high or very low, as is the case in our financial fraud scenario. In Figure 8 we show the distribution of intra-rater reliability throughout the expert team. Assuming our team of financial analysts is comprised of highly trained experts, performing a challenging task, this distribution seems realistic.

Testing L2D systems under variable intra-rater agreement can pose significant challenges, as these methods rely on modeling the expert behavior in order to predict their likelihood of being correct on a given instance. If a team is comprised of experts with low intra-rater consistency, they will have more unpredictable behaviour, harming the performance of L2D systems. In conclusion, having a team with variable levels of intra-rater consistency allows users to stress test L2D systems.

Inter-Rater Agreement

In high-stakes scenarios, experts are usually highly educated individuals, who have undergone extensive training to perform their decision making task. Consequently, we expect the decision processes of different experts to be somewhat similar, according to commonly accepted practices or criteria in their domain of expertise. In other words, we expect to observe substantial levels of inter-rater agreement, a measure of the likelihood that two separate experts will produce the same prediction for the same input.

In the same study referenced in the previous section, involving a team of cardiologists[34], the values of inter-rater agreement, as measured by Cohen's κ, were as high as $\kappa = 0.97$. This value is, again, highly dependent on the classification task at hand, and its difficulty. It was found that in a more challenging diagnostic task inter-rater reliability was, on average, $\kappa = 0.49$. The proportion of agreement between experts was also reported at $p_a = 66.29\%$.

To determine the inter-rater agreement of our team of experts, we calculate the Cohen's κ as well as the proportion of agreement between pairs of experts. In Figure 9, we display a heatmap showing the value of κ for each expert pair. As the classification task is highly imbalanced, κ values are expected to be lower, as the expected level of agreement by chance rises. We also report the values for the proportion of agreement, calculated as the fraction of instances where the predictions of a pair of experts are identical. These values, for each expert pair, shown in Figure 10. In this set of results we can immediately notice how some of the experts within our team, such as experts #3 and #4, have a particularly unique set of decisions, yielding very low agreement with most of the team's members (an average of $\kappa = 0.33$ and $\kappa = 0.28$, respectively), but having high agreement between them ($\kappa = 0.71$). Plotting the distribution for the two agreement metrics, in Figures 11,12, we can see that the agreement between experts varies substantially, with $\kappa \in [0.06, 0.79]$, and $p_a \in [0.32, 0.95]$. We also note that the average values for the inter-rater agreement are similar to the real-world values reported in the previously mentioned study[34], with $\mathbb{E}[\kappa] = 0.49$ and $\mathbb{E}[p_a] = 0.75$.

When implementing L2D methods, it is desirable for inter-expert agreement to not be perfect. Should the experts perfectly agree on every instance, there would be no benefit in attempting to intelligently assign within that expert team. It is therefore beneficial, for a L2D system, that expert agreement be low, as long as the intra-rater consistency and the rater's performance is high, as this would result in a high number of instances where we can benefit from selecting the expert that is most likely to be correct. Our dataset allows users to extensively test L2D systems under high or low values of inter-rater agreement, by sampling experts in order to create teams with said properties.

Expert Performance Distribution

In real-world high-stakes scenarios, highly-trained experts within a field have been shown to have varying degrees of expertise. In a study[24] involving a panel of 8 US board-certified ophthalmologists with the highest values for intra-rater consistency, the authors find that their decision processes vary significantly as measured by their Recall and FPR. While we can not derive a cost structure for the classification task, and can not, therefore, evaluate the difference in performance throughout the team, this study shows that the Recall and FPR of highly trained experts varies drastically throughout the team, with FPR varying

between $\approx 1\%$ and $\approx 10\%$, while the recall varies between $\approx 35\%$ and $\approx 99\%$.

In Equation 7, we establish a relationship between the expected misclassification cost for a given expert's decisions $\mathbb{E}[C]_j$ and their respective FPR_j and FNR_j. Let us consider the "Full Rejection" approach, where we reject all the alerts. If we substitute the expected cost of rejecting all the alerts into Equation 7, we obtain the set of FPR and FNR values that result in the same cost in the test split. As such, in a plot where the x axis is the FNR and the y axis is the FPR, all possible combinations of FPR and FNR leading to the same misclassification cost as rejecting all the alerts are represented by a single line. We exemplify this in Figure 4, where the green area represents the combinations of FPR and FNR corresponding to a lower expected misclassification cost, while the red area represents a higher misclassification cost. In this figure, the blue dots represent each expert's performance metrics.

Every expert's performance, as measured by the expected misclassification cost, is assumed to be better than rejecting all alerts. We choose to enforce this assumption due to the fact that, in some current applications of alert review systems, deferral is done randomly to humans based on their availability[1]. In these cases, if the experts do not perform, on average, better than just rejecting every single alert, they would not be useful for the system as a whole, and there would be no reason to integrate them in the system. As such, to make our setup more realistic, we assume that our experts perform better than simply rejecting all the alerted applications. In Figure 5 we present the expected misclassification cost distribution calculated over months 4-8 in the dataset. Our dataset allows users to test L2D systems under a variety of team performances: by sampling only the highest or lowest performing experts, for example, one could test the impact that the average human performance has on the performance gains of such systems.

Unfairness Against Social Groups

It is a well-known fact that even highly-trained human experts can be unfair. A recent meta-analysis of hiring practices[35] shows that discrimination against candidates with disabilities, older candidates, less physically attractive candidates and candidates with salient racial or ethnic traits is still an issue in Europe and the United States. The bias of human decision-makers is often influenced by the relationship between the social group and the ground-truth label in a given classification task, which is impacted by underlying social structures leading to differences across social groups[36]. Let us consider the case of financial loan applications, where we assume a ground truth label of $Y = 1$ is the undesirable outcome, signifying that the applicant is not able to pay off the loan (i.e. rejected application). In the lending setting, even if the sampling process of loan applicants' income is unbiased and accurate, the data could still suffer from societal bias if some social group, for example women, systematically received lower wages, making women less likely to be able to pay back a loan. In this case, a smaller fraction of women would have a positive ground truth label, meaning that a perfectly accurate algorithm would reject a higher proportion of women's applications when compared to men's applications, due to the underlying socio-economic factors that lead to women having lower incomes.

As such, to measure fairness, we need to take the prevalence of each group into account. To do so we will use a predictive equality (pe) metric, which, for a fair decision maker, should be $pe = 1$.

$$pe = \frac{\mathbb{P}(\hat{Y} = 1 | Y = 0, G = 0)}{\mathbb{P}(\hat{Y} = 1 | Y = 0, G = 1)}, \tag{13}$$

Enforcing $pe \approx 1$ is often referred to as *predictive equality*[37]. This requirement ensures that people from both privileged and underprivileged groups with a negative label are equally as likely to be incorrectly classified. In the case of loan applications, this requirement ensures that people who are able to pay off the loan are granted said loan with the same likelihood independently of the social group they belong to. Going back to the hiring discrimination example, if candidates with similar qualifications are more likely to be accepted/rejected by virtue of their social group, it can be concluded that $pe \neq 1$, as the outcome ($\hat{Y}$) given the qualifications of the worker (Y) are influenced by their social group.

In our dataset, two social groups are defined as "Older" customers (≥ 50 years old) and "Younger" customers (< 50 years old). The fraud prevalence within the alerts for the Older customers is $\approx 13\%$, while the prevalence for Younger customers is $\approx 11\%$. As such, we assume that experts can have varying levels of bias, but that most experts would exhibit bias against the older customers. In this case, we wish to ensure that no social group is being incorrectly denied a bank account at a higher rate than other social groups. In Figure 6, we demonstrate the distribution of each of the expert's predictions pe, as well as the group that they are biased against (the group with the highest false positive rate). Our expert team is consistently biased against older customers, at varying degrees, with some experts being almost 50% more likely to incorrectly reject an older customer's application. Therefore, our dataset can be used to explore the impact that L2D systems have in the overall fairness of the produced decisions. Furthermore, it may also be used to develop fairness aware methods, as L2D systems can have an active role in intelligently assigning instances in a way that promotes fairness (i.e. deferring the older customer's applications to the unbiased experts). While this is a concern that goes unaddressed by most recent publications in this area, it was explored by the original authors in their paper[10], where the authors consider adding a fairness regularization term to the loss function of their L2D system.

Feature Dependence and Algorithmic Bias

In this section we focus on validating whether our experts' feature dependence and algorithmic bias (dependence on the alert model's score) are realistic. To conduct this experiment, we will use a set of real-world financial analyst decisions. This dataset is private and, therefore, can not be made available in this publication. These analysts' decisions are made based on a set of features that are shown to them, including the score of an ML model trained to predict fraud. This dataset only contains one expert's prediction per instance, and, therefore, could not be used to evaluate the properties of our synthetic analysts in the past sections. Furthermore, as this dataset contains only the decisions made by the experts, and not the ground-truth label, we cannot analyse the dependency between the features and the probability of error, or the distribution of experts performance. We will use this dataset to evaluate the dependency between the decisions of each expert and the features presented to them.

To establish a level of dependency of an expert's decisions on a given feature, we fit a *LGBMClassifier* to the decisions of each expert, and measure each feature's importance within the resulting model. The relative importance of each feature is calculated as the proportion of information gain, in training, that can be attributed to splits based on said feature. We repeat this process with our synthetic experts' decisions, in order to compare the results. Each real expert model was trained by splitting the data pertaining to each expert into 60-20-20 splits. As the set of decisions pertaining to each of the 20 real-world experts had an average of 10K instances, we also trained the models for our synthetic experts with 10K predictions, following the same train-validation-test splitting strategy, ensuring that the training set was sampled from months 4-6, the validation set was sampled from month 7, and the test set was sampled from month 8. These models were trained by minimizing the binary cross-entropy, stopping training if the validation ROC-AUC did not improve for 10 rounds. The choice of hyperparameters is defined through random search on the hyperparameter space represented in table 3, where the optimization objective was to maximize the AUC in validation.

In Figures 13 and 14 we display the ROC curves in the test set for the models trained, where we observe that both sets of *LGBMClassifiers* were able to model the human and synthetic expert behavior. However, the values for the area under the curve (AUC) are much lower for the real-world experts. In Figures 15 and 16, we observe that the maximum value of AUC observed for the real experts is lower than the lowest AUC observed for the synthetic experts. The source of this disparity in the predictability of the expert behaviours may be due to several factors. It may be the case that the real world financial analysts have a lower intra-rater consistency than the generated experts, making their decision process inherently less predictable. However, it is also possible that the decision process of the real-world experts is significantly more complex than our simulated experts, making it harder to model with limited data. As we have no access to the ground truth labels, and each instance only has the prediction of a single expert, we are not able to determine the performance and intra-rater reliability of the real-world experts, making it impossible to identify the source of this disparity.

In Figures 17,19, we represent the distribution of feature importances for the *LGBMClassifiers* modeling the team of 20 real-world fraud analysts. The most important feature in predicting the decisions of the experts is the *Model Score*, which is to be expected as the value of the model score will not only influence the decision of financial analysts[20], but is also strongly correlated to the probability of fraud. We can see that although some features exhibit higher importance across all experts, the distribution of feature importances is relatively even, with some outliers, which can be interpreted as experts whose decisions are unusually dependent on said feature.

With our synthetic experts, we aimed to obtain a similar distribution, with a high dependence on the model score, a few features with higher feature importance, followed by a relatively even distribution of feature importance for the rest of the features. Our synthetic experts' feature dependence, as measured by the same method, is represented in Figures 18,20. In these results, we can see that the value of the proportion Gains are, across the board, higher for all features. This is explained by the fact that the BAF dataset contains 30 features, while the dataset of real-world financial analyst decisions contains 116, which means that the feature importance will be spread out over a much larger number of features. In both expert teams, the model score is the highest importance feature, followed by a similar distribution of feature importances. This experiment validates that our synthetic financial analysts behave in a similar feature dependence to the real-world experts, even though the decision process of a real analyst may be more complex.

L2D testing

To the best of our knowledge, only one paper [reference to the Arxiv of Deccaf (to be uploaded)] considers the multi-expert learning to defer setting under individually defined work capacity constraints. We will use the same baselines as said study (Full Rejection, Only Classifier, Random Assignment, OvA), as well as their newly proposed method (DeCCaF):

- Full Rejection - All alerts are automatically rejected.
- Only Classifier - All final decisions on the alerts are made by the classifier h, trained according to the L2D algorithm to predict alongside the experts.

- Random Assignment - Alert deferral is done randomly throughout the experts/model until each of their capacity constraints are met.

- OvA - This method involves training one classifier per expert, which predicts the likelihood that an expert will make the correct prediction on said instance. Each of these are trained on the subset of training data corresponding to the set of a given expert's decisions. A separate classifier h is trained to predict the likelihood that the instance belongs to the positive class. The classifier h is trained on all of the instances in the training set.

  In testing, this method defers to an expert if they are predicted as the most likely to be correct, or to classifier h if the probability of the instance being fraudulent/legitimate is higher than that of any expert being correct. Deferral is done instance-wise to the expert/model that is most likely to be correct, until their work capacity is exhausted, at which point it chooses to defer to the second most likely to be correct, and so forth.

- DeCCaF - This method trains a classifier for the entire team, by also taking as input the *expert_id* of the expert which made the prediction for a given instance in training. This classifier then outputs the probability that an expert will make the correct prediction on a given instance. This method also trains a separate classifier h, identically to the OvA method.

  In testing, this method maximizes the global probability of correctness by using constraint programming.

Training OvA and DeCCaF Baselines

Note that the training process of the classifier is identical in both methods, therefore, they will share the same classifier h, whose training process we detail in this section. We then describe the training process of the OvA Classifiers and DeCCaF's team model, as well as some results regarding the predictive capability of these models, and their ability to produce calibrated estimates for the experts' probability of correctness.

Classifier h We model the classifier h with a LightGBM[28] classifier. The model is trained on the alerts raised by the Alert Model, ranging from months 4-6 of the BAF dataset. The model is trained by minimizing the weighted log-loss, where the weight of a label positive instance is defined as $c_i = 1$, and the weight of a label negative instance is defined as $c_i = \lambda$. The choice of hyperparameters is defined through Bayesian search[31] on an extensive grid, for 300 total trials, with 200 start-up trials, with validation done on the seventh month. We also test several values for the initial probability estimate of the base predictor of the boosting model. This was done in order to test if introducing a bias towards predicting fraud can be beneficial to our model, as across all scenarios, false negatives incur a higher cost. Given instance-wise feature weights c_i, the default value g_{initial} of the initial estimator's prediction is

$$g_d = logit(\frac{\sum_i c_i y_i}{\sum_i c_i}) = 0.782. \tag{14}$$

When training the model, we run the hyper-parameter search independently for $g_{\text{initial}} \in \{0.782, 2.582\}$, the upper bound was selected as performance starts drastically deteriorating past said value. The optimization objective is to minimize the weighted log-loss. In a first series of experiments, we found that a low number of estimators and maximum depth resulted in the best results. As such, in our first thorough hyper-parameter search, we use the parameter space represented in Table 4. In this set of experiments, across all scenarios, LGBM classifiers with a maximum tree depth of 2 achieved the best performance. As such, we conducted a second experiment, consisting of a total of 1700 trials, with 1500 startup trials, with the same hyperparameter space detailed in Table 4, but fixing the maximum depth parameter at 2.

The values for the selected model are also shown in Table 4. To evaluate the ability of this model to rank the instances according to their probability of belonging to the positive class, we use the ROC AUC, which has a value of 0.71, indicating that the classifier h is able to model the probability of fraud. To evaluate the accuracy of the probability estimates, we use the *expected calibration error*[38], a measure of the mean absolute deviation of the probability estimates to the real probabilities of error. The ECE obtained in testing is of 4.9%.

OvA Classifiers and DeCCaF team model We model both the OvA Classifiers and the DeCCaF team model with LightGBM[28] classifiers. These models are trained on the alerts raised by the Alert Model, and the corresponding expert predictions, ranging from months four to six of the BAF dataset. Both methods are trained by minimizing the weighted log-loss. The choice of hyperparameters is defined through Bayesian search[31] on an extensive grid, for 120 total trials, with 100 start-up trials, with validation done on the seventh month. The hyperparameter search space is detailed in Table 5. The OvA classifiers and the DeCCaF team model were trained in the 25 distinct training scenarios.

The mean values of the OvA classifiers' AUC and ECE, with the 95% confidence intervals, are reported in Table 6. In the paper where the OvA multi-expert method is introduced[12], the authors also report the ECE in three separate datasets, with values rarely surpassing 4%. In this task, however, the miscalibration of the predictions is significantly higher, rarely falling

below 5%. This may be due to several properties of our dataset, as the decision process of our simulated experts is more complex, the dataset contains concept drift and shifts in prevalence throughout the months, and the high class imbalance pose significant challenges.

The mean values of the DeCCaF team model's AUC and ECE, with the 95% confidence intervals, are also reported in Table 6. It seems that in this scenario, jointly modeling the team's behavior leads to a slight increase in the average AUC (4% on average), and a significant improvement in the calibration of the estimates (17% on average). Note that for both methods, the values of the AUC for the prediction of expert correctness do not surpass 0.65, showing that under these realistic training conditions, it may be challenging to model human behaviour.

Deferral Results

In this section we evaluate the quality of deferral as measured by the average misclassification cost incurred over the entire test set. The results are displayed in Table 7, where the error bars denote a 95% confidence interval. The worst option is Full Rejection, as all possible decision makers (experts and classifier h), have a lower misclassification cost over the test set. This is evidenced by the "Only Classifier h" column, as well as Figure 4. Experts have, on average, a higher performance than the Classifier *h*, and, therefore the "Random Assignment" baseline is the best non-L2D baseline. Finally we observe that both OvA and DeCCaF obtain significantly better results than all non-L2D baselines, with the methods achieving an average 11% and 15% decrease, respectively, in the misclassification cost over the Random Assignment baseline. Note that while DeCCaF significantly outperforms the OvA approach for "team_1" and "team_5", there is overlap between the error bars within the rest of the scenarios. This demonstrates the importance of testing L2D systems with complex expert behavior under a wide variety of scenarios, as the performance of these algorithms are significantly impacted by changes in the human decision making properties.

Usage Notes

The FiFAR dataset is publicly accessible on Kaggle, available at `[URLhere]`. In order to ensure reproducibility, the alert model, as well as all models used in the L2D benchmark are also made available. The FiFAR dataset can be reproducibly re-generated by running the code detailed in Section "Code availability". Users are encouraged to modify the dataset and adapt it to their use case, making full use of the flexibility of our framework. Evaluation notebooks are included in order to validate the quality of the generated synthetic experts. These contain the code used to conduct the experiments presented in the Technical Validation Section, which will be detailed in the Code Availability Section.

Code availability

The OpenL2D synthetic data generation framework is made available for public use in GitHub (`tobeadded`). In this repository, we also include the code needed to train the Alert Model used, and to perform all the data processing needed for the generation of the FiFAR dataset. To ensure reproducibility of every step in the data generation process, we also give users access to the python environment specification file, as generated according to Section "Building Identical Conda Environments" in the following webpage (`https://conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html#activating-an-environment`). Detailed instructions on how to use the framework are available in the GitHub repository, in the "README.md" file.

References

1. De-Arteaga, M., Fogliato, R. & Chouldechova, A. A Case for Humans-in-the-Loop: Decisions in the Presence of Erroneous Algorithmic Scores. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, 1–12, 10.1145/3313831.3376638 (ACM, Honolulu HI USA, 2020).

2. Dellermann, D., Ebel, P., Soellner, M. & Leimeister, J. M. Hybrid Intelligence. *Bus. & Inf. Syst. Eng.* **61**, 637–643, 10.1007/s12599-019-00595-2 (2019). 2105.00691.

3. Inkpen, K. *et al.* Advancing human-ai complementarity: The impact of user expertise and algorithmic tuning on joint decision making. *arXiv preprint arXiv:2208.07960* (2022).

4. Charusaie, M.-A., Mozannar, H., Sontag, D. A. & Samadi, S. Sample Efficient Learning of Predictors that Complement Humans. In Chaudhuri, K. *et al.* (eds.) *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, vol. 162 of *Proceedings of Machine Learning Research*, 2972–3005 (PMLR, 2022).

5. Hemmer, P., Schellhammer, S., Vössing, M., Jakubik, J. & Satzger, G. Forming Effective Human-AI Teams: Building Machine Learning Models that Complement the Capabilities of Multiple Experts. In Raedt, L. D. (ed.) *Proceedings of*

the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022, 2478–2484, 10.24963/ijcai.2022/344 (ijcai.org, 2022).

6. Raghu, M. *et al.* Direct uncertainty prediction for medical second opinions. In *International Conference on Machine Learning*, 5281–5290 (PMLR, 2019).

7. Raghu, M. *et al.* The Algorithmic Automation Problem: Prediction, Triage, and Human Effort. *CoRR* **abs/1903.12220** (2019). 1903.12220.

8. Mozannar, H. & Sontag, D. A. Consistent Estimators for Learning to Defer to an Expert. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, vol. 119 of *Proceedings of Machine Learning Research*, 7076–7087 (PMLR, 2020).

9. Mozannar, H. *et al.* Who should predict? exact algorithms for learning to defer to humans. *arXiv preprint arXiv:2301.06197* (2023).

10. Madras, D., Pitassi, T. & Zemel, R. Predict Responsibly: Improving Fairness and Accuracy by Learning to Defer. In *Advances in Neural Information Processing Systems*, vol. 31 (Curran Associates, Inc., 2018).

11. Verma, R. & Nalisnick, E. T. Calibrated Learning to Defer with One-vs-All Classifiers. In Chaudhuri, K. *et al.* (eds.) *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, vol. 162 of *Proceedings of Machine Learning Research*, 22184–22202 (PMLR, 2022).

12. Verma, R., Barrejón, D. & Nalisnick, E. Learning to defer to multiple experts: Consistent surrogate losses, confidence calibration, and conformal ensembles. In *International Conference on Artificial Intelligence and Statistics*, 11415–11434 (PMLR, 2023).

13. Hemmer, P., Thede, L., Vössing, M., Jakubik, J. & Kühl, N. Learning to defer with limited expert predictions. *arXiv preprint arXiv:2304.07306* (2023).

14. Wang, X. *et al.* Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2097–2106 (2017).

15. Krizhevsky, A. & Hinton, G. Learning multiple layers of features from tiny images. (2009).

16. Zhu, Z., Liu, T. & Liu, Y. A second-order approach to learning with instance-dependent label noise. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 10113–10123 (2021).

17. Berthon, A., Han, B., Niu, G., Liu, T. & Sugiyama, M. Confidence scores make instance-dependent label-noise learning possible. In *International conference on machine learning*, 825–836 (PMLR, 2021).

18. Han, B. *et al.* Co-teaching: Robust training of deep neural networks with extremely noisy labels. *Adv. neural information processing systems* **31** (2018).

19. Jesus, S. *et al.* Turning the Tables: Biased, Imbalanced, Dynamic Tabular Datasets for ML Evaluation. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2022* (2022).

20. Amarasinghe, K. *et al.* On the importance of application-grounded experimental design for evaluating explainable ml methods. *arXiv preprint arXiv:2206.13503* (2022).

21. Levy, A., Agrawal, M., Satyanarayan, A. & Sontag, D. Assessing the impact of automated suggestions on decision making: Domain experts mediate model errors but take less initiative. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, 1–13 (2021).

22. Mitchell, T. J. & Beauchamp, J. J. Bayesian variable selection in linear regression. *J. american statistical association* **83**, 1023–1032 (1988).

23. Burden, R. L. & Faires, J. D. 2.1 the bisection algorithm. *Numer. analysis* **3** (1985).

24. Gulshan, V. *et al.* Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs. *Jama* **316**, 2402–2410 (2016).

25. Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M. & Bouchachia, A. A survey on concept drift adaptation. *ACM computing surveys (CSUR)* **46**, 1–37 (2014).

26. Elkan, C. The foundations of cost-sensitive learning. In *International joint conference on artificial intelligence*, vol. 17, 973–978 (Lawrence Erlbaum Associates Ltd, 2001).

27. Han, J., Huang, Y., Liu, S. & Towey, K. Artificial intelligence for anti-money laundering: a review and extension. *Digit. Finance* **2**, 211–239 (2020).

28. Ke, G. *et al.* LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In Guyon, I. *et al.* (eds.) *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, 3146–3154 (2017).

29. Shwartz-Ziv, R. & Armon, A. Tabular data: Deep learning is not all you need. *Inf. Fusion* **81**, 84–90 (2022).

30. Borisov, V. *et al.* Deep neural networks and tabular data: A survey. *IEEE Transactions on Neural Networks Learn. Syst.* (2022).

31. Akiba, T., Sano, S., Yanase, T., Ohta, T. & Koyama, M. Optuna: A Next-generation Hyperparameter Optimization Framework. In Teredesai, A. *et al.* (eds.) *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*, 2623–2631, 10.1145/3292500.3330701 (ACM, 2019).

32. Grimstad, S. & Jørgensen, M. Inconsistency of expert judgment-based estimates of software development effort. *J. Syst. Softw.* **80**, 1770–1777 (2007).

33. Danziger, S., Levav, J. & Avnaim-Pesso, L. Extraneous factors in judicial decisions. *Proc. Natl. Acad. Sci.* **108**, 6889–6892 (2011).

34. Remenyi, B. *et al.* Inter-rater and intra-rater reliability and agreement of echocardiographic diagnosis of rheumatic heart disease using the world heart federation evidence-based criteria. *Hear. Asia* **11** (2019).

35. Lippens, L., Vermeiren, S. & Baert, S. The state of hiring discrimination: A meta-analysis of (almost) all recent correspondence experiments. *Eur. Econ. Rev.* **151**, 104315 (2023).

36. Mitchell, S., Potash, E., Barocas, S., D'Amour, A. & Lum, K. Algorithmic fairness: Choices, assumptions, and definitions. *Annu. Rev. Stat. Its Appl.* **8**, 141–163 (2021).

37. Corbett-Davies, S., Pierson, E., Feller, A., Goel, S. & Huq, A. Algorithmic decision making and the cost of fairness. In *Proceedings of the 23rd acm sigkdd international conference on knowledge discovery and data mining*, 797–806 (2017).

38. Guo, C., Pleiss, G., Sun, Y. & Weinberger, K. Q. On calibration of modern neural networks. In *International conference on machine learning*, 1321–1330 (PMLR, 2017).

Author contributions statement

Must include all authors, identified by initials, for example: A.A. conceived the experiment(s), A.A. and B.A. conducted the experiment(s), C.A. and D.A. analysed the results. All authors reviewed the manuscript.

Competing interests

(mandatory statement)

The corresponding author is responsible for providing a competing interests statement on behalf of all authors of the paper. This statement must be included in the submitted article file.

Figures & Tables

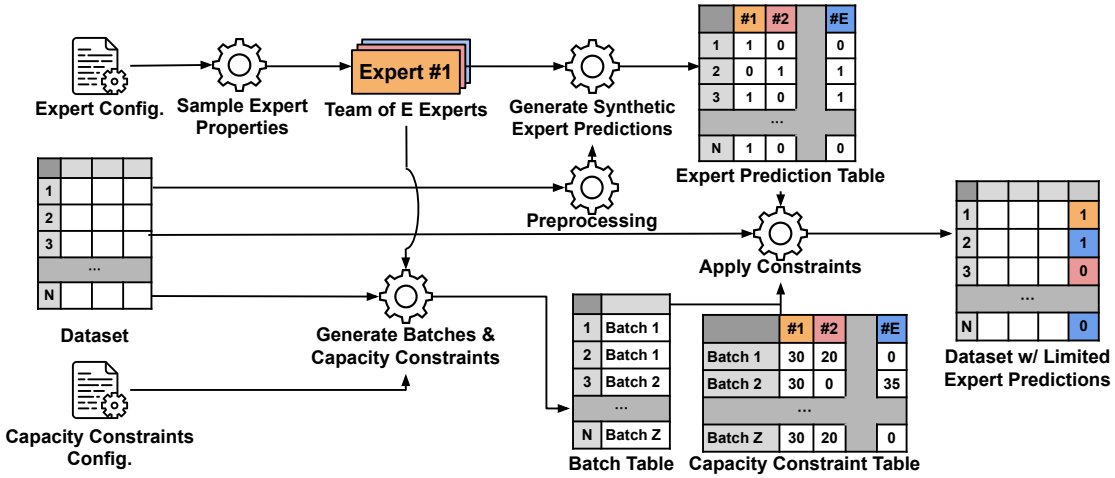

Figure 1. Generating Synthetic Expert Predictions and Capacity Constraints using our Framework

Hyperparameter	Values or Interval	Distribution	Selected
boosting_type	“goss”		“goss”
enable_bundle	False		False
n_estimators	[50,5000]	Log	94
max_depth	[2,20]	Unif.	2
num_leaves	[10,1000]	Log	145
min_child_samples	[5,500]	Log	59
learning_rate	[0.01, 0.5]	Log	0.3031421
reg_alpha	[0.0001, 0.1]	Log	0.0012637
reg_lambda	[0.0001, 0.1]	Log	0.0017007

Table 1. Alert Model: LightGBM hyperparameter search space

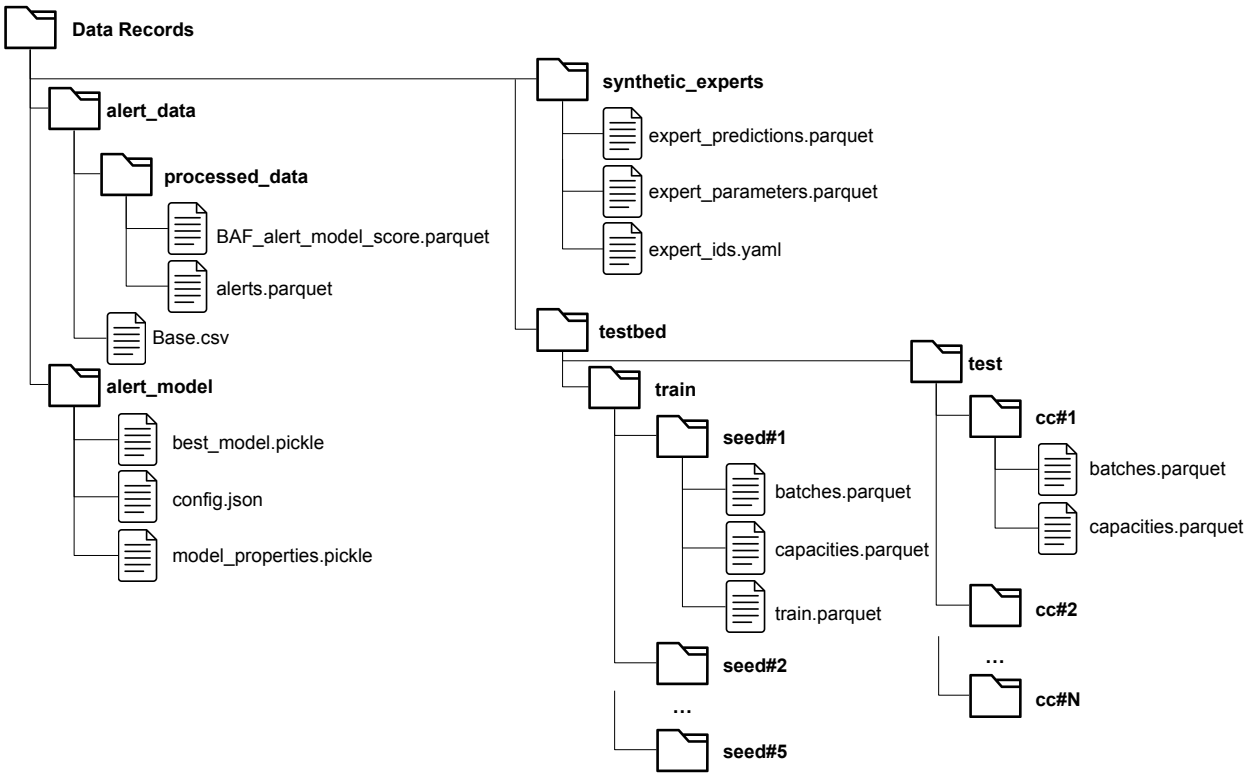

Figure 2. Folder Structure of the FiFAR Dataset

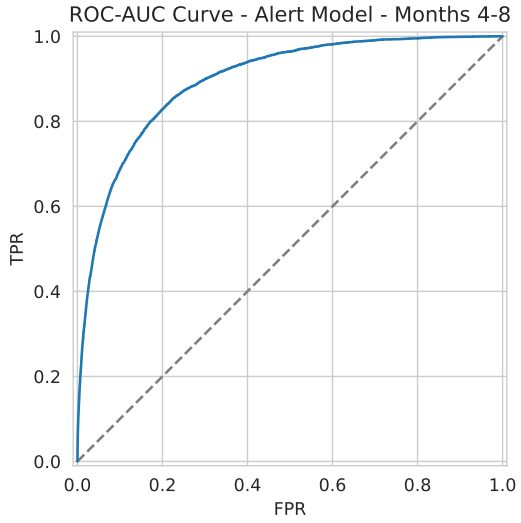

Figure 3. ROC-Curve - Alert Model - Months 4-8

feature	μ	σ
income	0.326	0.1
name_email_similarity	-0.388	0.1
prev_address_months_count	0	0.05
current_address_months_count	0	0.05
customer_age	0.07	0.02
days_since_request	0	0.05
intended_balcon_amount	0	0.05
payment_type	0.362	0.1
zip_count_4w	0	0.05
velocity_6h	-0.41	0.1
velocity_24h	0	0.05
velocity_4w	-0.447	0.1
bank_branch_count_8w	0	0.05
date_of_birth_distinct_emails_4w	-0.342	0.1
employment_status	0	0.05
credit_risk_score	0.626	0.1
email_is_free	0.383	0.1
housing_status	0	0.05
phone_home_valid	-0.347	0.1
phone_mobile_valid	0	0.05
bank_months_count	0	0.05
has_other_cards	0	0.05
proposed_credit_limit	0.735	0.1
foreign_request	0	0.05
source	0	0.05
session_length_in_minutes	0	0.05
device_os	0.354	0.1
keep_alive_session	-0.44	0.1
device_distinct_emails_8w	0.378	0.1
device_fraud_count	0	0.05
model_score	0.7	0.3

Table 2. Expert Feature Weight Distributions

Hyperparameter	Values or Interval	Distribution
boosting_type	“goss”	
enable_bundle	False	
n_estimators	[50,300]	Log
max_depth	[2,20]	Unif.
num_leaves	[10,100]	Log
min_child_samples	[5,500]	Log
learning_rate	[0.01, 0.5]	Log
reg_alpha	[0.0001, 0.1]	Log
reg_lambda	[0.0001, 0.1]	Log

Table 3. Feature Dependence Testing: LightGBM hyperparameter search space

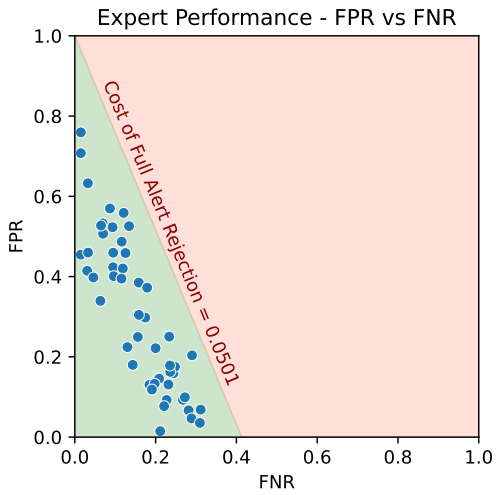

Figure 4. Expert Performance - FPR vs. FNR - Red line represents combinations of (FPR, FNR) resulting in the same cost as rejecting all customer applications.

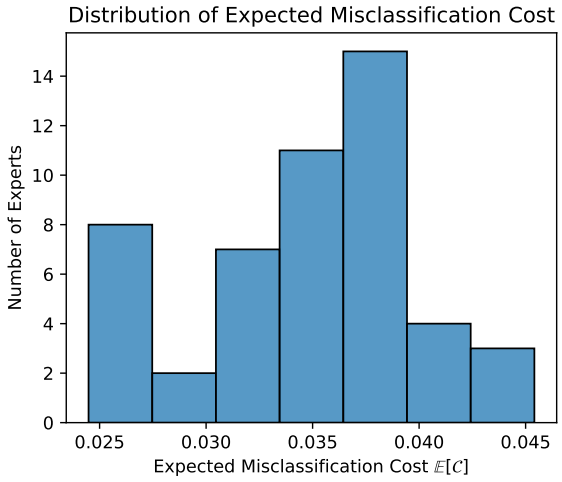

Figure 5. Distribution of Expected Misclassification Cost - months 4-8. FiFAR's expert team contains a wide array of expertise, allowing for testing of L2D systems with varying levels of human performance

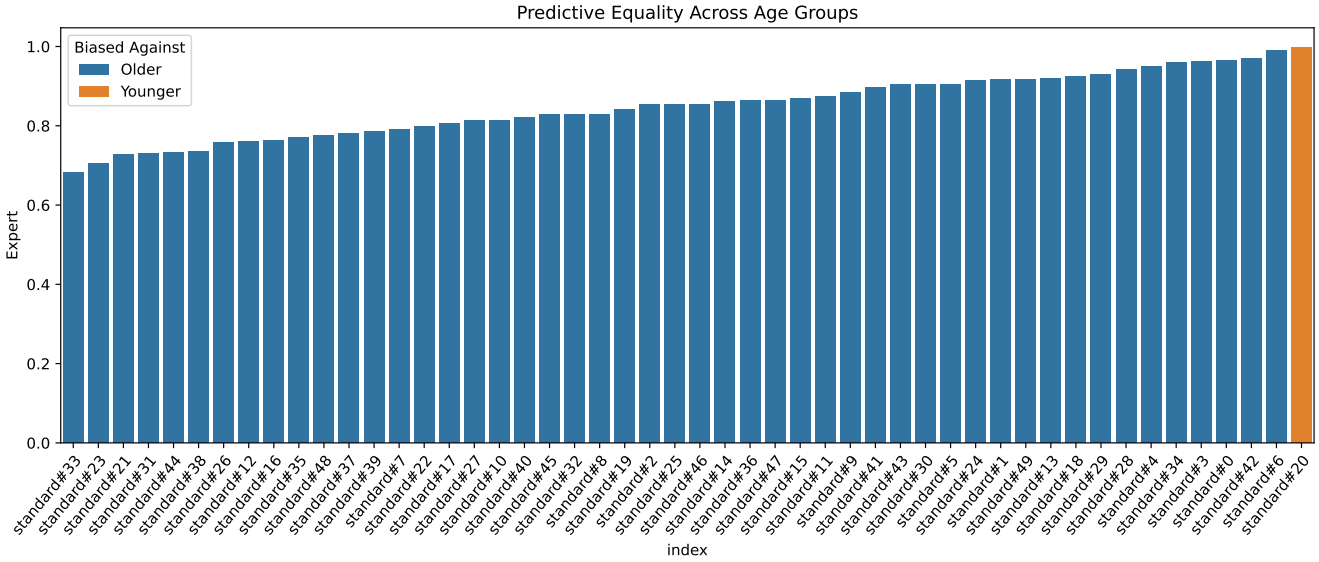

Figure 6. Expert Predictive Equality - Experts who are biased against a given group have a higher false positive rate (incorrectly rejecting an application) within said group.

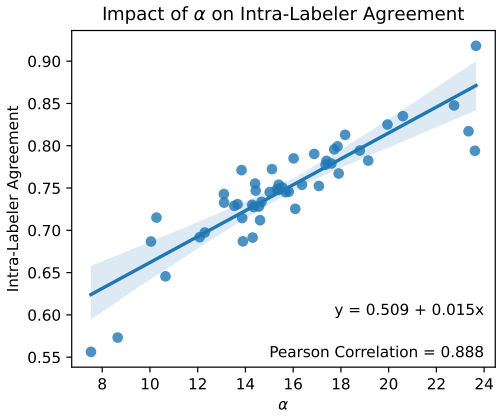

Figure 7. Relationship between the parameter α and the intra-rater reliability of each expert. Although the value of the intra-rater reliability is also impacted by β_0, β_1, the correlation with α is apparent.

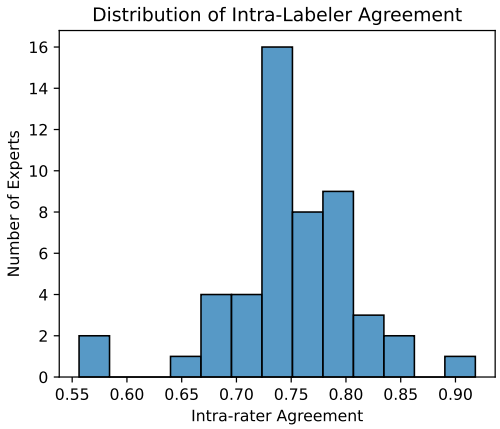

Figure 8. Distribution of Intra-Rater agreement, calculated in months 4-8. The expert team contains various levels of intra-rater consistency.

Table 4. ML Model: LightGBM hyperparameter search space

Hyperparameter	Values or interval	Distribution	Selected
init_score	{0.782,0.982,...,2.582 }		1.982
boosting_type	"dart"		"dart"
enable_bundle	[False,True]		False
n_estimators	[50,250]	Unif.	160
max_depth	[2,5]	Unif.	2
num_leaves	[100,1000]	Unif.	708
min_child_samples	[5,200]	Unif.	89
learning_rate	[0.001, 1]	Unif.	0.779
reg_alpha	[0.001, 2]	Unif.	1.673
reg_lambda	[0.001, 2]	Unif.	1.932

Table 5. LightGBM hyperparameter search space - OvA Classifiers

Hyperparameter	Values or Interval	Distribution
boosting_type	"dart"	
enable_bundle	[False,True]	
n_estimators	[50,250]	Unif.
max_depth	[2,20]	Unif.
num_leaves	[100,1000]	Log.
min_child_samples	[5,100]	Log.
learning_rate	[0.005, 0.5]	Log.
reg_alpha	[0.0001, 0.1]	Log.
reg_lambda	[0.0001, 0.1]	Log.

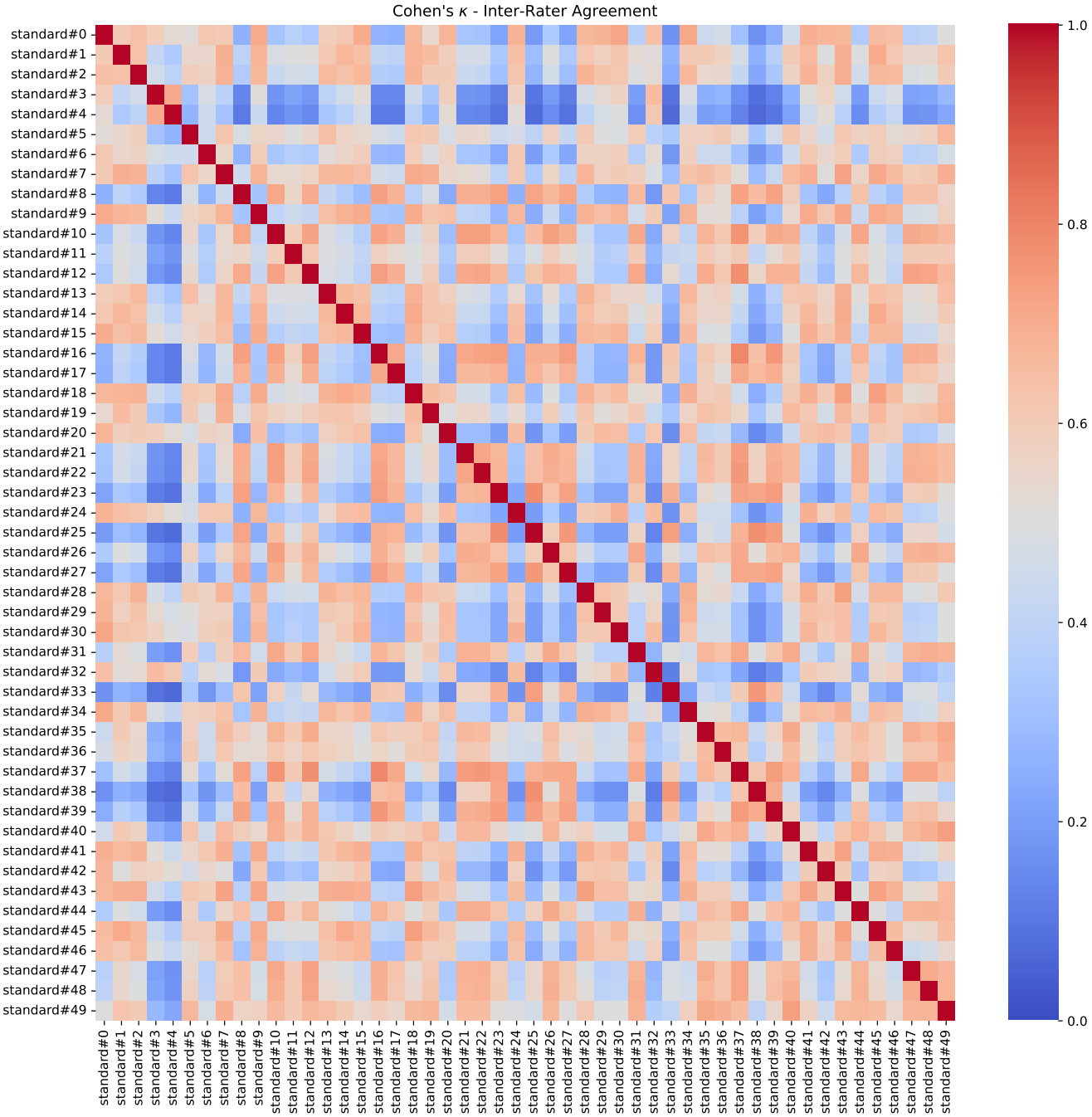

Figure 9. Heatmap depicting the value of Cohen's κ for each expert pair. These values were obtained using the expert decisions in months 4-8, made available in our dataset.

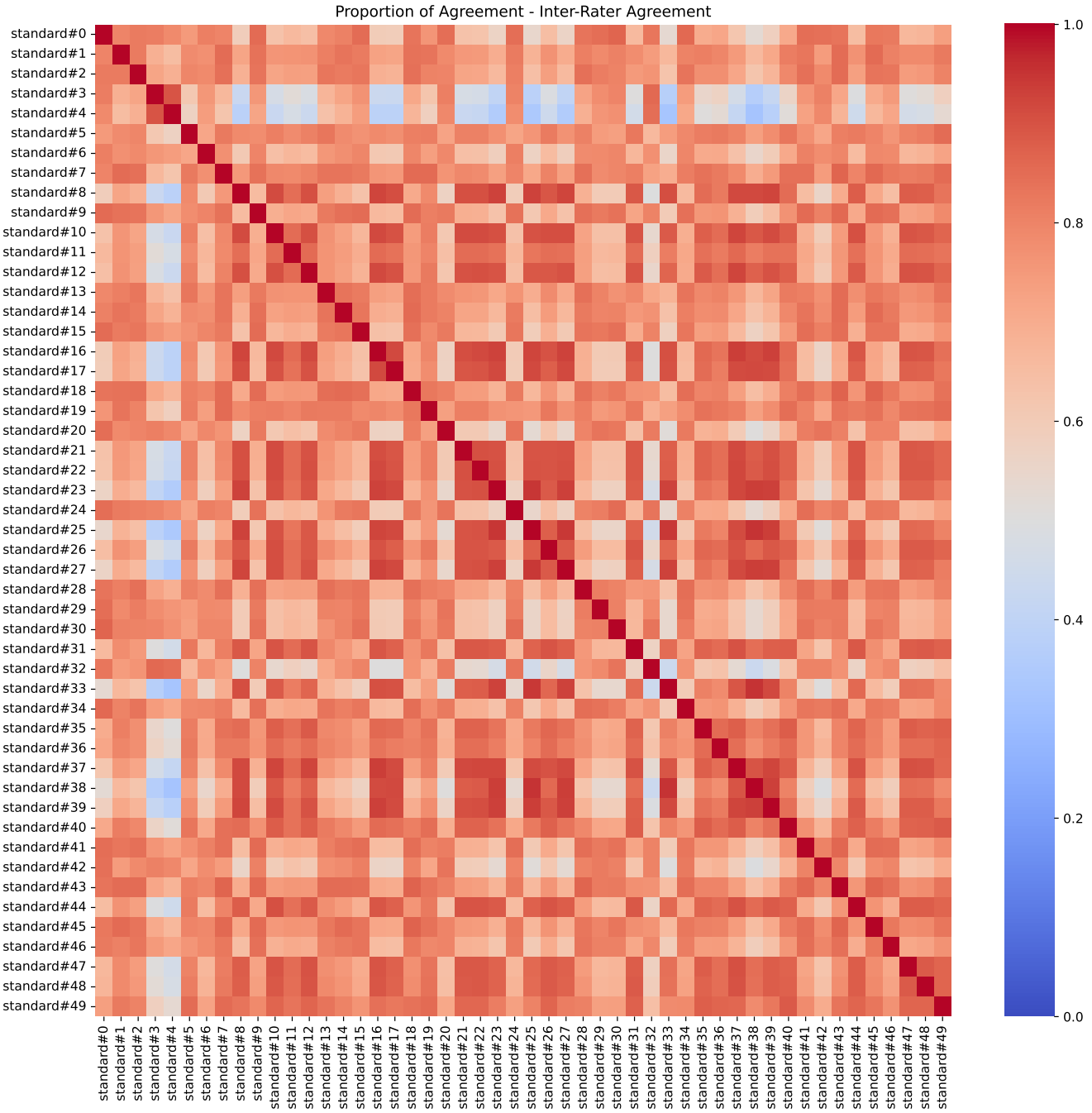

Figure 10. Heatmap depicting the proportion of agreement p_a for each expert pair. These values were obtained using the expert decisions in months 4-8, made available in our dataset.

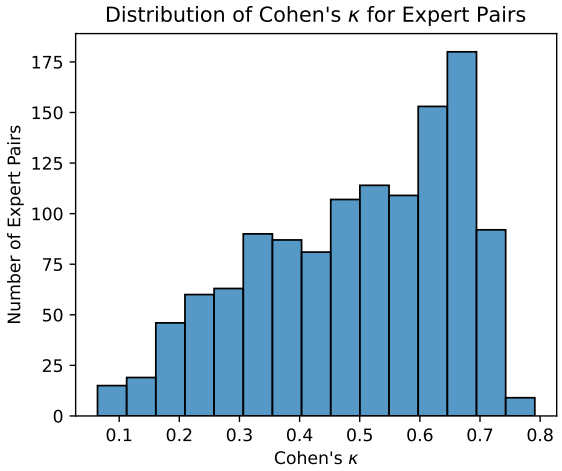

Figure 11. Distribution of Inter-Rater agreement as measured by Cohen's κ, calculated in months 4-8.

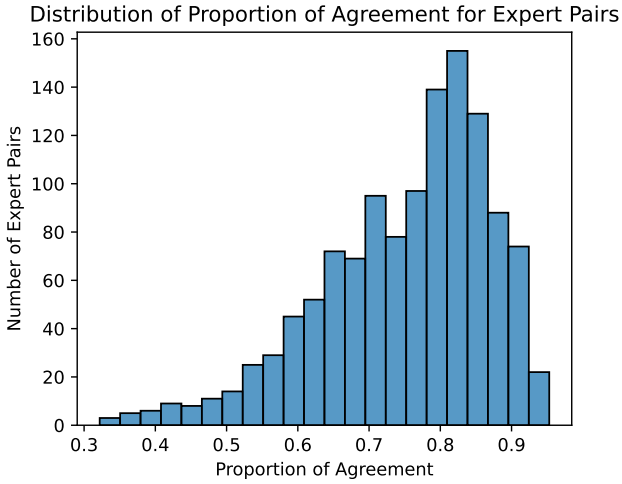

Figure 12. Distribution of Inter-Rater agreement as measured by proportion of agreement, calculated in months 4-8.

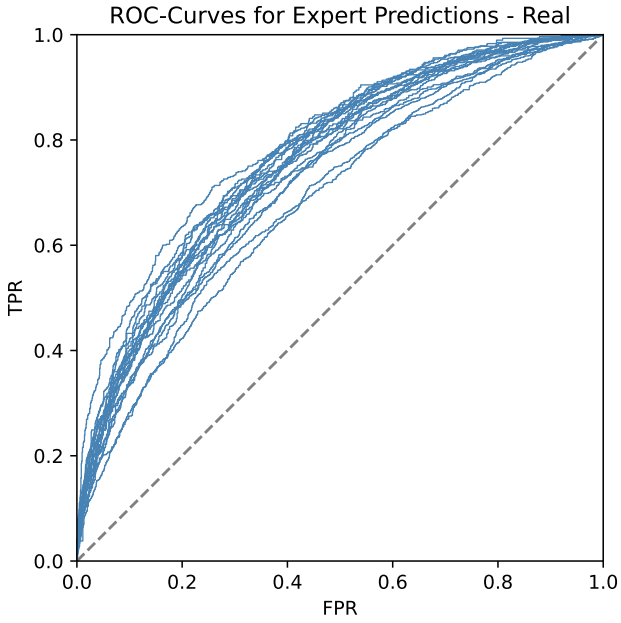

Figure 13. ROC Curves in Test Split - Real Experts

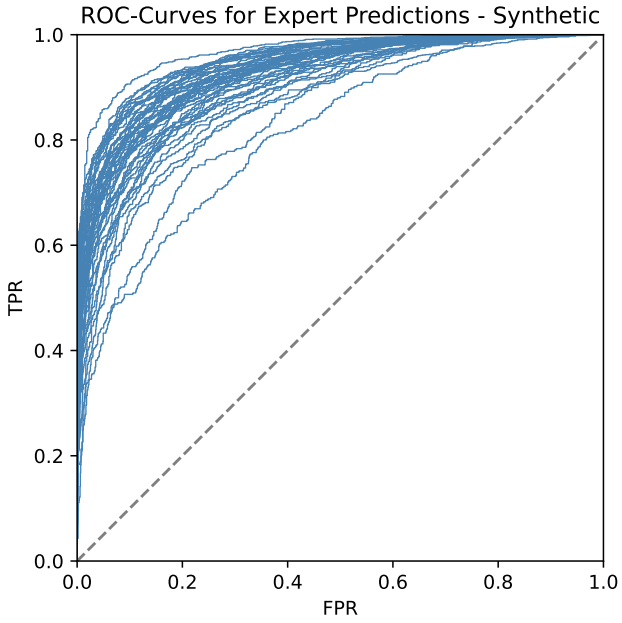

Figure 14. ROC Curves in Test Split - Synthetic Experts

Table 6. OvA Classifier Performances

team	auc_ova	ece_ova	auc_deccaf	ece_deccaf
team_1	0.61 ± 0.01	6.0 ± 0.5	0.64 ± 0.01	4.6 ± 0.5
team_2	0.58 ± 0.01	6.0 ± 0.6	0.6 ± 0.01	4.8 ± 0.4
team_3	0.6 ± 0.02	5.5 ± 0.5	0.62 ± 0.02	5.2 ± 0.6
team_4	0.59 ± 0.01	5.4 ± 0.6	0.6 ± 0.01	3.8 ± 0.4
team_5	0.59 ± 0.01	5.5 ± 0.7	0.61 ± 0.01	4.9 ± 0.6

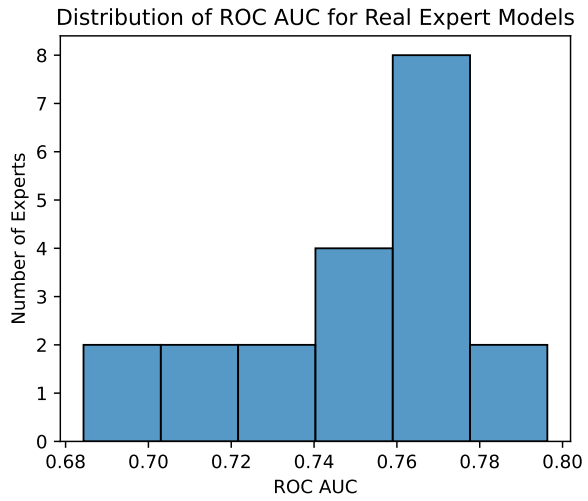

Figure 15. ROC-AUC Values in Test Split - Real Experts

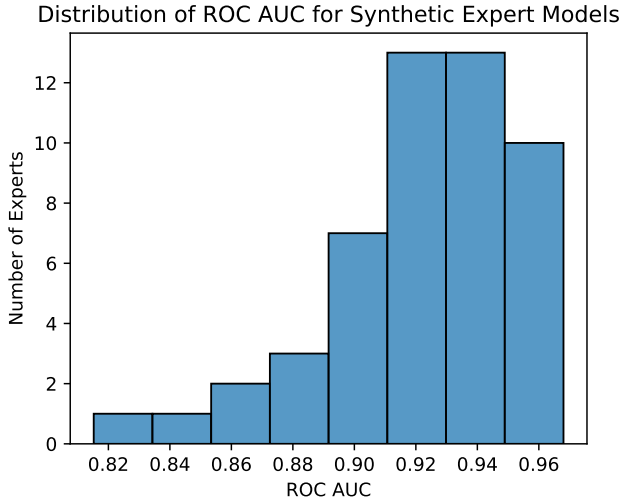

Figure 16. ROC-AUC Values in Test Split - Synthetic Experts

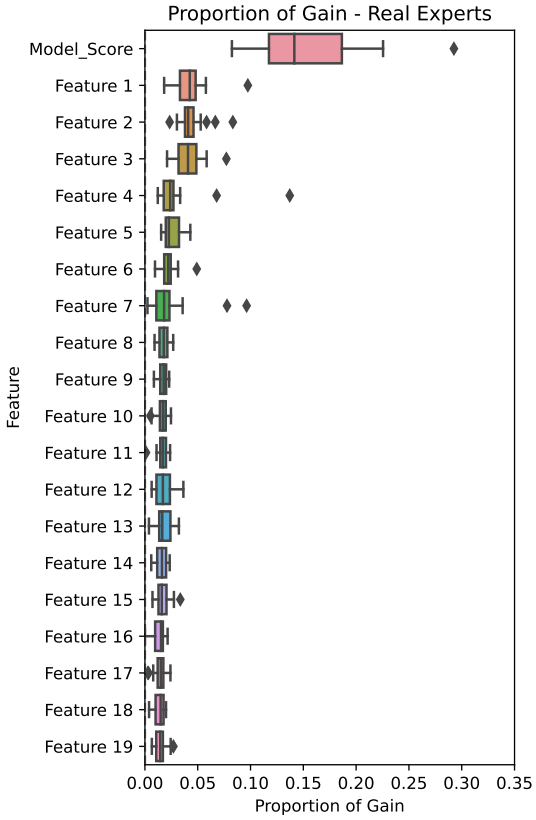

Figure 17. Distribution of Feature Importances - Real Experts

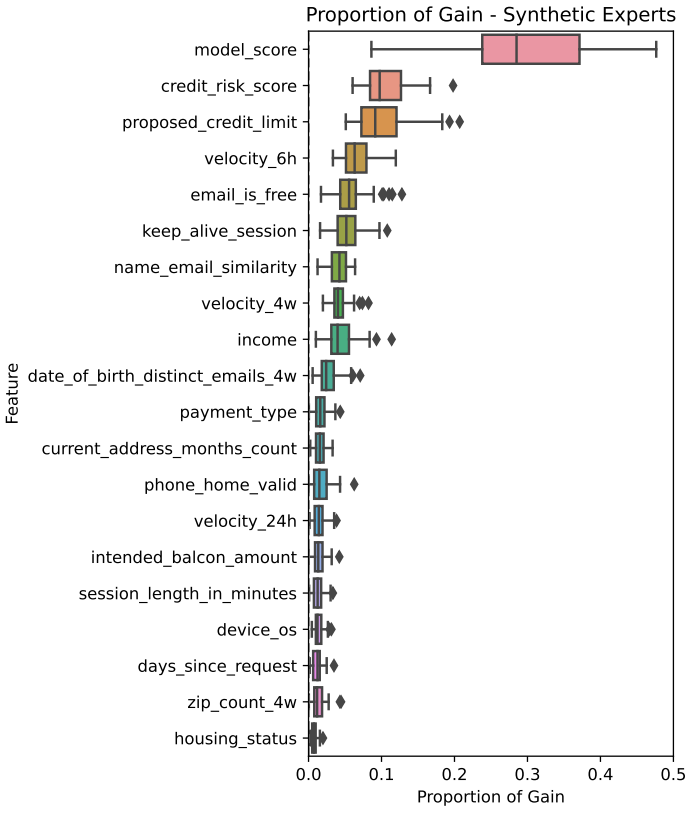

Figure 18. Distribution of Feature Importances - Synthetic Experts

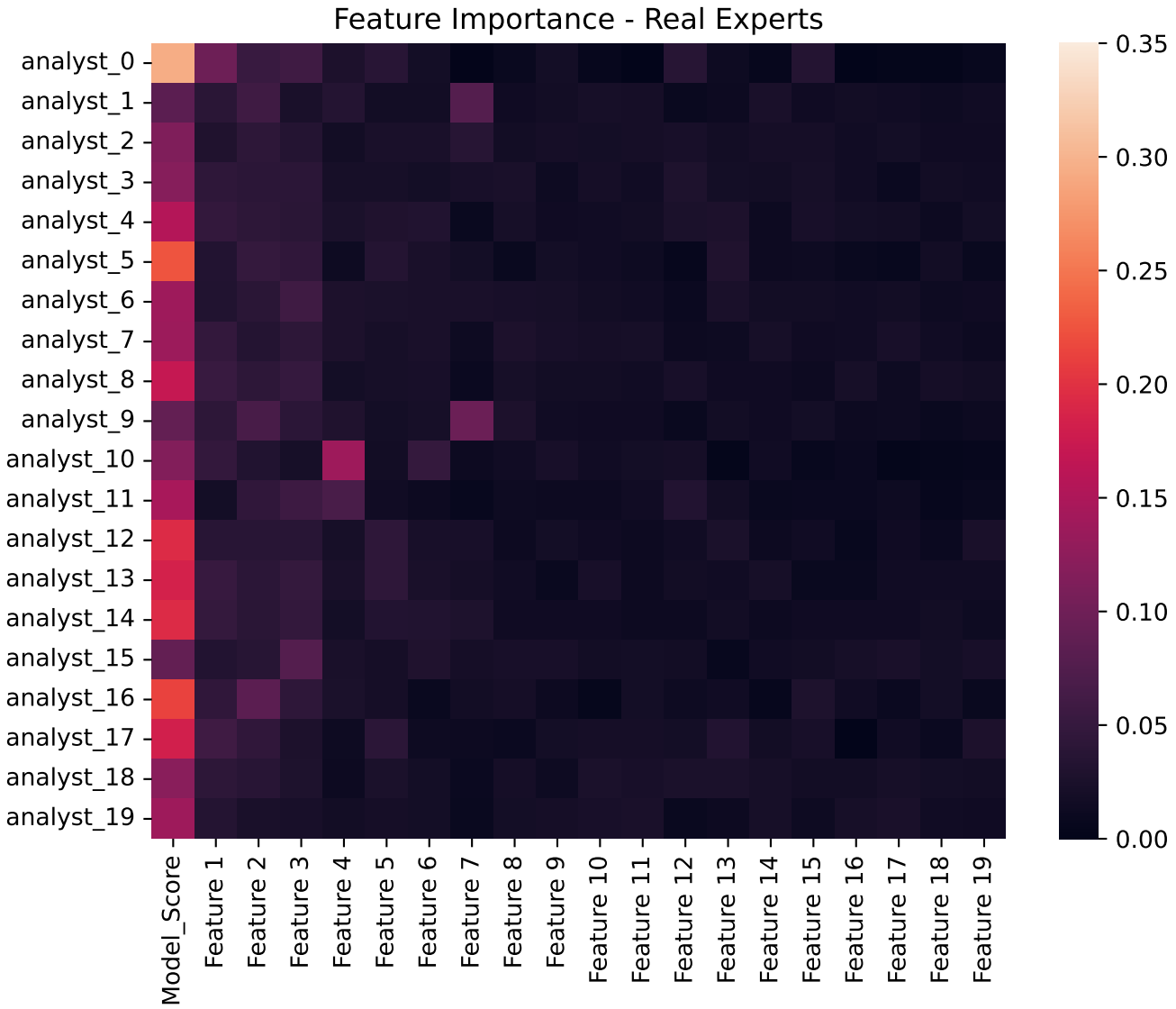

Figure 19. Heatmap of Feature Importances - Real Experts

Table 7. Deferral Results

Team	Full Rejection	Only Classifier h	Random Deferral	OvA	DeCCaF
team_1	213.5	204	169.6 ± 2.53	151.7 ± 1.99	$\mathbf{138.1} \pm 5.14$
team_2	213.5	204	157.8 ± 1.39	$\mathbf{142.0} \pm 1.61$	145.3 ± 4.28
team_3	213.5	204	151.2 ± 1.84	131.3 ± 2.19	$\mathbf{126.2} \pm 4.52$
team_4	213.5	204	163.1 ± 1.61	145.8 ± 3.51	$\mathbf{141.6} \pm 5.32$
team_5	213.5	204	163.3 ± 1.58	141.2 ± 2.57	$\mathbf{132.0} \pm 1.93$

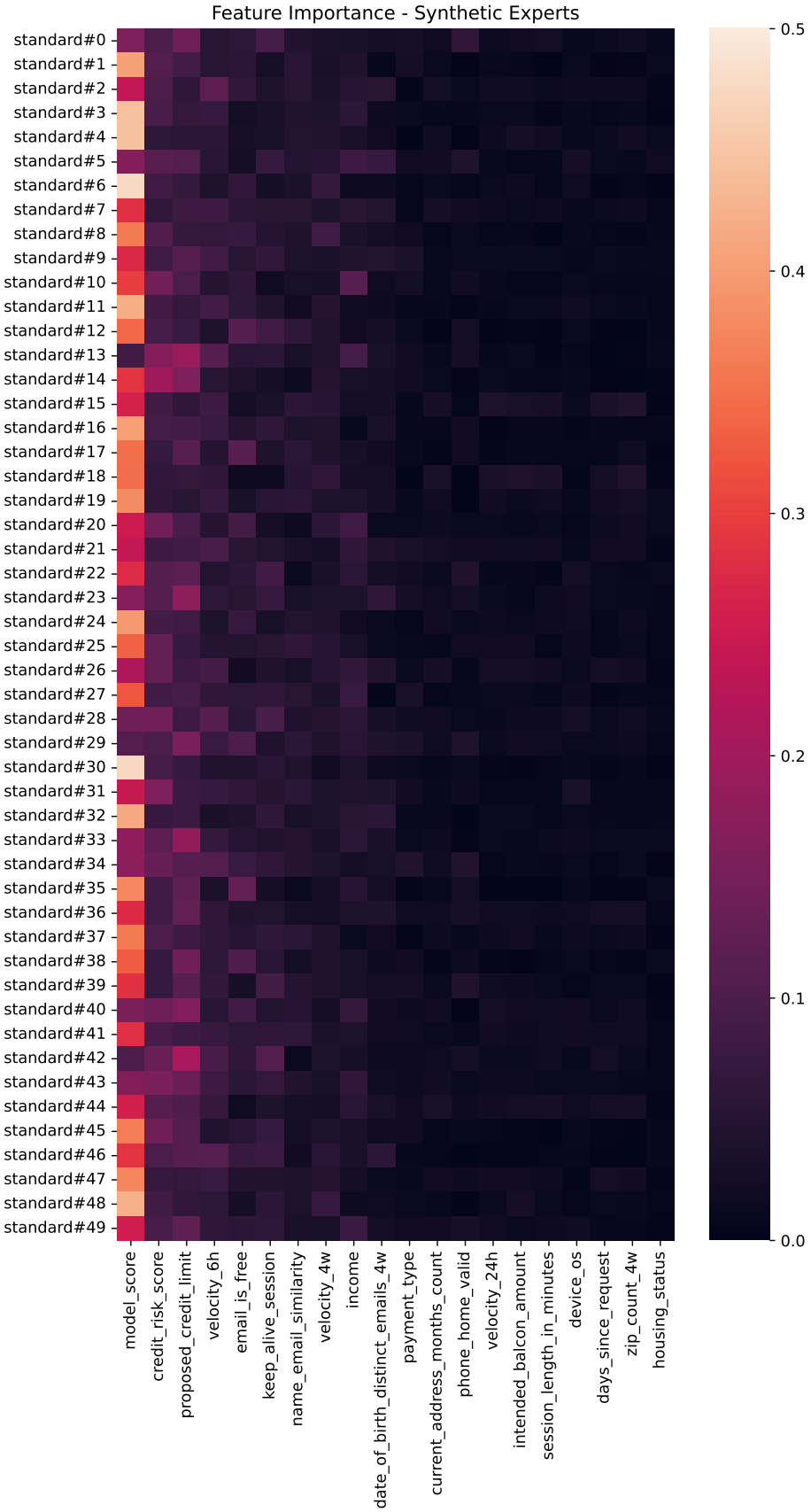

Figure 20. Heatmap of Feature Importances - Synthetic Experts