A Framework Details

A.1 Neyman-Pearson criterion and cost-sensitive learning - choice of λ

In our experiments, detailed in Section 4, we attempt to optimize assignments by modelling the probability of error of our human decision-makers, and combining it into a loss function to be minimized across a set of instances. Our choice of loss function depends on the final optimization objective. In our task, the optimization goal is expressed by a Neyman-Pearson criterion, aiming to maximize recall subject to a fixed FPR of 5%. This criterion is chosen due to the difference between error types. In bank account fraud prevention, the consequence of committing a false positive mistake, that is, rejecting a legitimate application, must be weighed against the cost of a false negative mistake, that is, accepting a fraudulent application. Due to these dynamics, a cost-sensitive approach [2] may be adequate. In a binary classification task, the possible outcomes of a prediction are the elements of a confusion matrix; true negative, false negative, false positive, true positive: [TN, FN, FP, TP]. Over a set of assignments, the loss is given by the total number of each outcome multiplied by its cost $\{c_{00}, c_{01}, c_{10}, c_{11}\}$, respectively:

$$\begin{aligned} L &= c_{00}\text{TN} + c_{01}\text{FN} + c_{10}\text{FP} + c_{11}\text{TP} \\ &= c_{00}(\text{LP} - \text{FN}) + c_{01}\text{FP} + c_{10}\text{FN} + c_{11}(\text{LN} - \text{FP}) \\ &= (c_{10} - c_{00})\text{FN} + (c_{01} - c_{11})\text{FP} + c_{00}\text{LP} + c_{11}\text{LN} \end{aligned}$$

where LP denotes the total number of label positives and LN denotes the total number of label negatives. Since LP and LN are constants, these do not affect resulting rankings when comparing methods. As such, c_{00} and c_{11} can be set to zero. Doing so, we reach an equation of the form:

$$L = \lambda\text{FP} + \text{FN} \quad \text{with} \quad \lambda = \frac{c_{01}}{c_{10}} \tag{1}$$

The predicted loss associated with deferring a case $\mathbf{X}_i$ to an expert e corresponds to the sum of the predicted probability of outcomes weighted by their cost, arriving at the loss function used in our methods.

$$L(\mathbf{X}_i, e) = \lambda\hat{\mathbb{P}}(\text{FP}) + \hat{\mathbb{P}}(\text{FN}) \tag{2}$$

However, in our task, we do not have access to the values of c_{10} and c_{01}. We must establish a relationship between the value of λ and the desired Neyman-Pearson criterion. According to Elkan [2], 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 $\mathbf{x_i}$ 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:

$$(1 - p)c_{10} \leq pc_{01} \tag{3}$$

Where $p = P(y = 1|\mathbf{x}_i)$, that is, the probability that x_i belongs to the positive class. An estimation of the value of p is given by our classifier, 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. In the case where the inequality is in fact an equality, then predicting either class is optimal. As such, the decision threshold t for making optimal decisions leads us to a value for λ:

$$(1 - t)c_{10} = tc_{01} \Leftrightarrow \lambda = \frac{t}{1 - t} \tag{4}$$

As the optimal threshold t for our ML 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 of lambda for our optimization goal. We chose to use this theoretically defined value such that the same value of λ is used across all methods' loss functions.

However, as shown by Sheng and Ling [4], choosing the theoretical threshold does not always work in practice. The authors show that testing several values for λ in validation works best. Secondly, the value of λ obtained through this method depends on the classifier trained. A different classifier would yield another value for the optimal threshold according to the Neyman-Pearson criterion, which would lead to a different λ, despite the task being the same.

B Benchmark Dataset

B.1 ML Model

As detailed in Section 4.1, our ML Model is a LightGBM [3] classifier. The model was trained on the first 3 months of the BAF dataset, and validated on the fourth month. The model is trained by minimizing binary cross-entropy loss. The choice of hyperparameters is defined through Bayesian search [1] 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.

Table 1: ML Model: LightGBM hyperparameter search space

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

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 the deployment split (months 4 to 8), used to train and test our assignment system, the model yields a recall of $M_{\text{TPR}} = 52.1\%$, using the same threshold. The false negative rate of the model, later used to generate our synthetic experts, is then $M_{\text{FNR}} = 47.9\%$. In Figure 1 we present the ROC curve for our model, calculated in the deployment split.

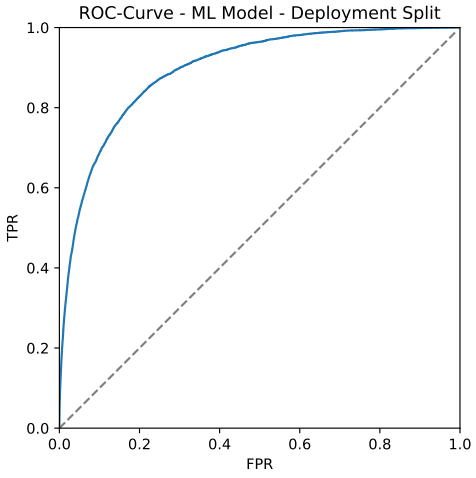

Figure 1: ROC-Curve - ML Model - Deployment Split

B.2 Synthetic Expert Properties

We created a team comprised of "standard","model-agreeing", and "unfair" experts. In Table 2 we present the distribution values defined for each expert group. Sampling from these distribution values, each expert's parameters - $T_{\text{FPR}}, T_{\text{FNR}}, \alpha, w_M$ and w_p - are presented in Table 4. Using these properties, the feature weight vector $\mathbf{w}$ is also sampled for each expert. To display each synthetic expert's feature dependence, the values of the normalized feature weights are displayed in in Figure

4, in a heatmap. As intended, the dominant feature for the unfair experts is the customer's age, which is our protected attribute, while the model agreeing experts are mostly influenced by the model score. Our standard experts display a more balanced weight distribution across features.

Table 2: Expert Group Properties

	Group		
Parameter	Standard	Model Agreeing	Unfair
n	20	15	15
group_seed	1	2	3
μ_w	0	0	0
σ_w	1	1	1
θ	0.3	0.3	0.3
μ_M	-2	-6	-2
σ_M	0.5	0.5	0.5
μ_p	-1	0	-4
σ_p	0.1	0.1	0.3
μ_α	4	12	4
σ_α	0.2	0.5	0.1
μ_{FNR}	$M_{\text{FNR}} - \sigma_{\text{FNR}}$	$M_{\text{FNR}} - \sigma_{\text{FNR}}$	$M_{\text{FNR}} - \sigma_{\text{FNR}}$
σ_{FNR}	0.05	0.05	0.05
μ_{FPR}	$M_{\text{FPR}} - \sigma_{\text{FPR}}$	$M_{\text{FPR}} - \sigma_{\text{FPR}}$	$M_{\text{FPR}} - \sigma_{\text{FPR}}$
σ_{FPR}	0.01	0.01	0.01

B.2.1 Generated Probabilities

The selected values for α were chosen so that every expert displays a wide range of probabilities of error. In Figures 2 and 3, we display the distribution of probabilities of error throughout instances, for our 3 different expert groups. The use of the logarithmic scale for the Y axis is due to the fact that a majority of instances will have a low probability of error.

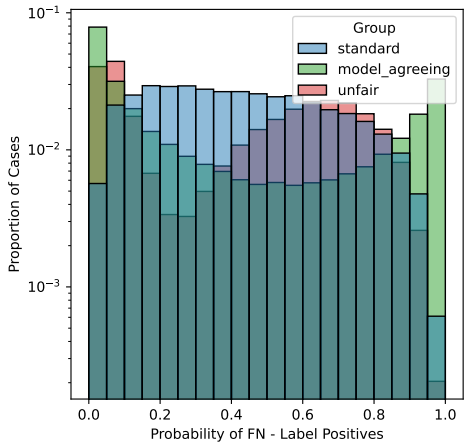

Figure 2: Probability Distribution FN - Label Positives

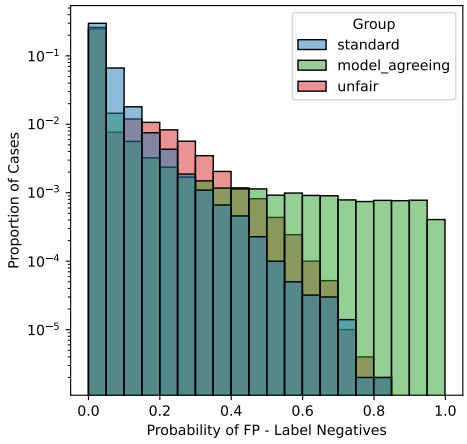

Figure 3: Probability Distribution FP - Label Negatives

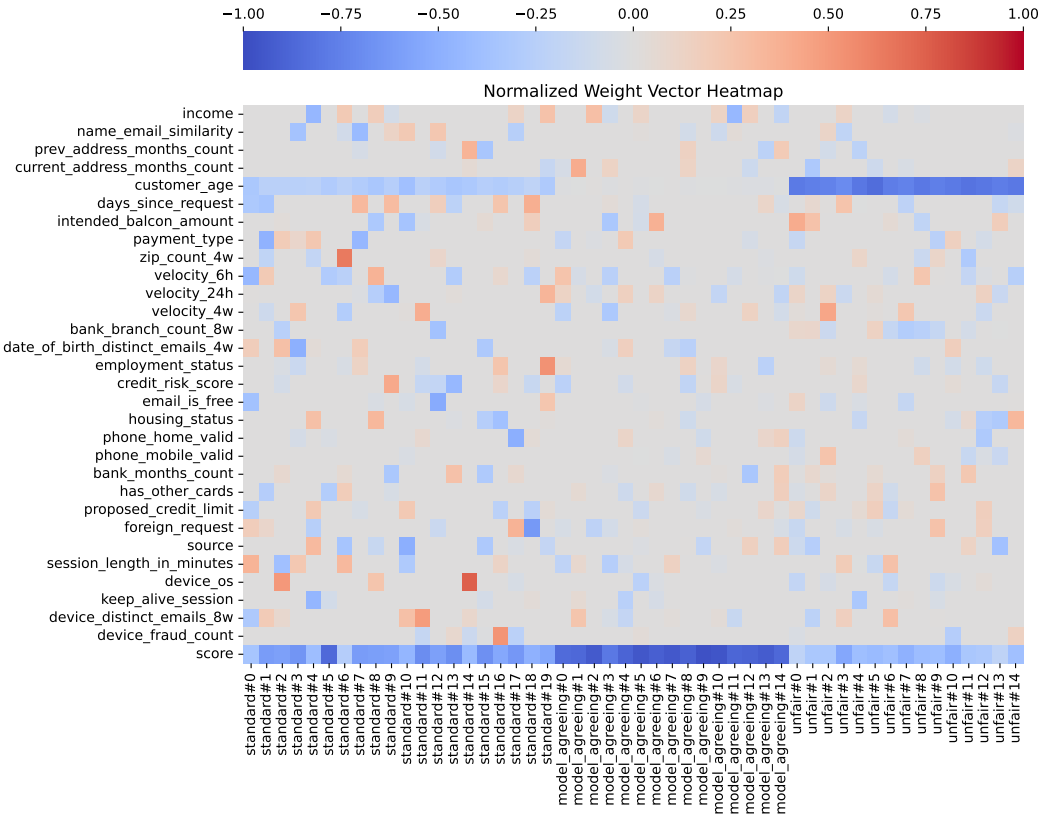

Figure 4: Normalized Feature weights

Each group's distributions are significantly different. The use of a larger value of α for model agreeing experts result in predictions saturating at the leftmost and rightmost bin, nearer the edges of the $[0, 1]$ interval.

B.2.2 Performance

When generating our team of synthetic experts, we aimed to obtain a team where most, but not all experts are better than the ML model. In a cost-sensitive task such as this, defining what "better" means is not trivial, as there is an implicit trade-off between FPR and FNR, imposed by our Neyman-Pearson criterion. In choosing our reported values for $\mu_{\text{FPR}}, \sigma_{\text{FPR}}, \mu_{\text{FNR}}, \sigma_{\text{FNR}}$, we aimed to obtain a team where most experts have both a lower FPR and a higher TPR than the model. A plot of each expert's Target FPR and recall when compared to the model is presented in Figure 5. Due to distribution shifts present across months in the bank-account-fraud dataset, the FPR and TPR obtained in the deployment split will deviate from the target values. The values of the FPR and TPR of each expert in comparison to the model, calculated in the deployment split, are presented in Figure 6. To compare the performance of the experts and the ML model using a single metric, we may use the loss function defined in section A.1, over the entire deployment split.

B.2.3 Unfairness and Model-Agreement

To introduce humans with a higher bias against older customers (age ≥ 50), we created a group of unfair experts. When setting a lower μ_p for the unfair experts, we expect to increase the probability of committing a FP mistake for older customers, as the customer's age will now be a primary factor in the probability of committing a mistake. The FPR disparities of each expert on the deployment split are displayed in Figure 8, confirming that our unfair experts exhibit a higher bias against older customers.

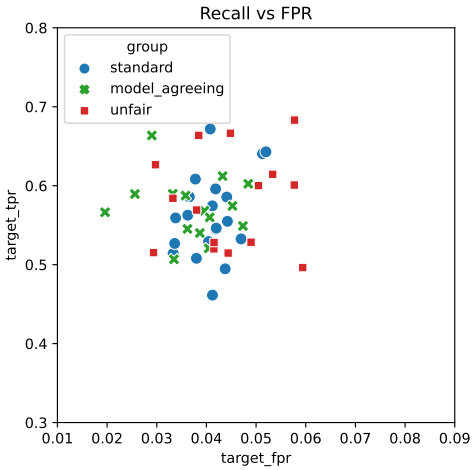

Figure 5: Target TPR and FPR

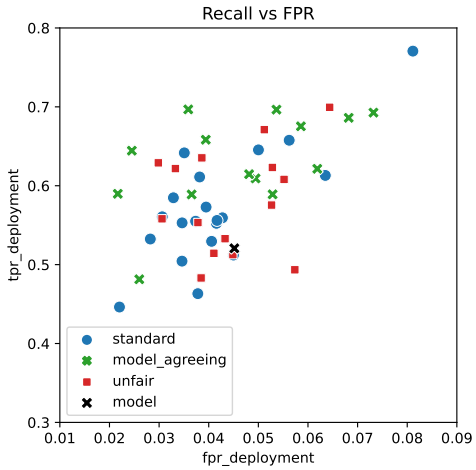

Figure 6: Deployment TPR and FPR

Table 3: Complementary Expertise: Fraction of instances within each class where expert and model disagree. y_x denotes the label, m_x the model's prediction, and e_x the expert's prediction. The subscript denotes the label/prediction value.

Expert Group	y_0, m_0, e_1	y_0, m_0, e_1	y_0, m_0, e_1	y_0, m_0, e_1
Standard	0.224	0.174	0.037	0.040
Model Agreeing	0.155	0.060	0.022	0.024
Unfair	0.229	0.169	0.038	0.039

Despite the higher average performance of our experts, we expect the model and the experts to have complementary expertise. We test if there is a significant number of instances where the model predicts correctly, and the expert incorrectly, or vice-versa. To do so, we calculate the fraction of instances within each label y where the expert e is correct and the model M is incorrect, or vice versa. These results are presented in Table 3. These show that both the expert and the model have a large amount of cases where disagreement takes place. We can also see that Model Agreeing experts exhibit less disagreement with the model. Using Cohen's Kappa, which aims to measure inter-labeler agreement between two sets of predictions, accounting for differences in prevalence, we can confirm that the model-agreeing experts' predictions are more similar to the model's. In Figure 8 we present the Cohen's Kappa for each expert in relation to the model's predictions.

B.3 Expert behavior modeling

To model each expert's behavior we again use the LightGBM algorithm to train an expertise model. The input features of this model are the instance's features, the ML Model score, and the expert's id. The ML Model score is included as it also influences expert decisions, while the expert id is included in order to allow for individual modelling of each individual expert's predictions. This model is trained in months 4 to 6, and validated on the 7th month. The task is a classification problem with classes [TP,TN,FN,FP], where we aim to estimate the probability that the expert's decision belongs to each of the classes. The model is trained by minimizing multi-class cross-entropy loss. The choice of hyperparameters is defined through Bayesian search [1] on an extensive grid, for 50 trials, with validation done on the 4th month, where the optimization objective is aligned with the loss function, aiming to minimize the cross entropy loss. This objective was selected to obtain accurate probability estimates. In table 5 we present the hyperparameter search space used, as well as the parameters of the selected model.

Table 4: Individual Expert's Parameters

	Parameters				
Expert	T_{FPR}	T_{FNR}	α	w_p	w_M
standard#0	0.392	0.038	3.96	-1.11	-1.19
standard#1	0.492	0.038	3.82	-0.89	-2.31
standard#2	0.455	0.042	3.85	-0.91	-2.26
standard#3	0.414	0.044	4.34	-0.95	-2.54
standard#4	0.454	0.042	4.01	-0.91	-1.57
standard#5	0.426	0.041	3.87	-1.07	-3.15
standard#6	0.486	0.033	4.04	-1.01	-1.13
standard#7	0.505	0.044	4.42	-1.09	-2.38
standard#8	0.539	0.041	4.02	-1.03	-1.84
standard#9	0.359	0.051	4.12	-0.95	-2.12
standard#10	0.357	0.052	4.06	-1.07	-1.27
standard#11	0.404	0.042	3.93	-1.04	-3.03
standard#12	0.437	0.036	3.77	-1.07	-2.16
standard#13	0.473	0.034	3.93	-1.08	-2.19
standard#14	0.445	0.044	3.96	-1.07	-1.43
standard#15	0.328	0.041	4.12	-1.00	-2.55
standard#16	0.414	0.037	4.17	-1.11	-2.09
standard#17	0.471	0.040	4.19	-0.98	-2.44
standard#18	0.441	0.034	4.06	-0.83	-1.98
standard#19	0.467	0.047	4.18	-0.93	-1.71
model_agreeing#0	0.413	0.036	11.87	-0.06	-6.21
model_agreeing#1	0.460	0.039	13.12	0.00	-6.03
model_agreeing#2	0.432	0.040	10.78	0.12	-7.07
model_agreeing#3	0.388	0.043	12.06	-0.07	-5.18
model_agreeing#4	0.434	0.020	12.19	0.00	-6.90
model_agreeing#5	0.479	0.040	12.68	-0.09	-6.42
model_agreeing#6	0.410	0.033	12.25	-0.02	-5.75
model_agreeing#7	0.411	0.026	11.58	0.03	-6.62
model_agreeing#8	0.426	0.045	12.00	-0.10	-6.53
model_agreeing#9	0.451	0.047	12.27	-0.03	-6.45
model_agreeing#10	0.493	0.033	11.84	-0.02	-5.72
model_agreeing#11	0.398	0.048	12.39	-0.06	-4.85
model_agreeing#12	0.455	0.036	11.07	-0.12	-5.98
model_agreeing#13	0.440	0.041	12.87	-0.14	-6.56
model_agreeing#14	0.336	0.029	12.73	-0.02	-5.73
unfair#0	0.400	0.050	4.07	-4.12	-1.11
unfair#1	0.386	0.053	4.20	-4.16	-1.78
unfair#2	0.431	0.038	3.88	-4.46	-1.95
unfair#3	0.317	0.058	3.94	-3.71	-2.93
unfair#4	0.416	0.033	3.92	-4.33	-2.14
unfair#5	0.480	0.042	3.76	-4.36	-2.18
unfair#6	0.472	0.042	3.91	-4.06	-2.04
unfair#7	0.485	0.029	3.90	-3.55	-2.31
unfair#8	0.485	0.044	4.11	-3.93	-2.02
unfair#9	0.504	0.059	3.99	-4.31	-2.24
unfair#10	0.373	0.030	3.84	-4.21	-2.66
unfair#11	0.472	0.049	4.06	-3.81	-1.56
unfair#12	0.336	0.038	3.96	-4.05	-1.56
unfair#13	0.399	0.058	3.83	-4.23	-1.15
unfair#14	0.334	0.045	3.94	-4.07	-1.97

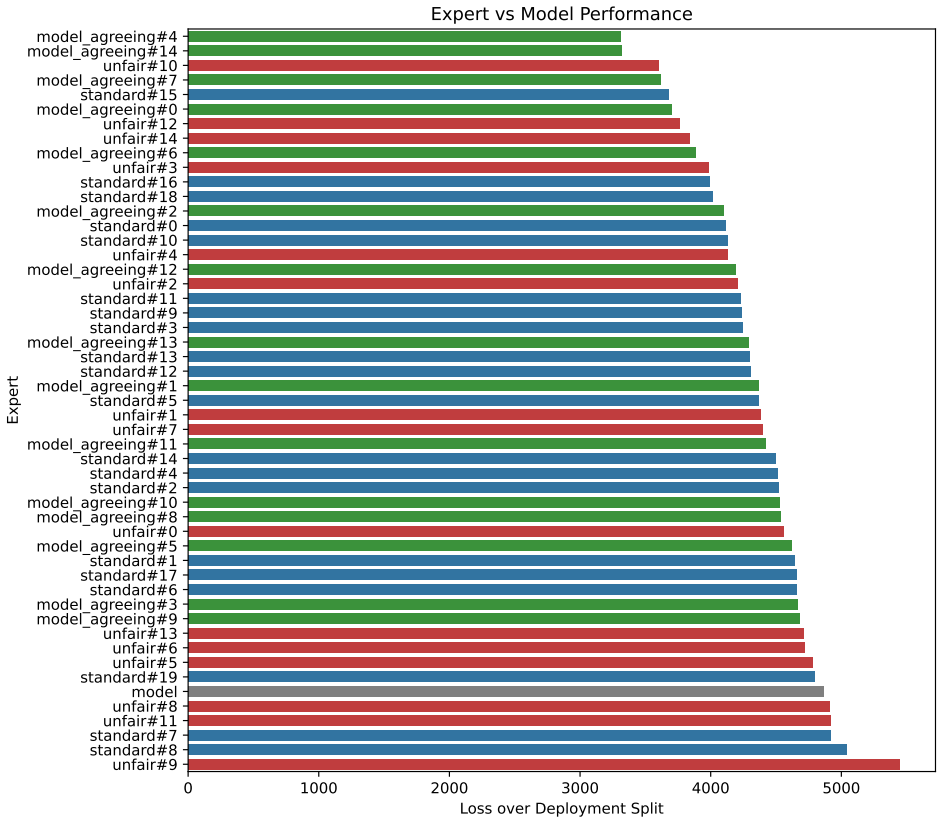

Figure 7: Normalized Feature weights

Table 5: Expertise Model: LightGBM hyperparameter search space

Hyperparameter	Values or Interval	Distribution	Selected value
boosting_type	{“goss”, “gbdt”}		“goss”
enable_bundle	{False, True }		True
n_estimators	[50,5000]	Logarithmic	303
max_depth	[2,20]	Uniform	5
num_leaves	[10,1000]	Logarithmic	158
min_child_samples	[5,500]	Logarithmic	152
learning_rate	[0.01, 0.5]	Logarithmic	0.021054
reg_alpha	[0.0001, 0.1]	Logarithmic	0.037153
reg_lambda	[0.0001, 0.1]	Logarithmic	0.002799

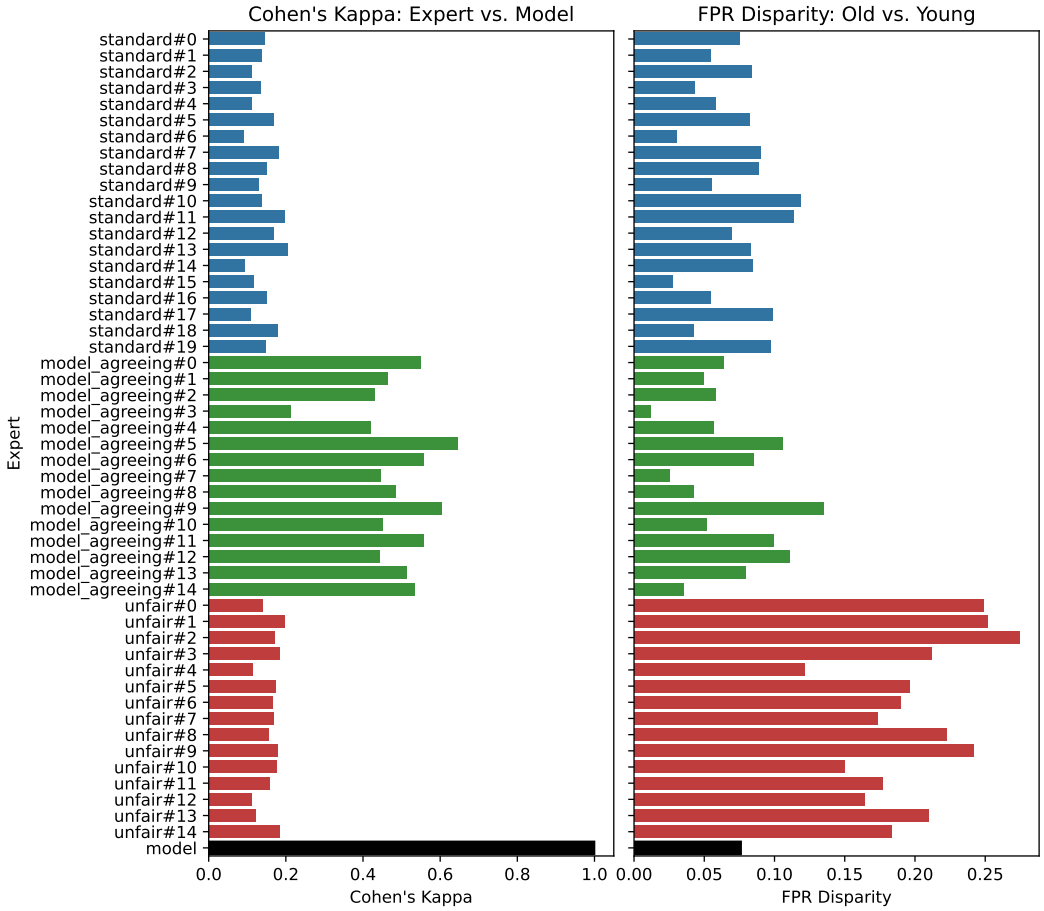

Figure 8: Cohen's Kappa and FPR Disparity across experts

We now verify that our model can predict human behavior. In order to use our assignment systems, the primary goal of our model is to predict the probability that a given expert will commit either a FP or FN mistake. To evaluate our model's performance, we plot the ROC Curves on a *one-versus-others* basis. To plot the ROC Curves pertaining to the prediction of False positives, we consider FP predictions as the positive class, and concatenate all other classes into the negative class. The same process is applied to the False negative predictions. The ROC curves obtained are presented in Figures 9 and 10

Regarding the FP ROC curves, we can observe some expected results. The regular experts display the least predictable decision process, the unfair experts are more predictable, while the model agreeing experts are the most predictable. We suspect this takes place due to a combination of factors. Firstly, the decision processes of the model agreeing and unfair experts are mostly dominated by one feature: the model score, and the customer's age, respectively. Furthermore, the distribution of the probabilities of error for each instance are distinct across groups, as shown in Figures 2 and 3. When observing the FN ROC curves, we cannot draw the same conclusions. This may be due, in part, to the fact that the entire deployment split only contains a total of 7133 human predictions, hindering the model's ability to capture behavioral patterns. Furthermore, the class imbalance between the four classes [FP, TP, FN, TN] may lead to worse predictions being output for False Negatives. Concluding, we verify that our model is capable of predicting the outcome of an expert's prediction.

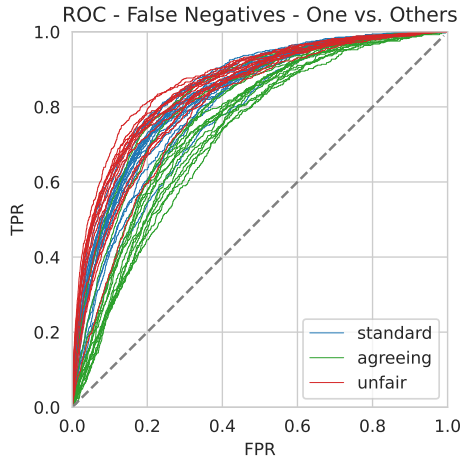

Figure 9: ROC-Curve ML Model - Deployment Split

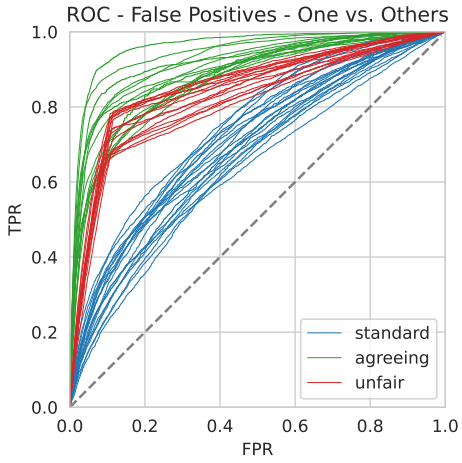

Figure 10: ROC-Curve ML Model - Deployment Split

C Experimental Results

Despite our attempts to develop an assignment system that takes the human's expertise into account, the value of the loss for our human expertise aware methods are surpassed by the *rejection learning* approach, which defers randomly to humans, without surpassing their work capacity. These results are presented in the paper, in Table 1. To further evaluate each method's performance, and compare the properties of each method, we present the values of the false positive rate (FPR) versus the true positive rate (TPR) for each of our baselines, in each of our scenarios. These results are available in Table C

Table 6: Baseline Results. FPR standard deviations are omitted due to low variability. Results for Full Automation are FPR $= 3.9\%$, TPR $= 49.8\%$. "Batch" refers to the batch size, "Def" to the deferral rate, "Abs" to the absence rate, and Homogeneous scenarios are denoted by omitting σ_d.

Scenario Properties				ReL(%)		$\text{ReL}_{\text{greedy}}(\%)$		$\text{ReL}_{\text{linear}}(\%)$	
Batch	Def.	Abs.	σ_d	FPR	TPR	FPR	TPR	FPR	TPR
1000	0.2	0.0	-	5.0	65.8±0.7	4.4	63.0±0.9	4.4	61.6±0.4
1000	0.2	0.0	0.2	5.0	66.1±0.7	4.4	62.9±0.6	4.4	61.7±0.3
1000	0.2	0.5	-	5.0	65.8±1.2	4.6	63.0±0.8	4.5	63.1±0.7
1000	0.2	0.5	0.2	5.0	66.0±0.8	4.5	63.1±0.7	4.5	62.9±0.8
1000	0.5	0.0	-	5.8	69.3±0.5	4.8	65.4±0.8	4.6	62.0±0.2
1000	0.5	0.0	0.2	5.8	69.2±0.8	4.9	66.1±0.9	4.6	62.3±0.4
1000	0.5	0.5	-	5.8	70.0±1.2	5.0	68.0±0.8	4.7	63.7±0.2
1000	0.5	0.5	0.2	5.8	69.9±0.9	4.9	67.8±1.4	4.7	63.8±0.3
5000	0.2	0.0	-	5.0	66.0±0.8	4.4	63.3±0.2	4.4	61.6±0.2
5000	0.2	0.0	0.2	5.0	65.9±0.6	4.4	63.1±0.4	4.4	61.6±0.3
5000	0.2	0.5	-	5.0	66.3±0.8	4.5	62.2±1.4	4.5	63.1±0.7
5000	0.2	0.5	0.2	5.0	66.0±0.9	4.5	62.6±1.0	4.5	63.0±0.7
5000	0.5	0.0	-	5.8	69.1±1.1	4.8	65.3±0.4	4.6	61.9±0.2
5000	0.5	0.0	0.2	5.8	69.5±1.0	4.9	65.7±0.9	4.6	62.2±0.3
5000	0.5	0.5	-	5.8	69.5±1.3	5.0	68.6±0.8	4.7	63.8±0.2
5000	0.5	0.5	0.2	5.8	69.6±1.3	4.9	68.3±1.7	4.7	63.9±0.3

We observe that the values for the false positive rate are lowered by our $\text{ReL}_{\text{greedy}}$ and $\text{ReL}_{\text{linear}}$ methods. This reduction in the amount of false positives entails a reduction in recall, resulting in a higher loss for the expertise-aware methods. Furthermore, in all cases, the linear method, which we

expected to yield better assignments, results in a more drastic reduction of the false positive rate, when compared to the greedy method.

As the loss minimized by our methods is the same one used in evaluation, we expected our methods to result in a lower total loss, as our human expertise model is able to predict the correctness of human predictions (see Figures 9, 10). However, the probability estimates obtained by using our model may be miscalibrated. In Figures 11 and 12, we display the calibration plots, or reliability diagrams, obtained for predictions in the test split. These are calculated separately for false positive and false negative probability estimates, in a *one-versus-others* basis. All expert's predictions and probability estimates, for all test split instances, were used to calculate these calibration plots.

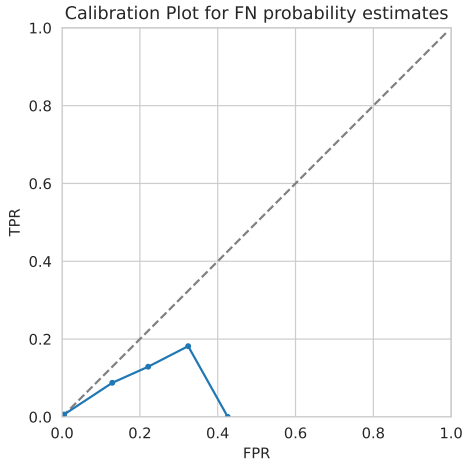

Figure 11: Calibration Plot - FN probability estimates

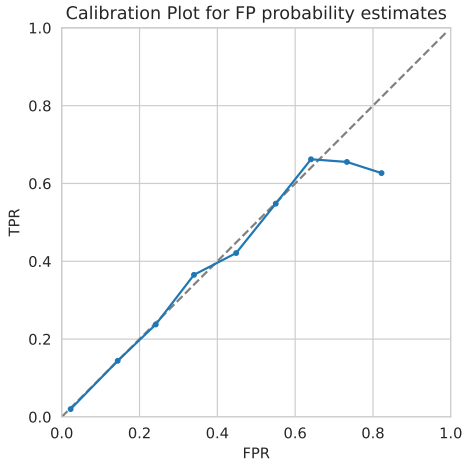

Figure 12: Calibration Plot - FP probability estimates

These calibration plots demonstrate that while the FP probability estimates are reasonably calibrated, the FN probability estimates are greatly underestimated. According to the output of our expertise model, the estimated probability of committing a false negative mistake never surpasses values around 0.4, despite the values of the generated probabilities of a FN error far surpassing this value, as shown in Figure 2.

The underestimation of the probability of a FN occurring leads our expertise-aware approaches to overestimate the contribution of False Positive mistakes to the overall Loss. This, in turn, leads our assignment systems to heavily mitigate FP mistakes, at the cost of an increase in false negatives. Attempts to calibrate our predictions using isotonic regression [5] on the validation split did not yield calibrated estimates in the test set. This is most likely due to the fact that on the validation set, a total of 54084 predictions per expert are available, with only 685 predictions for label positive instances. This low volume of label positive expert predictions hinders our ability to calibrate the false negative probability estimates.

As the parameter λ enforces a trade-off between TPR and FPR, a possible alternative is to test several values of λ on the validation set of the human expertise model, as mentioned in section A.1. To evaluate each value of λ, the human expertise model can be used to calculate the corresponding predicted FPR, thus not requiring extra predictions from humans. Then, the value of λ with the predicted FPR closest to, but below the desired level is chosen.

C.1 Total Compute and Resources Used

In Table 7 we list the total compute time for each of the most time consuming tasks: training the ML Model, generating synthetic experts, training the human expertise model and, finally, testing our three baseline methods in 220 training scenarios. All times displayed were obtained using 25

cores of Intel(R) Xeon(R) Gold 5120 CPU @ 2.20GHz. For each of our methods, assignments were computed in parallel, making us unable to provide a time for each experiment individually.

Table 7: Computing times for each stage of the experiments

Task	Compute Time
ML Model Training	00h27m
Synthetic Expert Generation	00h18m
Expertise Model Training	03h47m
All assignment experiments	25h32m

D Reproducibility

D.1 Benchmark Dataset

D.1.1 Data Pre-Processing

To pre-process the data in order to generate the experts we used the functions *StandardScaler*, *QuantileTransformer* and *OrdinalEncoder*, from the *Sci-kit Learn* package, version $0.24.2$. The *QuantileTransformer* function takes a subsample of data, so a random state was set. This ensures each run produces the same pre-processed data. In our experiments, we had not set this random seed, yielding a slightly different transformed dataset. We provide the transformed dataset utilized in our experiments in order to enable reproducible results across different versions and systems.

D.1.2 Capacity Constraints

When distributing cases throughout batches, the shuffling is done via a batch seed, defined by the user. This ensures that batches may be generated in the same manner throughout experiments. When generating capacity constraints for each batch, certain process involve random sampling. These are sampling each expert's capacity if the distribution is variable, or sampling which experts are absent in a given batch. A user is able to set a seed for the absence sampling, and a separate seed for the distribution sampling. This not only allows for the creation of different scenarios with the same absence rate or distribution properties, but it also ensures that capacity constraint generation is reproducible.

D.1.3 Expert Predictions

Given the same transformed data, and the same seeds defined in the group properties, the expert generation process will output the same predictions. As mentioned previously, the transformed data necessary to generate the expert decisions is available in the repository. With this training data, the experts generated should be identical to the ones used in our experiments.

D.1.4 Training Dataset

The training dataset used for our experiments involves randomly distributing 50% of the cases in each batch throughout our human team, uniformly. At the time of our experiments, this random distribution was done with no random seed. We have updated our code so that this random distribution is seeded. To enable users to obtain identical results to our experiments, we provide our training scenario data.

D.2 Benchmark Experiments

D.2.1 ML and Expertise Model

When training LightGBM models with the use of multi-threading, results may fluctuate slightly between successive runs. Furthermore, according to the documentation, for "different LightGBM versions, the binaries compiled by different compilers, or in different systems, the results are expected

to be different". As such, to obtain the same results in our experiments, users will have to use the provided ML and human expertise models.

D.2.2 Experiment Results

Given all the previously mentioned material, the results of an experiment run will yield the same results. The random assignment within Rejection learning is seeded, and the other methods obtain a solution through deterministic methods. The assignments are saved in files identified by the experiment's parameters.

E Available Materials

The code necessary to use our framework will be made publically available. To use our framework, users must install the package "autodefer", included with the available code. This package is necessary to handle the synthetic expert objects, and contains implementations of the L2D methods referred in the paper. To ensure complete reproducibility of our experiments, for the reasons stated in the previous section, we will also include:

- Transformed data
- ML Model and Expertise model
- Training Scenario

All other necessary data to run the experiments can be obtained by generating it with access to these materials. Due to the size of these, a link to a public drive containing the datasets and used models is available in our repository. Detailed instructions on how to use the provided resources and code are also made available in the repository.

F Accessibility and Long-Term Preservation Plan

Our code will be available in a public repository at `https://anonymous.4open.science/r/openl2d-7BD3`. If any change is made to the framework's code or new L2D algorithms are added, we will update the instructions for use accordingly. The data and code necessary to reproduce our experiments will be made available at a public drive, linked in the repository.

G License

The authors confirm the datasets provided are under the Creative Commons CC BY-NC-ND 4.0 license. The authors bear responsibility in case of violation of copyrights.

H Structured Metadata

Our code is available on GitHub, so structured metadata is automatically generated by the platform. Since we are using an existing data repository (Google drive?) to host our dataset, the structured metadata is automatically generated by the platform. [Nota]

I Datasheets for Datasets

Throughout the course of our experiments, we used a training dataset generated in order to simulate a realistic availability of human predictions. This training dataset was generated by utilizing the generated expert predictions of our expert team, respecting their work capacity constraints. Development of L2D algorithms can be conducted on this dataset, in order to test the performance of methods trained on limited human data. As such, we provide our training dataset and synthetic human predictions as

a public dataset. The datasheet for our training dataset and synthetic expert predictions is available on our GitHub repo.

References

[1] Akiba, T., Sano, S., Yanase, T., Ohta, T., and Koyama, M. (2019). Optuna: A Next-generation Hyperparameter Optimization Framework. In Teredesai, A., Kumar, V., Li, Y., Rosales, R., Terzi, E., and Karypis, G., editors, *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*, pages 2623–2631. ACM.

[2] Elkan, C. (2001). The Foundations of Cost-Sensitive Learning. In Nebel, B., editor, *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, IJCAI 2001, Seattle, Washington, USA, August 4-10, 2001*, pages 973–978. Morgan Kaufmann.

[3] Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., and Liu, T.-Y. (2017). LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In Guyon, I., von Luxburg, U., Bengio, S., Wallach, H. M., Fergus, R., Vishwanathan, S. V. N., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 3146–3154.

[4] Sheng, V. S. and Ling, C. X. (2006). Thresholding for Making Classifiers Cost-sensitive. In *Proceedings, The Twenty-First National Conference on Artificial Intelligence and the Eighteenth Innovative Applications of Artificial Intelligence Conference, July 16-20, 2006, Boston, Massachusetts, USA*, pages 476–481. AAAI Press.

[5] Zadrozny, B. and Elkan, C. (2002). Transforming classifier scores into accurate multiclass probability estimates. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 694–699.