

LEARNED REVIEW POLICIES VERSUS SCORE THRESHOLDS FOR BUDGETED FRAUD ALERT ALLOCATION**Ayush Ojha**

University of Cambridge, Cambridge, United Kingdom

ayushojha010@gmail.comORCID <https://orcid.org/0009-0001-9404-4930>**ABSTRACT**

We compare Double DQN, score thresholds and dynamic programs for budgeted fraud review on a chronological credit-card benchmark. Five refits of the selected zero-cost model find 47–64 frauds using 252–528 reviews. Validation utility falls by up to 38.5 after its selected checkpoint. At budget four, mean DQN utility is below the validation-tuned threshold at every tested cost. At budget two, results depend on the threshold grid. Changing the state inputs yields no clear improvement across five seeds. These exploratory findings show unstable DQN training; stronger claims require untouched data.

Keywords:

Financial fraud; Review budgets; Double deep Q-learning; Score thresholds; Sequential allocation.

I. INTRODUCTION

Fraud detection covers payment abuse, insurance claims and irregular records, each with different evidence and decisions. Statistical detection identifies candidates for examination; an unusual transaction is not, in itself, verified fraud. The present study uses recorded credit-card fraud labels and asks how an alert queue should be screened when reviews are scarce. It does not measure human investigation outcomes or the amount of money recovered [1], [2].

Card transactions combine severe class imbalance with changing behavior. Practitioner studies show why aggregate accuracy is inadequate, and distinguish immediate investigator feedback from supervision that arrives later. Streaming systems must also handle nonstationarity and feedback latency. These concerns motivate chronological evaluation and a capacity constraint, rather than treating an unlimited ranking of transactions as an operational policy [3], [4], [5].

A capacity limit and an investigation cost express different objectives. Minimum-risk fraud detection can combine the cost of examining an alert with an amount-dependent loss from missed fraud, while cost-sensitive learning requires an explicit accounting of the consequences of decisions. Here one review consumes one unit of capacity, and the optional review penalty is measured in units of one discovered fraud. It is not a monetary estimate [6], [7]. Related portfolio work evaluates transaction-cost objectives and downside risk against equal-weight rebalancing [8]. Both applications require decision costs to be stated explicitly, although their allocation tasks differ.

Deep Q-learning learns an action-value function from replayed transitions, and a reinforcement-learning policy has also been used for stream-based annotation selection. These mechanisms make sequential budget allocation plausible, but their success in games or language annotation does not establish fraud-detection effectiveness [9], [10].

We compare a quota-conditioned Double DQN with a fixed score threshold and a finite-horizon dynamic program. Each rule makes decisions from the current score and remaining capacity, without seeing future scores or labels. The evaluation separates the utility of a selected operating point from allocation quality at matched validation workload. Forward scoring, validation-selected optimization settings, low review budgets and prevalence controls test whether learned allocation improves on explicit online rules. On this benchmark, simple allocation rules match or exceed DQN, with comparisons at tighter budgets depending on the threshold grid. DQN training is unstable across seeds, and changing the state inputs does not establish a reliable improvement. These findings concern the tested procedure; stronger claims require untouched evaluation data.

II. LITERATURE SURVEY

Fraud scoring and decision costs have long been studied together. Example-dependent decision trees incorporate costs into learning, and ensembles can target financial savings rather than uniform classification accuracy [11], [12]. Random forests provide a standard nonlinear scorer, while regularized tree boosting offers another relevant tabular method; we evaluate the former and make no experimental claim about XGBoost [13], [14]. Transaction-history features can capture spending patterns, but require real card or account links and only past information. The anonymized benchmark used here supplies neither persistent identities nor a usable customer history, so those features cannot be reconstructed [15]. Drift-aware ensembles address changes over time, whereas a two-day benchmark can only provide a short chronological test of a fixed model [16]. Probability estimation is equally important: balanced sampling changes the training prior, undersampling can alter posteriors, and calibrated probabilities are useful when a decision depends on a cost [17], [18]. Modern neural networks can also be miscalibrated [19]. Accordingly, score learning,

held-out probability mapping and review allocation are separate components in our comparison. Better ranking alone does not determine how often a policy should investigate. Nor does an affine-sigmoid calibration map guarantee that its values remain calibrated after prevalence changes.

Active learning studies requests for labels, while budgeted naive-Bayes learning studies purchases of training features. These provide acquisition analogies, but their outcomes concern classifier learning rather than finding transaction fraud [20], [21]. The scorer in this study remains fixed during evaluation; reviewed labels do not improve its later predictions.

Q-learning updates action values using immediate reward and the best estimated continuation value. Its tabular convergence assumptions do not transfer automatically to neural approximation on a small, drifting dataset [22]. Published DQN combines a target network and replay with bounded temporal-difference gradients; Double DQN separates next-action selection from target evaluation to reduce an overestimation mechanism [23], [24]. We use those two mechanisms with a small feed-forward network, uniform replay and an undiscounted finite episode. Dueling value-advantage streams and prioritized replay are established alternatives, but neither is implemented here, and their reported performance supplies no evidence that this fraud policy would improve [25], [26]. The central question is whether learned state dependence improves the discovery-workload frontier over explicit thresholds, rather than whether a larger architecture can fit training returns. Observational deployment would require further safeguards: off-policy value evaluation and high-confidence policy improvement depend on logged trajectories and statistical assumptions absent from this static labeled benchmark [27], [28]. We therefore do not interpret replayed label outcomes as an off-policy guarantee or a field intervention.

III. METHODOLOGY

A. Data and chronological evaluation

The benchmark contains 284,807 transactions and 492 recorded frauds from approximately two days in September 2013. It originates from the ULB Machine Learning Group and Worldline. We use OpenML creditcard record 1597, version 1, released in June 2015 [17]. The data record is available at <https://www.openml.org/d/1597>. The supplement gives the input checksum, acquisition instructions and duplicate counts.

Rows are stably sorted by elapsed Time; timestamp groups remain intact at the approximately 60/10/10/20% train/calibration/validation/test boundaries. Time is an offset from the first transaction, not a verified local clock. The full span is 47.998 hours and test covers 7.651 hours near the end of the record; an evening or a specific market session cannot be inferred. Table 1 reports the partition sizes and fraud prevalence. Features are V1–V28, $\log(1+\text{Amount})$, and sine/cosine of elapsed Time with a 86,400-second period. The periodic features represent relative daily phase, not verified time of day. Training spans only 33.443 hours, limiting evidence for repeat daily patterns. Standardization uses only the fitting rows, with standardized features clipped to $[-10, 10]$.

Partition	Rows	Frauds	Prevalence	Alert fraction
Train	170888	360	0.211%	24.3%
Calibration	28480	24	0.0843%	20%
Validation	28478	33	0.116%	33.3%
Test	56961	75	0.132%	29.7%

TABLE 1. CHRONOLOGICAL PARTITIONS AND FINAL-SCORER ALERT FRACTIONS. POLICY TRAINING USES FORWARD SCORES WITHIN TRAINING; FINAL-SCORER FRACTIONS ARE REPORTED HERE TO SEPARATE THOSE PIPELINES.

The dataset contains 1,081 excess exact duplicate numeric rows, including 19 fraud rows. No exact duplicate group crosses a partition boundary. The duplicate-removal control keeps the first chronological copy, retains the original split time boundaries, and uses separately fitted scorer artifacts and newly trained allocation policies. This changes the selected population; its totals are reported separately.

B. Scorer and forward training scores

The neural scorer is a 31–32–16–1 network with ReLU hidden units, fan-in-scaled zero-mean initialization and a logistic loss [29]. Each batch samples 128 fraud and 128 legitimate fitting rows with replacement. Final epoch selection uses the last 10% of the training partition, while the scaler for that tuning fit uses only its earlier 90%. The chosen epoch count is then refitted on the full training partition. Calibration is reserved for a separate affine-sigmoid mapping of neural logits. We use regularized logistic fitting rather than claiming the complete original SVM calibration procedure [30]. The comparison scorers are balanced L2 logistic regression through LIBLINEAR and a 100-tree random forest; both use the same fitting features and separate affine-sigmoid calibration [31].

Allocation training uses four expanding-prefix scoring folds. After a 20% warm-up prefix, each successive 20% training block is scored using a model fitted only on earlier rows. Within that earlier prefix, its first 80% fits the scaler and scorer and its last 20% fits calibration; all target-block timestamps occur later. Ten scorer epochs are fixed for every fold. This preserves the

separation between fitting and held-out scoring while adapting it to chronology [32]. Validation and test labels never enter the forward scores.

Neural optimization uses Adam with $\beta_1=0.9$, $\beta_2=0.999$ and $\epsilon=10^{-8}$. The implementation follows Algorithm 1 of the January 2017 version, applying bias correction to both moments and adding epsilon outside the square root of the corrected second moment [33]. Scientific computation uses Python 3.6.4, NumPy 1.13.3, SciPy 1.0.0 and scikit-learn 0.19.1 [34].

Component	Setting
Neural scorer	31–32–16–1; ReLU; balanced batch 256; Adam learning rate 0.001; epoch candidates 1–10
Forward folds	20% warm-up; four expanding-prefix target blocks; ten epochs; earlier-prefix calibration 20%
Logistic / maps	LIBLINEAR C=1; balanced logistic scorer, max_iter=500; unweighted affine-sigmoid calibration
Random forest	100 trees; depth $\leq$ 12; leaf $\geq$ 5; sqrt feature sampling; balanced_subsample; one worker
Scorer seeds	Final/tuning 31415; forward folds 31416–31419
Queues / costs	H=128; B=1,2,3,4,8,16; $\lambda=0,0.002,0.01,0.05,0.1$
Q-network	5–32–32–2 ReLU; uniform replay 50,000; batch 128; warm-up 2,048; Huber $\delta=1$; gradient norm $\leq$ 10
Search learning rates	0.00025, 0.0005, 0.001, 0.002
Search target copies	Every 125, 250 or 500 optimizer updates
Search paired profiles	120/90, 240/180 or 480/360 training rounds / exploration-decay rounds
Rollouts / exploration	32 sampled episodes per round; ϵ starts 0.5, falls linearly to 0.05; discount 1
Selection / seeds	B=4 validation every ten rounds; search seeds 97,101,103; independent refits 11,23,37,51,71
Inference / intervals	Greedy actions; skip ties; paired circular blocks of five and ten queues; 10,000 replicates

TABLE 2. SCORER AND POLICY SETTINGS. FOR EACH COST, 36 CONFIGURATIONS ARE EVALUATED ON THREE SEEDS AND ONE IS SELECTED BY MEAN VALIDATION UTILITY AT B=4.

C. Alert queues and review objectives

The final neural scorer defines alerts with its held-out calibration 80th-percentile probability, 0.000374. This retains 20.0% of calibration rows, 33.3% of validation rows and 29.7% of test rows. Separately, each forward-scoring pipeline contributes training alerts above the pooled forward-score 80th percentile (0.0001931); a numerical cutoff is not transferred between the forward and final scorers. This equates the training selection rank rather than their probability scales. Alerts are grouped chronologically into complete blocks of 128; incomplete tails are counted but not evaluated. Training has 213 queues and 237 frauds, validation 74 and 31, and test 132 and 74. The test tail has 15 alerts and 0 frauds; one of the 75 test frauds lies below the alert cutoff. Median test queue duration is 182 seconds.

At each arrival, skip leaves capacity unchanged and review consumes one unit and reveals the recorded fraud label in the simulated reward. Skipped labels are unavailable to the policy but remain available to the evaluator. The scorer does not update. Quotas reset at the next complete queue and unused capacity expires. Each reviewed fraud gives unit reward; each review incurs penalty λ :

$$r_t = a_t (y_t - \lambda) \quad (1)$$

The primary endpoint is total utility $U = F - \lambda R$, where F is frauds found and R is reviews. The same objective selects thresholds, checkpoints and optimization settings. Review penalties 0, 0.002, 0.01, 0.05 and 0.1 are measured in units of one detected fraud; they do not estimate wages or recovered money. Discovery and workload are secondary outcomes. A label-informed ceiling, sum over queues of $\min(\text{frauds in queue}, B)$, shows what is reachable with complete knowledge and is excluded from policy comparisons.

D. Neural state and validation selection

Only forward-scored training alerts determine the mean μ and population standard deviation σ of the logit. The score input is:

$$q = \frac{\text{logit}(p) - \mu}{5\sigma} \quad (2)$$

Probabilities are restricted to $[10^{-12}, 1-10^{-12}]$ for finite logarithms; q itself is not clipped. With m alerts remaining including the current one, k reviews remaining, initial budget B and queue length H, the state is:

$$s = (q, \frac{m}{H}, \frac{k}{H}, \frac{k}{\max(m, 1)}, \frac{B}{H}) \quad (3)$$

The network has two action outputs; review is inadmissible at $k=0$. Double DQN separates next-action selection by the online network from evaluation by the target network [24].

$$y = r + Q_{\bar{\theta}}(s', \underset{a' \in A(s')}{\operatorname{argmax}} Q_{\theta}(s', a')). \quad (4)$$

Continuation is zero at queue end or after quota exhaustion. Training uses uniform replay, Huber loss with unit transition point, gradient-norm limit 10 and discount factor 1. Norm clipping follows the established control of large gradients [35]. Greedy evaluation reviews only when the review value strictly exceeds the skip value; ties skip.

For each cost, four learning rates, three target-copy intervals and three paired training/exploration profiles give 36 configurations. Every configuration is fitted with search seeds 97, 101 and 103. Each fit selects the checkpoint with highest $B=4$ validation utility, then fewer reviews and the earliest checkpoint. Configurations are ranked by mean selected utility across those three seeds, then mean reviews, mean selected round and configuration order. The selected configuration is independently refitted with seeds 11, 23, 37, 51 and 71 using the same checkpoint rule. Budgets other than four do not influence selection. A fixed maximum of 120, 240 or 480 rounds and validation every ten rounds were specified before this rerun; there is no adaptive extension or test-based stopping. The paired profiles change both training and exploration duration, so their effects are not separated. Selection at the last checkpoint does not establish convergence. Architecture and state remain fixed. Tables A9 and A13 and Figure S1 retain settings, endpoint diagnostics and complete refit curves.

The prevalence control samples training queue j with weight proportional to $\exp(-\alpha f_j)$, where f_j is its training fraud count. A bisection-selected $\alpha=0.4081$ makes the weighted mean 0.4189 frauds per queue, equal to validation rather than test, from the unweighted training mean 1.113. Effective episode sample size is 186.7 of 213. The selected full-state optimization settings are reused; five seeds are fitted without another configuration search. This changes sampled training prevalence while preserving within-queue chronology and recorded rewards.

E. Threshold and dynamic-programming comparators

Sequential stochastic assignment and stochastic knapsack compare immediate acceptance reward with the continuation value of preserving capacity [36], [37]. We estimate the arrival-score distribution from validation alerts and solve the equal-size, finite-horizon recursion under independent score arrivals. Calibrated probability p is the reward proxy:

$$V(m, k) = E[\max(V(m-1, k), p - \lambda + V(m-1, k-1))] \quad (5)$$

$$V(0, k) = V(m, 0) = 0 \quad (6)$$

A review occurs only when p strictly exceeds the reservation threshold:

$$\tau(m, k) = \lambda + V(m-1, k) - V(m-1, k-1) \quad (7)$$

This solves the specified independent-score model, not the realized transaction stream. A validation-recalibrated variant fits an L2 affine-sigmoid map to the validation alert logits and labels, then uses mapped probabilities both for the arrival distribution and for later review decisions. It is a validation-fitted control; its validation fit is not an independent calibration test. The fixed threshold reviews arrivals with score at least its cutoff until the quota is exhausted. The primary utility-selected and workload-matched thresholds share the dense candidate grid: every distinct validation alert score, zero and one. For each cost and budget, validation utility selects the cutoff, breaking ties by fewer reviews and then a higher cutoff. The workload comparator instead minimizes the difference from each DQN seed's validation review count, with the same fewer-review and higher-cutoff tie rules. Test workload can differ. The earlier primary grid, retained as the coarse-grid sensitivity, uses 0 and 1 plus calibration-alert percentiles 0, 25, 50, 60, 70, 75, 80, 85, 90, 92, 95, 97, 98, 99, 99.5, 99.8 and 100. The denser grid improves operating-point resolution but uses only 31 validation frauds and can overfit their realized ordering; both grids are reported.

F. Uncertainty and controls

Precision-recall and ROC curves use distinct score thresholds, grouping ties without arbitrary tie breaking; average precision uses the same tied groups. These measures complement utility under severe class imbalance [38], [39]. Paired uncertainty estimates resample consecutive test queues jointly across policies. Ordinary bootstrap motivates resampling, while block resampling retains local order [40], [41], [42]. The primary block length of five queues was fixed before this rerun to retain short local runs while permitting about 26 blocks per resample; ten-queue blocks provide a longer-dependence sensitivity analysis. Neither is claimed optimal. There are 10,000 replicates per length. Policy seeds are averaged before resampling;

training and model selection are not resampled. Tables A10–A12 report all budget and block-length comparisons; elapsed block spans follow Table A6.

State controls remove k/m, m/H or B/H separately, or retain only score, m/H and k/H, with unchanged network dimensions. They reuse the selected zero-cost full-state optimization settings and all five reporting seeds, without a separate search. Cost-workload increases, capacity-discovery decreases and cross-cost rescoring are reported as optimization diagnostics; none selects a replacement policy using test outcomes. Duplicate and independently retrained horizon controls appear in Tables A4–A5.

IV. RESULTS AND DISCUSSION

A. Scorer quality and queue dependence

Scorer	AP	ROC area	Brier score
Neural	0.7383	0.9773	0.0004667
Logistic	0.7657	0.9823	0.0004604
Forest	0.8099	0.9281	0.0004599

TABLE 3. CALIBRATED SCORER METRICS OVER ALL 56,961 HELD-OUT TEST TRANSACTIONS. ALLOCATION OUTCOMES USE THE SELECTED ALERT QUEUES.

The forest and logistic scorer exceed neural average precision (0.8099 and 0.7657 versus 0.7383). The forest nonetheless has lower global ROC area, 0.9281: three frauds have raw score zero tied with 4,405 legitimate transactions. Grouped-threshold recomputation reproduces all reported average precisions; the logistic curve has no tied block at its top score. The top neural transaction is legitimate, giving precision zero at its first real threshold. Figure 1 uses unsmoothed threshold steps rather than interpolation across a tied group.

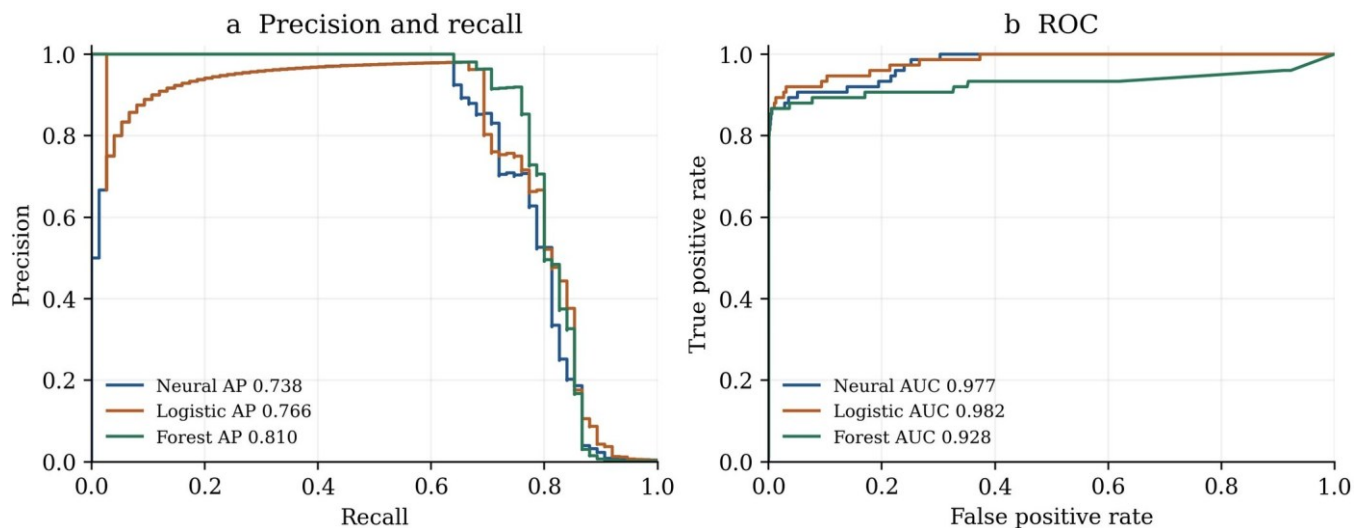


Figure 1. Test scorer curves. Precision–recall steps group equal scores consistently with average precision. The artificial (recall 0, precision 1) endpoint is retained; the first real neural threshold has precision 0. All test rows contribute to scorer metrics.

Fraud counts are overdispersed relative to a Poisson count reference. Training variance/mean is 5.1, with 117 zero-fraud queues versus 70 expected at the observed mean. Although its 21 queues containing at least three frauds are close to the 21.8 expected count, those queues contain 134 frauds, showing a severe tail. Test variance/mean is 1.37, with 6 queues containing at least three frauds versus 2.56 expected. Adjacent episode mean-score correlations are 0.628 in training and 0.124 in test, while test fraud-count correlation is 0.0918. These measured concentrations and dependence make independent arrivals an imperfect approximation. Label-count overdispersion alone does not establish score dependence, and these measurements do not identify its cause. Table A6 gives all counts.

B. Utility as the primary comparison

B	λ	Dense threshold	Coarse threshold	Score DP	Recalibrated DP	DQN
2	0.0	45.0	53.0	54.0	54.0	46.0
2	0.002	44.9	52.8	53.6	53.7	46.6
2	0.01	44.5	52.1	52.3	52.8	44.9
2	0.05	42.4	48.4	48.0	48.0	44.6
2	0.1	39.8	43.7	44.3	44.6	42.1
3	0.0	60.0	58.0	59.0	61.0	53.4
3	0.002	59.8	57.8	57.3	59.5	53.4
3	0.01	58.8	57.0	55.4	57.2	49.9
3	0.05	46.2	52.9	52.0	53.0	47.8
3	0.1	43.4	47.8	47.8	48.6	45.6
4	0.0	61.0	59.0	61.0	62.0	57.2
4	0.002	60.8	58.8	59.1	61.4	57.4
4	0.01	59.8	58.0	56.5	59.7	52.6
4	0.05	55.1	53.8	53.4	53.7	49.2
4	0.1	47.2	48.6	49.4	49.4	46.0

TABLE 4. TOTAL UTILITY F-AR AT B=2, 3 AND 4. DENSE THRESHOLDS AND DQN CHECKPOINTS USE VALIDATION UTILITY; DQN IS THE FIVE-SEED MEAN. THE COARSE GRID IS A SENSITIVITY COMPARATOR. DISPLAY USES ONE DECIMAL; RANKING USES UNROUNDED VALUES.

At B=4, DQN utility is below the dense threshold at all five costs and below all three primary comparators at every cost. At zero cost it finds 57.2 frauds with 342.6 reviews, versus 61/118 for the dense threshold. The recalibrated DP has the largest or tied-largest observed utility among the four primary methods at three of five costs (0.0, 0.002, 0.1). These are descriptive rankings, not simultaneous superiority claims. The dense and coarse thresholds differ because candidate resolution changes the validation-selected operating point.

B	Dense threshold F/R	Score DP F/R	Recalibrated DP F/R	DQN F/R	Oracle F
1	34/40	40/132	40/132	31.2/117.4	50
2	45/52	54/264	54/264	46/208.4	66
3	60/116	59/396	61/396	53.4/279.8	72
4	61/118	61/528	62/528	57.2/342.6	74
8	63/254	62/1056	63/1056	60/655	74
16	63/254	65/2112	65/2112	60.2/1642	74

TABLE 5. ZERO-COST OUTCOMES ON 132 TEST QUEUES CONTAINING 74 FRAUDS. F/R DENOTES FRAUDS FOUND/REVIEWS; DQN IS THE SEED MEAN. ORACLE IS THE LABEL-INFORMED CAPACITY CEILING.

At B=4 the label-informed ceiling reaches all 74 available frauds; this budget does not force a choice between recorded frauds. At B=2 and B=3 the ceilings are 66 and 72, so Table 4 and the paired comparisons in Tables 6 and 7 also report those capacity-limited settings. B=4 remains the prespecified selection endpoint for this rerun; tighter-budget results are secondary.

B	λ	DQN – dense	Recalibrated DP – dense
2	0.0	1.0 [-7.4, 9.0]	9.0 [2.0, 16.0]
2	0.002	1.7 [-5.7, 9.1]	8.8 [2.8, 15.8]
2	0.01	0.4 [-6.2, 7.2]	8.3 [3.3, 14.4]
2	0.05	2.2 [-1.4, 6.6]	5.6 [0.7, 11.6]
2	0.1	2.3 [-1.3, 6.8]	4.8 [0.0, 10.6]
3	0.0	-6.6 [-12.8, -1.0]	1.0 [-4.0, 6.0]
3	0.002	-6.4 [-12.6, -1.0]	-0.2 [-5.2, 3.8]
3	0.01	-8.9 [-13.7, -4.6]	-1.6 [-5.7, 1.5]
3	0.05	1.6 [-2.5, 6.7]	6.8 [1.2, 13.7]
3	0.1	2.2 [-1.8, 7.1]	5.2 [-0.1, 11.8]
4	0.0	-3.8 [-8.6, 0.6]	1.0 [-4.0, 6.0]
4	0.002	-3.4 [-8.8, 1.2]	0.6 [-4.4, 5.6]
4	0.01	-7.2 [-11.8, -3.2]	-0.1 [-1.2, 2.0]
4	0.05	-6.0 [-10.7, -1.9]	-1.4 [-4.4, 0.7]
4	0.1	-1.2 [-4.4, 2.1]	2.2 [-0.7, 6.7]

TABLE 6. UTILITY DIFFERENCES AGAINST THE DENSE THRESHOLD GRID AT B=2, 3 AND 4, WITH PAIRED FIVE-QUEUE CIRCULAR-BOOTSTRAP 95% INTERVALS. DQN USES FIXED SEED MEANS. DENSE IS THE REVISED PRIMARY GRID; COARSE IS THE EARLIER PRIMARY GRID. VALUES USE ONE DECIMAL; INTERVAL MEMBERSHIP IS ASSESSED BEFORE ROUNDING.

B	λ	DQN – coarse	Recalibrated DP – coarse
2	0.0	-7.0 [-12.6, -1.8]	1.0 [-4.0, 6.0]
2	0.002	-6.2 [-11.0, -1.6]	0.9 [-4.1, 5.9]
2	0.01	-7.2 [-11.1, -3.4]	0.8 [-3.3, 4.8]
2	0.05	-3.8 [-8.7, 0.3]	-0.4 [-3.6, 2.8]
2	0.1	-1.6 [-6.7, 2.5]	0.9 [-2.3, 4.1]
3	0.0	-4.6 [-11.0, 1.0]	3.0 [-2.0, 8.0]
3	0.002	-4.4 [-10.6, 1.2]	1.8 [-3.3, 6.7]
3	0.01	-7.1 [-11.6, -3.0]	0.2 [-4.7, 4.4]
3	0.05	-5.1 [-10.0, -1.5]	0.1 [-0.1, 0.3]
3	0.1	-2.2 [-7.2, 1.5]	0.8 [-1.2, 2.1]
4	0.0	-1.8 [-7.0, 3.0]	3.0 [-2.0, 8.0]
4	0.002	-1.4 [-7.2, 3.8]	2.6 [-2.5, 7.6]
4	0.01	-5.4 [-9.5, -1.2]	1.8 [-1.2, 4.9]
4	0.05	-4.7 [-9.0, -1.3]	-0.1 [-0.3, 0.0]
4	0.1	-2.6 [-8.0, 1.4]	0.8 [-1.2, 2.1]

TABLE 7. UTILITY DIFFERENCES AGAINST THE COARSE THRESHOLD GRID AT B=2, 3 AND 4, WITH PAIRED FIVE-QUEUE CIRCULAR-BOOTSTRAP 95% INTERVALS. DQN USES FIXED SEED MEANS. DENSE IS THE REVISED PRIMARY GRID; COARSE IS THE EARLIER PRIMARY GRID. VALUES USE ONE DECIMAL; INTERVAL MEMBERSHIP IS ASSESSED BEFORE ROUNDING.

The B=2 interpretation depends on threshold resolution. Relative to the dense grid, recalibrated-DP utility differences range from 4.8 to 9.0; against the earlier primary grid they range from -0.4 to 1.0. DQN likewise changes from positive differences against the dense grid to negative differences against the earlier grid. The dense grid has more validation candidates but poorer test utility in these cases. With only 31 validation frauds, this pattern is consistent with selection overfitting, although this single test stream cannot isolate its cause. The revised primary grid remains fixed; selecting a preferred grid from these test results would add further bias.

At zero cost and B=4, the DQN-minus-dense-threshold fraud difference is -3.8 with interval [-8.6, 0.6], and the review difference is 224.6 with interval [207.2, 242.0]. Intervals spanning zero do not support a directional claim. Tables A10–A12 give both block lengths at all budgets, including comparisons with each DP and the coarse threshold. These intervals condition on the fitted policies and are not adjusted for multiple comparisons; an isolated interval excluding zero does not carry the conclusion.

C. Operating points and matched validation workload

The workload-constrained threshold is a different comparator from the freely utility-selected threshold. At B=4 and zero cost, it finds 48.8 frauds with 359.4 test reviews versus selected DQN 57.2/342.6. At $\lambda=0.1$ the comparison is 55/79.8 versus 54.2/81.6. DQN has greater utility than its workload-constrained threshold in three of five settings at B=4. Matching is performed separately for every seed on validation review count, not on test workload; therefore this is a comparison of allocation patterns under a validation work constraint, not an exact test-work match or a replacement for Table 4.

At budget four and review cost 0.002, the recalibrated DP finds more frauds with fewer reviews than the DQN seed mean. At the other costs, neither dynamic program improves both components simultaneously. Utility ranking and two-component dominance are distinguished because a higher-discovery policy can spend more reviews. Validation recalibration tests the effect

of allowing the model-based rule to use validation labels; it does not establish reliable probability calibration on later institutions or days.

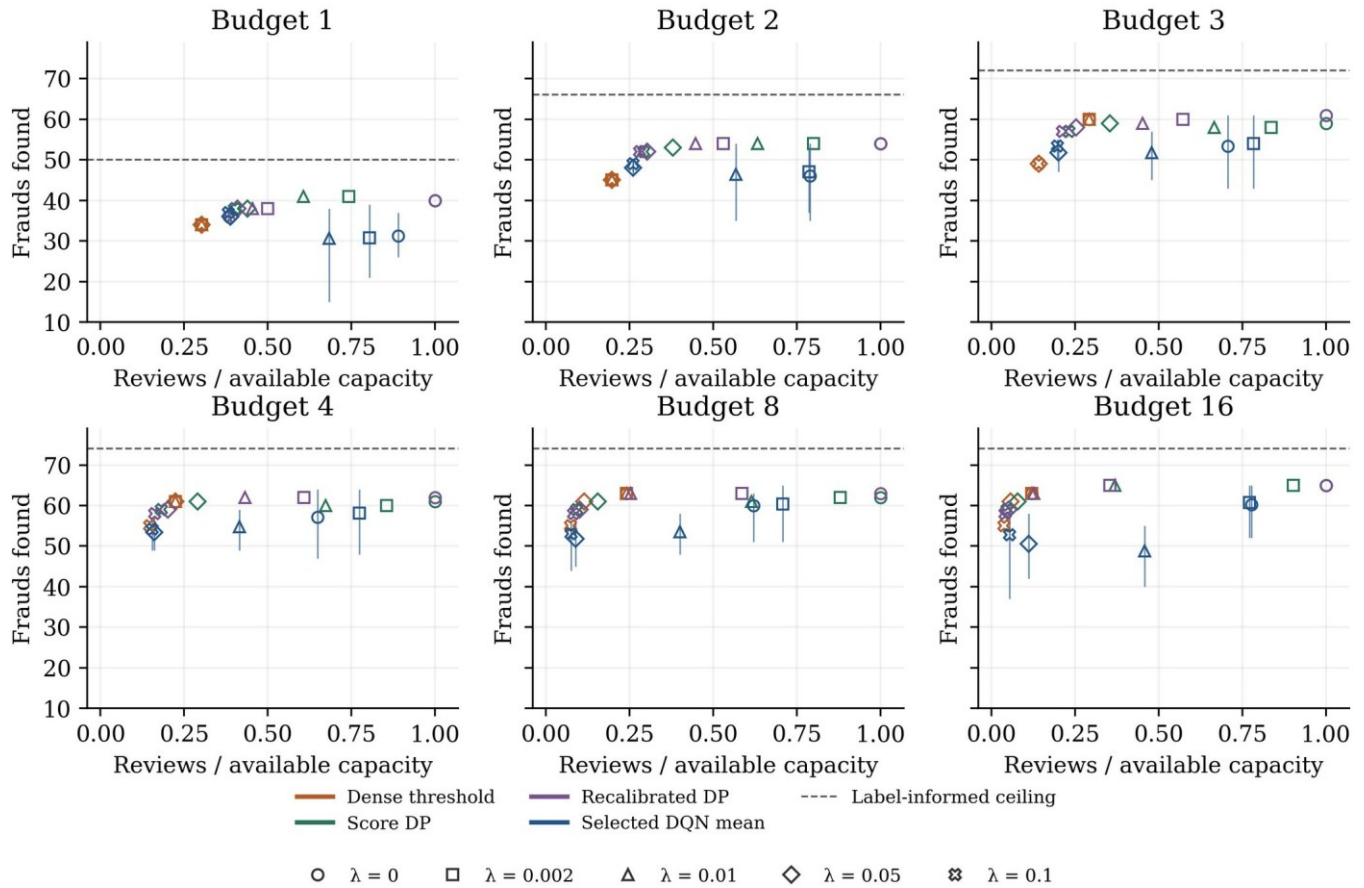


Figure 2. Discovery versus fraction of review capacity at six budgets. Color identifies the method and marker shape identifies λ for every method; coincident operating points overlap. DQN bars show seed ranges. Dashed lines mark label-informed oracle ceilings. The common vertical axis starts at 10, below the smallest seed outcome of 15, so every seed-range endpoint is visible; horizontal ticks are shown in every panel.

D. Optimization and component controls

Policy / state control	Mean frauds	SD frauds	Mean reviews	SD reviews
Selected full state	57.2	7.7	342.6	120.8
Prevalence-matched training	56.4	8.4	469.8	89.1
Remove k/m	56.6	6.8	370.2	136.6
Remove m/H	57.6	5.1	420.8	150.0
Remove B/H	59.0	3.8	239.8	105.3
Retain score, m/H, k/H	58.2	5.7	358.8	156.6
First B arrivals	4.0	0.0	528.0	0.0

TABLE 8. B=4 ZERO-COST CONTROLS USING THE SELECTED FULL-STATE OPTIMIZATION CONFIGURATION. CONTROLS ARE NOT SEPARATELY SEARCHED. SD IS THE SAMPLE STANDARD DEVIATION ACROSS FIVE REFIT SEEDS. FIRST B IS DETERMINISTIC; ITS ZERO SD IS NOT A FIVE-SEED ESTIMATE.

At B=4 and zero cost, prevalence-matched training gives 56.4/469.8 versus full-state 57.2/342.6; retaining only score, m/H and k/H gives 58.2/358.8, and removing B/H gives 59/239.8. The full-state runs find between 47 and 64 frauds across five seeds. Table 8 shows large variation within the controls as well. With only five refits and no independently powered comparison of state effects, these results do not distinguish a reliable state-design improvement. They do not establish equivalence either. A control with a better observed mean cannot be selected from this test stream as the preferred representation.

The search selects the 120-round profile at zero cost and the 480-round profile at the four positive costs. None of the twenty-five reporting refits selects its final checkpoint. The largest fall from selected-checkpoint to final-checkpoint validation utility is 38.5. Training duration and exploration decay change together across these profiles, so the longer profile may benefit from slower exploration rather than additional optimization alone. The experiment does not separate these effects or establish convergence. Figure S1 shows all reporting trajectories, and Table A13 gives each endpoint comparison.

The saved diagnostics flag three adjacent-cost increases in mean test workload and six adjacent-budget decreases in mean test discovery across the six budgets and five costs. At B=4 and evaluation cost 0.05, the policy trained at cost 0.1 has utility 50.1 versus 49.2 for the policy trained at 0.05. These are checks of optimization consistency, not permission to select a replacement from the test set. Independent approximate fits need not be monotone; observed reversals qualify any general claim about the learned policy. Table A14 identifies the mean reversals, and Table A15 reports fixed-policy cross-cost rescoring.

For horizon sensitivity, policies are retrained for H=64, 256 and 512 with the selected zero-cost settings and validation-selected checkpoints. Baseline operating points are reselected on each corresponding validation queue. Thus the supplement compares refitted policies rather than mixing transfer with reselection. Alternative logistic- and forest-defined queues use their own held-out calibration cutoffs; their populations differ and their results are not paired gains on identical frauds. Duplicate-removal, horizon and complete seed results are provided in Tables A1–A5.

E. Interpretation and limits

The test stream was reused across successive analysis versions, including changes to queue definitions, selection and threshold resolution. Those earlier runs disclose development history; they are not additional findings or independent replications. This evaluation is exploratory despite excluding test labels from fitting and checkpoint selection. Reusing validation across 36 configurations per cost and many threshold candidates risks selection overfitting, particularly with only 31 validation frauds [43]. The analysis is now frozen. Stronger comparisons of grids, states or learned policies require untouched data. Circular blocks preserve local order, but stationarity and coverage for a changing stream are not established. Intervals omit training, calibration and selection uncertainty and provide no multiplicity-controlled claim.

The benchmark covers two days and a 7.651-hour test interval. It has no persistent account identifiers, observed investigation times, delayed-label process or actual staffing constraint. The reward simulates access to recorded labels, not an investigator or intervention. Training, validation and test fraud rates differ; matching the validation mean does not recreate the later joint score-label distribution. The initial probability map uses only 24 calibration frauds; validation recalibration is another fitted approximation. Independent-score DP reservation values may be misspecified under clustered or changing arrivals, but the diagnostics do not establish clustering as the cause of any policy ranking. No monetary savings, institutional generalization or deployment-safety conclusion follows.

V. CONCLUSION

On this short chronological benchmark, simple allocation rules match or exceed DQN depending on the budget, cost and threshold grid. At budget four, mean DQN utility is below the revised primary threshold at all five costs. At budget two, comparison signs depend on the threshold grid, so the apparent recalibrated-DP advantage against the dense grid is not robust to the earlier primary grid. At budget four, recalibrated DP has higher observed utility than both threshold grids at three of five costs, but all paired utility intervals include zero and do not establish superiority. The main neural finding is instability: five zero-cost refits span seventeen discovered frauds, and a reporting run loses 38.5 validation utility after its selected checkpoint. The selected long profiles also confound training duration with exploration decay. State-control differences remain unresolved with five seeds. This frozen analysis supports a negative result about reliability under the tested procedure; it does not establish that all learned allocation methods fail. Stronger claims require a prespecified study on untouched data.

DECLARATIONS

Funding: none declared. Competing interests: none declared by the author. Ethics: the study analyzes a public anonymized transaction benchmark without participant recruitment, intervention or identification attempts.

Data availability: OpenML creditcard record 1597, version 1, <https://www.openml.org/d/1597>. Supplementary Archive S2 (fraud-allocation-reproducibility.zip), supplied with the manuscript, contains the fixed protocol, source, checkpoints, decisions, full results and checks. Its README gives executable commands and a file map. Supplement S1 contains Tables A1–A15 and Figure S1.

REFERENCES

- 1) [1] C. Phua, V. Lee, K. Smith, and R. Gayler, "A Comprehensive Survey of Data Mining-Based Fraud Detection Research," arXiv:1009.6119v1, 2010. <https://arxiv.org/abs/1009.6119v1>
- 2) [2] R. J. Bolton and D. J. Hand, "Statistical Fraud Detection: A Review," Statistical Science, vol. 17, no. 3, pp. 235–255, 2002. <https://doi.org/10.1214/ss/1042727940>
- 3) [3] A. Dal Pozzolo, O. Caelen, Y.-A. Le Borgne, S. Waterschoot, and G. Bontempi, "Learned Lessons in Credit Card Fraud Detection from a Practitioner Perspective," Expert Systems with Applications, vol. 41, no. 10, pp. 4915–4928, 2014. <https://doi.org/10.1016/j.eswa.2014.02.026>
- 4) [4] A. Dal Pozzolo, G. Boracchi, O. Caelen, C. Alippi, and G. Bontempi, "Credit Card Fraud Detection and Concept-Drift Adaptation with Delayed Supervised Information," in 2015 International Joint Conference on Neural Networks (IJCNN), 2015, pp. 1–8. <https://doi.org/10.1109/IJCNN.2015.7280527>
- 5) [5] F. Carcillo, A. Dal Pozzolo, Y.-A. Le Borgne, O. Caelen, Y. Mazzer, and G. Bontempi, "SCARFF: A Scalable Framework for Streaming Credit Card Fraud Detection with Spark," arXiv:1709.08920v1, 2017. <https://arxiv.org/abs/1709.08920v1>
- 6) [6] A. Correa Bahnsen, A. Stojanovic, D. Aouada, and B. Ottersten, "Cost Sensitive Credit Card Fraud Detection Using Bayes Minimum Risk," in 2013 12th International Conference on Machine Learning and Applications, 2013, pp. 333–338. <https://doi.org/10.1109/ICMLA.2013.68>
- 7) [7] C. Elkan, "The Foundations of Cost-Sensitive Learning," in Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, 2001, pp. 973–978. <https://cseweb.ucsd.edu/~elkan/rescale.pdf>
- 8) [8] A. Ojha, "Transaction Costs and Downside Risk in Deep Reinforcement Learning for Portfolio Allocation," International Journal of Engineering Technology Research and Management, vol. 1, no. 9, pp. 19–31, 2017. <https://doi.org/10.5281/zenodo.23212855>
- 9) [9] V. Mnih et al., "Playing Atari with Deep Reinforcement Learning," arXiv:1312.5602v1, 2013. <https://arxiv.org/abs/1312.5602v1>
- 10) [10] M. Fang, Y. Li, and T. Cohn, "Learning How to Active Learn: A Deep Reinforcement Learning Approach," in Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, 2017, pp. 595–605. <https://doi.org/10.18653/v1/D17-1063>
- 11) [11] A. Correa Bahnsen, D. Aouada, and B. Ottersten, "Example-Dependent Cost-Sensitive Decision Trees," Expert Systems with Applications, vol. 42, no. 19, pp. 6609–6619, 2015. <https://doi.org/10.1016/j.eswa.2015.04.042>
- 12) [12] A. Correa Bahnsen, D. Aouada, and B. Ottersten, "Ensemble of Example-Dependent Cost-Sensitive Decision Trees," arXiv:1505.04637v1, 2015. <https://arxiv.org/abs/1505.04637v1>
- 13) [13] L. Breiman, "Random Forests," Machine Learning, vol. 45, no. 1, pp. 5–32, 2001. <https://doi.org/10.1023/A:1010933404324>
- 14) [14] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 785–794. <https://doi.org/10.1145/2939672.2939785>
- 15) [15] A. Correa Bahnsen, A. Stojanovic, D. Aouada, and B. Ottersten, "Feature Engineering Strategies for Credit Card Fraud Detection," Expert Systems with Applications, vol. 51, pp. 134–142, 2016. <https://doi.org/10.1016/j.eswa.2015.12.030>
- 16) [16] J. Z. Kolter and M. A. Maloof, "Dynamic Weighted Majority: An Ensemble Method for Drifting Concepts," Journal of Machine Learning Research, vol. 8, pp. 2755–2790, 2007. <https://jmlr.org/papers/v8/kolter07a.html>
- 17) [17] A. Dal Pozzolo, O. Caelen, R. A. Johnson, and G. Bontempi, "Calibrating Probability with Undersampling for Unbalanced Classification," in 2015 IEEE Symposium Series on Computational Intelligence, 2015, pp. 159–166. <https://doi.org/10.1109/SSCI.2015.33>
- 18) [18] A. Correa Bahnsen, A. Stojanovic, D. Aouada, and B. Ottersten, "Improving Credit Card Fraud Detection with Calibrated Probabilities," in Proceedings of the 2014 SIAM International Conference on Data Mining, 2014, pp. 677–685. <https://doi.org/10.1137/1.9781611973440.78>
- 19) [19] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On Calibration of Modern Neural Networks," in Proceedings of the 34th International Conference on Machine Learning, vol. 70, 2017, pp. 1321–1330. <https://proceedings.mlr.press/v70/guo17a.html>
- 20) [20] B. Settles, "Active Learning Literature Survey," Department of Computer Sciences, University of Wisconsin–Madison, Technical Report 1648, 2009. https://burrsettles.com/pub/settles.activelearning_20090109.pdf
- 21) [21] D. J. Lizotte, O. Madani, and R. Greiner, "Budgeted Learning of Naive-Bayes Classifiers," in Proceedings of the 19th Conference on Uncertainty in Artificial Intelligence, 2003, pp. 378–385. <https://arxiv.org/abs/1212.2472v1>
- 22) [22] C. J. C. H. Watkins and P. Dayan, "Q-Learning," Machine Learning, vol. 8, nos. 3–4, pp. 279–292, 1992. <https://doi.org/10.1007/BF00992698>
- 23) [23] V. Mnih et al., "Human-Level Control Through Deep Reinforcement Learning," Nature, vol. 518, no. 7540, pp.

- 529–533, 2015. <https://doi.org/10.1038/nature14236>
- 24) [24] H. van Hasselt, A. Guez, and D. Silver, “Deep Reinforcement Learning with Double Q-Learning,” in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 30, no. 1, 2016, pp. 2094–2100. <https://doi.org/10.1609/aaai.v30i1.10295>
- 25) [25] Z. Wang, T. Schaul, M. Hessel, H. van Hasselt, M. Lanctot, and N. de Freitas, “Dueling Network Architectures for Deep Reinforcement Learning,” in Proceedings of the 33rd International Conference on Machine Learning, vol. 48, 2016, pp. 1995–2003. <https://proceedings.mlr.press/v48/wangf16.html>
- 26) [26] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, “Prioritized Experience Replay,” in International Conference on Learning Representations, 2016. <https://arxiv.org/abs/1511.05952v4>
- 27) [27] N. Jiang and L. Li, “Doubly Robust Off-Policy Value Evaluation for Reinforcement Learning,” in Proceedings of the 33rd International Conference on Machine Learning, vol. 48, 2016, pp. 652–661. <https://proceedings.mlr.press/v48/jiang16.html>
- 28) [28] P. Thomas, G. Theodorou, and M. Ghavamzadeh, “High Confidence Policy Improvement,” in Proceedings of the 32nd International Conference on Machine Learning, vol. 37, 2015, pp. 2380–2388. <https://proceedings.mlr.press/v37/thomas15.html>
- 29) [29] Y. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, “Efficient BackProp,” in Neural Networks: Tricks of the Trade, Lecture Notes in Computer Science, vol. 1524, pp. 9–50. Springer, Berlin, Heidelberg, 1998. https://doi.org/10.1007/3-540-49430-8_2
- 30) [30] J. C. Platt, “Probabilities for Support Vector Machines,” in Advances in Large-Margin Classifiers, A. J. Smola, P. Bartlett, B. Schölkopf, and D. Schuurmans, Eds., pp. 61–74. MIT Press, 2000. <https://doi.org/10.7551/mitpress/1113.003.0008>
- 31) [31] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, “LIBLINEAR: A Library for Large Linear Classification,” Journal of Machine Learning Research, vol. 9, pp. 1871–1874, 2008. <https://www.jmlr.org/papers/v9/fan08a.html>
- 32) [32] R. Kohavi, “A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection,” in Proceedings of the 14th International Joint Conference on Artificial Intelligence, 1995, pp. 1137–1143. <https://www.ijcai.org/Proceedings/95-2/Papers/016.pdf>
- 33) [33] D. P. Kingma and J. L. Ba, “Adam: A Method for Stochastic Optimization,” in International Conference on Learning Representations, 2015. <https://arxiv.org/abs/1412.6980v9>
- 34) [34] F. Pedregosa et al., “Scikit-Learn: Machine Learning in Python,” Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011. <https://jmlr.org/papers/v12/pedregosa11a.html>
- 35) [35] R. Pascanu, T. Mikolov, and Y. Bengio, “On the Difficulty of Training Recurrent Neural Networks,” in Proceedings of the 30th International Conference on Machine Learning, vol. 28, no. 3, 2013, pp. 1310–1318. <https://proceedings.mlr.press/v28/pascanu13.html>
- 36) [36] C. Derman, G. J. Lieberman, and S. M. Ross, “A Sequential Stochastic Assignment Problem,” Management Science, vol. 18, no. 7, pp. 349–355, 1972. <https://doi.org/10.1287/mnsc.18.7.349>
- 37) [37] J. D. Papastavrou, S. Rajagopalan, and A. J. Kleywegt, “The Dynamic and Stochastic Knapsack Problem with Deadlines,” Management Science, vol. 42, no. 12, pp. 1706–1718, 1996. <https://doi.org/10.1287/mnsc.42.12.1706>
- 38) [38] J. Davis and M. Goadrich, “The Relationship Between Precision-Recall and ROC Curves,” in Proceedings of the 23rd International Conference on Machine Learning (ICML ’06), 2006, pp. 233–240. <https://doi.org/10.1145/1143844.1143874>
- 39) [39] T. Saito and M. Rehmsmeier, “The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets,” PLOS ONE, vol. 10, no. 3, article e0118432, 2015. <https://doi.org/10.1371/journal.pone.0118432>
- 40) [40] B. Efron, “Bootstrap Methods: Another Look at the Jackknife,” The Annals of Statistics, vol. 7, no. 1, pp. 1–26, 1979. <https://doi.org/10.1214/aos/1176344552>
- 41) [41] D. N. Politis and J. P. Romano, “A Circular Block-Resampling Procedure for Stationary Data,” in Exploring the Limits of Bootstrap, R. LePage and L. Billard, Eds., pp. 263–270. John Wiley & Sons, New York, 1992. <https://mathweb.ucsd.edu/~politis/DPpublication.html>
- 42) [42] H. R. Künsch, “The Jackknife and the Bootstrap for General Stationary Observations,” The Annals of Statistics, vol. 17, no. 3, pp. 1217–1241, 1989. <https://doi.org/10.1214/aos/1176347265>
- 43) [43] G. C. Cawley and N. L. C. Talbot, “On Over-Fitting in Model Selection and Subsequent Selection Bias in Performance Evaluation,” Journal of Machine Learning Research, vol. 11, pp. 2079–2107, 2010. <https://jmlr.org/papers/v11/cawley10a.html>