CEROP Week 6: Finalizing Models - Financial and Operations Risk Layers

Ankit Kumar,

Aswin Shriram Thiagarajan

Arthunya Kanoklertwongse

Divya Gopal

Westcliff University

AIT 506: Machine Learning

Prof. Oueichek Ibba

August 8, 2026

CEROP Week 6: Finalizing Models - Financial and Operations Risk Layers

Introduction

Week 5 settled a model-selection question for each half of CEROP independently, which architecture at default settings, does the best separates risk from non-risk in each domain. XGBoost was shortlisted for the Financial layer which is bankruptcy prediction on 85 Taiwan-firm ratios and Random Forest for the Operations layer which is late delivery prediction on DataCo supply chain records. To narrow on Week questions, given the chosen architecture, what is its best configuration, and does finding it actually improve anything?

The two layers were tuned under the same protocol, a validation split carved out of training data, hyperparameter search scored by a metric matched to each layer's imbalance, threshold calibration on validation only and one sealed evaluation on the test set so that “before” and “after” are comparable within each layer. The two layers giving different answers. That contrast is the subject of the Cross Layer Synthesis that closes this report, after both layers are presented in full below.

Note. Full Finalizing Models code Week6_Financial_Model_Finalization.ipynb AIT506_Week6_Operations_Model_Tuning.ipynb

Part I: Financial Risk Layer

Method

Experimental Protocol

A before/after comparison is only meaningful if both sides are measured identically. Two design decisions enforce this. First, a stratified 20% validation split was carved out of the training rows, yielding 4,364 rows for fitting (141 bankrupt) and 1,091 for selection (35 bankrupt). Every subsequent decision — the hyperparameter search, the Random Forest re-tune,

the threshold calibration, and the ensemble comparison — was made using only these two partitions. The test set was not read until the final evaluation. Selecting a hyperparameter or a threshold against test-set performance would convert the test set into a training set and render the reported figure a fiction.

Second, the Week 5 configuration was refit on the identical 4,364-row split to serve as the baseline. Week 5's published numbers came from a model trained on all 5,455 rows, and comparing that directly against a tuned model trained on 4,364 rows would confound the effect of tuning with the effect of training-set size. The published Week 5 result is retained in the exported tables as a labelled reference row, but the refit baseline is the comparison that carries the argument. Table 1 documents the partition.

Table 1 Data Partition and the Role of Each Split

Split	Rows	Bankrupt	Rate	Used for
Tuning fit	4,364	141	0.0323	Cross-validated hyperparameter search
Validation	1,091	35	0.0321	Model choice and threshold calibration
Test (sealed)	1,364	44	0.0323	One final evaluation

Hyperparameter Search

XGBoost was tuned with RandomizedSearchCV over 40 sampled configurations, each scored by 5-fold stratified cross-validation, for a total of 200 model fits. The scoring function was average_precision rather than accuracy, for the reason given above. The search space contained 15,360 combinations, so the 40 draws cover 0.26% of the grid:

```
param_dist = {
    "n_estimators":     [100, 200, 300, 500],
    "max_depth":        [3, 4, 5, 6, 8],
    "learning_rate":    [0.01, 0.05, 0.1, 0.2],
    "subsample":        [0.7, 0.8, 0.9, 1.0],
    "colsample_bytree": [0.7, 0.8, 0.9, 1.0],
    "scale_pos_weight": [1, 5, 10, 20],
    "min_child_weight": [1, 3, 5],
}
```

Randomized search was chosen over exhaustive grid search on cost grounds: GridSearchCV over the same space at 5-fold would require 76,800 fits. The standard justification for the randomized alternative is that few hyperparameters dominate performance, and random draws sample every axis at full resolution rather than spending the budget on a coarse lattice (Bergstra & Bengio, 2012). Section 'Why Tuning Did Not Help' revisits whether that assumption held here.

Random Forest was re-tuned under an identical budget 40 draws, 5-fold, same scorer, same split because a tuned model outperforming an untuned rival demonstrates nothing about the underlying ranking. The Random Forest grid additionally varied class_weight, which the team's operations-layer specification omits. That layer is near-balanced; this one runs at roughly 30:1, so withholding a re-weighting option would have handicapped Random Forest against an XGBoost whose scale_pos_weight was being tuned, and made the comparison unfair in the opposite direction.

Threshold Calibration

Week 5 established that the 0.5 decision threshold is an arbitrary default rather than a property of a model, and that apparent recall advantages dissolve once models are held to equal recall. That argument applies reflexively to XGBoost's own threshold. Two operating points were therefore selected on the validation split: the F1-optimal threshold, which balances missed

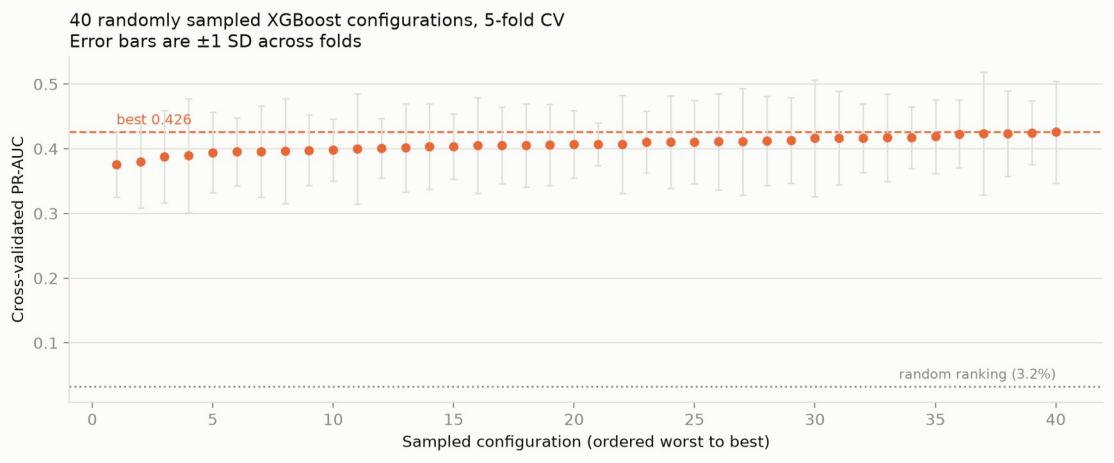
bankruptcies against false alarms without importing an external cost assumption, and the most precise threshold achieving at least 80% recall, which matches the recall band Week 5 used as a reference and reflects a lender's asymmetric loss function. Both were fixed before the test set was opened.

Results

The Search Behaved Normally

The search converged on a configuration scoring 0.4257 cross-validated PR-AUC, against 0.3895 for the Week 5 baseline scored under the identical cross-validation object an improvement of +0.0362. By the conventional reading, the search succeeded. Figure 1 shows the full distribution of sampled configurations and is the first indication that this reading is too generous.

Figure 1 Cross-Validated PR-AUC Across 40 Sampled XGBoost Configurations



Note. Configurations are ordered worst to best; error bars are ±1 standard deviation across the 5 folds. Every error bar overlaps the dashed line marking the best score, which is the visual signature of a search whose ranking is not statistically identified.

Table 2 Selected Hyperparameters and Cross-Validated Scores

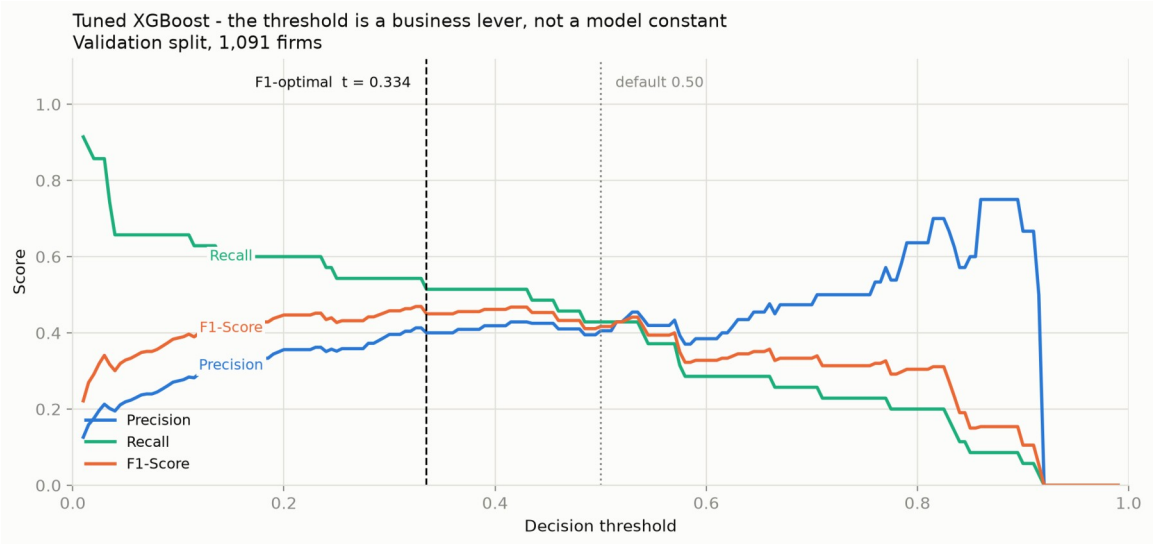
Model	Selected configuration	CV PR-AUC

XGBoost (tuned)	colsample_bytree=0.8, learning_rate=0.05, max_depth=8, min_child_weight=1, n_estimators=100, scale_pos_weight=10, subsample=0.9	0.4257 (SD 0.0791)
Random Forest (tuned)	class_weight=balanced, max_depth=None, max_features=sqrt, min_samples_leaf=8, min_samples_split=10, n_estimators=800	0.3994
XGBoost (Week 5 baseline)	colsample_bytree=0.8, learning_rate=0.1, max_depth=4, min_child_weight=1, n_estimators=300, scale_pos_weight=29.95, subsample=0.8	0.3895 (SD 0.0590)

Threshold Calibration on the Validation Split

Figure 2 shows precision, recall, and F1 for the tuned model across the full range of decision thresholds. The F1-optimal threshold is 0.335, well below the 0.5 default, which is the expected direction under class imbalance: the default is calibrated for balanced data and is systematically too conservative when positives are rare.

Figure 2 *Precision, Recall, and F1 Across Decision Thresholds*



Note. Tuned XGBoost on the 1,091 firm validation split. The threshold is a business lever rather than a model constant; the model fixes the attainable frontier and the threshold selects a point on it.

Test Set Performance

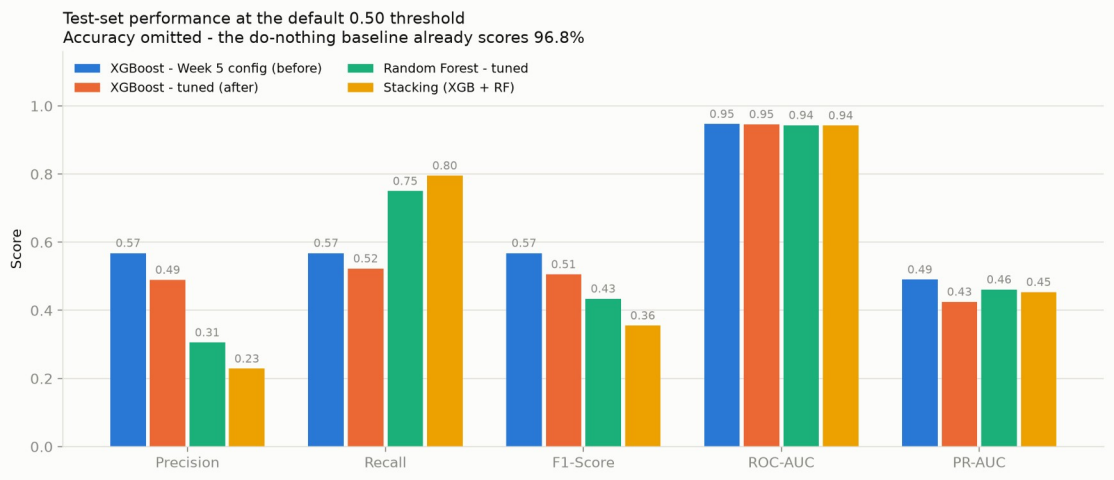
With all hyperparameters and thresholds locked, the test set was opened once. Table 3 reports the result. Tuning reduced F1 from 0.5682 to 0.5055 (-0.0627) and PR-AUC from 0.4902 to 0.4250 (-0.0652). Threshold calibration recovered part of the recall from 0.523 to 0.591, catching 26 of 44 bankruptcies rather than 23 but purchased it with false alarms rising from 24 to 42, so F1 fell further to 0.4643. The aggressive recall-targeted threshold catches 35 of 44 at the cost of 124 false alarms.

Table 3 Test-Set Performance Before and After Tuning

Model	Thr.	Prec.	Recall	F1	ROC-AUC	PR-AUC	Caught
XGB - Week 5 published	0.500	0.535	0.523	0.529	0.950	0.476	23/44
XGB - Week 5 config	0.500	0.568	0.568	0.568	0.947	0.490	25/44
XGB - tuned	0.500	0.489	0.523	0.505	0.946	0.425	23/44
XGB - tuned @ calibrated	0.335	0.382	0.591	0.464	0.946	0.425	26/44
XGB - tuned @ recall>=80%	0.030	0.220	0.795	0.345	0.946	0.425	35/44
RF - tuned	0.500	0.306	0.750	0.434	0.944	0.461	33/44
RF - tuned @ calibrated	0.791	0.429	0.477	0.452	0.944	0.461	21/44
RF - tuned @ recall>=80%	0.331	0.233	0.795	0.361	0.944	0.461	35/44
Stacking (XGB + RF)	0.500	0.229	0.795	0.355	0.943	0.454	35/44
Stacking (XGB + RF) @	0.949	0.415	0.500	0.454	0.943	0.454	22/44

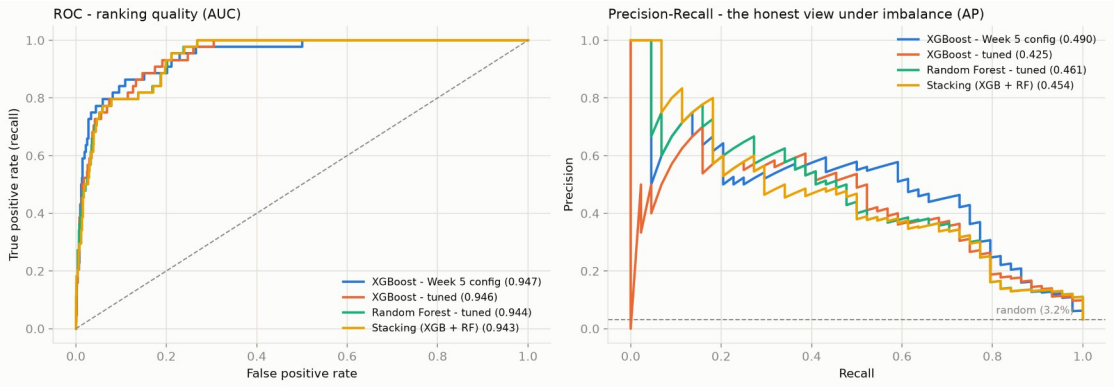
calibrated							
Stacking (XGB + RF) @ recall>=80%	0.518	0.235	0.795	0.363	0.943	0.454	35/44

Figure 3 Test-Set Metrics by Model at the Default Threshold



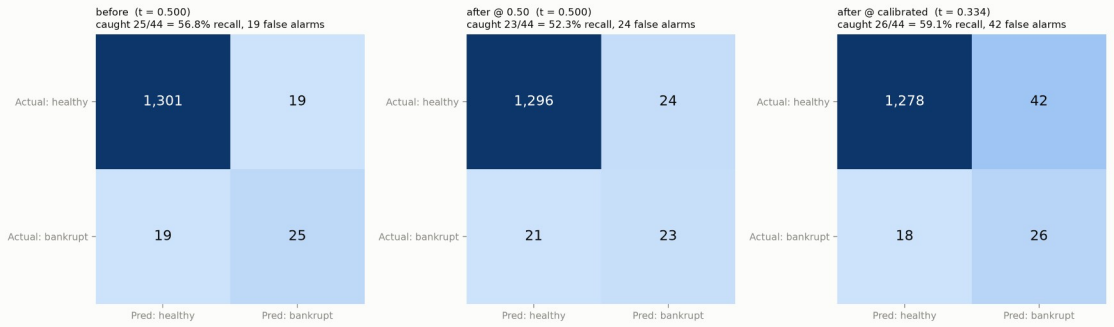
Note. Accuracy is omitted because the majority-class baseline already scores 96.8%, so it cannot distinguish a useful model from a useless one.

Figure 4 *ROC and Precision-Recall Curves on the Test Set*



Note. ROC-AUC separates the four models by less than 0.005, whereas the precision-recall view the appropriate one under imbalance separates them more visibly.

Figure 5 *Confusion Matrices Before Tuning, After Tuning, and After Calibration*



Note. Cell shading is a single-hue sequential ramp on a logarithmic scale; the true negative cell dominates every panel by two orders of magnitude, which is itself the reason accuracy is uninformative here.

The Random Forest Check and the Ensemble

Random Forest did not overtake XGBoost on either surface available for selection: cross-validated PR-AUC of 0.3994 against 0.4257, and validation PR-AUC of 0.3794 for the baseline against 0.3780 for the tuned model. On the sealed test set, however, tuned Random Forest does edge tuned XGBoost on PR-AUC (0.4610 against 0.4250) and catches 33 of 44 bankruptcies against 23, though at 75 false alarms against 24. This reversal is reported because it happened, not because it changes the recommendation: it appears only on the surface with the smallest positive count and the widest error bars, and it does not appear on either surface the protocol permits for selection.

The optional stacking ensemble tuned XGBoost and tuned Random Forest as base learners with a Logistic Regression meta-learner trained on out-of-fold probabilities did not beat the tuned single model on validation PR-AUC and was not adopted. On the test set it scores 0.4536, between the two base learners. This is a valid negative finding and is reported as one.

Why Tuning Did Not Help

A negative result is only useful if its cause is identified, because the cause determines the remedy. Three explanations were tested, each implying a different corrective action.

The first hypothesis — that the search simply failed to find anything better — is rejected. The search located a configuration scoring +0.0362 PR-AUC above the baseline in cross-validation. Whatever went wrong occurred between cross-validation and deployment, not during the search itself.

The second hypothesis is that the search's ranking signal is smaller than its own measurement noise, and it holds decisively. The spread between the best and worst of the 40 configurations is 0.0498 PR-AUC, while the mean fold-to-fold standard deviation within a single configuration is 0.0649 — a signal to noise ratio of 0.77. All 40 configurations lie within one standard deviation of the winner. The mechanism is arithmetic: 141 bankrupt firms divided across 5 folds leaves roughly 28 positives per fold, and average precision estimated on 28 positives is inherently high-variance. Taking the maximum over 40 such estimates therefore selects, in large part, whichever configuration drew the most favourable fold assignment. This is the winner's curse, and it predicts precisely the pattern observed — a cross-validation score that looks best and does not transfer.

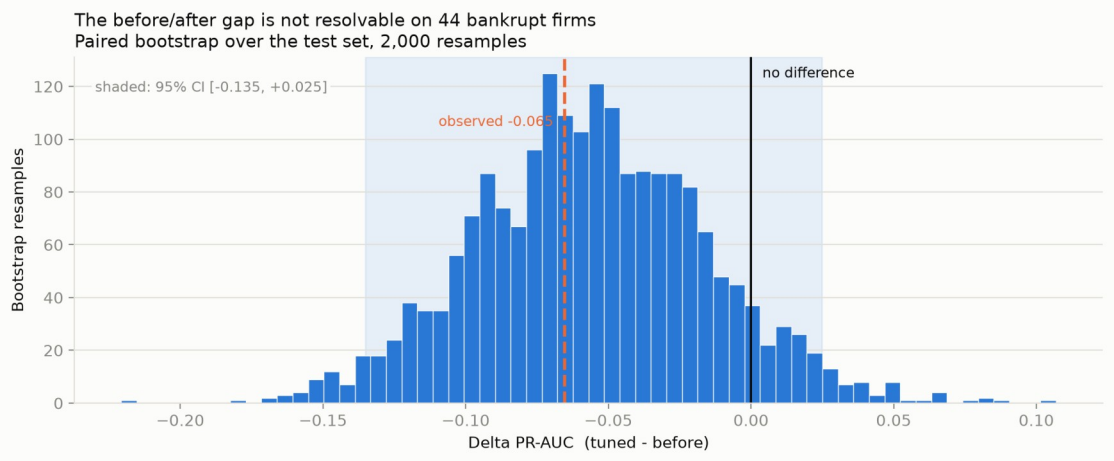
The third hypothesis is that the specified grid could not reach the baseline, and it also holds. The grid caps scale_pos_weight at 20 while the empirical class ratio on the tuning split is 30.0. Every other baseline parameter is inside the grid; this one is not. No number of draws could have recovered the Week 5 configuration, because it lies outside the space being searched. A diagnostic rerun with that single axis widened confirms the mechanism: the search then selects scale_pos_weight = 30 and reaches validation PR-AUC of 0.3985, ahead of both the baseline

(0.3794) and the specified-grid winner (0.3780). That diagnostic was scored on validation only and is deliberately excluded from the exported deliverables; the test set had already been spent, and re-tuning against it until the number improved is the exact failure this protocol exists to prevent.

How Much of This Is Real?

The preceding section explains the direction of the result. This section establishes its magnitude, and the answer is that the magnitude is not distinguishable from zero. A paired bootstrap over the test set with 2,000 resamples places the before/after difference in PR-AUC at -0.0652, with a 95% confidence interval of [-0.1351, +0.0249]. The interval straddles zero. The tuned model outperforms the baseline in 8.8% of resamples a minority, but far from a decisive one. Expressed in units that matter to a credit committee, the entire before/after difference amounts to 2 firms out of 44.

Figure 6 Bootstrap Distribution of the Before/After Difference in PR-AUC



Note. The shaded band is the 95% confidence interval, which includes zero. The observed difference sits near the centre of a distribution wide enough to accommodate improvements and deteriorations of similar size.

The correct claim is therefore not that tuning degraded the model. It is that tuning produced no measurable improvement, and that this test set cannot resolve differences of the size under discussion. The 44 firm positive count flagged as a caveat in Week 5 is, on closer inspection, the binding constraint on the entire exercise. It limits what the search can detect, what the validation split can select, and what the test set can confirm.

Limitations

- The validation split contains only 35 bankrupt firms. Thresholds calibrated on it carry substantial uncertainty, which is one reason the calibrated model does not outperform the default on test data.
- Model selection used a single stratified split and a single 5-fold cross-validation. Repeated cross-validation with several seeds would give a more stable ranking, at proportionally higher compute cost.
- The bootstrap resamples test rows but treats the fitted models as fixed. It quantifies evaluation uncertainty, not the additional variance that would come from refitting on resampled training data.
- Results are reported for a single random seed (random_state = 42). Given the noise levels documented above, seed-to-seed variation in the selected configuration should be expected.

Part II: Operations Risk Layer

Overall Process

The new updated dataset contains 137569 training rows and 34393 test rows, and a late delivery target percentage of 54.82% roughly in both the datasets. We also performed a stratified split of 80-20 on the training data to create 27514 validation rows and 110055 training rows. The validation data was used to adjust the prediction threshold only and not involved in hyperparam search. For the hyper parameter search we sampled 50000 records out of the 110055, to save on computational cost and training efficiency. Because RandomizedSearchCV can be exhaustive on large datasets.

In RandomizedSearchCV, we tested with 25 random combinations from a distribution of **n_estimators**: 200 to 800; **max_depth**: [10, 15, 20, unlimited]; **min_samples_leaf**: 1 to 8; **min_samples_split**: 2 to 10; **max_features**: [sqrt, log2, or 0.5]. We also used the F1 score as the target metric and performed a **3-fold cross validation**. We then applied prediction thresholds from 0.1 to 0.9 on the resulting models on the validation dataset and selected the thresholds that maximized the F1 score. In the end, we ran both the untuned baseline model and the fine tuned model in the training data, and generated the final models to compare against on the test dataset. The performance was compared with threshold of 0.5 for the baseline model and the calibrated threshold for the fine tuned model.

Results

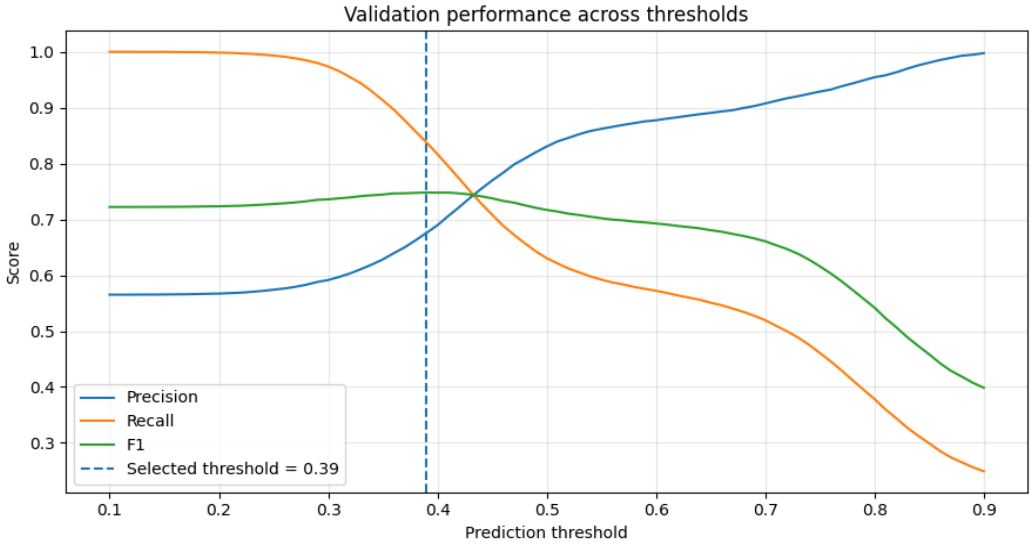
Table 1 shows the final selected hyperparameters of the random search experiment for our Random Forest model.

Table 1 *Selected Random Forest Hyperparams.*

Hyperparameter	Selected Value
n_estimators	500
max_depth	unlimited
min_samples_split	5
min_samples_leaf	1
max_features	0.5
Best CV F1 score	0.7051

Our threshold experiment showed precision and recall moving in opposite directions as we changed the threshold. A lower threshold flags more cases as late, catching more true late deliveries resulting in higher recall but also producing more false alarms causing low precision. The best validation F1 of 0.748 occurred at a threshold of 0.39, with precision of 0.676 and recall of 0.838. This performance was better than the standard 0.5 threshold. Figure 1 shows the trade-off across the threshold range vs the score.

Figure 1 Validation Precision, Recall, F1 score across different precision thresholds



The table 2 depicts the final test set performance comparison between the baseline RF model, the fine tuned RF model, the fine tuned RF model with a calibrated 0.39 threshold.

Table 2 Final Test performance by model config

Model	**Threshold**	**Accuracy**	**Precision**	**Recall**	**F1**	**ROC-AUC**	**PR-AUC**
Corrected Baseline RF	0.50	0.7315	0.8261	0.6461	0.7251	0.8138	0.8620
Tuned RF at 0.50 (default)	0.50	0.7337	0.8456	0.6290	0.7214	0.8258	0.8730
Tuned RF at 0.39 (calibrated)	0.39	0.7124	0.6944	0.8490	0.7639	0.8258	0.8730

At default threshold, tuning produced a mixed result: F1 dipped slightly from 0.7251 to 0.7214, but ROC-AUC increased from 0.8138 to 0.8258 and PR-AUC improved from 0.8620 to 0.8730. This means the tuned model ranks positive cases better overall even the F1 was marginally lower. The real improvement is from the threshold calibration: When we moved the tuned model decision threshold from 0.50 to 0.39, this improved the recall from 0.6290 to 0.8490 and F1 from 0.7214 to 0.7639, but the cost of precision dropped from 0.8456 to 0.6944.

Discussion

In our point of view, even though the F1 gain over the baseline doesnt look significantly high we still feel the calibrated tuned model is the best suited for our case. From an operations risk context, missing a late delivery is generally more costly than flagging an on time delivery as "at risk". That is, a false negative is expensive than a false positive. So it's better to double check an order that is totally fine, rather than letting a risky order go unnoticed. The calibrated model

cut false negatives from 6994 to 2847, eventhough the false positives increased from 2166 to 7045. Table 3 shows the confusion matrices of each final model that we evaluated.

The practical effect of the tradeoff depends on CEROP's stakeholder tolerance. If reviewing a flagged order is cheap, recall oriented calibration is the best approach and if each flag triggers a costly manual process then this may not be the best practical model. Another thing to note is we used RandomizedSearchCV instead of the exhaustive GridSearchCV because the parameter space: $4 \times 4 \times 4 \times 3 \times 3 = 576$ combinations would have made a full grid search more expensive for the gain we obtained.

Table 3 Confusion matrix of all the models

Model	Threshold	TN	FP	FN	TP
Corrected Baseline RF	0.50	12975	2564	6672	12182
Tuned RF at 0.50 (default)	0.50	13373	2166	6994	11860
Tuned RF at 0.39 (calibrated)	0.39	8494	7045	2847	16007

Cross Layer Synthesis

CEROP produces two independent risk signals this week, and they are not equally ready to hand to a business stakeholder. The Operations side improved in a way a supply chain manager can act on right now. By tuning plus threshold calibration pushed the model's F1 from 0.725 to 0.764, mostly by catching more late deliveries. False negatives dropped from 6,994 to 2,847 out of 34,393 orders. That's about 4,100 additional at-risk shipments now getting flagged for review instead of just showing up late with no warning.

The cost is more false alarms 2,166 jumped to 7,045 but in a shipping context a false alarm just means a review that turns out fine. A missed flag means an unhappy customer and

zero chance to fix anything beforehand. That's the whole reason to recommend the calibrated, lower-threshold model over the raw tuned one at its default cutoff.

The Financial layer tells a different story, and it's just as useful for being a "no." Tuning didn't improve bankruptcy detection. The Week 5 configuration still catches more failing firms with fewer false alarms than anything RandomizedSearchCV turned up. For a lender or procurement team deciding which vendors need closer scrutiny. Current model is already close to the best this data and budget can produce. What would help is sourcing more bankruptcy examples the whole search was hamstrung by having only 141 failing firms to learn from.

Combined, CEROP's two halves are at different points of business readiness. Operations is ready to deploy with a concrete, quantified improvement. Fewer missed late deliveries, a known and acceptable increase in manual reviews. Financial is ready to deploy as is, with a clear recommendation that the next investment should go toward data, which more labeled bankruptcies or additional financial signals, rather than more search time on the same 5,455 firms.

Neither layer can be merged into a single score a firm's balance sheet and a shipment's delivery risk answer different business questions for different stakeholders. However, a combined CEROP report can present both signals side by side. Such as financial risk as a stable model to monitor as-is and next to delivery risk as a newly calibrated model that will surface more flags with review capacity planned for it.

Conclusion

Week 6 fine tuned both shortlisted models under a shared protocol and the two landed in different places. Hyperparameter search improved the Operations layer's ranking ability and combined with threshold calibration, its practical F1 as well. Run the same search strategy on the Financial layer, though, and it optimized noise instead of signal. A result traced to the 30:1 class imbalance and a positive class count too small for the search to resolve reliably. Both outcomes are reported as found rather than adjusted to look more better. Which has been the standard since the accuracy trap discussion in Week 2.

For Week 7, the path forward is to finalize deployment specifications for both layers. The Financial layer at its Week 5 configuration, the Operations layer at its Week 6 tuned and calibrated configuration to define how the two independent scored risk signals are surfaced together in a single CEROP report, since they can't be merged into one model. Their units of analysis are just too different.

References

Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. Journal of Machine Learning Research, 13, 281–305.

Breiman, L. (2001). Random forests. Machine Learning, 45(1), 5–32. https://doi.org/10.1023/A:1010933404324

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 785–794. https://doi.org/10.1145/2939672.2939785

Davis, J., & Goadrich, M. (2006). The relationship between precision-recall and ROC curves. Proceedings of the 23rd International Conference on Machine Learning, 233–240. https://doi.org/10.1145/1143844.1143874

Efron, B., & Tibshirani, R. J. (1993). An introduction to the bootstrap. Chapman & Hall.

Géron, A. (2022). Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems (3rd ed.). O'Reilly Media.

Liao, J. G., & Chin, K.-V. (2007). Logistic regression for disease classification using microarray data: Model selection in a large p and small n case. Bioinformatics, 23(15), 1945–1951. https://doi.org/10.1093/bioinformatics/btm270

Liaw, A., & Wiener, M. (2002). Classification and regression by randomForest. R News, 2(3), 18–22.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12, 2825–2830.

Saito, T., & Rehmsmeier, M. (2015). The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. PLOS ONE, 10(3), Article e0118432. https://doi.org/10.1371/journal.pone.0118432

Varma, S., & Simon, R. (2006). Bias in error estimation when using cross-validation for model selection. BMC Bioinformatics, 7, Article 91. https://doi.org/10.1186/1471-2105-7-91

Wolpert, D. H. (1992). Stacked generalization. Neural Networks, 5(2), 241–259. https://doi.org/10.1016/S0893-6080(05)80023-1