p-adic Prefix-Consensus Ensembles for Branch-Point Prediction in Donker's Athanasius Apostolos Data

Gregory D. Baker

June 12, 2026

Abstract

This chapter gives a methodology for using exact p-adic linear models in a high-dimensional, low-sample textual-variant setting. Donker's derived Athanasius Apostolos matrices are used as a deliberately limited test case: they are small, already interpreted, and unsuitable for historical reconstruction without renewed collation. The supervised target is the parent branch point of the held-out witness, $y(w) = \text{parent}(\text{leaf}(w))$, rather than the unseen held-out leaf. Branch paths are encoded as base-5 integers and evaluated by p-adic loss, so errors within a nearby part of the assumed tree are penalised less than errors across a distant branch. The experiment compares dummy prediction, opaque-class logistic regression, path-wise logistic regression, nested random forests, and exact through-points p-adic linear ensembles aggregated by a branch-risk medoid. A non-nested exploratory replay can tune greedy p-adic ensembles to very low leave-one-out losses, but nested leave-one-out selection removes that advantage. Nested logistic has the lowest binary loss, nested random forest has the lowest multistate loss, and the p-adic ensembles use many fewer active parameters while yielding interpretable feature-usage diagnostics. The result is therefore methodological and diagnostic, not stemmatic, source-reconstructive, or authorship-related.

1 Motivation and contribution

Before printing presses, photocopiers, and the internet, documents were copied by hand. Scribes sometimes believed that the text in front of them should be preserved exactly. At other times they believed that correction or updating was appropriate. They might correct what they took to be a grammatical error. They might alter a word to reflect a political, religious, or stylistic judgement. They also made ordinary copying mistakes, and they often copied from older documents whose damaged surfaces could leave a lacuna, or gap, in the exemplar.

A scribe a generation later, copying from that document, could preserve many of the previous generation's mistakes and changes while introducing some new ones. One way to study a document's provenance is therefore documentary genealogy: compare patterns of difference, ask which changes are shared, and infer which witnesses may be close to one another. Shared changes may indicate shared ancestry, and distinctive combinations of readings may place a witness near one part of a documentary tree rather than another.

There are several complications. Some scribes had access to multiple exemplars and could incorporate readings from two different parts of the tree. A later scribe could correct a previous error back towards an earlier form. Two scribes could also independently make the same change. These cases mean that a variant state is not automatically a clean genealogical marker.

Fortunately, multiple changes per document were not uncommon. Some changes can therefore be treated as noise while others carry signal about the document's position in an assumed tree. This makes textual-variant data an interesting candidate for machine-learning approaches, provided the models are kept cautious and interpretable.

The available datasets are often minuscule. Very few documents survive from antiquity, and the witnesses available for a specific textual problem may number in the tens or fewer.

Training efficiency is usually a low priority, because a source dataset may change only when new collation work appears. Exhaustive search can still fail: once the number of features is larger than the number of data points, a brute-force exact linear search over all feature subsets has too many cases. This suggests methods that can learn from small numbers of documents, expose the reasoning behind each classification, and respect the tree structure implied by copying relationships.

Since each surviving document has a parent exemplar, even if that parent has been lost, and those parent relationships continue back towards an original document, the natural prediction geometry is hierarchical. This chapter studies that problem with p-adic branch labels and an ultrametric p-adic loss. The tree used here is a modelling scaffold derived inside the project from Donker's data products and associated distance matrices. The "ground truth" is at best an operational target that relies heavily on Donker's analysis and on the project-local tree-building choices. It is not treated as a proven historical stemma, and none of the following results should be read as source reconstruction or as evidence about Athanasian authorship.

The first version of the experiment tried to predict the exact held-out leaf under leave-one-out cross-validation. That target is structurally mis-specified. If `Ath` is held out, a classifier trained on the other eleven rows has never seen the class label `Ath`. The corrected target is the parent branch point:

$$y(w) = \text{parent}(\text{leaf}(w)).$$

If a witness leaf has path $d_1.d_2.\cdots.d_k$, the supervised target is $d_1.d_2.\cdots.d_{k-1}$.

The contribution is a coherent branch-point prediction workflow. Duplicate feature columns are collapsed; branch paths are encoded as base-5 integers; models are evaluated by p-adic loss; exact through-points affine certificates are sampled from random feature subsets; and candidate hyperplanes are aggregated by a p-adic medoid that acts as prefix consensus over their raw predictions. The empirical result is a small case study showing both the attraction and the danger of this setup: greedy through-points ensembles can be tuned to strong apparent leave-one-out performance, but a nested redo shows that this was optimistic. What remains useful is the sparse p-adic certificate machinery and the ability to ask which textual features repeatedly enter the selected local explanations.

2 Textual and data background

Gerald J. Donker's work on the text of the Apostolos in Athanasius produced derived matrices for Athanasius and comparison witnesses [1]. The files used here include binary, multistate, and dissimilarity encodings. They are not fresh manuscript transcriptions. The present project therefore starts from Donker's already selected and encoded variation units, and the analysis cannot say whether an observed feature-state is the best philological representation of a passage.

Scribal variation is heterogeneous. A reading may be a shared innovation copied by descendants, a correction towards another known form, a harmonisation, an accidental omission, a substitution, a transposition, or the result of mixture between exemplars. Standard introductions to New Testament textual criticism stress both accidental and intentional sources of variation [2, 3]. Royse's study of early Greek New Testament papyri, especially through singular readings, is a useful warning against treating every difference as equally genealogical [4]. Head likewise uses singular readings in early papyri of John to study copyist behaviour rather than only manuscript grouping [5].

For the present method, the central distinction is between propagated branch markers and recurrent states. A propagated state is the ideal case for tree prediction: a change occurs once and descendants inherit it. A recurrent state can appear in disconnected parts of the tree because the same omission is easy, because a familiar phrase invites harmonisation, because scribes correct towards a known reading, or because a witness is mixed. Epp's variant-conscious approach and the Coherence-Based Genealogical Method both make the same broad point: witnesses and

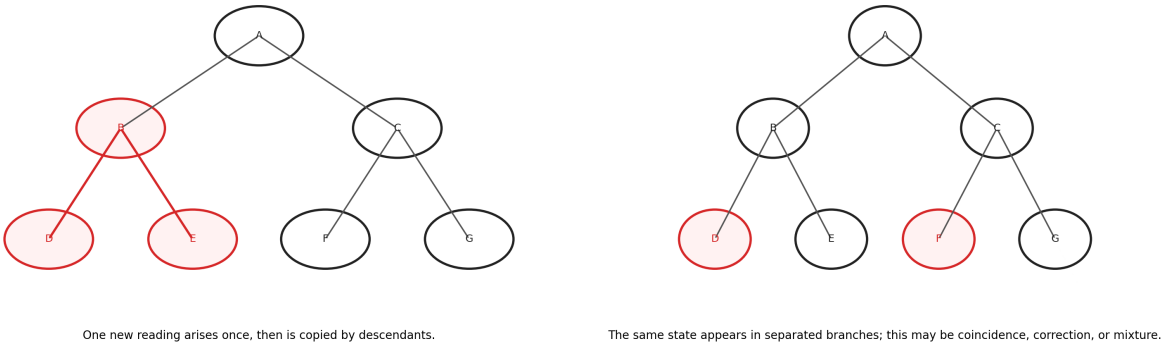

Figure 1: Left: a change that arises once and is propagated. Right: the same observed state appears in disconnected parts of the tree. Only the first case is a straightforward branch marker.

readings need local treatment, and manuscript relationships are often more complex than a single clean tree [6, 8, 7].

This chapter does not resolve those philological ambiguities. It asks a narrower modelling question: if the derived feature flags are accepted as input, can a p-adic method predict the parent branch point of a held-out witness in an assumed tree?

3 Data, target tree, and p-adic encoding

Athanasius of Alexandria was a fourth-century Christian author whose works quote the Apostolos. The question here is not whether those works are Athanasian, nor whether the project can recover a complete manuscript history. The narrower question is whether Donker's derived variation-unit matrices contain enough signal to place a held-out witness near the correct branch point of an assumed tree. The analysis is limited to passages where Athanasius quotes the biblical text and where Donker's derived comparison data are available.

The case study has twelve witness labels: `Ath`, `P46`, `U1`, `A`, `B`, `C`, `D`, `F`, `G`, `M223`, `M1739`, and `M2423`. There are only twelve because this is the present Donker-derived Athanasius Apostolos design matrix, not a fresh collation of every possible witness. Reconstructing genealogical proximity matters because a nearby branch is a less severe historical error than a branch on the other side of the root, even when both are wrong as flat labels.

Exact duplicate feature columns are collapsed before modelling. The retained design matrices have 36 binary features and 30 multistate features. Table 1 gives the binary training table so that the object being modelled is visible in the chapter; the corresponding CSV outputs are written under `outputs/tables/` with the `branchpoint_training_table` prefix.

Witness	Leaf path	Branch	Branch path	Target	Rom.8.22.1	Rom.8.22.2	Rom.9.5.1.b1	Rom.15.19.1.b2	Rom.15.19.1.b1	1Cor.1.24.1	1Cor.2.4.1.b3	1Cor.2.4.1.b2	1Cor.2.4.1.b1	1Cor.4.6.3.b3	1Cor.4.6.3.b2	1Cor.4.6.3.b1	1Cor.9.16.1	1Cor.9.16.2	1Cor.9.22.1.b2	1Cor.11.2.1	1Cor.11.2.2.b2	1Cor.15.47.1.b2	1Cor.15.53.1.b1	1Cor15.54.1.b2	1Cor15.54.1.b1	1Cor.15.55.1.b2	1Cor.15.55.1.b1	2Cor.1.10.1.b2	2Cor.1.10.1.b1	Eph.3.18.1	Eph.3.19.2	Phil.2.5.1	Phil.2.5.2	Phil.2.6.1	Phil.2.9.1	Col.1.16.1.b2	Col.1.16.1.b1	Col.1.16.2	Col.1.18.2	Col.1.18.3
Ath	1.2	N7	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P46	1.1.1	N6	1.1	6	1	0	1	0	1	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	1	1	0	1	1	1
U1	1.1.4.2	N3	1.1.4	106	0	0	0	0	1	1	0	1	1	0	0	1	0	0	0	0	0	0	0	0	1	0	1	0	0	1	0	0	1	0	1	1	0	1	0	1
A	1.1.2	N6	1.1	6	0	0	0	0	1	1	0	1	0	0	0	1	0	0	0	0	0	1	1	1	0	1	0	0	1	1	0	0	1	0	1	0	1	0	0	0
B	1.1.3.1	N1	1.1.3	81	1	0	0	0	1	1	0	1	1	0	0	1	0	1	0	0	0	0	1	1	1	0	1	0	0	0	1	0	1	0	0	0	1	0	1	0
C	1.1.3.2	N1	1.1.3	81	0	0	0	0	1	1	0	1	0	0	0	1	0	1	1	0	1	0	1	1	1	0	1	0	0	0	0	0	1	0	1	0	0	0	0	0
D	2.1	N5	2	2	0	0	0	1	0	1	0	1	1	0	1	1	0	1	1	1	1	0	1	1	1	1	1	0	1	0	0	0	1	1	0	0	1	0	0	0	0
F	2.2.1	N2	2.2	12	1	1	0	1	0	1	0	0	1	1	0	0	0	1	1	1	1	0	0	0	0	1	1	1	0	0	0	1	1	1	0	1	0	0	0	0
G	2.2.2	N2	2.2	12	1	1	0	1	0	1	0	0	1	1	0	0	0	1	1	1	1	0	1	0	0	1	1	1	0	0	0	1	1	1	0	1	0	0	0	0
M223	1.3.1	N4	1.3	16	0	0	0	0	1	1	1	0	0	1	0	1	1	0	1	1	0	1	1	1	1	1	0	1	0	1	0	1	0	0	0	0	1	0	0	0
M1739	1.1.4.1	N3	1.1.4	106	0	0	0	0	1	1	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0	1	1	0	1	0	1	1	0	1	1	0	1	1	0
M2423	1.3.2	N4	1.3	16	0	0	0	0	1	1	0	1	0	1	1	0	1	0	1	1	0	1	1	1	1	1	0	1	0	1	0	1	0	0	0	0	1	0	0	0

Table 1: Collapsed binary branch-point design matrix. The target is the parent branch point of the witness leaf, encoded in base 5; the passage-flag columns are exact-duplicate representatives used by the model.

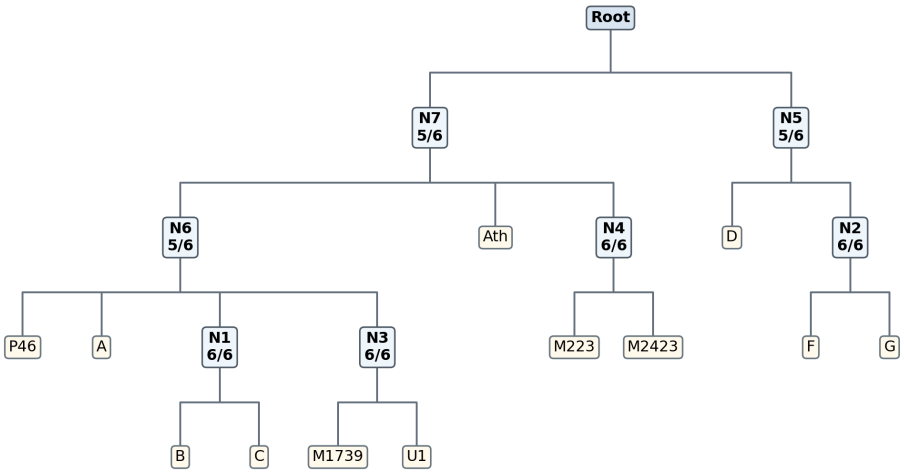

Figure 2: Consensus tree used as the modelling scaffold. Branch-point prediction uses the parent internal node of each witness leaf as the target.

The consensus tree used for targets is:

$$(((P46, A, (B, C), (M1739, U1)), Ath, (M223, M2423)), (D, (F, G))).$$

Figure 2 shows the same structure. The internal node labels are project-local identifiers used for prediction and reporting.

Paths are encoded in base 5. For a branch path $d_1.d_2.\cdots.d_k$, define

$$\mathrm{enc}(d_1, \ldots, d_k) = \sum_{r=1}^{k} d_r 5^{r-1}.$$

The base is 5 because the assumed tree has no node with more than five children. The p-adic loss between two encoded branch labels a and b is

$$L_5(a, b) = |a - b|_5 = 5^{-v_5(a-b)}$$

with $L_5(a, a) = 0$. A wrong first digit has loss 1. A correct first digit but wrong second digit has loss $1/5$. A correct first two digits but wrong third digit has loss $1/25$.

4 Medoid through-points ensembles

The p-adic models use exact through-points affine certificates. For a feature subset of size s, an affine model has $s+1$ coefficients. A candidate certificate is fitted through $s+1$ observed witness points where the corresponding affine design matrix is non-singular. The procedure samples many random feature subsets, enumerates the available through-points hyperplanes for each subset, and keeps the candidate with the lowest training p-adic loss for the held-out fold. This is a discrete exact search over anchored certificates rather than gradient-based optimisation.

The ensemble aggregates raw rational predictions from selected hyperplanes. The aggregation is a branch-risk medoid: each possible branch label is scored by its average bounded p-adic distance to the raw predictions, and the lowest-risk branch is selected. The resulting model is a prefix-consensus procedure over predictions, not an arithmetic average of coefficients.

This setup is aimed at the case where there are more feature columns than data points and where there is no obvious regularisation term intrinsic to the p-adic exact search. A full brute-force search over all feature subsets would grow exponentially in the number of retained readings.

Target branch	Path	Witnesses whose parent is this branch
N7	1	Ath
N6	1.1	A, P46
N1	1.1.3	B, C
N3	1.1.4	M1739, U1
N4	1.3	M223, M2423
N5	2	D
N2	2.2	F, G

Table 2: Corrected leave-one-out branch-point targets. The full machine-readable target table is written to `outputs/tables/branchpoint_targets.csv`.

Random feature subsets and greedy ensemble selection are therefore not just computational conveniences; they are the mechanism that makes exact through-points fitting usable in this small-n, larger-p setting.

4.1 The p-adic medoid as prefix consensus

Proposition 1. *Let C be a finite subset of $\mathbb{Z}_p$ of candidate branch labels and let $z_1, \dots, z_m \in \mathbb{Z}_p$ be raw predictions. Define*

$$\hat{c} = \arg\min_{c \in C} \sum_{j=1}^{m} |c - z_j|_p.$$

Writing $v_j(c) = v_p(c - z_j)$, with $p^{-\infty} = 0$, this is equivalent to maximising

$$\sum_{r \geq 1} p^{-(r-1)} \#\{j : z_j \equiv c \pmod{p^r}\}.$$

For finite-depth integral branch codes, the same equivalence holds with the usual exact-match terminal convention by treating full-code equality as $v_j(c) = \infty$.

Proof. For a fixed c and z_j, the congruence $z_j \equiv c \pmod{p^r}$ holds exactly when $r \leq v_j(c)$. Therefore

$$\sum_{r \geq 1} p^{-(r-1)} \mathbf{1}\{z_j \equiv c \pmod{p^r}\} = \sum_{r=1}^{v_j(c)} p^{-(r-1)} = \frac{1 - p^{-v_j(c)}}{1 - p^{-1}},$$

where the final value is interpreted as $(1 - p^{-1})^{-1}$ when $v_j(c) = \infty$. Since $|c - z_j|_p = p^{-v_j(c)}$, summing over j gives

$$\sum_{j=1}^{m} |c - z_j|_p = m - (1 - p^{-1}) \sum_{r \geq 1} p^{-(r-1)} \#\{j : z_j \equiv c \pmod{p^r}\}.$$

The first term is constant in c, so minimising the left-hand side is equivalent to maximising the weighted prefix count. □

The proposition gives the model interpretation. A first-digit agreement contributes the largest term to the score; second-digit agreement adds a smaller term; third-digit agreement adds a smaller term again. A through-points hyperplane supplies a congruence certificate determined by witness points. The ensemble samples many such certificates, and the medoid chooses the candidate branch whose p-adic ball receives the strongest geometrically weighted support.

5 Experimental design

All experiments use leave-one-out cross-validation over the twelve witnesses. The target is the parent branch point from Table 2. The comparison includes dummy, logistic, decision-tree, random-forest, and p-adic model families.

The dummy predictor is the training-fold p-adic medoid of observed branch labels and has one parameter. The opaque logistic baseline treats branch labels such as `N1` and `N6` as unrelated classes. The path-wise logistic baseline predicts the successive path digits and then maps the predicted path back to a branch label. Both logistic variants are run with L1 and L2 regularisation settings; the tables report the best-by-loss settings for each encoding. Decision trees and random forests are also run in opaque and path-wise forms, with a small nested grid over depth, feature subsampling, and leaf-size settings; the forest grid uses 100 trees.

Hierarchical classification and hierarchical loss have a substantial literature, including large-margin hierarchical classification and surveys of hierarchical evaluation measures [9, 10, 11]. Wu, Tygert, and LeCun's work on hierarchical loss is especially relevant to the distinction between using an ultrametric for evaluation and treating it as an ordinary smooth training loss [12]. Recent hierarchy-aware baselines make the same caution concrete: hierarchical cross-entropy and soft labels, CHAMP, and HAF optimise mistake severity through surrogate losses or representation constraints rather than by routing through a tree [13, 14, 15]. Operating-point and lowest-common-ancestor evaluations also show that the reported result can change with the level of prediction specificity and with the chosen hierarchical metric [16, 17]. Recent hierarchical text-classification work likewise stresses that evaluation and inference choices matter, and that strong simple baselines can remain competitive with specialised hierarchy-aware methods [18, 19]. For large e-commerce categorisation, flat models can outperform hierarchical pipelines in some settings [20]; this is a direct warning against treating top-down structure as automatically superior. The closest modern p-adic-native comparison point is not this small exact-linear method but the newer van der Put neural-network family, which uses p-adic balls as model primitives for ultrametric representation learning [29].

The p-adic model is the exact through-points ensemble described above. Two size parameters are kept separate. K is the generated candidate-library size: in the repeated experiment $K = 200$ random feature subsets are generated for each encoding, subset size, repeat, and fold. m is the number of candidate hyperplanes retained in the final ensemble. Candidate generation follows the random-subspace intuition [24]; selection follows ensemble pruning and library selection ideas [21, 25, 26, 23, 22, 27]. The p-adic interpretation is also consistent with the broader motivation for non-Archimedean representations in machine learning [28].

Two ensemble replays are reported. The prefix replay uses the first m candidates in random order and is retained as a diagnostic. Greedy branch-risk selection builds an ensemble by repeatedly adding the candidate that gives the lowest mean leave-one-out p-adic branch loss under the medoid rule, with ties broken by lower active parameter count and then candidate id. For each encoding and method, the repeated curves are averaged over ten seeds. The selected row is the most parsimonious row within one standard error of the best repeated mean loss, ordered by lower active parameter count, smaller m, smaller s, and then lower loss. This replay is useful for understanding the candidate-library geometry, but it uses the same leave-one-out surface for selection and reporting.

To separate selection from evaluation, the branch-point experiment is also rerun with nested leave-one-out selection. Each outer fold holds out one witness for evaluation only. Inside the remaining eleven witnesses, the p-adic procedure selects subset size s, ensemble size m, and greedy candidate order by inner leave-one-out; the selected protocol is then refit on the eleven outer-training witnesses and evaluated on the held-out witness. The nested p-adic run uses $K = 200$, ten random repeats, $s = 1, \ldots, 10$, and a predeclared greedy replay cap of $m \leq 50$. The cap is not binding in the reported run: the largest selected m is 15. The logistic comparator

Dataset	Baseline	Accuracy	Mean p-adic loss	Params used	Active used
Binary	dummy	0.000	0.320	1.0	1.0
Binary	opaque logistic	0.500	0.060	252.8	103.8
Binary	path logistic	0.417	0.050	185.0	68.5
Multistate	dummy	0.000	0.320	1.0	1.0
Multistate	opaque logistic	0.500	0.047	211.8	34.0
Multistate	path logistic	0.500	0.113	170.5	13.7

Table 3: Best branch-point dummy and logistic baselines by mean p-adic loss.

is nested in the same way, selecting model family, penalty, and C inside each outer fold. The decision-tree and random-forest comparators are nested over opaque/path-wise form and tree hyperparameters.

Parameter counts are reported in two ways. "Parameters used" counts the intercept and coefficients of selected models that are available for inference in a fold. "Active parameters" counts coefficients that are non-zero and multiply non-zero feature values for that held-out witness. The active count is the more direct measure of inference sparsity, while the total count is retained to keep comparison with logistic regression auditable.

6 Results

Table 3 gives the non-nested dummy and logistic baselines used in the exploratory replay. The dummy loss is 0.320 in both encodings. The best binary logistic comparator is path-wise L2 logistic regression with mean p-adic loss 0.050 and about 68.5 active parameters. The best multistate logistic comparator is opaque L1 logistic regression with mean loss 0.047 and about 34.0 active parameters.

Table 4 is the non-nested exploratory comparison. The selected binary greedy ensemble uses $s = 9$, $K = 200$, and $m = 4$, with mean loss 0.009, exact branch accuracy 0.867, and about 9.4 active parameters. The selected multistate greedy ensemble uses $s = 4$, $K = 200$, and $m = 4$, with mean loss 0.005, exact branch accuracy 0.933, and about 17.6 active parameters. These rows beat the best non-nested logistic baselines by mean p-adic loss and active parameter count, but they are selected on the same leave-one-out loss that they report.

Dataset	Model	s	K	m	Loss	Active used
Binary	dummy medoid				0.320	1.0
Binary	best path logistic				0.050	68.5
Binary	single-seed endpoint	10	200	200	0.023	82.5
Binary	prefix one-SE	7	200	94	0.038	249.2
Binary	greedy one-SE	9	200	4	0.009	9.4
Multistate	dummy medoid				0.320	1.0
Multistate	best opaque logistic				0.047	34.0
Multistate	single-seed endpoint	9	200	200	0.043	734.4
Multistate	prefix one-SE	5	200	194	0.046	1022.1
Multistate	greedy one-SE	4	200	4	0.005	17.6

Table 4: Compact branch-point model comparison. The single-seed endpoint uses the older $K = 200$ diagnostic; the one-SE rows use repeated randomisation and selected ensemble size m.

Table 5 makes the parameter comparison explicit. The binary greedy p-adic ensemble uses 24.0 mean total parameters at inference, versus 185.0 for the binary path-logistic comparator, and 9.4 active parameters versus 68.5. The multistate greedy p-adic ensemble uses 19.6 mean total parameters, versus 211.8 for the multistate opaque-logistic comparator, and 17.6 active

parameters versus 34.0.

Dataset	p-adic ensemble	Logistic comparator	Metric	p-adic	Logistic	Lower?
Binary	$s = 9, m = 4$	path L2 $C = 0.1$	Loss	0.009	0.050	yes
Binary	$s = 9, m = 4$	path L2 $C = 0.1$	Params used	24.0	185.0	yes
Binary	$s = 9, m = 4$	path L2 $C = 0.1$	Active params	9.4	68.5	yes
Multistate	$s = 4, m = 4$	opaque L1 $C = 10$	Loss	0.005	0.047	yes
Multistate	$s = 4, m = 4$	opaque L1 $C = 10$	Params used	19.6	211.8	yes
Multistate	$s = 4, m = 4$	opaque L1 $C = 10$	Active params	17.6	34.0	yes

Table 5: Direct parsimony comparison between the selected greedy p-adic ensembles and the best-by-loss logistic baseline for the same encoding. Parameter counts are mean parameters used at inference over leave-one-out folds.

The nested redo gives a different answer. Table 6 keeps the outer folds evaluation-only. Under that protocol, nested logistic has lower mean p-adic loss than nested p-adic greedy in both encodings: 0.047 versus 0.223 for binary, and 0.113 versus 0.170 for multistate. The p-adic ensembles remain much sparser at inference, using about 10.5 active parameters in binary and 16.3 in multistate, but the low non-nested losses do not survive honest nested selection.

Dataset	Model	Rows	Accuracy	Loss	Active params
Binary	Nested logistic	12	0.500	0.047	38.8
Binary	Nested p-adic greedy	120	0.417	0.223	10.5
Multistate	Nested logistic	12	0.500	0.113	91.2
Multistate	Nested p-adic greedy	120	0.442	0.170	16.3

Table 6: Nested leave-one-out branch-point comparison. Outer folds are evaluation-only; model selection is performed inside the remaining eleven witnesses. The p-adic rows average ten random candidate-library repeats per outer fold.

Table 7 adds the nested decision-tree and random-forest checks. The result is mixed rather than a simple win for one non-p-adic baseline. In the binary encoding, nested logistic remains slightly better than nested random forest by mean p-adic loss, 0.047 versus 0.050, while the selected decision tree is smaller but worse at 0.160. In the multistate encoding, nested random forest has the lowest mean p-adic loss, 0.047, ahead of nested logistic at 0.113 and the selected decision tree at 0.127. The difference between exact branch accuracy and p-adic loss is possible because the loss rewards near-branch mistakes more than exact-label accuracy does.

Dataset	Model	Exact acc.	Mean p-adic loss
Binary	Nested logistic	0.500	0.047
Binary	Nested random forest	0.417	0.050
Binary	Nested decision tree	0.333	0.160
Multistate	Nested random forest	0.500	0.047
Multistate	Nested logistic	0.500	0.113
Multistate	Nested decision tree	0.500	0.127

Table 7: Nested logistic, decision-tree, and random-forest branch-point prediction performance under the same outer leave-one-out folds. Tree models are selected by inner p-adic loss from opaque and path-wise candidates.

The value $K = 200$ is a candidate-library size, not the selected ensemble size. Table 8 estimates how many random candidates are needed to see at least one useful single-candidate model under two thresholds. For the selected binary setting $s = 9$, a candidate that beats the dummy threshold is observed often enough that $K_{95} = 69$, while a candidate that beats the

logistic threshold is rare enough that $K_{95} = 1996$. For the selected multistate setting $s = 8$, the corresponding estimates are $K_{95} = 14$ against the dummy threshold and $K_{95} = 998$ against the logistic threshold. This supports $K = 200$ as a reasonable exploratory library size for finding useful candidates, while also showing that it is not large enough to guarantee a logistic-beating single candidate in every setting.

Dataset	s	Min loss	π_{dummy}	K_{95} dummy	π_{logistic}	K_{95} logistic
Binary	1	0.120	0.050	59	0.000	–
Binary	2	0.030	0.292	9	0.003	1197
Binary	3	0.033	0.392	7	0.003	998
Binary	4	0.023	0.495	5	0.012	249
Binary	5	0.013	0.501	5	0.005	598
Binary	6	0.020	0.444	6	0.006	460
Binary	7	0.033	0.275	10	0.004	748
Binary	8	0.057	0.139	21	0.000	–
Binary	9	0.013	0.043	69	0.002	1996
Binary	10	0.097	0.009	351	0.000	–
Multistate	1	0.087	0.065	45	0.000	–
Multistate	2	0.030	0.431	6	0.005	544
Multistate	3	0.023	0.530	4	0.007	398
Multistate	4	0.023	0.573	4	0.010	299
Multistate	5	0.003	0.583	4	0.013	221
Multistate	6	0.003	0.510	5	0.008	373
Multistate	7	0.010	0.366	7	0.004	665
Multistate	8	0.010	0.199	14	0.003	998
Multistate	9	0.010	0.084	35	0.004	855
Multistate	10	0.000	0.015	199	0.002	1996

Table 8: Candidate-library sizing diagnostic. π is the observed rate at which a single random subset candidate beats the stated loss threshold across leave-one-out folds. K_{95} is the candidate-library size needed for a 95% chance of seeing at least one such candidate, using $1 - (1 - \pi)^K$. A dash means no such candidate was observed in the present repeated library.

The value m is selected separately. Table 9 shows that the best repeated binary greedy loss occurs at $s = 9, m = 5$, but the one-standard-error rule selects $s = 9, m = 4$. The best multistate greedy loss occurs at $s = 8, m = 14$, but the one-standard-error rule selects the smaller $s = 4, m = 4$ row. This is the basis for the small selected ensembles in the main comparison.

Dataset	Method	Best s	Best m	Best loss	Selected s	Selected m	Selected loss
Binary	greedy	9	5	0.008	9	4	0.009
Binary	prefix	7	94	0.038	7	94	0.038
Multistate	greedy	8	14	0.003	4	4	0.005
Multistate	prefix	5	194	0.046	5	194	0.046

Table 9: Selected ensemble-size diagnostic. The best row is the lowest repeated mean loss for an encoding and method; the selected row is the most parsimonious row within one standard error of that best loss.

Figures 3 and 4 show the repeated loss curves. The selected greedy ensembles are small because most of the gain occurs early in the greedy sequence. The prefix replay needs much larger m and remains less parsimonious.

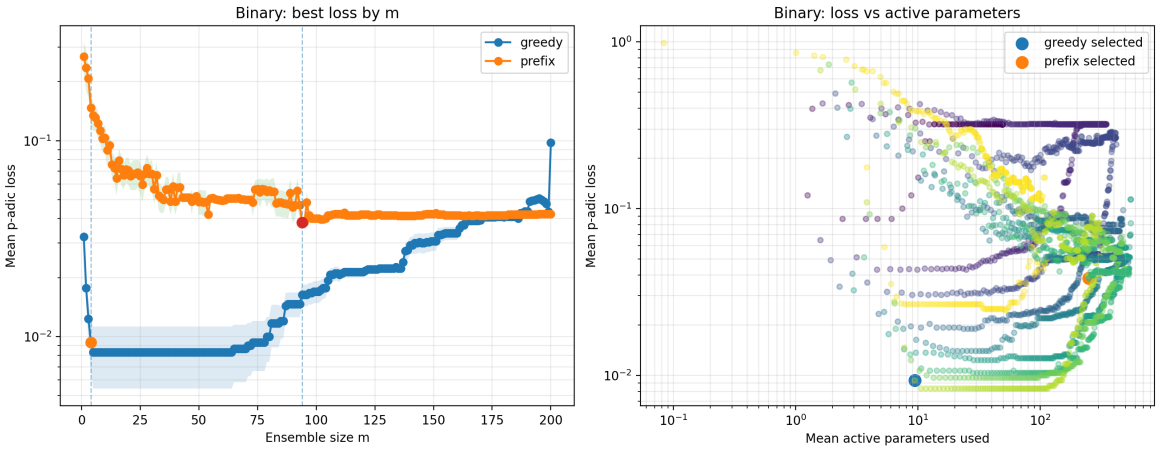

Figure 3: Binary repeated ensemble-selection curves. Ribbons show standard error over ten repeats; markers show the one-standard-error selections.

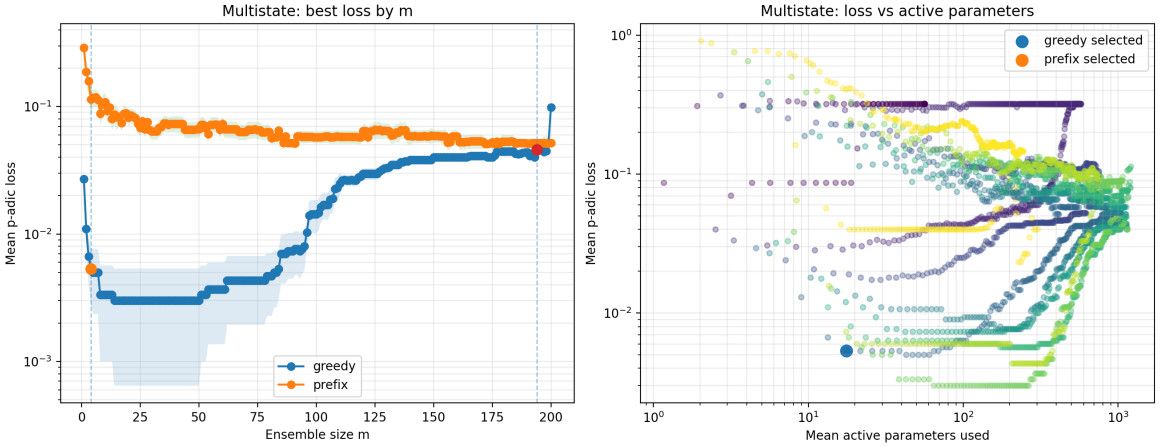

Figure 4: Multistate repeated ensemble-selection curves. Greedy selection reaches its useful range with small m; prefix replay does not.

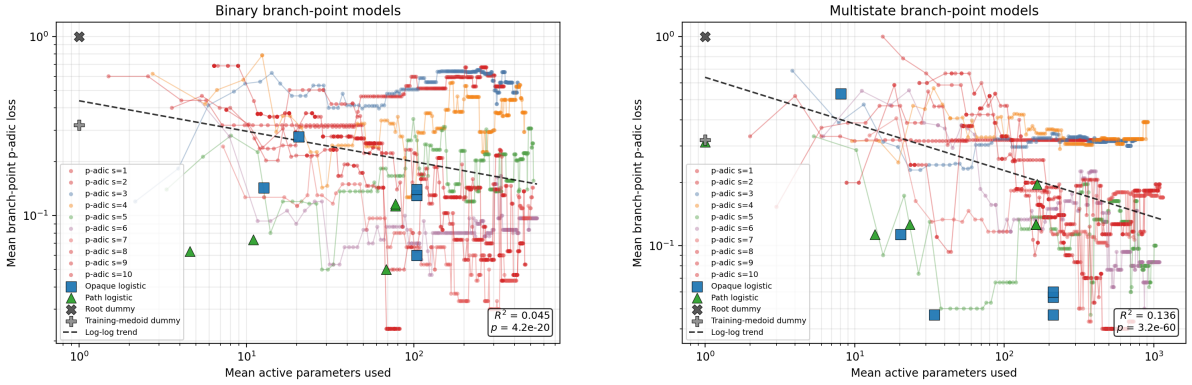

Figure 5: Mean p-adic loss against active parameters used. The plotted regression lines are descriptive diagnostics for the generated model families, not independent hypothesis tests.

Dataset	Loss	L1 margin	L2 margin	Exact margin	Min gap	Min stability
Binary	0.009	2.3	1.9	1.0	0.000	-0.400
Multistate	0.005	3.0	2.5	1.8	0.000	-0.333

Table 10: Prefix-vote and medoid-stability diagnostics for the selected greedy ensembles. Margins are vote counts for the true residue class minus the strongest alternative.

Dataset	Exact	Sibling	Same top	Cross-root	Unavail.
Binary	104	1	15	0	0
Multistate	113	0	7	0	0

Table 11: Selected greedy ensemble branch-error types over ten repeated leave-one-out runs.

7 Prefix, stability, and conditioning diagnostics

Table 10 summarises fold-level diagnostics for the selected greedy ensembles over ten repeated leave-one-out runs. The diagnostics record true and predicted branches, p-adic loss, level-1, level-2, and exact-branch margins, and residue counts modulo 5, 25, and 125.

The margins are mostly positive, but the minimum medoid gap is zero and the sufficient one-step stability margin remains negative. The condition used here is conservative: if the runner-up risk gap g_m is greater than $2/(m+1)$, one additional bounded-loss ensemble member cannot flip the medoid. Negative margins mean that this sufficient local condition is not met. They do not prove that the selected branch is wrong, but they do show that the small selected ensembles sit near medoid boundaries in some folds.

Table 11 summarises the branch-error types. In these selected greedy repeats, no error crosses the root split. The binary model has one same-parent or sibling error and fifteen further errors within the same top-level branch; the multistate model has seven same-top-level errors. This pattern is consistent with the low p-adic losses: the wrong predictions are usually nearby in the assumed tree.

For each selected candidate where the defining design matrix can be reconstructed, the determinant and $v_5(\det A)$ are recorded. Table 12 summarises those rows. Most selected systems have $v_5(\det A) = 0$. The multistate selected rows include a maximum of 1, which indicates some divisibility by the chosen base, but this diagnostic does not currently explain the remaining fold fragility by itself.

Dataset	Rows	Missing det.	Median $v_5(\det A)$	Max $v_5(\det A)$	Mean raw loss
Binary	288	0	0.0	0.0	0.047
Multistate	395	0	0.0	1.0	0.054

Table 12: Conditioning diagnostics for selected through-points systems. Larger $v_5(\det A)$ indicates stronger divisibility by the chosen p-adic base and therefore a less well-conditioned exact certificate.

8 Selected feature usage

The selected greedy ensembles can also be dissected by counting which feature representatives appear in the one-standard-error selected models. Across ten repeats, the binary selection uses 40 selected hyperplanes: $m = 4$ models for each of ten seeds. Those models use all 36 collapsed binary feature representatives at least once. The multistate selection uses 50 selected hyperplanes and likewise uses all 30 collapsed multistate feature representatives at least once.

Dataset	Feature representative	Selected models containing feature
Binary	`1Cor.4.6.3.b2`	17
Binary	`Eph.3.18.1`	17
Binary	`Rom.15.19.1.b2`	15
Binary	`1Cor.15.53.1.b1`	14
Binary	`1Cor.4.6.3.b1`	13
Binary	`1Cor.4.6.3.b3`	12
Multistate	`1Cor.10.13.1`	25
Multistate	`1Cor.1.24.1`	20
Multistate	`Rom.15.19.1`	18
Multistate	`1Cor.15.47.1`	18
Multistate	`Rom.15.19.2`	18
Multistate	`Eph.3.18.1`	17

Table 13: Most frequent feature representatives in the one-standard-error selected greedy ensembles. Counts are over 40 selected binary models and 50 selected multistate models across ten repeated selection seeds.

The counts in Table 13 should not be read as a claim that one passage alone explains the tree. They show the opposite: the selected ensembles reuse some readings more often, but the selected models are distributed across the available collapsed feature set. First Corinthians supplies the largest number of selected feature slots, but it also supplies the largest number of available collapsed features in both encodings. Romans, Colossians, Philippians, and Ephesians provide repeated secondary support. The model is therefore best interpreted as a sparse ensemble of local reading certificates rather than as a single decisive branch marker.

The nested run gives a stricter feature-usage view because it counts features in outer-fold models whose hyperparameters were selected inside the training data. Table 14 ranks feature representatives by active use: a feature is active only when it appears in a selected outer-fold certificate, has a non-zero coefficient, and has a non-zero value for the held-out witness. In the binary encoding the most active representatives are `1Cor.15.55.1.b1`, `Phil.2.5.2`, `1Cor.1.24.1`, and `Col.1.16.1.b2`. The most frequently selected binary representative, `Rom.15.19.1.b2`, appears in more selected certificates but is active less often because its non-zero support is narrow. In the multistate encoding, `1Cor.10.13.1`, `1Cor.15.47.1`, `Gal.4.8.1`, and `1Cor15.54.1` dominate active use; these variables have non-zero support across all witnesses and should be interpreted as broad multistate separators rather than as isolated branch markers.

The logistic and tree-model importance rankings only partly agree. Table 15 compares the top eight active held-out features in each encoding. In the binary encoding, logistic shares four top-eight features with the selected decision-tree ranking and three with the selected random-forest ranking. In the multistate encoding, logistic shares four top-eight features with each tree-model ranking. The agreement is therefore real but not strong enough to claim a model-independent feature ranking.

Table 16 compares the non-p-adic rankings with the nested p-adic feature-usage rankings. The strongest overlap is the binary random forest against the p-adic active-feature ranking, with five of the top eight shared. Agreement with the p-adic selected-certificate ranking is weaker. This reinforces the interpretation that "important feature" depends on the fitted model and on

Dataset	Feature	Active	Models	Folds	Support branches
Binary	1Cor.15.55.1.b1	54	72	12	5
Binary	1Cor.1.24.1	53	60	12	6
Binary	Col.1.16.1.b2	52	85	12	4
Binary	Phil.2.5.2	52	70	12	5
Binary	1Cor15.54.1.b2	49	95	12	4
Binary	1Cor15.54.1.b1	44	93	12	4
Binary	1Cor.2.4.1.b2	40	79	12	5
Binary	1Cor.4.6.3.b1	34	65	12	5
Multistate	1Cor.10.13.1	118	118	12	7
Multistate	1Cor.15.47.1	101	102	12	7
Multistate	1Cor15.54.1	88	95	12	7
Multistate	Gal.4.8.1	81	81	12	7
Multistate	Col.1.16.2	70	70	12	7
Multistate	Col.1.16.1	67	72	12	7
Multistate	Eph.3.18.1	66	66	12	7
Multistate	2Cor.1.10.1	65	73	12	7

Table 14: Nested p-adic feature representatives ranked by active use in outer predictions. "Active" requires selection into an outer-fold certificate, a non-zero fitted coefficient, and a non-zero held-out feature value.

Dataset	Model	Top features	Logistic–model overlap
Binary	Nested decision tree	8	4/8
Binary	Nested random forest	8	3/8
Multistate	Nested decision tree	8	4/8
Multistate	Nested random forest	8	4/8

Table 15: Agreement between nested logistic coefficient rankings and tree-model impurity-importance rankings. Features are ranked by active held-out weight.

whether importance means a coefficient/impurity score, selected certificate frequency, or active use in the held-out prediction.

Dataset	Model	Active overlap	Selected overlap
Binary	Nested decision tree	3/8	3/8
Binary	Nested logistic	3/8	2/8
Binary	Nested random forest	6/8	2/8
Multistate	Nested decision tree	2/8	3/8
Multistate	Nested logistic	3/8	4/8
Multistate	Nested random forest	3/8	4/8

Table 16: Overlap between the top eight non-p-adic feature-importance rankings and the nested p-adic feature-usage rankings. Active overlap uses the p-adic active-feature ranking; selected overlap uses selected certificate frequency.

9 Discussion

The method shows how p-adic loss can be used as an evaluation geometry for hierarchical branch-point prediction. The branch-risk medoid has a concrete prefix-consensus interpretation: each raw hyperplane prediction votes for congruence classes modulo 5, 25, 125, and so on, with geometrically smaller rewards for deeper agreement. This gives an interpretable alternative to flat class accuracy when the labels are positions in an assumed tree.

The experiment on Donker's derived data shows that selected greedy through-points ensembles can be competitive on the training-selected leave-one-out surface, but also that this surface is too optimistic to use as a final performance estimate. Under nested selection, the p-adic ensembles do not beat the strongest non-p-adic comparator by mean p-adic loss: logistic is best for binary features, while random forest is best for multistate features. The p-adic method's remaining advantages are different: it uses substantially fewer active parameters, it produces exact local certificates, and it gives an auditable feature-usage trail. The small values of m are not arbitrary endpoints; they come from repeated greedy curves and remain small in the nested run, where no selected repeat reaches the $m = 50$ cap.

The experiment does not establish a historical stemma, identify sources, or make an authorship claim. The features are derived from Donker's analysis rather than directly recollated from manuscript images or editions. There are only twelve witnesses, and singleton branch-point classes such as `Ath` and `D` make the leave-one-out setting especially brittle: when either singleton is held out, its true branch label is absent from the training data. Source-level reconstruction from the underlying apparatus, or renewed collation of the relevant passages, is the next philological step before any historical interpretation should be attempted.

10 Conclusion

The main result is a workable recipe for exact p-adic branch-point prediction when the feature space is larger than the witness set, together with a warning about selection on very small data. Encode branch paths in base 5, fit exact through-points hyperplanes on random feature subsets, and aggregate the raw predictions by a p-adic medoid. On this Donker-derived case study, non-nested greedy selection can tune the p-adic ensembles to strikingly low apparent loss, but nested leave-one-out evaluation favours simpler non-p-adic baselines by mean p-adic loss. The p-adic ensembles remain sparse and interpretable, and the nested feature-usage counts show that the result is not carried by a single reading; it is an ensemble of small local certificates. The logistic, decision-tree, and random-forest rankings give useful corroboration for some features,

but not a single stable importance ordering. The limitations remain decisive for interpretation: the target tree is an assumed modelling scaffold, the features are derived rather than freshly collated, and twelve witnesses are not enough for a historical reconstruction of Athanasius' textual tradition.

A Reproducibility

The project repository is `https://github.com/solresol/athanasius`. The README describes the full workflow, and the current repository Makefile exposes a single top-level rebuild command:

```
make all
```

The repeated ensemble-selection command is the expensive step inside that workflow. The present chapter uses the existing targeted run with subset sizes $s = 1, \ldots, 10$, ten repeats, and $K = 200$ candidates per repeat. The nested redo is generated by `scripts/run_branchpoint_nested_selection.py`; the run reported here uses the same subset sizes and repeats, $K = 200$, $m \leq 50$, and per-fold checkpointing under `outputs/tables/branchpoint_nested_checkpoints/`. The logistic, decision-tree, and random-forest feature comparison is generated by the Makefile target `branchpoint-model-feature-comparison`.

References

[1] Gerald J. Donker. *The Text of the Apostolos in Athanasius of Alexandria*. Society of Biblical Literature, 2011.

[2] Bruce M. Metzger and Bart D. Ehrman. *The Text of the New Testament: Its Transmission, Corruption, and Restoration*. 4th edition. Oxford University Press, 2005.

[3] D. C. Parker. *An Introduction to the New Testament Manuscripts and their Texts*. Cambridge University Press, 2008. `https://www.cambridge.org/core/books/an-introduction-to-the-new-testament-manuscripts-and-their-texts/contents/9A683AA00CBDE0FF71A7D20ABDE6B37C`

[4] James R. Royse. *Scribal Habits in Early Greek New Testament Papyri*. New Testament Tools, Studies and Documents 36. Brill, 2008. `https://catalog.libraries.psu.edu/catalog/43335410`

[5] Peter M. Head. The habits of New Testament copyists: singular readings in the early fragmentary papyri of John. *Biblica* 85, 399–408, 2004. `https://www.bsw.org/biblica/index-by-authors/v1-g/v2-l/`

[6] Eldon Jay Epp. It's all about variants: a variant-conscious approach to New Testament textual criticism. *Harvard Theological Review*, 2007. DOI: 10.1017/S0017816007001599.

[7] Tommy Wasserman. The Coherence Based Genealogical Method as a tool for explaining textual changes in the Greek New Testament. *Novum Testamentum* 57(2), 206–218, 2015. DOI: 10.1163/15685365-12341486.

[8] Institute for New Testament Textual Research. The Coherence-Based Genealogical Method. `https://www.uni-muenster.de/INTF/Genealogical_method.html`

[9] Ofer Dekel, Joseph Keshet, and Yoram Singer. Large margin hierarchical classification. In R. Greiner and D. Schuurmans, editors, *Proceedings of the Twenty-First International Conference on Machine Learning*, 209–216, 2004. `https://collaborate.princeton.edu/en/publications/large-margin-hierarchical-classification/`

[10] Carlos N. Silla Jr. and Alex A. Freitas. A survey of hierarchical classification across different application domains. *Data Mining and Knowledge Discovery* 22(1–2), 31–72, 2011. DOI: 10.1007/s10618-010-0175-9.

[11] Aris Kosmopoulos, Ioannis Partalas, Eric Gaussier, Georgios Paliouras, and Ion Androutsopoulos. Evaluation measures for hierarchical classification: a unified view and novel approaches. *Data Mining and Knowledge Discovery* 29(3), 820–865, 2015. DOI: 10.1007/s10618-014-0382-x.

[12] Cinna Wu, Mark Tygert, and Yann LeCun. A hierarchical loss and its problems when classifying non-hierarchically. *PLOS ONE* 14(12), 2019. DOI: 10.48550/arXiv.1709.01062.

[13] Luca Bertinetto, Romain Mueller, Konstantinos Tertikas, Sina Samangooei, and Nicholas A. Lord. Making better mistakes: leveraging class hierarchies with deep networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020. `https://arxiv.org/abs/1912.09393`

[14] Ashwin Vaswani, Gaurav Aggarwal, Praneeth Netrapalli, and Narayan G. Hegde. All mistakes are not equal: comprehensive hierarchy aware multi-label predictions. arXiv:2206.08653, 2022. `https://arxiv.org/abs/2206.08653`

[15] Ashima Garg, Depanshu Sani, and Saket Anand. Learning hierarchy aware features for reducing mistake severity. In *European Conference on Computer Vision*, 2022. `https://arxiv.org/abs/2207.12646`

[16] Jack Valmadre. Hierarchical classification at multiple operating points. In *Advances in Neural Information Processing Systems*, 2022. `https://arxiv.org/abs/2210.10929`

[17] Jia Shi, Gautam Gare, Jinjin Tian, Siqi Chai, Zhiqiu Lin, Arun Vasudevan, Di Feng, Francesco Ferroni, and Shu Kong. LCA-on-the-Line: benchmarking out-of-distribution generalization with class taxonomies. In *International Conference on Machine Learning*, 2024. `https://arxiv.org/abs/2407.16067`

[18] Roman Plaud, Matthieu Labeau, Antoine Saillenfest, and Thomas Bonald. Revisiting hierarchical text classification: inference and metrics. In *Proceedings of the 28th Conference on Computational Natural Language Learning*, 2024. `https://arxiv.org/abs/2410.01305`

[19] Nan Li, Bo Kang, and Tijl De Bie. Your next state-of-the-art could come from another domain: a cross-domain analysis of hierarchical text classification. arXiv:2412.12744, 2024. `https://arxiv.org/abs/2412.12744`

[20] Abhinandan Krishnan and Abilash Amarthaluri. Large scale product categorization using structured and unstructured attributes. arXiv:1903.04254, 2019. `https://arxiv.org/abs/1903.04254`

[21] Thomas G. Dietterich. Ensemble methods in machine learning. In *Multiple Classifier Systems*, Lecture Notes in Computer Science, 1–15. Springer, 2000. DOI: 10.1007/3-540-45014-9_1.

[22] Rich Caruana, Alexandru Niculescu-Mizil, Geoff Crew, and Alex Ksikes. Ensemble selection from libraries of models. In *Proceedings of the 21st International Conference on Machine Learning*, 2004. DOI: 10.1145/1015330.1015432.

[23] Zhi-Hua Zhou, Jianxin Wu, and Wei Tang. Ensembling neural networks: many could be better than all. *Artificial Intelligence* 137(1–2), 239–263, 2002. DOI: 10.1016/S0004-3702(02)00190-X.

[24] Tin Kam Ho. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20(8), 832–844, 1998. DOI: 10.1109/34.709601.

[25] Leo Breiman. Bagging predictors. *Machine Learning* 24, 123–140, 1996. DOI: 10.1007/BF00058655.

[26] Leo Breiman. Random forests. *Machine Learning* 45(1), 5–32, 2001. DOI: 10.1023/A:1010933404324.

[27] Philipp Probst and Anne-Laure Boulesteix. To tune or not to tune the number of trees in random forest? *Journal of Machine Learning Research* 18(181), 1–18, 2017. `http://jmlr.org/papers/v18/17-269.html`

[28] André F. T. Martins. Learning with the p-adics. arXiv:2512.22692, 2025. `https://arxiv.org/abs/2512.22692`

[29] Gnankan Landry Regis N'guessan. v-PuNNs: van der Put neural networks for transparent ultrametric representation learning. arXiv:2508.01010, 2025. `https://arxiv.org/abs/2508.01010`