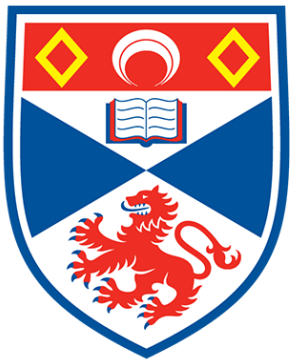
Applying Active Learning Algorithms for Population Linkage

James Ross [redacted]
Supervisor: Dr. Ozgür A gün

January 16th 2026



University of
St Andrews

Abstract

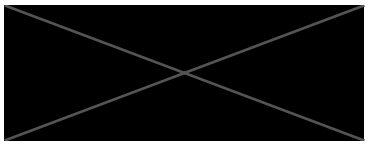
Population linkage refers to the task of identifying and linking records which concern the same or related individuals across large-scale population datasets. Due to the expertise required to accurately identify such relationships, applying traditional supervised machine learning methodologies to these linkage problems can prove highly expensive, as they rely on vast quantities of labelled training data. This project proposes and implements a system for solving both identity linkage and sibling bundling problems using active learning algorithms, which aim to minimise the quantity of training data required by strategically selecting the most uncertain pairs for labelling. Using synthetic and real-world datasets, the effectiveness of different classifiers and sampling strategies, as well as the impact of data corruption, is evaluated; concluding that the system achieves performance comparable to that of costly passive learning systems, with only a fraction of the labelled data.

Declaration

I hereby certify that this dissertation, which is approximately 12,000 words in length, has been composed by me, that it is the record of work carried out by me and that it has not been submitted in any previous application for a degree.

This project was conducted by me at the University of St Andrews from September 2025 to January 2026 towards fulfilment of the requirements of the University of St Andrews for the degree of Master of Science (MSci) in Computer Science under the supervision of Dr. Özgür Akgün.

In submitting this project report to the University of St Andrews, I give permission for it to be published online. I retain the copyright in this work.



January 16th 2026

Acknowledgements

I'd like to thank both my supervisor, Dr. Özgür Akgün, and the guest-star of my weekly supervisor meetings, Prof. Graham Kirby, not only for their consistent support and encouragement throughout this project, but for providing me with the opportunity to explore such a unique field of research. I'd also like to thank my family, friends and my cat, Fluffy, for their unwavering love and support and for always listening; even when I've been at my most incomprehensible. Finally, I'm incredibly grateful to the School of Computer Science for providing me with the facilities and resources that have made this project, and the last five years of my studies, possible.

Contents

Acronyms . . . . . **iv**

1. Introduction **1**
1.1. Objectives 2
1.2. Revision of Original Objectives 3

2. Context-survey **5**
2.1. Background 5
2.1.1. Population Linkage 5
2.1.2. Machine Learning 6
2.1.3. Active Learning 7
2.1.4. Python Active Learning Frameworks 8
2.1.5. Classifiers 9
2.1.6. Synthetic Populations 10
2.2. Related works 11
2.2.1. Splink 11
2.2.2. Febrl 12
2.2.3. Enhancing Demographic Linkage with Active Learning in Machine Learning 12

3. Requirements Specification **15**
3.1. Implementation Requirements 15
3.2. Experimental Requirements 17

4. Ethics **19**

5. Design and Implementation **20**
5.1. Design Plan 20
5.2. Defining Sibling Bundling 21
5.3. Data Preprocessing 21
5.4. Ground-Truth Generation 21
5.5. Blocking 22
5.5.1. Identifying Fields for Blocking 22
5.6. Comparison 24
5.6.1. Identifying Fields for Comparison 24
5.7. Active Learning 25
5.7.1. Initial Labelling 25

5.7.2. The Automatic Stopping Criterion . . . 25
5.8. Evaluating a Learning System . . . 26
5.8.1. Classifier-Only vs. Combined Evaluation . . . 26
5.8.2. Measuring Uncertainty . . . 27
5.9. Parallelism . . . 27

6. Experimental Setup and Results . . . 28
6.1. Datasets . . . 28
6.2. Experimental Setup . . . 29
6.3. Blocking Experiments . . . 31
6.3.1. Best-Case Blocking (Clean Data) . . . 31
6.3.2. Blocking on Corrupted Data . . . 33
6.3.3. Blocking with Real-World Data . . . 34
6.3.4. Exploring Lower Thresholds . . . 36
6.4. Passive Learning Experiments . . . 39
6.4.1. Classifier vs. Combined Performance . . . 39
6.4.2. Passive Learning on Corrupted Data . . . 41
6.4.3. Comparing Classifiers for Passive Learning . . . 43
6.4.4. Passive Learning with Real-World Data . . . 44
6.4.5. Identifying Significant Thresholds . . . 46
6.5. Active Learning Experiments . . . 46
6.5.1. Initial Labelling . . . 47
6.5.2. Active Learning Results . . . 50
6.5.3. Passive vs. Active Learning . . . 52
6.5.4. Active Learning on Clean and Corrupted Data . . . 54
6.5.5. Random Sampling . . . 56
6.5.6. Comparing Classifiers for Active Learning . . . 59
6.5.7. Initial Labelling Retrospective . . . 60
6.5.8. Active Learning with Real-World Data . . . 60

7. Evaluation and Critical Appraisal . . . 62
7.1. Evaluating against Primary Objectives . . . 62
7.2. Evaluating against Revised Secondary and Tertiary Objectives . . . 63
7.2.1. Comparing Passive and Active Learning . . . 64
7.2.2. Exploring Different Configurations . . . 64
7.2.3. Evaluating Performance with Real-World Data . . . 64
7.2.4. Evaluating the Stopping Condition . . . 65
7.3. Comparison of Related Works . . . 65
7.3.1. Splink . . . 65
7.3.2. Enhancing Demographic Linkage with Active Learning in Machine Learning . . . 65

8. Conclusions 67
References 69
A. Source Code Repository 73
B. Ethics Approval Form 74
C. Valipop Configuration 76
D. Crptr Configuration 77
E. Crptr Standard Profiles 78
F. Blocking Plots for TD5k (Clean) 79
G. Random Seeds 80
H. Learner Scores 81
I. Passive Learner Plots 96
J. Passive Learner Confusion Matrixes 103
K. Active Learner Plots 110

Acronyms

AL Active Learning 1, 7, 20, 28, 62, 67

LSH Locality-Sensitive Hashing 20, 31, 67, 75

ML Machine Learning 6, 20

OCR Optical Character Recognition 11, 28, 73, 74

PL Passive Learning 1, 3, 6, 29, 64, 67

SHiPP Scottish Historic Population Platform 3, 67

1. Introduction

Population linkage (also known as *demographic linkage*) refers to the process of identifying records in population data which correspond to the same or related individuals. By drawing connections between records, linkage offers significant potential across a diverse range of contexts, from helping medical researchers trace conditions through lineage to aiding historians in following the migration of populations over time, or even simply grouping present-day records by geographical area. While this is traditionally a manual and resource-intensive process, many population datasets are prohibitively large.

Traditional supervised machine learning approaches, such as Passive Learning (PL), present one potential solution to this: learning patterns from a labelled set of training data and applying them to accurately identify the remaining unlabelled data. However, in the context of population linkage, obtaining a sufficiently large training set can prove highly expensive, both financially and timewise, due to the domain-expertise required to accurately label potential links.

This project proposes and implements a solution to this issue through Active Learning (AL). AL algorithms strategically select the most uncertain pairs of records for labelling, aiming to minimise the quantity of training data required, and thus input from the domain-expert. The implemented system addresses two of the many types of population linkage: sibling bundling from birth records and birth-death identity linkage (see Section 2.1.1).

To evaluate the performance of this system, and the impact of different variables, both synthetic and real-world datasets were utilised. The use of synthetic data facilitated a greater degree of control over errors and corruption present in the records, allowing the impact of these to be isolated and measured. A series of experiments were devised to compare the effectiveness of different classifiers, Random Forest and XGBoost, as well as uncertainty-based and random sampling strategies for selecting data points to label.

The experimental results (see Chapter 6) show that the active learning system achieved performance scores comparable to, although not exceeding, that of a passive learner, while requiring only a fraction of the training data (approximately $\leq 0.1\%$). The uncertainty-based query strategy reached these scores significantly faster than random sampling, with the latter proving computationally infeasible for several experiments, given the time and hardware available. However, independent of the learning method, the results conclude that the design

of the blocking algorithm, used to filter candidate pairs, is the most significant factor limiting performance.

1.1. Objectives

As part of the initial proposal, a set of primary, secondary and tertiary objectives were devised for this project to achieve.

Primary Objectives

1. Develop a system which leverages active learning algorithms to efficiently label and train a model for population linkage, starting with two basic types of linkage: sibling bundling from birth records and birth-death identity linkage (see Section 2.1.1).
2. Evaluate the accuracy and effectiveness of this system on clean synthetic populations data generated using `Valipop` [13].
3. Explore the impact to performance of introducing data corruption, induced using `Crptr` [3], to the synthetic data.

Secondary Objectives

1. Extend the system to cover an additional three types of linkage: birth-marriage identity linkage, marriage-death identity linkage and sibling bundling from death records.
2. Subject to accurate results being achieved with the synthetic data, evaluate the system's performance with real-world data with available ground-truth, using the POPLINK (Umeå) dataset (see Section 6.1) [15].

Tertiary Objectives

1. Implement consistency-checking within chains of linked records (e.g. if a marriage record has both birth and death records linked for the groom, it follows that those two records should also be linked).

2. Extend the system to support additional missing linkage types, for example linking parents to their children's birth records.
3. Explore the behaviour of the system on a real-world dataset without ground-truth. For this purpose, the Scottish Historic Population Platform (SHiPP) dataset would be used, with an external collaborator providing responses to labelling queries (subject to additional ethics approval for external collaborator involvement) [25].

1.2. Revision of Original Objectives

In response to the results of the primary objectives, the project deviated from several of its original secondary and tertiary objectives. Initial findings demonstrated disproportionately positive results, irrespective of the data provided and blocking threshold used. These results were attributed to the use of `scikit-learn`'s evaluation methodology, which assessed only the performance of the classifier instead of the system as a whole (see Section 5.8.1). Consequently, a bespoke approach to evaluation, that explicitly accounted for the effects of the blocking stage, was required.

Rather than implementing additional types of linkage, priority shifted towards developing an in-depth understanding of the overall effectiveness of the AL system. This included exploring the impact of different configurations and comparing performance against Passive Learning (PL) approaches. Additionally, experiments with the SHiPP dataset were deprioritised in favour of the synthetic and POPLINK datasets (see Section 6.1), due to the availability of ground-truth values facilitating more-informative evaluation.

Revised Secondary Objectives

1. Subject to accurate results being achieved with the synthetic data, evaluate the system's performance with real-world data with available ground-truth, using the POPLINK (Umeå) dataset [15].
2. Implement a passive learning system and compare its performance against that of the AL alternative.
3. Evaluate the impact of different classifiers (Random Forest and XGBoost, see Section 2.1.5) and query strategies (uncertainty-based and random sampling).

Revised Tertiary Objectives

1. Implement an automatic stopping condition to prevent the active learner from querying additional data once performance has stabilised.
2. Extend the system to perform sibling bundling from death records.

2. Context-survey

2.1. Background

2.1.1. Population Linkage

Data linkage (also known as *data matching*, *record linkage*, *entity resolution*, *object identification* or *field matching*) refers to the process of identifying, matching, and merging records corresponding to the same data-subject across one or many datasets. In a world with an ever-expanding focus on data-collection, linkage is a crucial issue for efficient processing, management and analysis of large datasets. Linkage systems face many challenges, such as how to deal with duplicates, uncertain matches, and variations in schema between different data-sources [9] [10].

Population linkage (also known as *demographic linkage*) refers specifically to the application of linkage methods for population datasets, usually consisting of personal values such as names, dates of birth, addresses, and other relevant attributes [4]. While often viewed as a single problem, population linkage comprises several linkage tasks which, only when performed in conjunction, will provide an accurate and comprehensive reconstruction of a population. Each of these tasks address different relationships between records or the individuals that they pertain to; from linking families (identifying marital and parent-child relationships) to geographic linkage (grouping individuals living at the same address or in the same area). Two common linkage problems, identity linkage and sibling bundling, will be focused on in this project.

Identity Linkage

Arguably the most intuitive of these linkage tasks is **identity linkage**. This refers to the process of linking multiple records (for example birth and death records) which correspond to the same individual.

Sibling Bundling

Generally regarded as one of the easier types of linkage (as it requires less-precise matching of attributes, than identity linkage for example), **sibling bundling** is the process of grouping records which correspond to siblings. This is generally done by matching on common fields, such as parents' names or family home address.

2.1.2. Machine Learning

Machine Learning (ML) is a subset of artificial intelligence and can be considered the fundamental hypothesis behind much of modern AI research. It focuses on algorithms that learn patterns from labelled training data and use these to make accurate inferences about unlabelled data. The training process involves identifying and extracting key features from data items, which can be expressed numerically for comparison. Using these, the model learns to map different inputs to a desired output. In the context of population linkage, these feature vectors may be similarity scores between values in a record pair, for example comparing the name fields of a birth and death record [17].

Unsupervised Machine Learning

Unsupervised ML algorithms are a subset of these which do not require any human participation or labelled data during training. These algorithms instead use methods such as clustering to group unlabelled data points, based on their locality or similarity. This process is best suited for tasks where patterns aren't necessarily visible to human observers and there is no "correct" answer. Thus, these algorithms are generally unsuitable for population linkage [17].

Supervised Machine Learning

Supervised ML refers to another subset of algorithms which learn from an annotated training dataset to perform tasks, such as classification, with accuracy. Traditionally, these models require a human in the loop to provide ground-truth by accurately labelling a subset of the data [17].

One of the most basic implementations of this is **Passive Learning** (**PL**). Passive learners are initialised using a large set of labelled data, which the algorithm

learns in its entirety. This approach does not involve any human participation beyond providing this labelled subset.

2.1.3. Active Learning

A key disadvantage of supervised ML methods is their reliance on labelled data for training. For optimal results, training data should be well-balanced, containing comparison vectors corresponding to both matches and non-matches (assuming that this is a binary classification problem, and uncertain matches are not also being considered), as well as representing the overall characteristics of the dataset being classified [7].

In the context of population linkage, this presents a new issue as labelling training data will likely require the input of a human participant with sufficient expertise to identify matches and non-matches with a reasonable degree of accuracy. Depending on the data being worked with, evaluating potential links may require this participant to undertake additional background research, thus rendering the process expensive both financially and timewise. Population linkage is also an inherently unbalanced problem, as any given dataset will likely have a significantly higher number of non-matches than matches, increasing the difficulty of obtaining balanced training data.

Active Learning (**AL**) offers a potential means of mitigating, or even resolving, these issues. Active learning enables an ML algorithm to evaluate the uncertainty of unlabelled instances and selectively query the data most likely to improve performance. This approach optimises the number of queries given to the human participant, thus minimising the cost of reaching high accuracy [7] [28].

While the underlying hypothesis of AL is relatively straightforward, a vast range of implementations are proposed by different research literature. Due to the diverse nature of the problems which AL can be applied to, there is no universal best option. The performance of the system can be impacted by several attributes including the query strategy, which defines how the learner should select data to label. This could be through a measure of uncertainty (**uncertainty-based sampling**), density of data, or even having a group of different ML models voting (query-by-committee). Another key consideration is the choice of classifier for predicting unlabelled data (see Section 2.1.5) [28] [2].

2.1.4. Python Active Learning Frameworks

A number of frameworks exist for implementing active learning systems in Python. Some of these specialise in particular use-cases such as `small-text` [26], a lightweight text classifier, while other focus on optimising specific classifiers, for example Bayesian Active Learning (`baal`) [5], a `PyTorch`-based library for deep-learning using Bayesian uncertainty. This survey shall however focus on four of the more general options.

modAL

The framework chosen for this project is `modAL` [14], a modular library implementing 9 AL strategies. Built on `scikit-learn` [24], it is compatible with several pre-defined classifiers and models, lightweight compared to other frameworks, and supports Bayesian methods. Its definition of density (the sum of the distance to all other samples) may however result in poor performance when less than 20% of the dataset is labelled. While a full experimental framework is not provided, the global `ActiveLearner` object, which wraps around a `scikit-learn` estimator, can be used to manage the unlabelled dataset and incrementally train on new labelled instances [1].

libact

`libact` [32] is a pool-based AL libary, lacking native support for stream-based applications. It implements a limited set of AL strategies, prioritising performance via code written in C, and provides an adapter for compatibility with `scikit-learn` estimators. It also offers an Active-Learning-By-Learning (ALBL) meta-algorithm, which adaptively selects a query strategy, but in practice this performs poorly compared to selecting the right algorithm from the offset [1].

ALiPy

`ALiPy` [29] is the largest of these frameworks, implementing 25 AL algorithms, as well as compatibility for classifiers from both `scikit-learn` and `TensorFlow` [22]. A full experimental framework is provided, facilitating storing intermediate results and generating visualisations of results. Similar to `libact`, it also implements meta-algorithms for selecting query strategies, although these only provide performance benefits for a limited subset of problems. These additional features

also introduces a trade-off in efficiency, with experiments taking longer to run than in other frameworks [1].

Scikit-learn

For specialised experiments requiring greater flexibility, or when working on resource-constrained systems, researchers may choose to avoid AL frameworks and instead build directly on `scikit-learn`. While simple examples of this approach exist [27], more complex applications require significant additional boilerplate code and are therefore not advised unless full control of the pipeline is essential.

2.1.5. Classifiers

Classifiers (also known as *classification algorithms*) are fundamental to supervised ML classification tasks. These algorithms extract key features which characterise each label during training, deriving a prediction function which is then applied to unlabelled data. A range of classifiers exist; some focus on linear decision boundaries, such as logistic regression classifiers, while others are designed to represent more complex relationships. This project shall explore two non-linear classifiers [16].

Random Forest

Random Forest is a common non-linear ML classifier, which combines the outputs of multiple decision trees to reach a single result. It requires three hyperparameters: node-size, number of trees, and the number of features to sample. The algorithm evenly splits the labelled training data, using bootstrap-sampling, and trains each decision tree on a different subset. For each prediction, each tree votes on the classification to apply to the unlabelled instance. This averaging process is a key feature of the algorithm as it helps lower overall variance and prediction error, as-well as reducing the risk of overfitting. However, this comes at a cost in terms of time and computational resources [18].

Extreme Gradient Boosting (XGBoost)

XGBoost is a Python implementation of an optimised distributed gradient-boosting classifier. It implements a sparsity-aware algorithm, leveraging parallel and distributed computation, to solve problems significantly quicker than other algorithms and also supports GPU acceleration. The algorithm adopts a different approach to multiple decision trees, starting with a single weak decision tree. New trees are created iteratively, each trained on the gradient of the loss function derived from the error residuals of the previous model. Once fully trained, the final classification result is calculated by combining the weighted contributions of all trees [6] [19].

2.1.6. Synthetic Populations

To evaluate the quality of linkage, common measures such as Precision, Recall and F1 can be used (see Section 5.8). These however rely on the availability of ground-truth. This can prove challenging in the context of population linkage research, where the available data is often historic and lacking ground-truth labels. While more recent datasets exist and are more likely to contain an accurate mapping of truth, these will pertain to individuals still living or recently deceased and consequently are usually deemed sensitive and made unavailable for both legal and ethical reasons.

Synthetic populations present a solution to this problem, through generating datasets which mimic the probabilities and distributions present in real-world population datasets. Synthetic individuals, and the relationships between them, can be assigned unique identifiers during the generation process, providing a reliable source of truth. Additionally, as these records do not directly correspond to any real individuals, the data and results derived from them can (in most circumstances) be published without any ethical or legal concerns.

Valipop

`valipop` is a simulation model used to generate realistic synthetic populations. The model takes a dataset defining distributions and probabilities, over a range of years, for various properties drawn from real-world populations data as an input. Given a starting population size and a timeframe to generate records for, the simulation mirrors trends and patterns from the input data (scaled to the set timeframe), and outputs a set of synthetic birth, death and marriage records

with unique identifiers for providing ground-truth. Once generated, a statistical analysis is performed on the resulting data to verify that the desired properties and distributions are present in the synthetic population [13].

Crptr

Developed to aid in the evaluation of record-linkage systems, `crptr` is a data corruption tool designed to simulate a wide range of real-world data errors. These range from basic forms of data corruption (e.g. missing records and values, data stored in incorrect fields, or duplicate records) to more nuanced variations arising from human-error (such as common typographical errors and phonetic variations) [3].

As part of the same research which produced `valipop`, an extension package was developed for corrupting synthetic populations data. This extension provides classes for parsing and modifying synthetic datasets, whilst ensuring all ground-truth values are kept intact for evaluation. An additional two types of corruption are included: common name variations and Optical Character Recognition (OCR) errors (mistaking similar looking characters, such as 'u' and 'v'). The latter of these can be used to simulate the errors that would be expected in population data that has been transcribed to a digital format from original paper records. The accompanying thesis also proposes three "corruption profiles" for configuring the tool for use with synthetic populations (see Appendix E) [12] [3].

2.2. Related works

2.2.1. Splink

Developed by the Ministry of Justice, `splink` is an open-source Python data-linkage package for performing probabilistic and highly-scalable linkage on administrative datasets across the justice space. It implements a Fellegi-Sunter linkage model, using an Expectation-Maximisation algorithm for estimating parameters. The package is optimised for high-performance with Apache Spark and also provides built-in functionality for generating a range of graphical visualisations, presenting the performance of linkage models in an accessible and user-friendly format [20].

Expanding beyond the original use-case, researchers have developed extensions for `splink` to implement an active learning approach for linking medical data. The implemented AL strategy calculated a usefulness measure for unlabelled data items, using a combination of Shannon entropy and frequency-based uncertainty measures. This system was evaluated by performing linkage between two registries, with around 5,000 entries each. By labelling just 126 training samples and using a classification threshold of -4, the study achieved impressive results, with a mere 29 false-positives across the full dataset [23].

2.2.2. Febrl

Freely Extensible Biomedical Record Linkage (FEBRL) is an open-source application for solving parallelised large-scale data-linkage problems. Designed to handle variations commonly found in real-world data, the system presents innovative features for data-cleaning and standardisation, based on a Hidden Markov Model (HMM). Unlike traditional rule-based cleansing, these models apply a probabilistic approach and learn likely sequences and patterns in fields from training data then use this to accurately segment and process input data. This method proved highly effective for data which followed rigid and established structures, such as addresses, but was outperformed by traditional rule-based approaches for more nuanced fields, such as names [11].

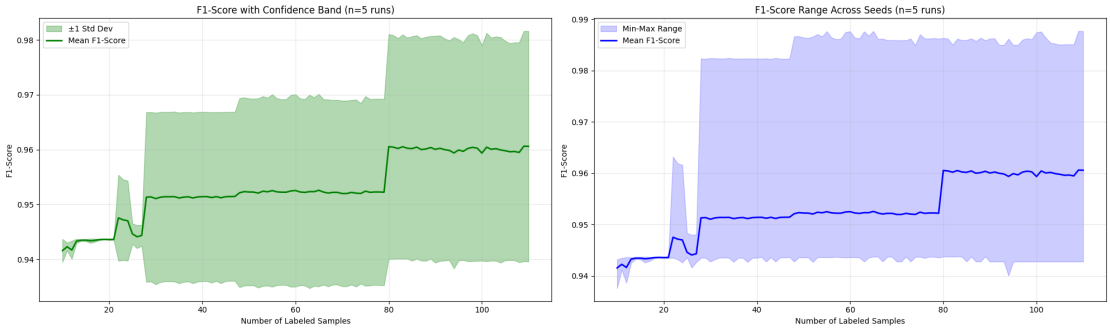
2.2.3. Enhancing Demographic Linkage with Active Learning in Machine Learning

A similar active learning approach for population linkage is proposed and implemented in the paper "Enhancing Demographic Linkage with Active Learning in Machine Learning". This study focuses on the comparison of passive learning and active learning algorithms for birth-death identity linkage on a synthetic dataset. The results in Table 2.1 show that, given an initial labelled subset consisting of 10 labelled pairs (equally balanced between positive and negative), the AL algorithm was able to match the passive learner with only 100 queries, on corrupted synthetic data blocked using a threshold value of 0.7 [21].

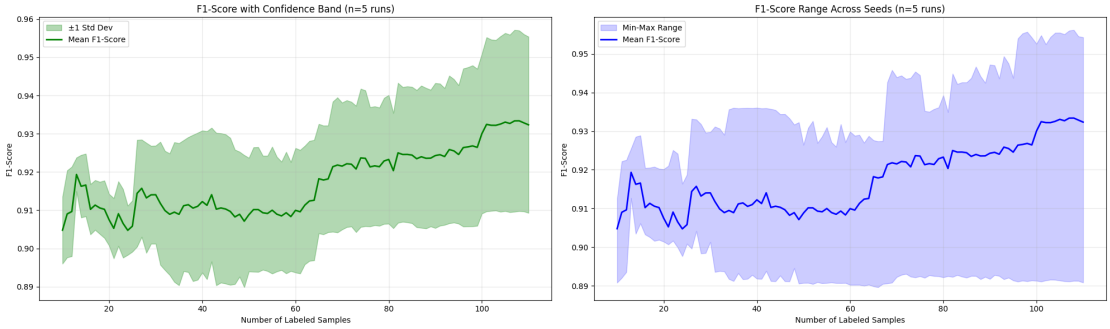
	F1 Scores		
Dataset	Passive Learner	Active Learner (Initial)	Active Learner (100 queries)
Clean	0.999	0.938	0.999
Corrupt	0.989	0.927	0.985

Table 2.1.: Table comparing F1 scores for a passive learner against an active learner on clean and corrupted datasets, using a blocking threshold of 0.7 (adapted from [21, Tab. 4.1 - Tab. 4.3]).

The results of this study also suggest that uncertainty-based querying offers no significant advantage over random sampling, reaching comparable F1 scores on both clean and corrupted synthetic data. However, the evaluation methods used in this study take only the post-blocking data into consideration when assessing performance, resulting in unrealistic results (as seen in Figure 2.1). If true-positive links lost during the blocking process were also to be taken into consideration, the results would likely show lower performance overall and a greater disparity between clean and corrupted datasets.



(a) Clean dataset.



(b) Corrupted dataset.

Figure 2.1.: Figure from [21, Fig. 4.9] comparing results of an active learner, using a Random Forest classifier, with both uncertainty-based (left) and random sampling (right) query strategies on clean and corrupted synthetic datasets (blocked at threshold of 0.7).

3. Requirements Specification

The following requirements were devised from the objectives outlined in Section 1.1.

3.1. Implementation Requirements

Description	Justification
Implement a preprocessor module with functions for reading records from synthetic datasets into a standardised format and handling incomplete data.	Standardising record formats allows code to be written independent of datasets, while appropriate handling of missing values prevents errors arising in later stages of the pipeline.
Implement a preprocessor for real-world POPLINK records.	As above, this facilitates applying the existing system to real-world data without adjustment.
Implement an LSH MinHash blocking function to generate a set of candidate pairs.	Due to the scale of population data, it is computationally infeasible to evaluate all possible pairs without filtering first.
Implement a comparison function for evaluating candidate pairs.	This generates an array of feature vectors, used by the system for learning and prediction, from relevant values in a record pair.
Implement a function to compare birth and death years against age at death for birth-death identity linkage.	Ensuring consistency between registered birth and death years and age at death helps identify implausible links between records corresponding to otherwise similar individuals.

Continued on next page...

Description	Justification
Implement an initial labelling function for the active learner which mirrors the manual labelling process.	As it is unlikely that an expert will produce a perfectly balanced subset, imitating this provides a more accurate representation of how the system would perform in a real-world use-case.
Implement the main AL training loop function.	Abstracting the training process into a single function, which manages querying data, learning and performance evaluation for each iteration, simplifies experimental setup.
Implement functionality to evaluate the combined performance of the blocking and learning stages.	Provides an accurate overview of the overall effectiveness of the system, accounting for all stages of the pipeline.
Implement an end-to-end experiment for birth-death identity linkage.	Provides a runnable in-context example of identity linkage for use in experimental procedure.
Implement an end-to-end experiment for sibling bundling from birth records.	Provides a runnable in-context example of sibling bundling for comparison.
Implement an end-to-end experiment for sibling bundling from death records.	Facilitates comparison of sibling bundling performance using birth and death records.
Design an “auto-stop” condition to end the training loop once performance has stabilised.	An adaptive stop condition optimises the system by preventing the learner from running longer than necessary.

Table 3.1.: Table of functional requirements for the proposed system.

3.2. Experimental Requirements

Description	Justification
Implement CI/CD unit-tests for the `al_linker` Python package.	Ensures all functions behave as expected and that future development does not disrupt existing functionality.
Conduct an experiment to measure the number of positive and negative links retained at blocking threshold values ranging from 0.05 to 0.95 for each dataset.	The effectiveness of the blocking stage has a significant impact on any subsequent learning, rendering this crucial to understanding the learner's performance.
Evaluate the performance of a passive learner across blocking threshold values ranging from 0.05 to 0.95 for each dataset.	Provides a benchmark to compare results of an active learner against and informs the selection of blocking thresholds to explore with AL.
Compare the performance of passive learners using Random Forest and XGBoost classifiers.	Provides an indication of the relative performance of each classifier and generates additional benchmarks for comparison.
Compare passive learner performance using classifier-only and combined evaluation methods (see Section 5.8.1).	Emphasises the limitations of classifier-only evaluation and isolates the impact of the blocking stage on overall performance.
Evaluate the performance of an active learner using only an initial labelled subset prior to training.	Establishes a starting point for AL, providing context to subsequent performance gains during training.
Compare initial active learner performance when using balanced and unbalanced labelled subsets.	Determines whether a smaller but balanced initial subset offers any advantages over a larger, unbalanced, set.
Evaluate active learner performance throughout training.	Provides insight into performance improvements made at each iteration and when the learner begins to stabilise.

Continued on next page...

Description	Justification
Compare active learner performance on clean and corrupted versions of the same synthetic data.	Facilitates isolating and visualising the effects of data corruption on AL performance.
Compare active learner performance with Random Forest and XGBoost classifiers.	Evaluates whether classifier choice has a significant impact on AL results.
Compare the performance of active learners using uncertainty-based and random sampling query strategies.	Determines whether uncertainty-based strategies offer advantages over simple random selection.
Compare active learning performance when initialised with balanced and unbalanced labelled subsets.	Evaluates the influence that the initial labelled subset has on the final performance of an AL system.
Compare the performance of active and passive learners.	Explore whether AL systems can achieve comparable or even superior performance to passive learners, and how much data must be labelled to achieve this.
Evaluate the effectiveness of the active learner on real-world POPLINK data.	Provides insight into how the learner performs in an uncontrolled environment with potentially ambiguous, flawed and incomplete data.

Table 3.2.: Table of requirements for proposed experiments for the project.

4. Ethics

As the proposed project involves work with two sensitive external datasets which correspond to real data subjects, some of whom may still be alive at the time of research, ethical considerations do apply (see Appendix B for the submitted *Ethics Approval Form*). However, no data contained in or derived from these datasets will be published as part of this project, nor can any sensitive information be inferred from the results reported.

The work with both POPLINK and SHiPP datasets (as outlined in Chapters 1 and 3) is covered by existing, approved, ethics applications; therefore no additional applications were required for this project.

5. Design and Implementation

This chapter presents an overview of the design and notable features of the Active Learning (AL) system implemented to fulfil the objectives and requirements set out in Chapters 1 and 3.

5.1. Design Plan

The Machine Learning (ML) pipeline for the proposed system can be divided into five main stages, as illustrated in Figure 5.1:

1. **Preprocessing** - Records are read from input files and converted into a standardised format, handling missing values.
2. **Ground-Truth Generation** - ID values within the records are used to construct a lookup table of all true-positive links.
3. **Blocking** - Locality-Sensitive Hashing (LSH) MinHash blocking is used to filter candidate pairs of records.
4. **Comparison** - Fields expected to match or remain consistent between linked records are compared for all candidate pairs.
5. **Active Learning** - The AL system incrementally queries data points for labelling, evaluating its performance on test data at each iteration.

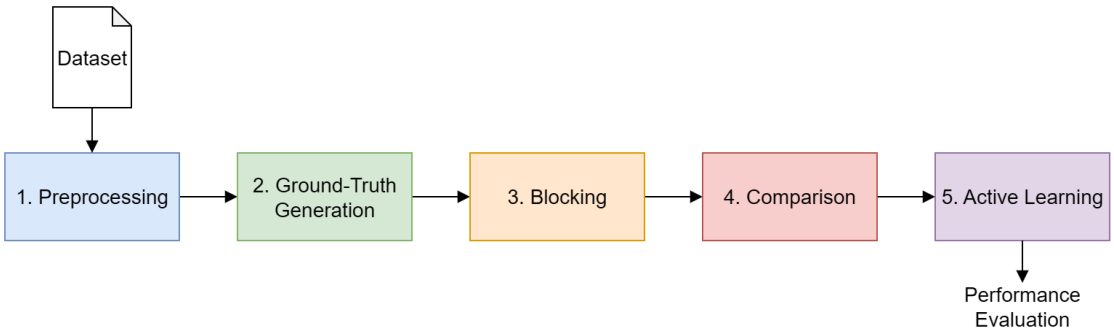


Figure 5.1.: Diagram illustrating the five main stages of the active learning pipeline.

While minor variations exist between identity linkage and sibling bundling problems, such as the generation of ground-truth lookup tables, much of the core functionality remains unchanged. Therefore, a modular design was adopted to minimise unnecessary code repetition and aid the experimental procedure for both this project and future work, allowing different configurations to be easily interchanged. This resulted in the Python package `al_linker`, which implements functionality for most stages of the pipeline, except ground-truth generation.

5.2. Defining Sibling Bundling

There exist multiple interpretations of the sibling bundling linkage task. In some literature, only complete groups of siblings within a family are considered positive links, and there is additional ambiguity around half-siblings. For the purposes of this project, a positive sibling bundling link shall be defined as a bidirectional link between a pair of records corresponding to individuals sharing the same mother and father.

5.3. Data Preprocessing

The `al_linker` package implements preprocessors for both the synthetic datasets generated by Valipop and the POPLINK dataset (see Section 6.1), standardising column names and dropping fields that are not required. These functions also ensure the correct datatypes are set for columns and replace missing values in text columns with empty Strings.

The omission of a ground-truth field for families within the POPLINK dataset presented an issue for sibling bundling. To address this, an identifier was synthesised for each birth record by hashing the identifiers for the mother and the father.

5.4. Ground-Truth Generation

For evaluating the total number of true-positive and true-negative links and assigning labels to data points, a ground-truth lookup set is constructed, containing the record identifiers of all true-positive pairs. For sibling bundling, canonicity is

enforced to prevent double-counting or other issues by sorting record identifiers in ascending order for each pair.

5.5. Blocking

As population datasets are typically of significant size, it is computationally infeasible to evaluate all possible pairs. Thus, LSH MinHash blocking is used to group similar records using hashing functions, returning a list of candidate pairs. This functionality is encapsulated within a single public function, `al_linker.blocking.lsh_minhash.block()`, which accepts up to two dataframes, a list of column names to hash, and a sensitivity threshold as parameters.

The behaviour of this function adapts depending on whether the optional second dataframe is provided. When two dataframes are passed, such as in birth-death identity linkage, the function only returns cross-dataframe candidate pairs. Whereas for sibling bundling tasks, candidate pairs are generated within a single dataframe, with canonicity enforced to prevent double-counting by ordering the record identifiers in each pair.

5.5.1. Identifying Fields for Blocking

The hashing process itself is computationally expensive, scaling with the number of columns selected. As such, it's crucial to select a minimal set of columns which can be expected to match for true-positive links. To identify suitable fields, a data-analysis script was implemented to calculate the probability that values match for each column within true-positive pairs. This was run with a clean synthetic dataset (see TD100k (Clean) in Section 6.1) to return the best-case results, which would be expected in the absence of data corruption or any other ambiguity.

While blocking columns were selected primarily using the clean results, scores for all three datasets are reported in Table 5.1. These probability values can however be misleading, as they do not consider the opposite perspective of which columns are also matches for non-links. For example, "mother_occupation" is missing for all columns in the POPLINK dataset so, despite being a guaranteed match, is not suitable for use in blocking or comparison.

	Probability of Match		
Column	**Clean**	**Corrupt**	**POPLINK**
sex	1.000	0.964	0.991
mother_surname	1.000	0.766	0.248
father_surname	1.000	0.759	0.448
mother_forename	1.000	0.744	0.393
father_forename	1.000	0.743	0.234
surname	1.000	0.738	0.609
forename	1.000	0.725	0.659
mother_occupation	0.000	0.003	1.000
father_occupation	0.000	0.003	0.350

(a) Birth-Death Identity linkage.

	Probability of Match		
Column	**Clean**	**Corrupt**	**POPLINK**
surname	1.000	0.703	0.627
father_forename	1.000	0.721	0.737
mother_surname	1.000	0.742	0.809
mother_forename	1.000	0.724	0.680
father_surname	1.000	0.741	0.697
address	0.980	0.785	0.815
mother_occupation	0.695	0.659	1.000
father_occupation	0.648	0.617	0.528
sex	0.618	0.603	0.596
birth_year	0.324	0.324	0.289
forename	0.319	0.322	0.289

(b) Sibling bundling from birth records.

Table 5.1.: Table detailing probability that values match for each column within true-positive links in TD100k (Clean), TD100k (Corrupt) and POPLINK datasets.

From the results shown in Table5.1, and manual experiments observing the number of positive links lost with different combinations of columns, the following columns were selected:

- **Birth-Death Identity Linkage** - “forename”, “surname”, “father_surname”, “mother_surname”
- **Sibling Bundling from Birth Records** - “father_forename”, “mother_forename”, “father_surname”, “mother_surname”

As there is no clean version of the real-world POPLINK data available, identifying optimal blocking fields for this is a more difficult process. For the purpose of the experiments conducted in Chapter 6, the blocking columns selected using the clean synthetic data were used for all datasets.

5.6. Comparison

Once candidate pairs have been generated, the String fields in these records are compared using the following three similarity measures to generate an array of feature vectors for each pair:

1. Jaro-Winkler similarity
2. Damerau-Levenshtein distance (normalised)
3. Jaccard similarity

For birth-death identity linkage, an additional consistency score is calculated using the year of birth, year of death, and age at death.

5.6.1. Identifying Fields for Comparison

The probabilities reported in Table 5.1 were also used to select the following String columns for comparison:

- **Birth-Death Identity Linkage** - “forename”,“surname”, “sex”, “father_forename”, “father_surname”,“mother_forename”, “mother_surname”
- **Sibling Bundling from Birth Records** - “surname”, “father_forename”, “father_surname”, “mother_forename”,“mother_surname”, “address”

While additional columns pertaining to parents’ marriage details are available in all the datasets experimented with in Chapter 6, these were left out to avoid issues around handling “illegitimate” children.

5.7. Active Learning

`ALWrapper` abstracts the `ModAL.ActiveLearner` class and handles the full experimental procedure for active learning, from initial labelling to training and evaluation. An extension of this wrapper, `ALMultiWrapper`, was also implemented to optimise running multiple learners on the same dataset.

5.7.1. Initial Labelling

While the availability of ground-truth does allow the system to filter candidate pairs by their true labels and construct a perfectly balanced initial labelled subset, `ALWrapper` instead simulates the process an expert would follow when manually constructing such a subset. `ALWrapper.label_initial_data()` accepts a target number of pairs to aim to label for each class and randomly selects candidate pairs to label until the target is met for both classes.

As population linkage is an inherently unbalanced problem, this usually results in an unbalanced initial labelled subset. An optional Boolean parameter, `rebalance`, can be enabled to discard surplus data points and enforce balance in the initial subset.

5.7.2. The Automatic Stopping Criterion

An optional "auto-stop" condition is implemented in the function `ALWrapper.run_training_loop()` to avoid unnecessary iterations when the learner has already stabilised and therefore optimise the runtime of the system. This condition monitors a performance metric (F1 score by default), and after a minimum number of iterations (30 by default), evaluates stability over a window of recent iterations.

The metric is determined to be stable if, for all iterations within the most-recent window (15 iterations by default), the difference between successive values is less than a set threshold (1×10^{-3} by default). If this condition is enabled and the observed metric is determined stable, the training loop exits early.

5.8. Evaluating a Learning System

Precision, Recall and F1 scores are used to evaluate the performance of the learning system.

Precision measures the quality of positive predictions and is defined as the number of true-positive predictions over all positive predictions. **Recall** (also known as *hit rate*, *true-positive rate* and *sensitivity*) measures the proportion of true-positives that have been correctly predicted [8].

$$\text{Precision} = \frac{TP}{TP + FP} \qquad \text{Recall} = \frac{TP}{TP + FN}$$

For an overall representation of system performance, the **F1** measure combines both through a harmonic mean [8].

$$\text{F1} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2TP}{2TP + FP + FN}$$

5.8.1. Classifier-Only vs. Combined Evaluation

Calculating these performance scores using only the classifier outputs, or via the standard evaluation functions provided in `scitkit-learn`, can lead to misleadingly high scores, particularly when paired with high blocking thresholds (see Section 6.4.1). These evaluations ignore ground-truth positive links which are discarded during blocking and consequently never reach the classifier.

To obtain an accurate end-to-end evaluation of the system, performance metrics must account for these discarded pairs. Ground-truth positive links lost during blocking should be counted as false-negatives, while ground-truth negative links removed during blocking contribute to true-negatives. Although true-negatives do not directly affect the performance scores, they are reported in confusion matrixes.

For the purposes of the experiments in Chapter 6, these two approaches to evaluation are referred to as **classifier-only** and **combined**. Unless stated otherwise, the combined approach will be used for results.

5.8.2. Measuring Uncertainty

Uncertainty is calculated using the **least-confident** measure [28], defined for a data point x as:

$$U(x) = \min_{y \in \mathcal{Y}} P(y \mid x)$$

It should be noted that, for data point x, the probabilities over all possible labels $y \in \mathcal{Y}$ sum to 1. Thus, as this is a binary-classification task, the maximum uncertainty value is $U_{max} = 0.5$.

As the active learner can be configured to query any number of instances at each iteration during training, the uncertainty value recorded is the mean of the prediction uncertainty across all data points queried during that iteration.

5.9. Parallelism

To reduce computational load on a single core, several functions implemented in `al_linker` offer an optional `parallel` parameter which can be enabled to assign work in batches across all available cores, using `multiprocessing.Pool`.

In practice, however, due to the overhead of pickling large objects and aggregating results, this is detrimental to performance. Consequently, this parameter is only enabled during the preprocessing stage, where it is used for generating the family ground-truth column (see Section 5.3).

Instead, task-level parallelism was utilised through `GNUParallel` [30] for the experiments described in Chapter 6. `ModAL.ActiveLearner` was also configured to use multiple workers during predictions (capped at eight workers for each of the aforementioned experiments to avoid over-allocating when running several tests in parallel).

6. Experimental Setup and Results

6.1. Datasets

For evaluating the performance of the Active Learning (AL) system described in Chapter 5, three different datasets have been used. Each of these contains 200,000+ records, amounting to over 4×10^{10} possible pairs, and have been chosen as they facilitate exploring the effects of different factors, such as data corruption.

TD100k (Clean)

TD100k (Clean) is a synthetic dataset generated using the `valipop` simulation model [13], with a starting population size of 100,000 and a timeframe of 113 years (see Appendix C for full configuration). The resulting dataset contains 262,723 pairs of birth and death records, and 115,862 marriage records (although the latter will not be used in any of the following experiments). As this is a "perfect" synthetic dataset, TD100k (Clean) will be used to explore the best-case performance of the system, before introducing corruption.

TD100k (Corrupt)

TD100k (Corrupt) was generated using the `crptr` tool [3] to simulate real-world variations and data corruption (whilst retaining accurate ground-truth) on the records comprising TD100k (Clean). `crptr` was configured with profile A (see Appendix E), using the Optical Character Recognition (OCR) corruptor to simulate human transcription errors (see Appendix D for full configuration details). Since this variation is synthetically introduced, the results derived from working with this dataset can be compared against those from its clean counterpart, isolating and meaasuring the impact of data corruption on the quality of linkage.

POPLINK (Umeå)

Unlike the previous two datasets, **POPLINK** is a real-world dataset, developed by the Demographic Data Base at Umeå University, Sweden. The data is adapted from digitised Swedish population registers and represents the populations of four agricultural regions in Northern Sweden between 1700 - 1950, comprising 15 generations and over 350,000 individuals. To facilitate its usage for both research purposes and linking to modern registries, the dataset introduces unique identifiers for both individuals and records, which can be utilised as ground-truth for population linkage [31].

The subset of data which will be used for the following experiments corresponds to individuals in the Umeå region between 1900 and 1950. This subset consists of 270,581 birth records, 125,736 death records and 50,568 marriage records. As the data is derived from real-world records, it contains natural errors and data corruption, providing an insight into how the unpredictability of real-world data may effect linkage performance [15].

It should also be noted that due to the brief period recorded, birth-death identity linkage yields relatively low numbers of positive pairs (9.32×10^4 positive pairs against 3.40×10^{10} negative pairs). This period also spans both world wars and while Sweden remained neutral throughout these, the population still experienced disruption and migration, which may influence the completeness of this dataset.

6.2. Experimental Setup

To understand and evaluate the performance of the active learning system, three main tests (and scripts to run them) have been devised. The first of these focuses on the blocking stage and the impact of different thresholds, as the results at this stage define the upper bounds of performance for any subsequent learning stage. The second test evaluates the performance of a Passive Learning (PL) system on the blocked-data, providing a benchmark against which the results of the final test, evaluating the AL system itself, can be compared.

Due to the significant scale of the datasets used, the preprocessing stage for these experiments can be extremely computationally expensive. To reduce experimental cost, the tests have been broken down, separating preprocessing and blocking stages into individual scripts which pickle their results. This avoids

unnecessary repetition of expensive operations, as well as reducing the overall memory footprint of the experiments by discarding intermediary results and other variables that are no longer required at the end of each stage.

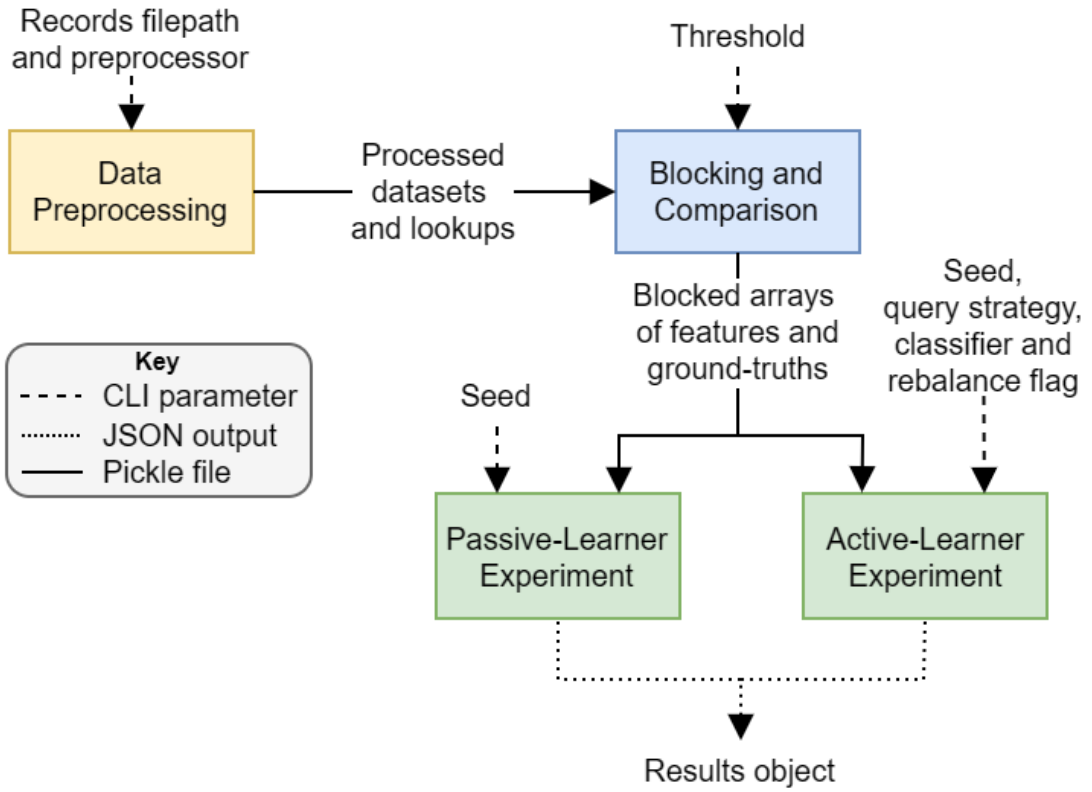


Figure 6.1.: Diagram depicting the experimental pipeline, showing pickled results and parameters used at each stage.

Note on Computational Infeasibility

The experiments described above were conducted on a machine with 1 TiB of RAM and 160 CPU cores. Whilst efforts were made to optimise the implemented system as much as possible, a small number of tests could not be completed with the available resources. For the purposes of the following results, **computational infeasibility** shall be defined as the inability to complete a task within 24 hours on the aforementioned machine.

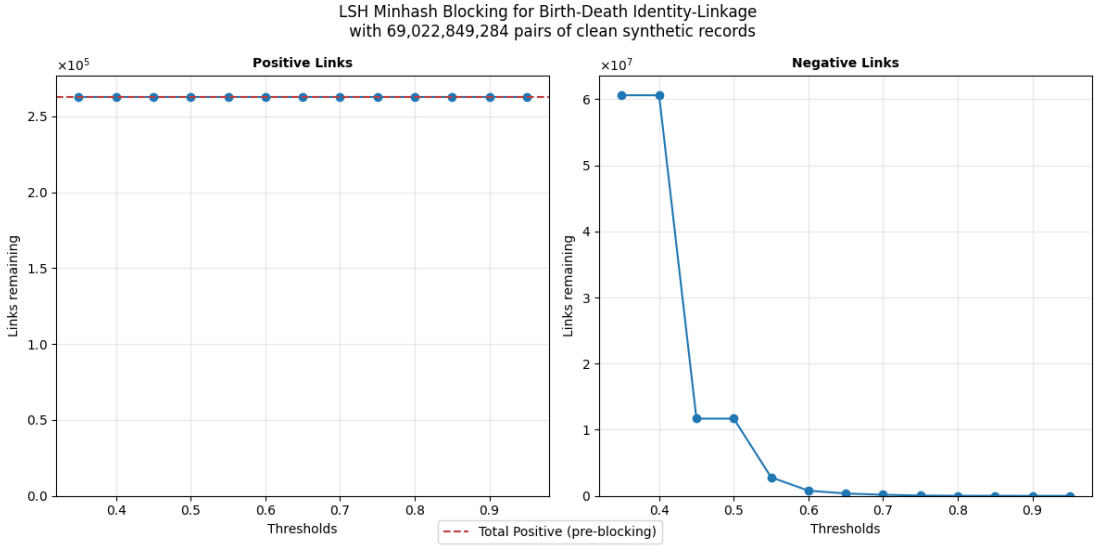
6.3. Blocking Experiments

To evaluate the accuracy of the Locality-Sensitive Hashing (LSH) MinHash blocking stage, an experiment was devised to measure the number of positive and negative links retained and discarded at each threshold value. This experiment was carried out with a sliding scale of blocking thresholds from 0.35 to 0.95, in increments of 0.05. Although thresholds as low as 0.05 were originally intended to be explored, these configurations proved computational infeasible (even when allocated 1 TB of RAM).

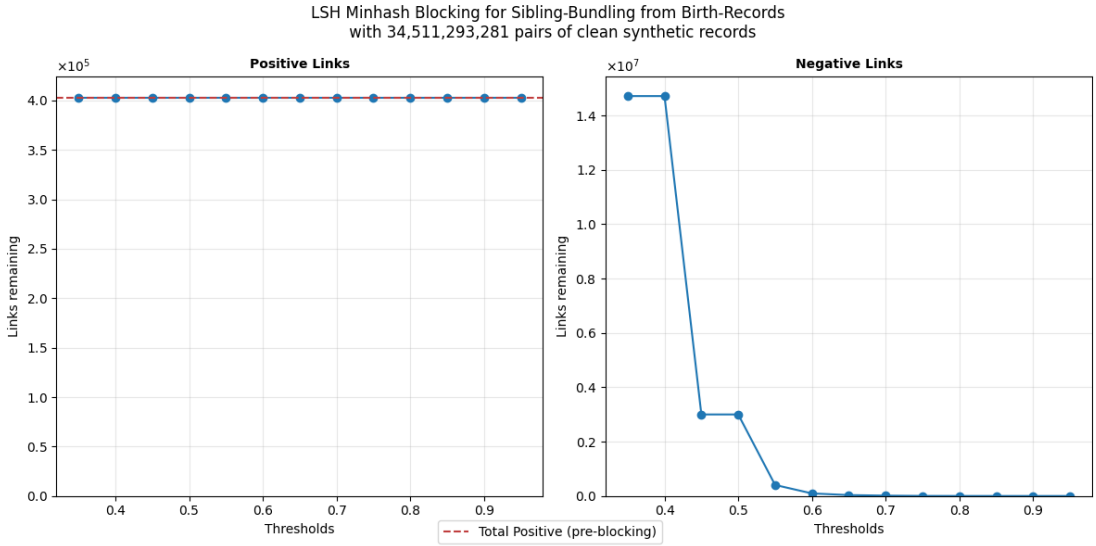
6.3.1. Best-Case Blocking (Clean Data)

To verify that the blocking function has been implemented correctly and suitable columns selected, the experiments were run first with TD100k (Clean). As the data does not contain any variations, it stands that true-positive pairs should generate identical hash signatures (see Section 5.5.1) and consequently all positive links will be retained, regardless of the threshold value set.

These best-case results are exhibited in Figure 6.2, where the number of negative links decreases as the threshold increases, while the number of positive links remains unaffected. It should also be noted that, as population linkage is an unbalanced problem, there is a significant difference in scale (two orders of magnitude) between the plots depicting positive and negative links.



(a) Birth-death identity linkage.

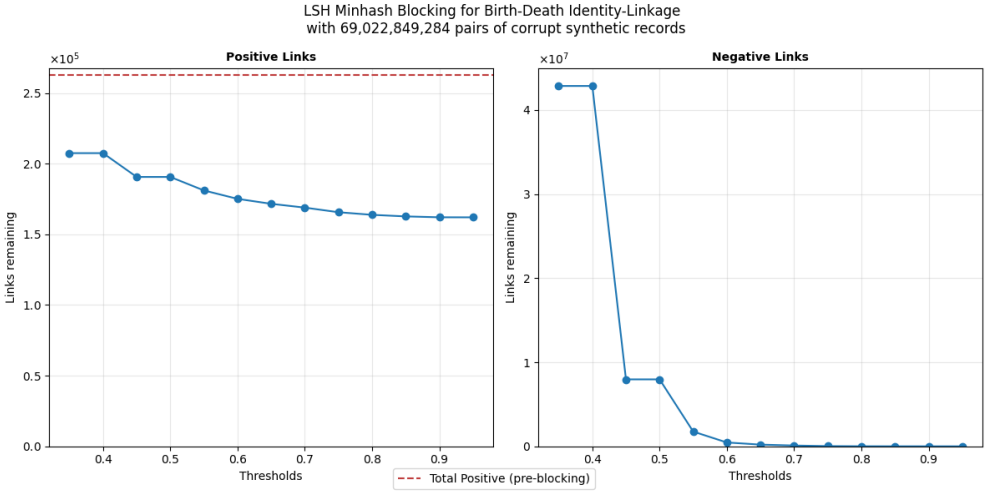


(b) Sibling Bundling from Birth Records.

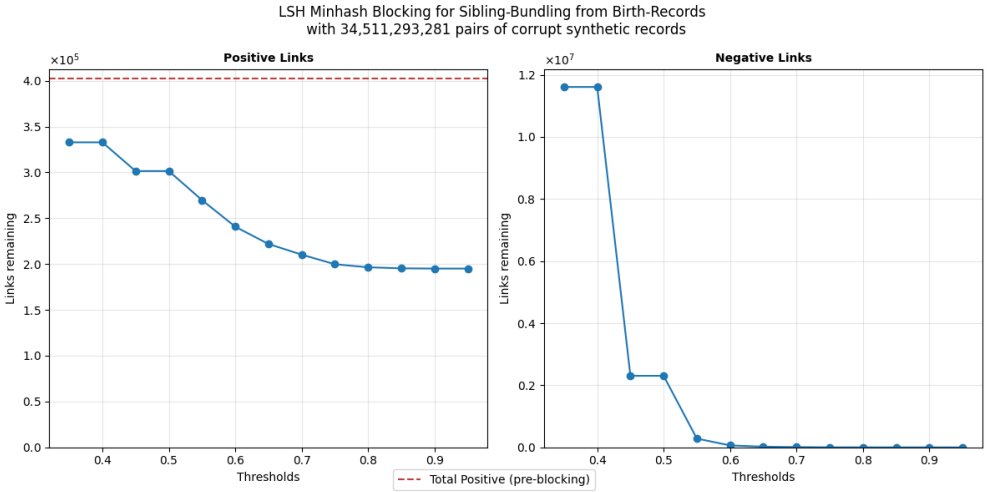
Figure 6.2.: Line graphs plotting the number of positive and negative links retained after blocking against a range of threshold values, on the synthetic TD100k (Clean) dataset.

6.3.2. Blocking on Corrupted Data

These perfect results are unlikely to be achieved in real-world usage due to data corruption in the values of the selected blocking columns. By repeating this experiment with TD100k (Corrupt), which represents the same underlying records, the impact of such variations can be measured in a controlled setting.



(a) Birth-death identity linkage.



(b) Sibling bundling from birth records.

Figure 6.3.: Line graphs plotting the number of positive and negative links retained after blocking against a range of threshold values, on the synthetic TD100k (Corrupt) dataset.

By introducing variations into the dataset, the plot seen in Figure 6.3 depicts positive links following a similar curve to the negative plot, albeit much shallower. In the context of birth-death identity linkage, 21% of positive links have already been lost with a threshold as low as 0.35 (or 17% for sibling bundling from birth records).

		Links Retained		
Threshold	**Positive (Clean)**	**Positive (Corrupt)**	**Negative (Clean)**	**Negative (Corrupt)**
Total	262,722	–	69,022,586,562	–
0.35	262,722	207,477	60,591,918	42,859,313
0.5	262,722	190,635	11,685,774	7,965,660
0.8	262,722	163,822	21,470	10,686

(a) Birth-death identity linkage.

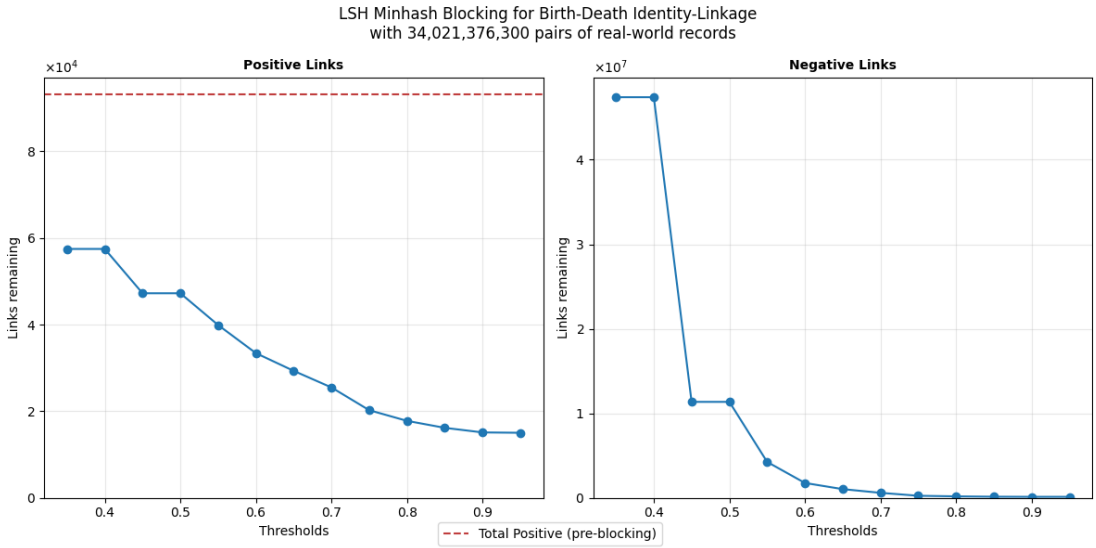
		Links Retained		
Threshold	**Positive (Clean)**	**Positive (Corrupt)**	**Negative (Clean)**	**Negative (Corrupt)**
Total	402,547	–	34,510,890,734	–
0.35	402,547	332,856	14,711,824	11,608,232
0.50	402,547	301,502	2,999,791	2,304,536
0.80	402,547	196,475	1,056	545

(b) Sibling bundling from birth records.

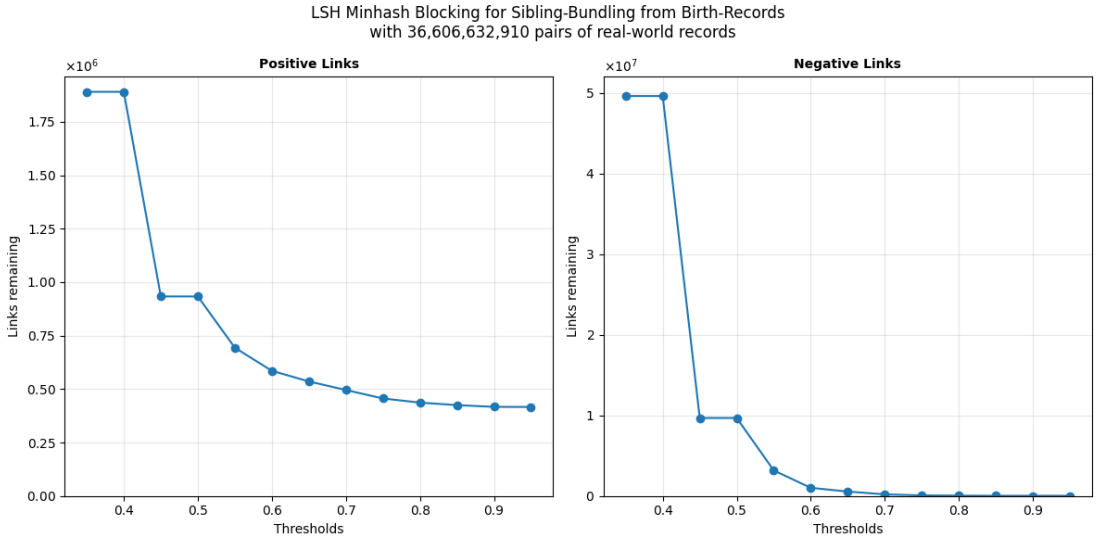
Table 6.1.: Tables reporting the number of positive and negative links retained at various thresholds, for both clean and corrupted versions of the synthetic TD100k data. Note that values marked '–' are identical to the value in the previous column.

6.3.3. Blocking with Real-World Data

As described in Section 5.5.1, since there is no clean version of the POPLINK data, it is difficult to select optimal columns for blocking. Thus, the following results assume that patterns similar to those observed in the synthetic datasets will also be present in this real-world data.



(a) Birth-death identity linkage.



(b) Sibling bundling from birth records.

Figure 6.4.: Line graphs plotting the number of positive and negative links retained after blocking against a range of threshold values, on the real-world POPLINK dataset. The total positive links benchmark has been omitted for sibling bundling as it did not fit on the scale of the plot due to the high proportion of links lost (99.5%).

Figure 6.4 depicts extremely poor blocking performance with the real-world POP-

LINK data, losing 38% and 99.5% of positive pairs at a 0.35 threshold for birth-death identity linkage and sibling bundling from birth records respectively. As a result of this, performance at the learning stage with this data will be severely limited for sibling bundling tasks.

The results for sibling bundling (compared against those for identity linkage) suggest that the patterns observed in the synthetic data are not reflected within the POPLINK data for this task. Thus, it is likely that these results could be improved by reevaluating the selected blocking columns.

	Links Retained	
Threshold	**Positive**	**Negative**
Total	93,182	34,021,283,118
0.35	57,458	47,395,422
0.50	47,231	11,358,944
0.80	17,779	182,269

(a) Birth-death identity linkage.

	Links Retained	
Threshold	**Positive**	**Negative**
Total	417,793,044	36,188,839,866
0.35	1,889,817	49,593,862
0.50	933,099	9,672,141
0.80	436,692	42,560

(b) Sibling bundling from birth records.

Table 6.2.: Tables reporting the numbers of positive and negative links retained at various thresholds, using the real-world POPLINK data.

6.3.4. Exploring Lower Thresholds

Ignoring the optimal performance for positive links in the clean synthetic data, a consistent pattern can be observed across Figures 6.2, 6.3, 6.4. Between thresholds of 0.35 and 0.4, the number of retained links remains stable, before dropping sharply at 0.45. The number of links stabilises again at 0.5, then starts to decline more gradually from 0.55, before it finally plateaus around 0.9.

To gain insight into how thresholds below 0.35 may affect the blocking results, a smaller synthetic dataset, **TD5k**, was created using the same configuration as TD100k (see Appendices C and D) but with a reduced population size of just 13,104 individuals (compared to 262,723 in TD100k).

The same blocking experiments were repeated with the TD5k (Corrupt) dataset, this time extending the range of thresholds to start at 0.05 and ran without issue (see Appendix F for results with TD5k (Clean)).

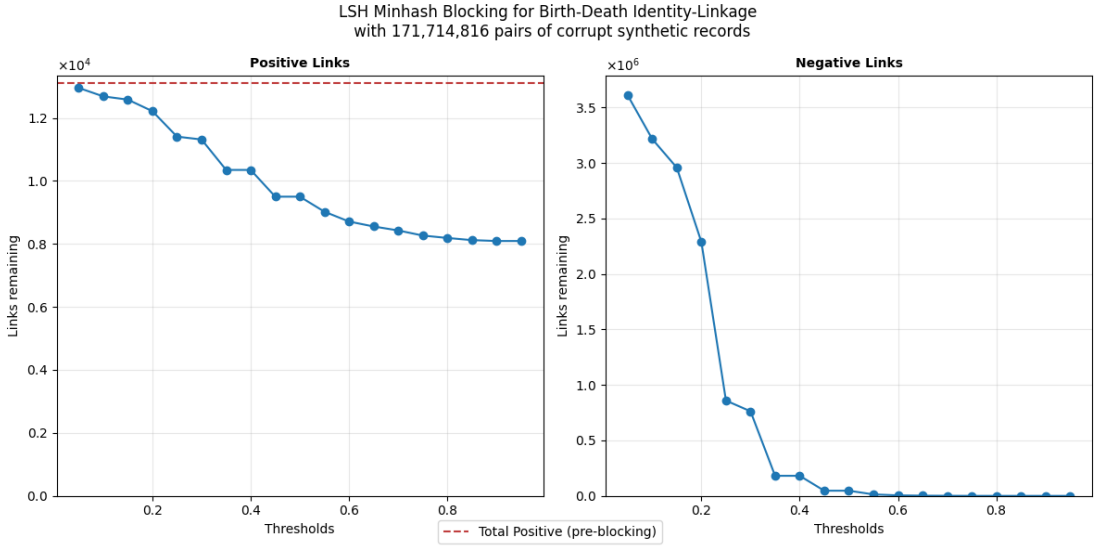
	Links Retained	
Threshold	**Positive**	**Negative**
Total	13,104	171,701,712
0.05	12,952	3,610,296
0.15	12,580	2,958,061
0.25	11,405	859,031
0.35	10,348	181,154

(a) Birth-death identity linkage.

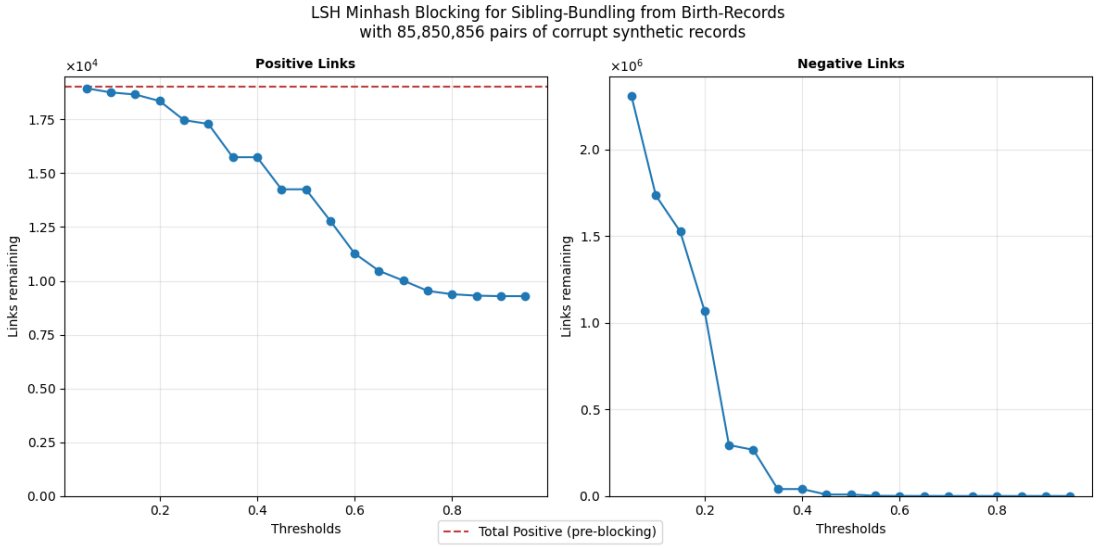
	Links Retained	
Threshold	**Positive**	**Negative**
Total	19,028	85,831,828
0.05	18,945	2,307,723
0.15	18,658	1,525,755
0.25	17,467	294,038
0.35	15,744	40,063

(b) Sibling bundling from birth records.

Table 6.3.: Tables reporting the number of positive and negative links retained after blocking with thresholds between 0.05 and 0.35, using the smaller TD5k (Corrupt) dataset.



(a) Birth-death identity linkage.



(b) Sibling bundling from birth records.

Figure 6.5.: Line graphs plotting the number of positive and negative links retained after blocking against a range of threshold values, on the smaller synthetic dataset, TD5k (Corrupt).

In Figure 6.5, a similar pattern is exhibited for the thresholds between 0.35 and 0.95. Between 0.05 and 0.3, the number of positive links retained decreases gradually, starting from just below the total positive links benchmark. However,

the number of negative links retained is significantly larger within this range, explaining the higher resource usage for these values.

As the number of links retained at thresholds greater than 0.35 follows the same pattern in each of these experiments, it could be hypothesised that similar results for the lower thresholds would also be achieved with the larger datasets, were the computational power available to evaluate these.

6.4. Passive Learning Experiments

To provide a benchmark for comparison with the AL system, a second experiment was conducted to evaluate the performance of a passive learner on the blocked data. The same range of blocking threshold values was used and the resulting candidate pairs are split into training and test sets with a 4:1 split (to facilitate potential future extension to 5-fold validation). The model is trained using the full training set and performance is then evaluated using the test set.

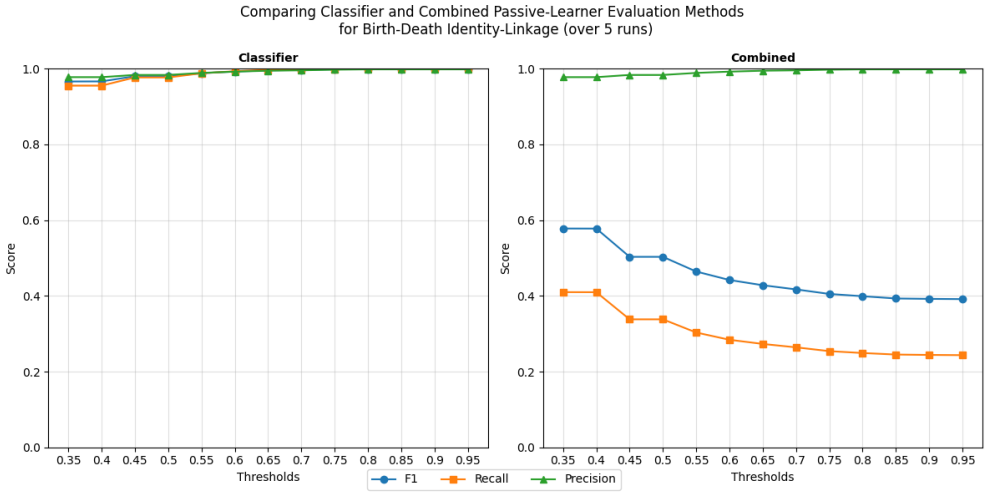
Two different classifiers, Random Forest and XGBoost, were used to assess whether classifier choice has a significant impact on the outcome. Each experiment was repeated five times with randomly generated seeds (see Appendix G), and thus the results presented below report mean values, with variability visualised through shaded area. While significant results will be included in the following sections, additional results can be found in Appendices H, I and J.

6.4.1. Classifier vs. Combined Performance

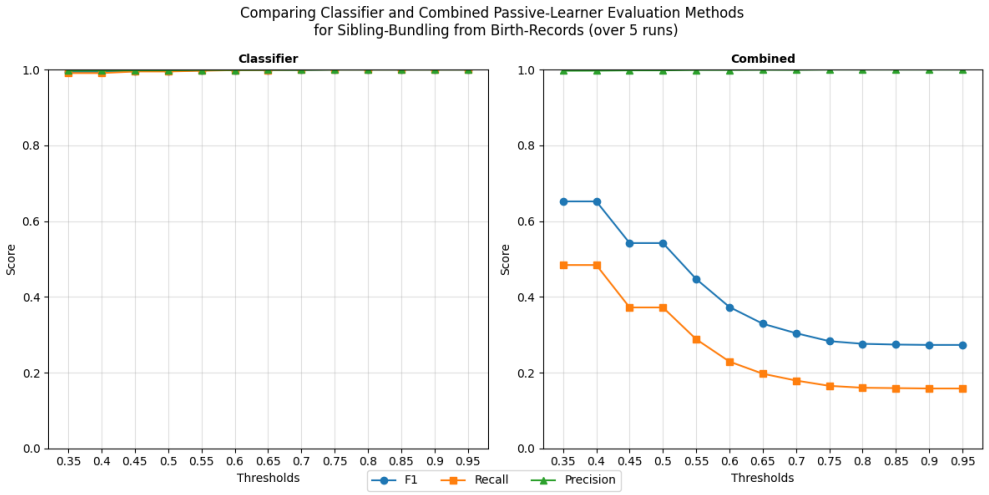
As discussed in Section 5.8.1, evaluation on blocked data alone does not provide an accurate representation of overall performance. When considering just the classifier, performance metrics improve as the blocking threshold increases. However, when accounting for the blocking stage, overall performance decreases at higher thresholds due to the number of positive links lost during blocking (except when linking clean data).

In the classifier-only evaluation in Figure 6.6, both Precision and Recall scores are high at lower threshold values and increase gradually, reaching near-perfect results at thresholds 0.7 and above. In contrast, under the combined evalution, the Recall score, and consequently F1, are heavily impacted by the increased

number of false-negatives, now including positive links discarded during blocking. Both scores follow trends similar to those observed in the blocking experiment, highlighting the impact this stage has. Precision, however, remains unaffected, as this measures only the output of the classifier (see Section 5.8).



(a) Birth-death identity linkage.



(b) Sibling bundling from birth records.

Figure 6.6.: Line graphs comparing classifier and combined evaluation methods for a passive learner, using a Random Forest classifier, on the synthetic TD100k (Corrupt) dataset.

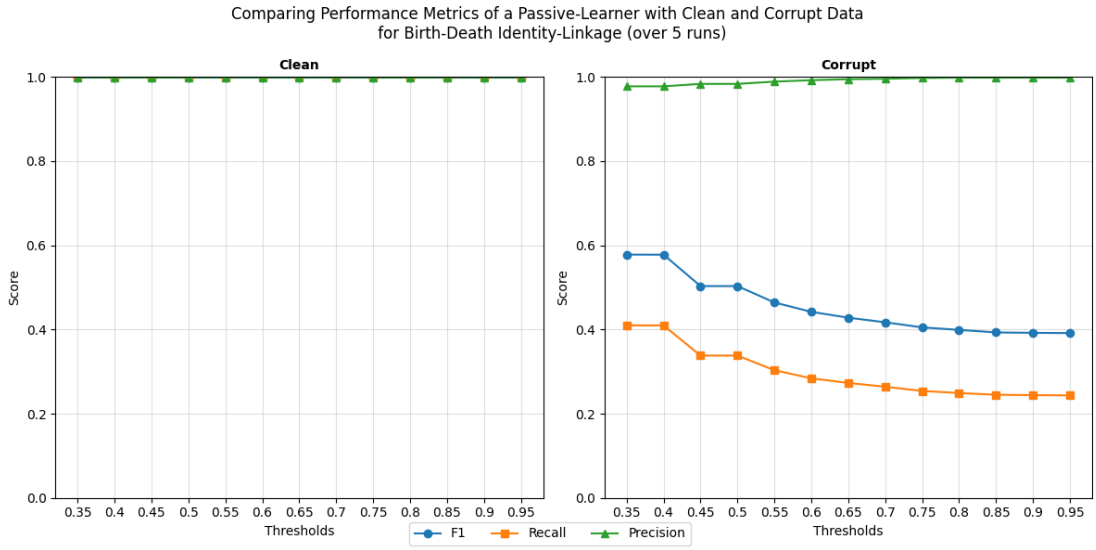
Under both evaluations, sibling bundling outperforms identity linkage at lower blocking thresholds, with the difference between linkage types diminishing at

higher thresholds. It can also be observed that randomisation has minimal effect on PL, since no error bars are clearly visible in Figure 6.6.

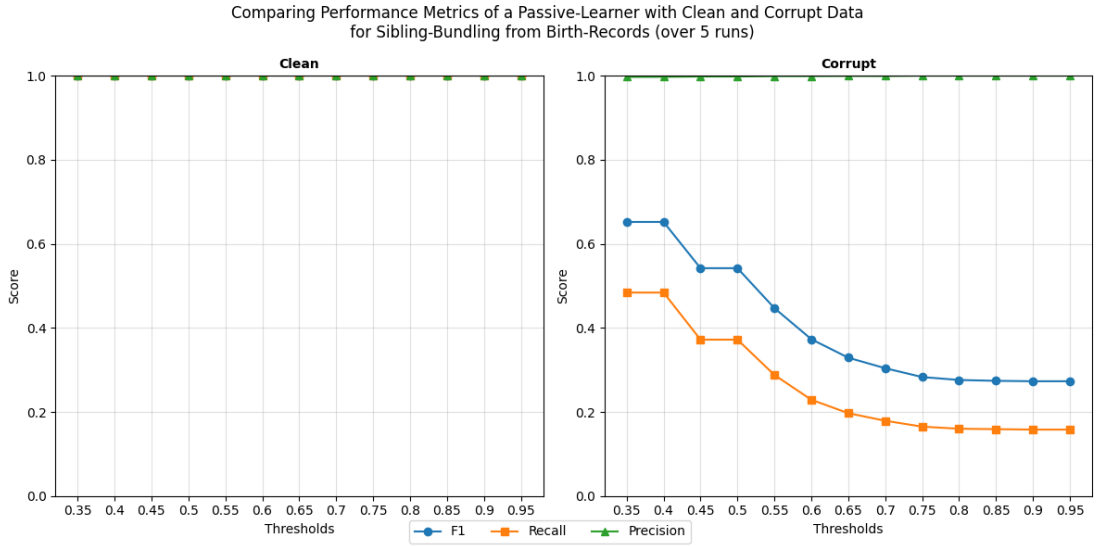
Unless stated otherwise, all subsequents results will use a combined evaluation, as the classifier-only method is misrepresentative of overall performance (however these are still included in Appendix I).

6.4.2. Passive Learning on Corrupted Data

In the absence of data corruption, clean synthetic data achieves near-perfect scores (F1 $=\sim 0.999$) due to the optimal performance of the blocking stage (see Section 6.3.1). These remain constant at all threshold levels, rendering the choice of threshold value largely insignificant for this data (although an impact may be seen in resource usage).



(a) Birth-death identity linkage.

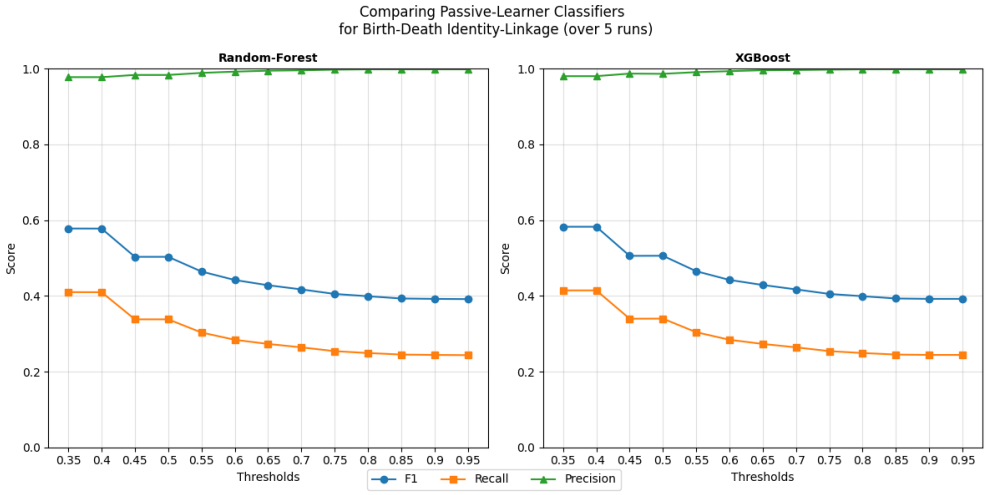


(b) Sibling bundling from birth records.

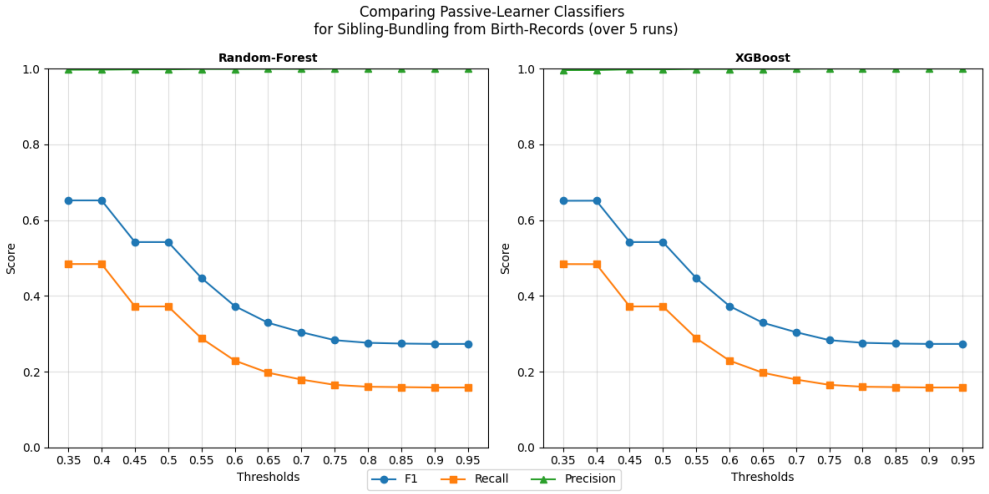
Figure 6.7.: Line graphs comparing performance of a passive learner, using a Random Forest classifier, on both clean and corrupted versions of the synthetic TD100k dataset.

6.4.3. Comparing Classifiers for Passive Learning

For both passive and active learning experiments, Random Forest (results shown in the previous sections) and XGBoost classifiers were configured using the default parameters, albeit with parallelisation enabled and a random seed set. As shown in Figure 6.8, the choice of classifier proved insignificant for PL performance, with both producing almost identical results.



(a) Birth-death identity linkage.

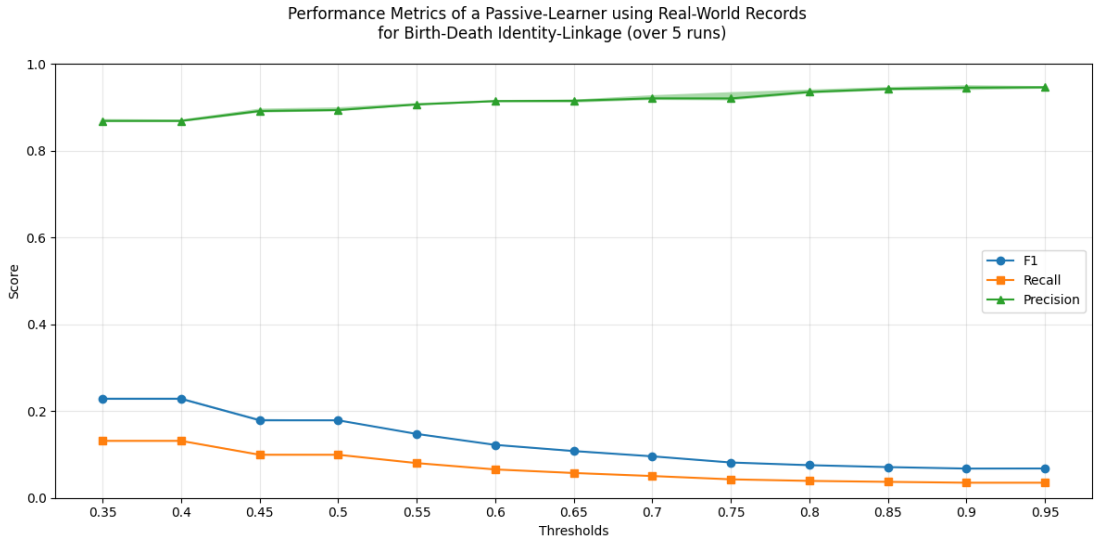


(b) Sibling bundling from birth records.

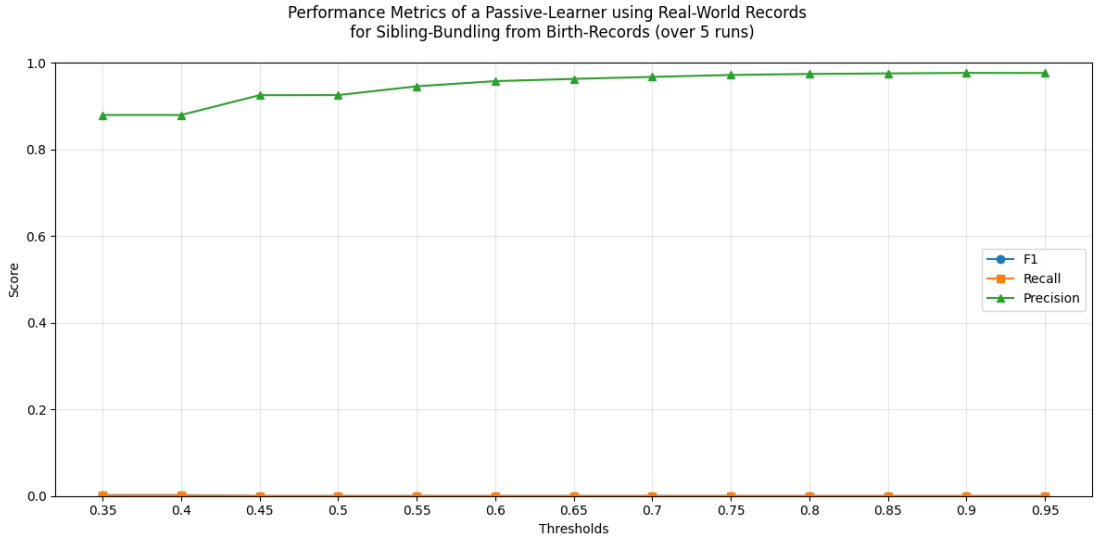
Figure 6.8.: Line graphs comparing performance of Random Forest and XGBoost classifiers, for passive learning on the synthetic TD100k (Corrupt) dataset.

6.4.4. Passive Learning with Real-World Data

As expected from the blocking results (see Section 6.3.3), the POPLINK dataset performed poorly in this experiment. For sibling bundling, the model can be considered unsuccessful, with F1 scores ≤ 0.002. An interesting result is recorded by Precision, which is lower than the score achieved with the corrupted synthetic data ($0.879 < 0.997$ for sibling-bundling, blocked at threshold 0.35). This is noteable as Precision is not directly impacted by the blocking stage (see Section 5.8), so this difference is a reflection of the data itself, rather than the results of earlier stages.



(a) Birth-death identity linkage.



(b) Sibling bundling from birth records.

Figure 6.9.: Line graphs plotting performance metrics of a passive learner, using a Random Forest classifier, on the real-world POPLINK dataset. Shaded area represents the range of values across 5 runs.

6.4.5. Identifying Significant Thresholds

From the above results, 0.35 and 0.5 were chosen as significant thresholds to explore in the active learning experiments. These values were selected due to being either side of a significant difference in performance scores, after which performance begin to gradually stabilise. Confusion matrixes for PL using these thresholds can be found in Appendix J.

6.5. Active Learning Experiments

To evaluate the active learning system, a similar experimental setup was used, with a 4:1 test-training data split and five random seeds (see Section 6.4).

For greater control, `ALWrapper` was used instead of `ALMultiWrapper` (see Section 5.7), to run each learner in an isolated experiment. The wrapper was configured to aim to label five positive and five negative links in the initial labelling process. During the training loop, 100 instances were queried per iteration, to account for the scale of the datasets. The "auto-stop" function (see Section 5.7.2) was enabled, and the window-size parameter increased from the default value of 15 to 50, to ensure that the F1 metric has properly stabilised.

Two blocking thresholds, 0.35 and 0.5, were chosen from the results of the PL experiments (see Section 6.4.5) to conduct experiments with. In addition to these and the different datasets (see Section 6.1), three other variables were explored:

- Classifier - Random Forest or XGBoost.
- Query Strategy - Uncertainty-based sampling or random sampling.
- Rebalance Initial Labels - True or False (see Section 5.7.1).

As before, significant results will be included in the subsequent sections, with full results found in Appendices H and K.

6.5.1. Initial Labelling

Using only an initial labelled subset, the model achieves performance reasonably close to that of the PL system, with F1 scores only ~ 0.05 lower. The values reported in Table 6.4 are means of up to ten runs (five random seeds across two query strategies), although random sampling results for identity linkage blocked at threshold 0.35 were excluded due to computational infeasibility (see Section 6.2). The exact numbers of failed runs for each experiment can be found in Appendix H.

Learner	Threshold	n labelled	Precision	Recall	F1
Passive	0.35	34,453,432	0.977	0.410	0.578
Active (Initial)	0.35	1,131	0.986	0.340	0.506
Passive	0.50	6,525,036	0.983	0.338	0.503
Active (Initial)	0.50	334	0.988	0.298	0.458

(a) Birth-death identity linkage.

Learner	Threshold	n labelled	Precision	Recall	F1
Passive	0.35	9,552,870	0.997	0.484	0.652
Active (Initial)	0.35	179	0.942	0.457	0.615
Passive	0.50	2,084,830	0.998	0.372	0.542
Active (Initial)	0.50	40	0.851	0.332	0.476

(b) Sibling bundling from birth records.

Table 6.4.: Tables comparing the performance scores for an active learner after initial labelling against a passive learner, both using a Random Forest classifier on the synthetic TD100k (Corrupt) dataset. Values are means over all completed runs.

Balanced vs. Unbalanced Initial Labelling

As described in Section 5.7.1, an optional parameter for the initial labelling function can be set to rebalance the number of positive and negative links, by dis-

carding surplus data. Due to the non-deterministic nature of the initial labelling process,there is no consistent effect of this setting. However, Table 6.5 indicates that results are often worse than their unbalanced counterparts, as the learner performs better with more initial data. Unless indicated otherwise, all AL results in subsequent sections will be for an unbalanced initial subset.

Rebalance?	Threshold	n labelled	Precision	Recall	F1
False	0.35	1,131	0.986	0.340	0.506
True	0.35	10	0.244	0.391	0.292
False	0.50	334	0.988	0.298	0.458
True	0.50	10	0.367	0.334	0.340

(a) Birth-death identity linkage.

Rebalance?	Threshold	n labelled	Precision	Recall	F1
False	0.35	179	0.942	0.457	0.615
True	0.35	10	0.728	0.463	0.541
False	0.50	40	0.851	0.332	0.476
True	0.50	10	0.765	0.331	0.458

(b) Sibling bundling from birth records.

Table 6.5.: Tables comparing performance scores for balanced and unbalanced initial labelled subsets, using a Random Forest classifier on the TD100k (Corrupt) dataset. Values are means over all completed runs.

Comparing Classifiers for Initial Results

Similar to the PL experiments (see Section 6.4.3), the choice of classifier had a largely insignificant effect on results with an initial unbalanced labelled subset, although Random Forest outperforms XGBoost by a very small margin.

Classifier	Threshold	n labelled	Precision	Recall	F1
Random Forest	0.35	1,131	0.986	0.340	0.506
XGBoost	0.35	1,131	0.941	0.318	0.471
Random Forest	0.50	334	0.988	0.298	0.458
XGBoost	0.50	334	0.897	0.304	0.453

(a) Birth-death identity linkage.

Classifier	Threshold	n labelled	Precision	Recall	F1
Random Forest	0.35	179	0.942	0.457	0.615
XGBoost	0.35	179	0.962	0.426	0.590
Random Forest	0.50	40	0.851	0.332	0.476
XGBoost	0.50	40	0.831	0.289	0.426

(b) Sibling bundling from birth records.

Table 6.6.: Tables comparing initial performance scores for Random Forest and XGBoost classifiers, on the synthetic TD100k (Corrupt) dataset. Values are means over all completed runs.

However, an interesting result was observed with rebalanced subsets. As seen in Table 6.7 (and Appendix H), the XGBoost classifier performs comparatively poorly with limited data.

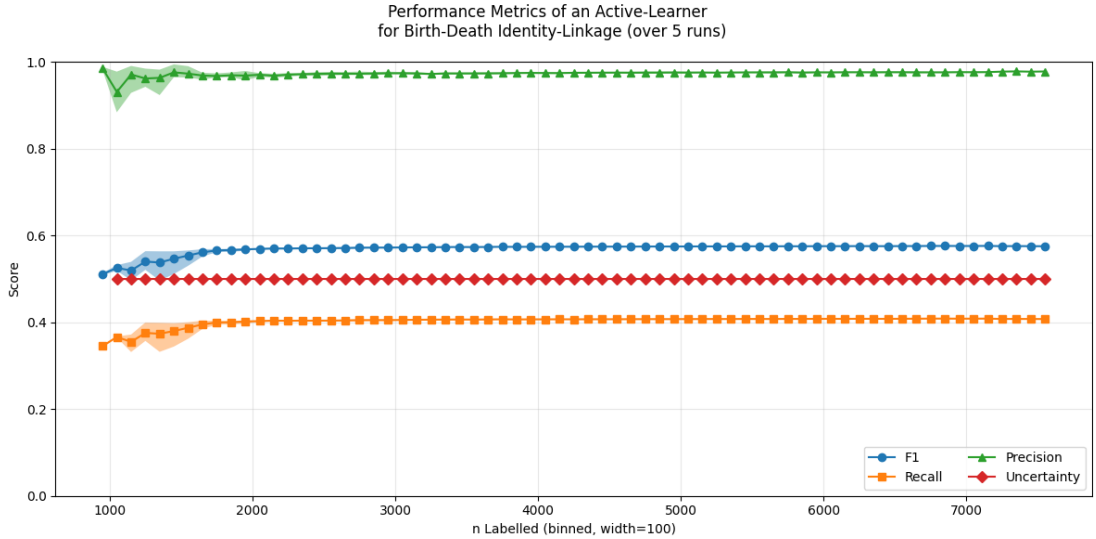
Classifier	Threshold	n labelled	Precision	Recall	F1
Random Forest	0.35	10	0.728	0.463	0.541
XGBoost	0.35	10	0.133	0.409	0.195

Table 6.7.: Table comparing initial performance scores with a balanced subset, for Random Forest and XGBoost classifiers, performing sibling bundling from birth records on the TD100k (Corrupt) dataset. Values are means over 10 runs.

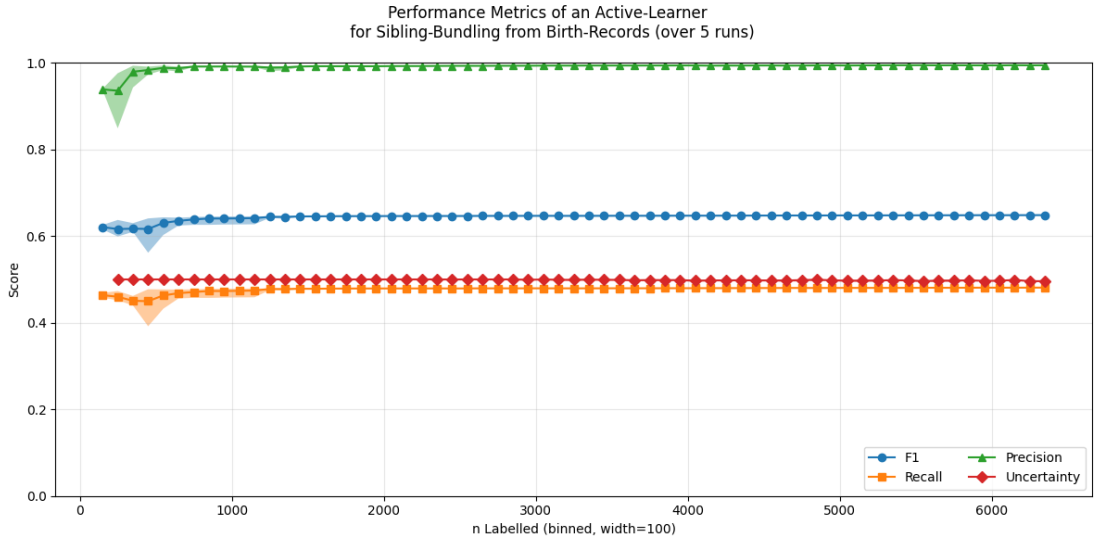
6.5.2. Active Learning Results

As observed in Figure 6.10, the AL system improves quickly from it's initial results, reaching a stable F1 value with only a fraction of the available data labelled (approximately 2,500 data points for birth-death identity linkage and 1,500 for sibling bundling from birth records). The active learner exhibits greater variability than the passive learner, with small fluctuations clearly visible across Precision, Recall and F1. In contrast, the uncertainty metric remains constant at the maximum value of 0.5 (see Section 5.8.2) for the duration of training. This indicates that even when performance has stabilised, there are still highly uncertain unlabelled pairs. However, this metric represents the mean prediction uncertainty across the 100 instances queried at each iteration, and thus the individual instances may still exhibit variation in uncertainty.

As both the size of initial labelled subset and the number of iterations required for stabilisation vary due to randomisation, there is no uniform x-axis scale across runs. To address this, the "n_labelled" values were grouped into bins of width=100 (unless stated otherwise), with each bin corresponding approximately to a single iteration. To prevent long-running outliers from misrepresenting results, bins represented in fewer than half of the completed runs were omitted from the plots.



(a) Birth-death identity linkage.



(b) Sibling bundling from birth records.

Figure 6.10.: Line graphs plotting performance against the number of data points labelled for an active learner, using a Random Forest classifier, on the TD100k (Corrupt) dataset blocked at threshold 0.35. Shaded areas represent the range of values across 5 runs.

While these experiments have been configured with an increased window-size for the “auto-stop” function, subsequent plots will truncate this window to focus

more closely on the phase where performance metrics fluctuate.

6.5.3. Passive vs. Active Learning

As visualised in Figure 6.11, both AL and PL systems stabilise at almost identical F1 values. Despite not offering superior performance, the active learner presents itself as a cheaper alternative to the passive learner, requiring only a fraction of the data to produce the same results (see Table 6.8).

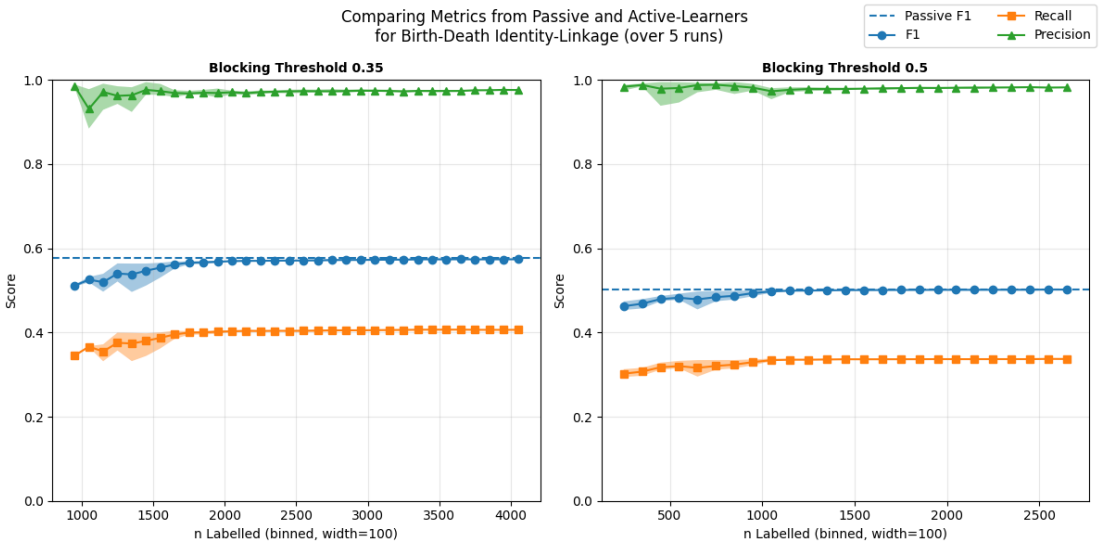
Learner	n labelled	F1
Passive	34,453,432	0.578
Active	7,331	0.576

(a) Birth-death identity linkage.

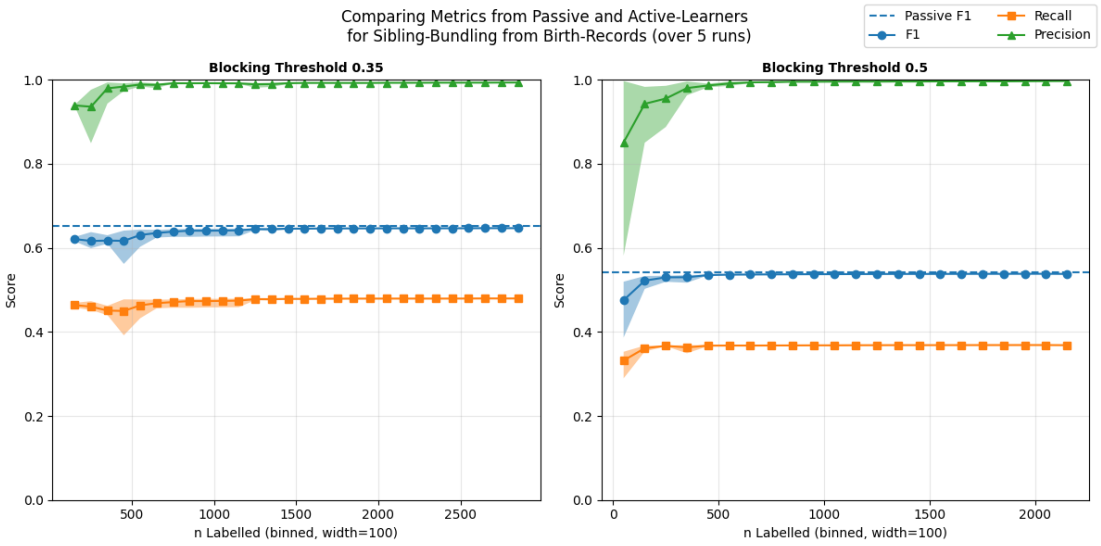
Learner	n labelled	F1
Passive	9,552,870	0.652
Active	6,059	0.648

(b) Sibling bundling from birth records.

Table 6.8.: Tables comparing the number of data points labelled by a passive learner and a stabilised active learner (with uncertainty-based sampling), both using Random Forest classifiers, on the TD100k (Corrupt) dataset at blocking threshold 0.35. Values are means over 5 runs.



(a) Birth-death identity linkage.



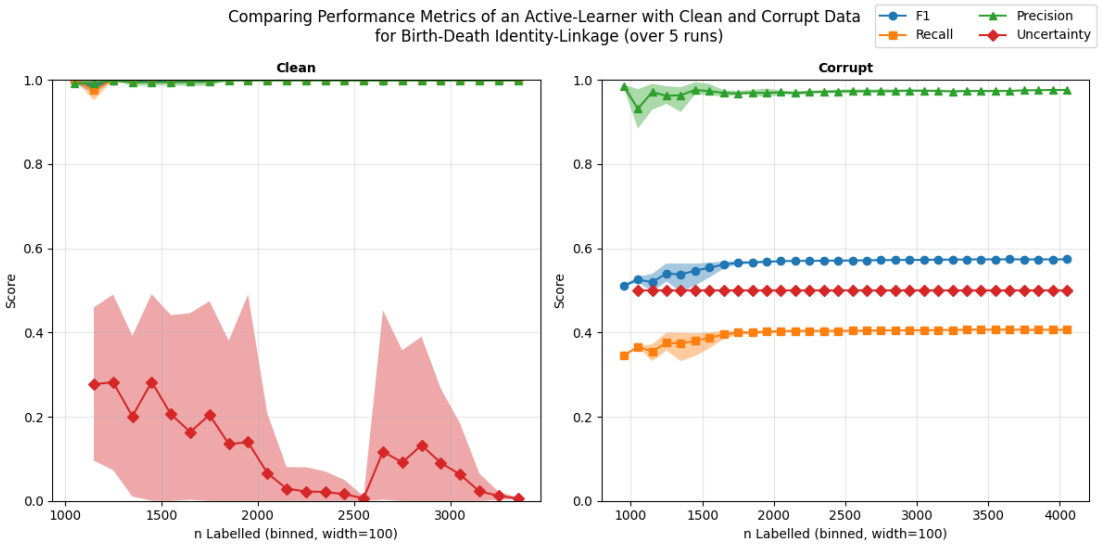
(b) Sibling bundling from birth records.

Figure 6.11.: Line graphs comparing performance for passive and active learners (with uncertainty-based sampling), both using Random Forest classifiers, on the TD100k (Corrupt) dataset blocked at threshold 0.35. Shaded areas represent the range of values across 5 runs.

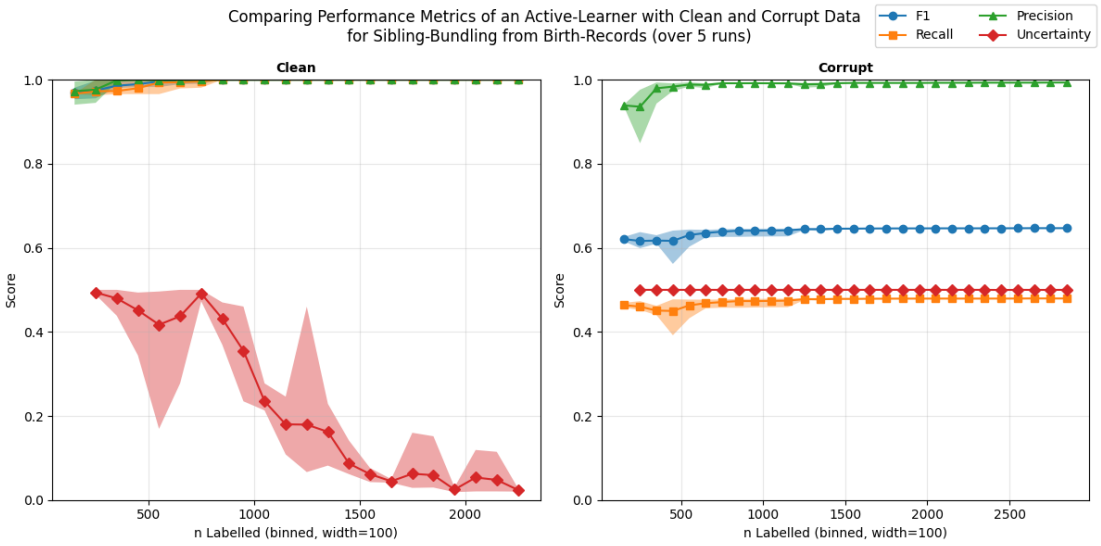
6.5.4. Active Learning on Clean and Corrupted Data

Owing to its optimal blocking results (see Section 6.3.1), the active learner achieves near perfect results on the clean synthetic data, with the F1 value stabilising at 0.999. Due to the less ambiguous data, the clean synthetic dataset requires significantly less labelled data than its corrupted counterpart, stabilising with approximately 1,642 and 714 pairs labelled for birth-death identity linkage and sibling bundling from birth records respectively.

As observed in Figure 6.12, the prediction uncertainty measure exhibits significantly more dynamic and variable behaviour with the clean synthetic data. Aside from occasional spikes, uncertainty decreases gradually, converging towards values close to zero by the end of training.



(a) Birth-death identity linkage.



(b) Sibling bundling from birth records.

Figure 6.12.: Line graphs comparing performance metrics of an active learner, using a Random Forest classifier with uncertainty-based sampling, on clean and corrupted versions of the synthetic TD100k dataset blocked at threshold 0.35. Shaded areas represent the range of values across 5 runs.

6.5.5. Random Sampling

Substituting uncertainty-based sampling for the less methodical approach of random sampling produces an immediate effect, with experiments taking considerably longer to stabilise, often resulting in computational infeasibility (see Section 6.2). As reported in Table 6.9, the random sampling learner still achieves performance comparable to that of uncertainty-based sampling but requires significantly more data to do so.

Learner	n labelled	Precision	Recall	F1
Passive	9,552,870	0.997	0.484	0.652
Uncertainty Sampling	6,059	0.995	0.481	0.648
Random Sampling	16,099	0.985	0.475	0.641

Table 6.9.: Table comparing performance and the number of data points required to stabilise for different learners, using Random Forest classifiers, for sibling bundling from birth records on the synthetic TD100k (Corrupt) dataset blocked at threshold 0.35. Values are means over 5 runs.

In Figure 6.13, Precision exhibits a more gradual incline for random sampling with greater variability visible throughout. The mean prediction uncertainty remains extremely low over the course of training, indicating that this strategy is ineffective, as it frequently queries instances for which the model is already confident in it's predictions. The x-axis scale is also much greater for random sampling, with the bin-width adjusted to 400 to fit the plot.

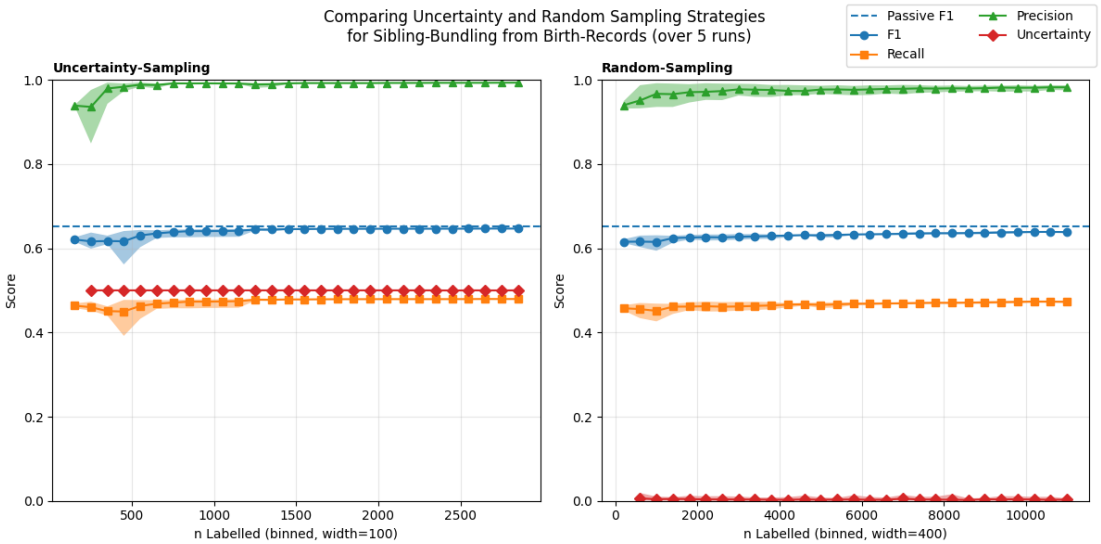


Figure 6.13.: Line graphs comparing performance of uncertainty-based and random sampling query strategies, both using Random Forest classifiers, for sibling bundling from birth records on the synthetic TD100k (Corrupt) dataset at blocking threshold 0.35. Shaded area represents the range of values across 5 runs.

A more significant impact is exhibited for birth-death identity linkage. Likely due to the sparsity of positive links, random sampling sees an increase of 5-10 times for the number of data points required to stabilise (see Table 6.10). As observed in Figure 6.14, none of the metrics exhibit significant improvement over the lengthened duration of training. Furthermore, this increase rendered all random sampling experiments at threshold 0.35 for birth-death identity linkage with both the TD100k (Corrupt) and POPLINK datasets computationally infeasible (see Section 6.2).

Learner	n labelled	Precision	Recall	F1
Passive	6,525,036	0.983	0.338	0.503
Active (Initial)	334	0.988	0.298	0.458
Uncertainty Sampling	6,094	0.984	0.338	0.503
Random Sampling	26,254	0.984	0.321	0.485

Table 6.10.: Table comparing performance and the number of data points required to stabilise for different learners, all using Random Forest classifiers, for birth-death identity linkage on the TD100k (Corrupt) dataset blocked at threshold 0.5. Values are means over 5 runs.

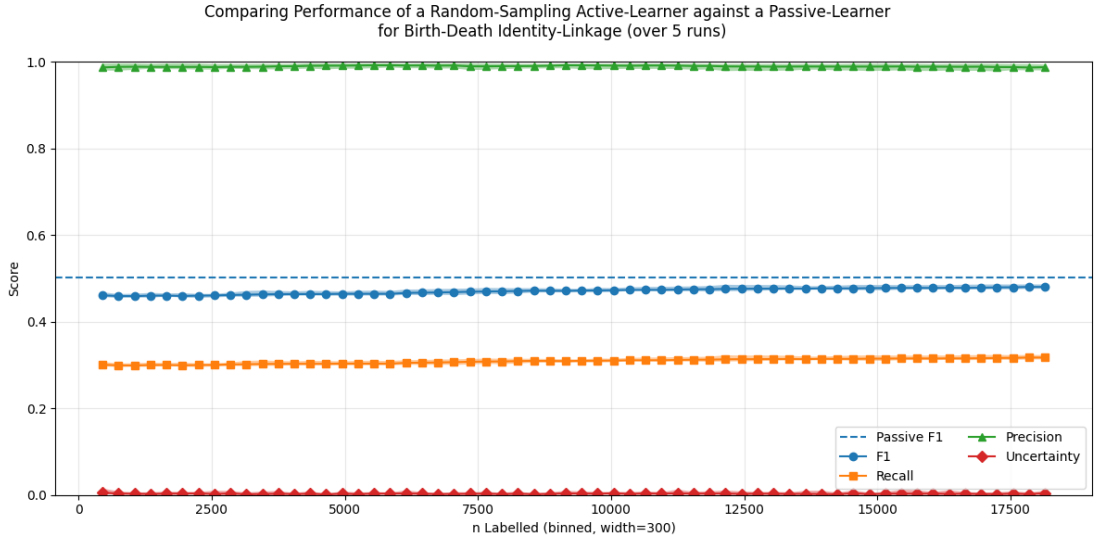
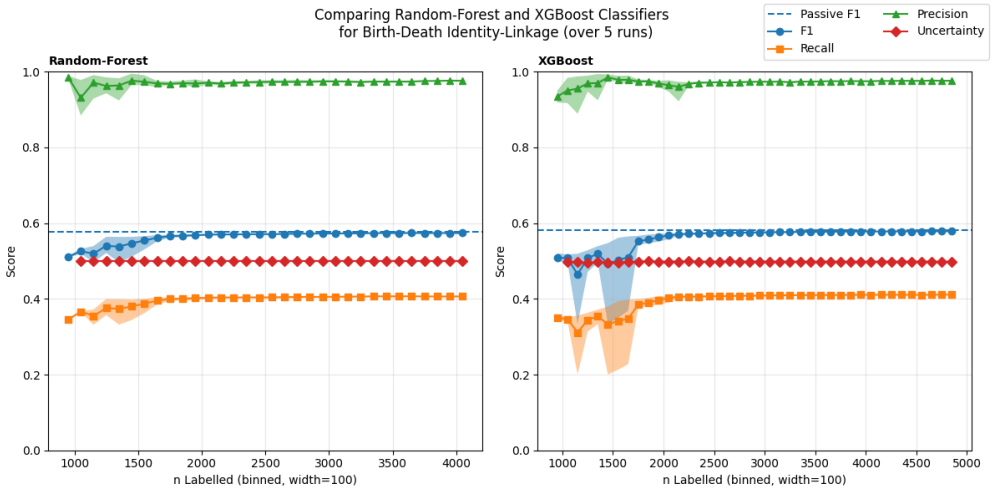


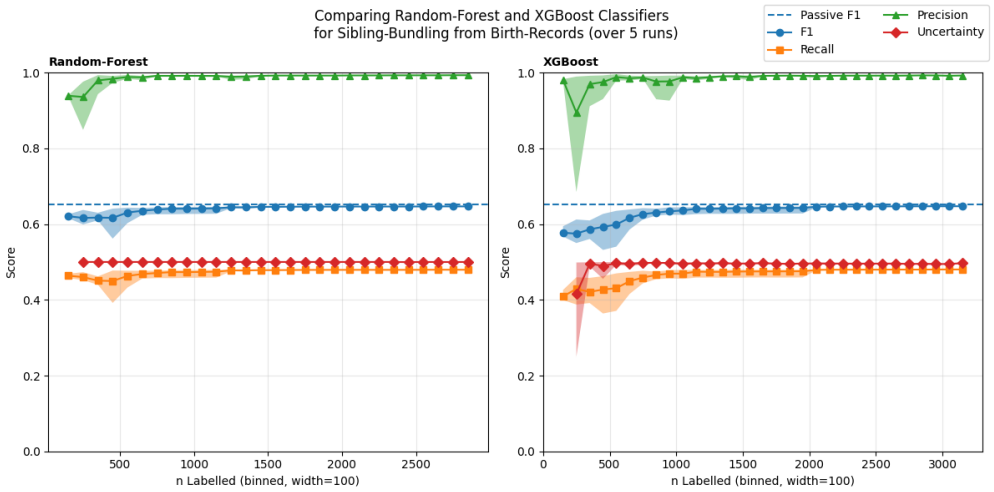
Figure 6.14.: Line graph plotting the performance metrics of a random sampling AL system, using a Random Forest classifier, for birth-death identity linkage on the synthetic TD100k (Corrupt) dataset with blocking threshold 0.5. Shaded areas represent the range of values across 5 runs.

6.5.6. Comparing Classifiers for Active Learning

Similar to both the PL experiments and initial results for AL (see Sections 6.4.3 and 6.5.1), the choice of classifier proved inconsequential for the final results. However, as observed in Figure 6.15, XGBoost is more sensitive to random variations and generally requires more data to stabilise (see Table 6.11).



(a) Birth-death identity linkage.



(b) Sibling bundling from birth records.

Figure 6.15.: Line graphs comparing AL performance using Random Forest and XGBoost classifiers, both with uncertainty-based sampling, on the synthetic TD100k (Corrupt) dataset blocked at threshold 0.35. Shaded areas represent the range of values across 5 runs.

Learner	n labelled	F1
Random Forest	7,331	0.576
XGBoost	8,221	0.580

(a) Birth-death identity linkage.

Learner	n labelled	F1
Random Forest	6,059	0.648
XGBoost	6,719	0.65

(b) Sibling bundling from birth records.

Table 6.11.: Tables comparing the number of data points required for Random Forest and XGBoost classifiers to stabilise, both using uncertainty-based sampling, on the synthetic TD100k (Corrupt) dataset at blocking threshold 0.35. Values are means over 5 runs.

6.5.7. Initial Labelling Retrospective

Although early results differ (see Section 6.5.1), balancing the initial subset has little impact on the final performance of the AL system. In some instances, learners trained from a balanced subset can stabilise slightly more quickly but this is not always the case (as exemplified in 6.12).

Balanced	n labelled	F1
False	7,331	0.576
True	6,670	0.576

(a) Birth-death identity linkage.

Balanced	n labelled	F1
False	6,059	0.648
True	6,570	0.649

(b) Sibling bundling from birth records.

Table 6.12.: Tables comparing the number of data points required to reach stable AL performance when initialised with balanced and unbalanced subsets, using a Random Forest classifier with uncertainty-based sampling, on the TD100k (Corrupt) dataset at blocking threshold 0.35. Values are means over 5 runs.

6.5.8. Active Learning with Real-World Data

As observed in the PL experiments, performance with the real-world POPLINK dataset is severely limited by ineffective blocking (see Sections 6.3.3 and 6.4.4), rendering sibling bundling non-functional (results included in Appendices H and K). While this could be addressed using more careful selection of blocking

columns (see Section 5.5.1), it is still unlikely that real-world data would perform as well as the synthetic data due to ambiguities and human-error present.

In Figure 6.16, plotting identity linkage with the POPLINK data, the learner exhibits a high degree of variability throughout training, particularly in the Precision metric. A unique result is observed in the F1 value, which surpasses the benchmark from PL on two occasions for up to seven iterations before declining again. The system eventually stabilises at a lower value, just below the benchmark, consistent with that observed with the synthetic datasets.

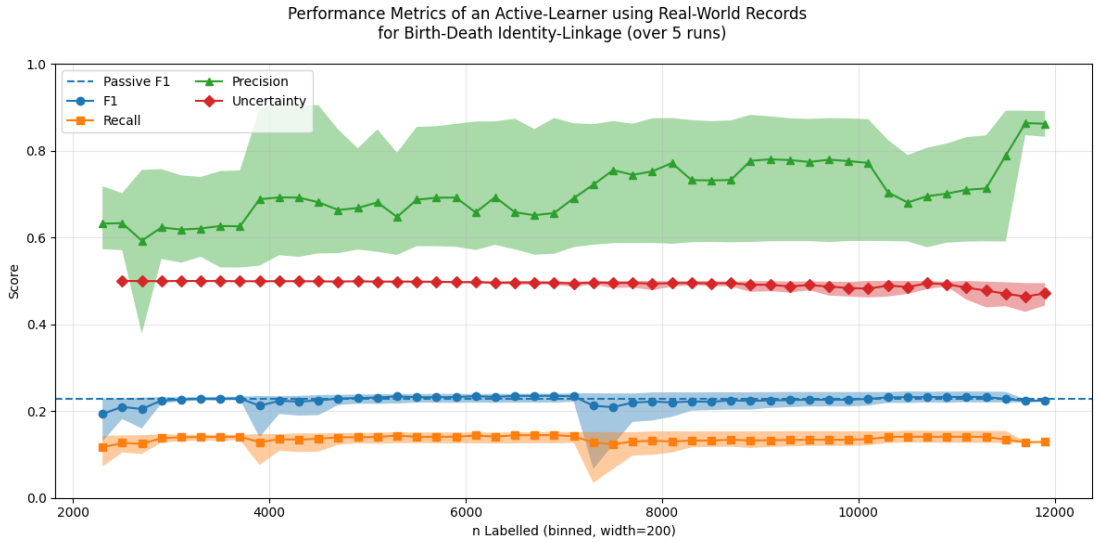


Figure 6.16.: Line graph plotting the performance metrics of an active learner, using a Random Forest classifier with uncertainty-based sampling, for birth-death identity linkage on the real-world POPLINK data with blocking threshold 0.35. Shaded areas represent the range of values across 5 runs.

7. Evaluation and Critical Appraisal

The following chapter aims to evaluate the implemented system, and the experimental results, against the objectives set out in Section 1.1.

7.1. Evaluating against Primary Objectives

The implemented Active Learning (AL) solution fulfils all the primary objectives, addressing both linkage types: birth-death identity linkage and sibling bundling from birth records. The system is relatively efficient, both in making use of available hardware resources to reduce computational load through parallelism, and in minimising the volume of training data required via uncertainty-based sampling. As evidenced in Table 7.1, the system generally achieved stable performance with $\leq 0.1\%$ of the training data labelled. Furthermore, the random sampling learner consistently failed to finish across all five runs in this example (see Section 6.2), whereas the uncertainty-based learner required just over 7,000 labelled pairs, out of a total training set of 34,000,000+.

		n Labelled	
Linkage Type	**PL (all training data)**	**AL (random sampling)**	**AL (uncertainty-based)**
Birth-Death Identity Linkage	34,453,432	Timeout	7,331
Sibling Bundling from Birth Records	9,552,870	16,099	6,059

Table 7.1.: Tables comparing the number of data points required to stabilise for passive learning, random sampling AL and uncertainty-based AL; each using a Random Forest classifier on the synthetic TD100k (Corrupt) dataset, blocked at a threshold value of 0.35. Values are means over 5 runs.

The AL system was evaluated using both clean and corrupted versions of the same synthetic data. When applied to the clean data, the blocking stage achieved

perfect results, retaining all positive links regardless of the threshold value set. As a result, the learner quickly reached near-perfect performance using this data, with F1 scores converging on 0.999. In contrast, performance on the corrupted dataset was significantly worse. Setting a blocking theshold value of 0.35 (the lowest threshold that could be evaluated with the available resources), the F1 value stabilised at just 0.576. While this result might initially suggest that the learner itself performs poorly with corrupted data, this is not the case.

As seen in Table 7.2, when the AL classifier is evaluated in isolation, without considering results of the blocking stage, the observed performance difference between the two datasets is significantly reduced. This proves that the poor performance is primarily attributable to suboptimal blocking configuration. The blocking columns used were selected from a basic analysis of patterns in the clean synthetic data (see Section 5.5.1) and this same configuration was applied to all three datasets. Thus, it is likely that improved performance could be achieved through configuring blocking more carefully, with columns tailored individually to each dataset.

Dataset	**F1 (classifier)**	**F1 (combined)**
Clean	0.999	0.999
Corrupt	0.964	0.576

(a) Birth-Death Identity linkage.

Dataset	**F1 (classifier)**	**F1 (combined)**
Clean	1.000	1.000
Corrupt	0.989	0.503

(b) Sibling Bundling from Birth Records.

Table 7.2.: Tables comparing the F1 scores from classifier-only and combined evaluation methods, for active-learners using Random Forest classifier with uncertainty-based sampling. Values are means over 5 runs and both the clean and corrupted synthetic datasets are blocked using a threshold value of 0.35.

7.2. Evaluating against Revised Secondary and Tertiary Objectives

Almost all of the revised secondary and tertiary objectives were met, with the exception of implementing sibling bundling from death records, which was not completed due to time-constraints.

7.2.1. Comparing Passive and Active Learning

The experiments described in Section 6.5.3 concluded that, when using an uncertainty-based query strategy, the AL system consistently achieved performance comparable to, although not exceeding, that of Passive Learning (PL). Fewer than 0.01% of the candidate pairs in the training set required labelling to reach this level of performance, confirming that AL is a more cost-effective approach to solving population linkage tasks.

7.2.2. Exploring Different Configurations

Several configurations of the AL system were explored, evaluating different combinations of classifier, query strategy and initial labelled subset. The most notable differences were observed when using random sampling. While final performance was only slightly lower than that of uncertainty-based query strategies, significantly more labelled instances were required to reach this level. Random sampling experiments were also prone to timeout when performing identity linkage at low blocking thresholds (see Section 6.2).

The choice of classifier proved less influential, with both Random Forest and XGBoost exhibiting similar overall performance. However, XGBoost often required a larger volume of labelled data to stabilise. The nature of the labelled subset used to initialise the system (balanced or unbalanced) was found to be inconsequential to final performance.

7.2.3. Evaluating Performance with Real-World Data

As discussed in the evaluation of primary objectives, the main limitation of this system is the design of the blocking stage. As no clean version of the real-world POPLINK data is available to analyse, the blocking setup devised for the synthetic datasets was reused instead of a bespoke configuration (see Section 5.5.1).

This is particularly evident in the POPLINK results where, for birth-death identity linkage, over $1/3$ of positive links were lost during blocking (using threshold values ≥ 0.35). This severely limits performance for the AL system, although it was still able to match the passive learner, with an F1 score of 0.229 (at a blocking threshold of 0.35). A more severe impact was observed for sibling bundling from birth records, where 99.5% of positive links were lost, rendering both learning systems entirely ineffective.

7.2.4. Evaluating the Stopping Condition

An automatic stopping condition was successfully implemented to exit training early once the F1 measure has remained constant for a predefined number of iterations. While the default configuration uses a relatively small number of iterations, the experimental results in Section 6.5.2, which were configured with a larger window of 50 iterations rather than 15, indicate that the smaller window size was sufficient for determining stability.

It is important to note that this condition detects general stability in the F1 score, rather than specifically a lack of improvement. An interesting edge-case result is observed for birth-death identity linkage with the POPLINK dataset (see Figure 6.16), where AL performance temporarily surpasses the PL benchmark before stabilising at a lower value. This exposes a potential flaw in the condition, presenting an argument for a more sophisticated mechanism. This hypothetical solution could track maximum performance attained during training and roll-back to this state if subsequent iterations degrade performance, potentially improving the end results.

7.3. Comparison of Related Works

7.3.1. Splink

The open-source Splink project implements a different approach to record-linkage through using the Felligi Sunter model, a rule-based system which labels pairs based on the probability that values do or do not match for a positive link. A subsequent extension leveraged AL algorithms to link two medical datasets, each containing approximately 5,000 records. The AL `splink` model was able to perform this task with just 29 false-positives, having labelled only 126 pairs. Comparatively, the solution implemented in this project is less effective and generally requires significantly more data to reach stable performance.

7.3.2. Enhancing Demographic Linkage with Active Learning in Machine Learning

The research presented in “Enhancing Demographic Linkage...” sets out similar aims to those of this project, applying active learning to the problem of birth-

death identity linkage. A key difference lies in the evaluation methodology used. In the aforementioned paper, the classifier is evaluated in isolation, using relatively high blocking thresholds (≥ 0.65) and a fixed budget of 100 queries, each for a single instance. This results in consistently high F1 scores ($0.9+$), regardless of the query strategy applied or dataset used.

In contrast, this project implements an end-to-end evaluation, assessing the combined performance of the entire system. This approach concluded that, unless perfect blocking is achieved, such high performance scores are unlikely to be achieved in practice.

The related work also explores hyperparameter tuning through Bayesian optimisation to enhance the performance of the Random Forest classifier. By comparison, this project instead prioritises experimental reproducibility through using the default parameters for both Random Forest and XGBoost classifiers.

8. Conclusions

This project set out with the aim to explore the potential use of Active Learning (AL) algorithms for solving population linkage problems. While the original end goal was to apply the system on real unlinked populations data, using the Scottish Historic Population Platform (SHiPP) dataset, the initial objectives of this project were not met in their entirety. In a manner somewhat fitting for a study on active learning, the aims of the project instead evolved in response to its findings, adapting to explore new questions as they presented themselves.

The implemented system addresses two basic forms of linkage: birth-death identity linkage and sibling bundling from birth records. By leveraging an efficient learning process which pairs uncertainty-based queries with an automatic stopping criterion, the system proved just as effective as Passive Learning (PL) alternatives, while using less than 0.1% of the available training data. When evaluated with a clean synthetic dataset, the learner achieved near-perfect performance, with F1 scores converging on 0.999. Lower performance (~ 0.5) was observed with corrupted synthetic data, primarily due to a poorly configured Locality-Sensitive Hashing (LSH) MinHash blocking stage.

The research evaluated multiple learner configurations and found that the Random Forest classifier was more efficient than XGBoost, requiring fewer labelled instances for comparable performance. The results also concluded that replacing uncertainty-based querying with random sampling was highly ineffective and, in many cases, computationally infeasible.

Any future development should primarily focus on improving blocking functionality. This could be achieved through in-depth data-analysis or even by developing a companion system capable of evaluating and selecting optimal blocking and comparison fields from a small subset of labelled data. There is also scope for a more adaptive approach, in which blocking thresholds are dynamically adjusted throughout training based on observed performance. This would however require careful handling to ensure pairs selected for the training set under one blocking threshold do not later appear in the test set under another.

At present, no consistency can be enforced between the two linkage types implemented in this project. Therefore, future work could also explore additional linkage types and introducing consistency constraints between them, enabling evaluation of the system's overall linkage quality rather than assessing several linkage tasks in isolation.

In conclusion, this project does not present a definitive solution to applying active learning algorithms for population linkage, nor does it conclude that a single optimal solution exists. Instead, it serves as a proof-of-concept, highlighting the potential of active learning as a cost-effective approach to data linkage. The insights gained from this work provide a foundation for future research and development, paving the way toward more refined and robust systems that may one day solve real-world unlinked populations data.

References

[1] Alexandre Abraham. *A Proactive Look at Active Learning Packages*. Accessed: 2025-09-26. July 2020. URL: https://blog.dataiku.com/a-proactive-look-at-active-learning-packages.

[2] ALiPy developers. *Introduction to Active Learning*. Accessed: 2025-09-30. 2025. URL: https://parnec.nuaa.edu.cn/_upload/tpl/02/db/731/template731/pages/huangsj/alipy/intro2al.html.

[3] Ahmad Alsadeeqi, Tom Dalton and James Ross. *Crptr-fork*. 2025. URL: https://github.com/stacs-srg/crptr-fork/.

[4] W. Brian Arthur. 'The analysis of linkages in demographic theory'. In: *Demography* 21.1 (Feb. 1984), pp. 109–128. ISSN: 0070-3370. DOI: 10.2307/2061031. eprint: https://read.dukeupress.edu/demography/article-pdf/21/1/109/905667/109arthur.pdf. URL: https://doi.org/10.2307/2061031.

[5] Parmida Atighehchian et al. *Baal, a Bayesian Active Learning Library*. 2022. URL: https://github.com/baal-org/baal/.

[6] Tianqi Chen and Carlos Guestrin. 'XGBoost: A Scalable Tree Boosting System'. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '16. San Francisco, California, USA: ACM, 2016, pp. 785–794. ISBN: 978-1-4503-4232-2. DOI: 10.1145/2939672.2939785. URL: http://doi.acm.org/10.1145/2939672.2939785.

[7] Peter Christen. 'Classification'. In: *Data Matching: Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 129–162. ISBN: 978-3-642-31164-2. DOI: 10.1007/978-3-642-31164-2_6. URL: https://doi.org/10.1007/978-3-642-31164-2_6.

[8] Peter Christen. 'Evaluation of Matching Quality and Complexity'. In: *Data Matching: Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 163–184. ISBN: 978-3-642-31164-2. DOI: 10.1007/978-3-642-31164-2_7. URL: https://doi.org/10.1007/978-3-642-31164-2_7.

[9] Peter Christen. 'Front Matter'. In: *Data Matching: Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. i–xix. ISBN: 978-3-642-31164-2. DOI: 10.1007/978-3-642-31164-2_6. URL: https://doi.org/10.1007/978-3-642-31164-2.

[10] Peter Christen. 'Introduction'. In: *Data Matching: Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 3–22. ISBN: 978-3-642-31164-2. DOI: 10.1007/978-3-642-31164-2_1. URL: https://doi.org/10.1007/978-3-642-31164-2_1.

[11] Peter Christen, Tim Churches and Markus Hegland. 'Febrl - A Parallel Open Source Data Linkage System'. In: vol. 3056. Apr. 2004, pp. 638–647. ISBN: 978-3-540-22064-0. DOI: 10.1007/978-3-540-24775-3_75.

[12] Thomas Stanley Dalton. 'Evaluating data linkage algorithms with perfect synthetic ground truth'. PhD thesis. University of St Andrews, June 2022. DOI: 10.17630/sta/247. URL: https://research-repository.st-andrews.ac.uk/handle/10023/26784.

[13] Tom Dalton, Alan Dearle, Graham Kirby et al. *Valipop*. 2025. URL: https://github.com/stacs-srg/valipop.

[14] Tivadar Danka and Peter Horvath. *modAL: A modular active learning framework for Python*. Available on arXiv at https://arxiv.org/abs/1805.00979. URL: https://github.com/cosmic-cortex/modAL.

[15] Demographic Data Base. *POPLINK Database*. Dataset. U18012. 2018.

[16] IBM. *What is classification in machine learning?* Accessed: 2025-09-30. 2025. URL: https://www.ibm.com/think/topics/classification-machine-learning.

[17] IBM. *What is machine learning?* Accessed: 2025-09-30. 2025. URL: https://www.ibm.com/think/topics/machine-learning.

[18] IBM. *What is random forest?* Accessed: 2025-09-30. 2025. URL: https://www.ibm.com/think/topics/random-forest.

[19] IBM. *What is XGBoost?* Accessed: 2025-10-01. 2025. URL: hhttps://www.ibm.com/think/topics/xgboost.

[20] Robin Linacre et al. 'Splink: Free software for probabilistic record linkage at scale.' In: *International Journal of Population Data Science* 7.3 (Aug. 2022). Available at https://ijpds.org/article/view/1794. DOI: 10.23889/ijpds.v7i3.1794. URL: https://github.com/moj-analytical-services/splink.

[21] Natthapong Lueangphumphitthaya. 'Enhancing Demographic Linkage with Active Learning in Machine Learning'. Not publicly available. Master's dissertation. University of St Andrews, July 2025. URL: https://github.com/natthapong2100/demographic-linkage-active-learning.

[22] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[23] Marko Miletic and Murat Sariyar. 'Implementing Informative-Based Active Learning in Biomedical Record Linkage for the Splink Package in Python'. In: *Studies in health technology and informatics* 305 (June 2023), pp. 509–512. DOI: 10.3233/SHTI230545.

[24] F. Pedregosa et al. 'Scikit-learn: Machine Learning in Python'. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[25] SCADR and University of Edinburgh. *Scottish Historic Population Platform (SHiPP)*. Accessed: 2025-09-27. 2025. URL: https://www.scadr.ac.uk/our-research/shipp.

[26] Christopher Schröder et al. 'Small-Text: Active Learning for Text Classification in Python'. In: *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*. Dubrovnik, Croatia: Association for Computational Linguistics, May 2023, pp. 84–95. URL: https://aclanthology.org/2023.eacl-demo.11.

[27] Scikit-Learn developers. *Label Propagation digits: Active learning*. Accessed: 2025-09-29. 2025. URL: https://scikit-learn.org/stable/auto_examples/semi_supervised/plot_label_propagation_digits_active_learning.html.

[28] Burr Settles. *Active Learning Literature Survey*. Tech. rep. 2009. URL: http://digital.library.wisc.edu/1793/60660.

[29] Ying-Peng Tang, Guo-Xiang Li and Sheng-Jun Huang. *ALiPy: Active Learning in Python*. Tech. rep. available as arXiv preprint https://arxiv.org/abs/1901.03802. Nanjing University of Aeronautics and Astronautics, Jan. 2019. URL: https://github.com/NUAA-AL/ALiPy.

[30] Ole Tange. *GNU Parallel 20251122 ('Mamdani')*. GNU Parallel is a general parallelizer to run multiple serial command line programs in parallel without changing them. Nov. 2025. DOI: 10.5281/zenodo.17692695. URL: https://doi.org/10.5281/zenodo.17692695.

[31] Anders Westberg, Erik Engberg and Sören Edvinsson. 'A Unique Source for Innovative Longitudinal Research: The POPLINK Database'. In: *Historical Life Course Studies* 3 (2016), pp. 20–31. DOI: 10.51964/hlcs9351.

[32] Yao-Yuan Yang et al. *libact: Pool-based Active Learning in Python*. Tech. rep. available as arXiv preprint https://arxiv.org/abs/1710.00379. National Taiwan University, Oct. 2017. URL: https://github.com/ntucllab/libact.

A. Source Code Repository

The full source code for this project (including the `al_linker` package, example Jupyter Notebooks for different linkage types, and all other experiments run) is available in the following GitHub repository:

https://github.com/jamesross03/al-population-linkage

B. Ethics Approval Form

School of Computer Science
Student Project Ethics Form

Title of project

Applying Active-Learning Algorithms for Population Linkage

Name of student

James Ross

Name of supervisor

Özgür Akgün

To be completed by student

I confirm that:

(select one of the following options)

[1] ☐ My **intended project** raises no ethical issues: I have agreed with **all** of the statements on the following page, in relation to my full project.

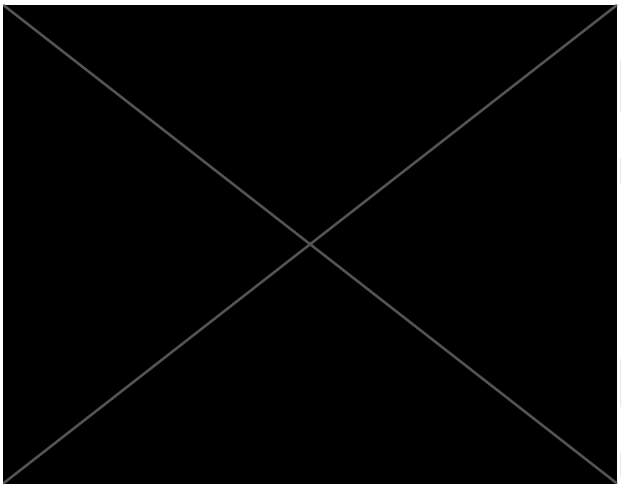
[2] ☐ My **intended project** is covered by the Computer Science generic artifact evaluation ethical application (approval code CS15727).

[3] ☐ My **fall-back project** raises no ethical issues: I have agreed with **all** of the statements on the following page, in relation to my fall-back project. I will submit a full ethics application for my intended project.

[4] ☐ My **fall-back project** raises no ethical issues: I have agreed with **all** of the statements on the following page, in relation to my fall-back project. My supervisor will submit an ethics amendment application for my intended project.

[5] ☐ My **fall-back project** is covered by approval CS15727. I will submit a full ethics application for my intended project.

[6] ☐ My **fall-back project** is covered by approval CS15727. My supervisor will submit an ethics amendment application for my intended project.



Delete this page or leave blank if you are relying on the generic artifact evaluation approval

☐ I confirm that **all** of the following statements are correct, in relation to:

(select <u>one</u> of the following options)

☐ my intended project

☐ my fall-back project

- My project does **not** involve interacting with human subjects.
- My project does **not** involve collecting personal data on human subjects.
- My project does **not** have potential adverse consequences for human welfare and wellbeing.
- My project does **not** involve use of secondary data in a way that requires ethical review according to the University's guidance on secondary data.

> *Consider:*
>
> - *Will you survey, observe or interview human subjects?*
> - *Could your project have a significant negative effect on people in the study area?*
>
> *If your project involves secondary datasets, please list them with links in your DOER.*

- My project does **not** present any foreseeable risks to me or to any participants.

> *Consider:*
>
> - *Is there any potential for physical harm for anyone involved in the project?*
> - *Is there any potential for psychological harm, discomfort or stress for anyone involved in the project?*

- There are **no** potential conflicts of interest.

> *Consider:*
>
> - *Might research objectivity be compromised by sponsorship?*
> - *Might any issues of intellectual property or roles in research be raised?*

- My project is **not** funded externally, **or** the external funder appears on the Ethical Funder List on the UTREC website, **or** I have received ethical approval of the funder.

> *A project **cannot** commence with an unapproved external funder.*

- My project does **not** involve the use of living animals.

> *A fall-back project **cannot** involve the use of living animals. If your proposed full project involves the use of living animals, a proposal must be referred to the University's Animal Welfare and Ethics Committee (AWEC).*

C. Valipop Configuration

The following configuration file was used with `valipop` [13] to generate the synthetic populations dataset, *TD100k*, used for the experiments detailed in Chapter 6.

```
# Properties File

var_data_files = src/main/resources/valipop/inputs/synthetic-
    scotland
tS = 1687-01-01
t0 = 1855-01-01
tE = 1973-01-01
t0_pop_size = 100000

output_record_format = TD
output_tables = true

results_save_location = results
run_purpose = TD100k
```

Listing C.1: The Valipop configuration file used to generate *TD100k*.

For generating the smaller synthetic dataset, *TD5k* used in Section 6.3.4, a similar configuration was used (adapted to use a significantly lower starting population).

```
#Properties File

var_data_files = src/main/resources/valipop/inputs/synthetic-
    scotland
tS = 1687-01-01
t0 = 1855-01-01
tE = 1973-01-01
t0_pop_size = 5000

output_record_format = TD
output_tables = true

results_save_location = results
run_purpose = TD_5k
```

Listing C.2: The Valipop configuration file used to generate *TD5k*.

D. Crptr Configuration

The following values were set in `populations_crptr/config.py` to generate the corrupted version of the synthetic populations dataset, *TD100k*, using `Crptr`. This configuration uses profile A, designed to simulate Optical Character Recognition (OCR) errors (see Appendix E. [3].

```
#!/usr/bin/python
# Author: James Ross (jar35@st-andrews.ac.uk)
# 24/06/2025
#
# Config object for population_corruptor.py example population
    corruptor.

from populations_crptr.example_profiles.profile_a import
    CorruptionProfileA
from populations_crptr.example_profiles.profile_b import
    CorruptionProfileB
from populations_crptr.example_profiles.profile_c import
    CorruptionProfileC
import populations_crptr.example_corruptors.standard_corruptors_td
     as StandardCorruptorsTD
import populations_crptr.example_corruptors.ocr_corruptors_td as
    OcrCorruptorsTD

class Config:
    PROFILE = CorruptionProfileA
    CORRUPTORS = OcrCorruptorsTD
    OUTPUT_DIR = "results"
    PURPOSE = "TD100k"
    LOOKUP_FILES_DIR = "src/main/resources/lookup-files"
    DETERMINISTIC = True
    SEED = 42
```

Listing D.1: The values set in Config for corrupting the *TD100k* dataset.

The same configuration file (with the purpose String adjusted accordingly) was used to corrupt the smaller *TD5k* dataset used in sub-Section 6.3.4.

E. Crptr Standard Profiles

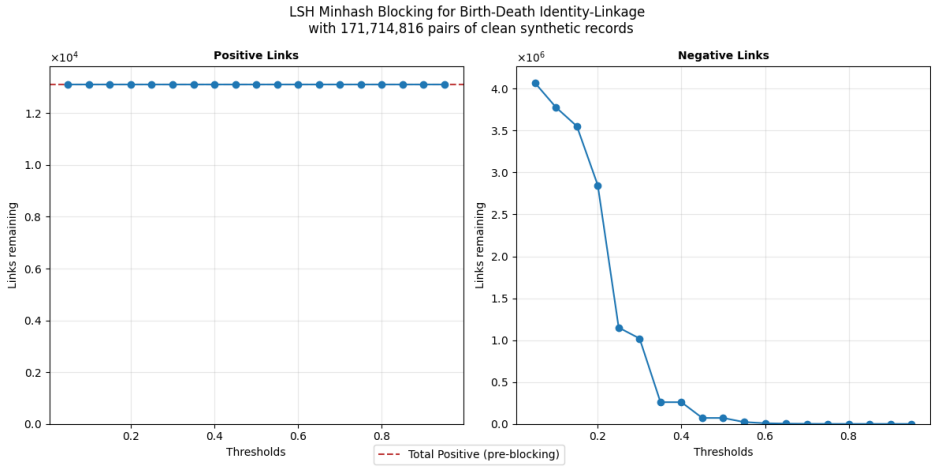
The following standard configurations were defined for corrupting synthetic populations with the `crptr` tool [12].

	Corruption Profiles		
Configuration Parameter	**A** OCR Transcription	**B** Human Transcription	**C** Human Transcription
Proportion of Records	0.40	0.40	0.90
Attribute Max Modifications	2.00	2.00	2.00
Modifications Per Record	4.00	4.00	8.00
Attribute Level Proportion	0.75	0.75	0.50
Record Level Proportion	0.25	0.25	0.50

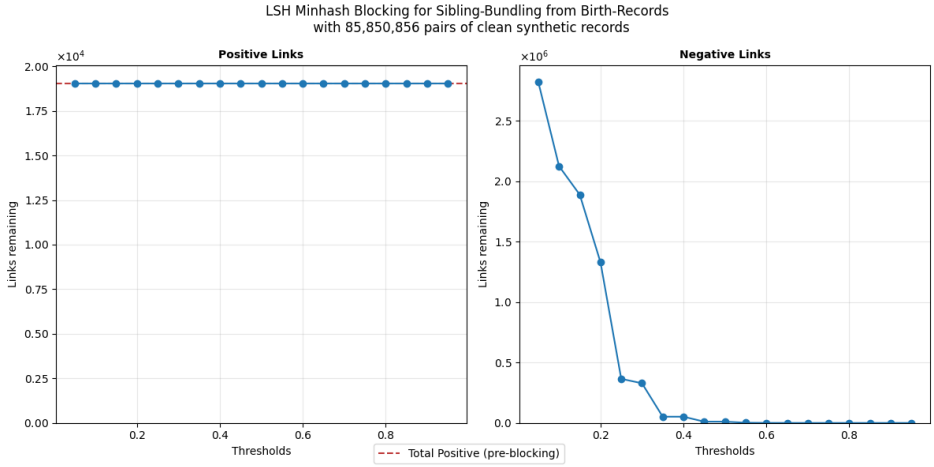
Table E.1.: This table shows the configuration parameters used for each corruption profile in *Crptr* (adapted from [12, Tab. 7.2]).

F. Blocking Plots for TD5k (Clean)

As with the Locality-Sensitive Hashing (LSH) MinHash blocking results for the larger synthetic TD100k (Clean) dataset (see Section 6.2), the TD5k (Clean) dataset exhibits best-case performance, retaining all positive links regardless of threshold value set.



(a) Birth-Death Identity linkage.



(b) Sibling bundling from Birth Records.

Figure F.1.: Line graphs plotting the number of positive and negative links retained by LSH MinHash blocking, with an expanded range of threshold values, on the clean TD5k synthetic dataset.

G. Random Seeds

To ensure reproducibility, whilst also properly evaluating the effects of randomisation on the learning systems, five random seeds were generated using `numpy.random` and used consistently throughout the experimental procedure.

```
296050970
1239748595
1979413803
3282447458
4082546634
```

Listing G.1: Random seeds used for the experiments in Chapter 6.

H. Learner Scores

The following tables detail the performance scores of different learner configurations using blocking thresholds 0.35 and 0.50 (see Section 6.4.5).

Note that all values are means over 5 runs (or 10 for initial results, as repeated for both query strategies), unless stated otherwise. Computationally infeasible runs have been ommitted (see Section 6.2).

Acronyms and Abbreviations

The following acronyms and abbreviations are used to identify learners in the results tables:

- Passive Learning (PL)
- Active Learning (AL)
- Random Forest (RF)
- XGBoost (XGB)
- Uncertainty (Uncert)
- Rebalanced (Rb)

TD100k (Clean)

Birth-Death Identity Linkage

Learner	Threshold	n labelled	Precision	Recall	F1
PL RF	0.35	48,683,712	0.999	1.000	0.999
PL XGB	0.35	48,683,712	0.999	1.000	0.999
AL RF (Initial)	0.35	1,402	0.987	1.000	0.994

Continued on next page...

Learner	Threshold	n labelled	Precision	Recall	F1
AL XGB (Initial)	0.35	1,372	0.929	0.901	0.902
AL RF (Initial Rb)	0.35	10	0.377	1.000	0.499
AL XGB (Initial Rb)	0.35	10	0.110	1.000	0.188
AL RF Uncert	0.35	6,642	0.999	1.000	0.999
AL XGB Uncert	0.35	7,142	0.999	1.000	0.999
AL RF Random	0.35	7,242	0.993	1.000	0.996
AL XGB Random	0.35	11,134	0.992	1.000	0.996
AL RF Uncert (Rb)	0.35	5,470	0.999	1.000	0.999
AL XGB Uncert (Rb)	0.35	6,230	0.999	1.000	0.999
AL RF Random (Rb)	0.35	7,185	0.992	1.000	0.996
AL XGB Random (Rb)	0.35	9,570	0.994	1.000	0.997
PL RF	0.50	9,558,796	0.998	1.000	0.999
PL XGB	0.50	9,558,796	0.998	1.000	0.999
AL RF (Initial)	0.50	309	0.985	1.000	0.992
AL XGB (Initial)	0.50	309	0.996	1.000	0.998

Continued on next page...

Learner	Threshold	n labelled	Precision	Recall	F1
AL RF (Initial Rb)	0.50	10	0.363	1.000	0.532
AL XGB (Initial Rb)	0.50	10	0.220	1.000	0.338
AL RF Uncert	0.50	5,629	0.998	1.000	0.999
AL XGB Uncert	0.50	5,869	0.998	1.000	0.999
AL RF Random	0.50	6,509	0.993	1.000	0.996
AL XGB Random	0.50	11,809	0.992	1.000	0.996
AL RF Uncert (Rb)	0.50	5,490	0.998	1.000	0.999
AL XGB Uncert (Rb)	0.50	5,790	0.998	1.000	0.999
AL RF Random (Rb)	0.50	7,030	0.995	1.000	0.998
AL XGB Random (Rb)	0.50	11,310	0.993	1.000	0.996

Table H.1.: Table comparing performance scores for assorted learner configurations for birth-death identity linkage, using the TD100k (Clean) dataset. Values are means over all completed runs.

The following runs were deemed computationally infeasible:

- Active learning Random Forest random sampling (rebalanced) at threshold 0.35 - 1/5 failed
- Active learning XGBoost random sampling at threshold 0.35 - 1/5 failed

Sibling Bundling from Birth Records

Learner	Threshold	n labelled	Precision	Recall	F1
PL RF	0.35	12,091,496	1.000	1.000	1.000
PL XGB	0.35	12,091,496	1.000	1.000	1.000
AL RF (Initial)	0.35	175	0.950	0.966	0.958
AL XGB (Initial)	0.35	175	0.982	0.964	0.973
AL RF (Initial Rb)	0.35	10	0.488	1.000	0.602
AL XGB (Initial Rb)	0.35	10	0.165	1.000	0.281
AL RF Uncert	0.35	5,715	1.000	1.000	1.000
AL XGB Uncert	0.35	6,135	1.000	1.000	1.000
AL RF Random	0.35	9,095	0.999	1.000	0.999
AL XGB Random	0.35	13,135	0.994	1.000	0.997
AL RF Uncert (Rb)	0.35	5,570	1.000	1.000	1.000
AL XGB Uncert (Rb)	0.35	6,230	1.000	1.000	1.000
AL RF Random (Rb)	0.35	9,150	0.999	1.000	1.000
AL XGB Random (Rb)	0.35	17,910	0.998	1.000	0.999

Continued on next page...

Learner	Threshold	n labelled	Precision	Recall	F1
PL RF	0.50	2,721,870	1.000	1.000	1.000
PL XGB	0.50	2,721,870	1.000	1.000	1.000
AL RF (Initial)	0.50	43	0.983	0.974	0.978
AL XGB (Initial)	0.50	43	0.921	0.972	0.938
AL RF (Initial Rb)	0.50	10	0.921	1.000	0.955
AL XGB (Initial Rb)	0.50	10	0.598	1.000	0.798
AL RF Uncert	0.50	5,563	1.000	1.000	1.000
AL XGB Uncert	0.50	5,943	1.000	1.000	1.000
AL RF Random	0.50	7,363	0.999	1.000	1.000
AL XGB Random	0.50	10,283	0.997	0.999	0.998
AL RF Uncert (Rb)	0.50	5,530	1.000	1.000	1.000
AL XGB Uncert (Rb)	0.50	5,690	1.000	1.000	1.000
AL RF Random (Rb)	0.50	6,130	0.998	1.000	0.999

Continued on next page...

Learner	Threshold	n labelled	Precision	Recall	F1
AL XGB Random (Rb)	0.50	9,910	0.998	1.000	0.999

Table H.2.: Table comparing performance scores for assorted learner configurations for sibling bundling from birth records, using the TD100k (Clean) dataset. Values are means over all completed runs.

TD100k (Corrupt)

Birth-Death Identity Linkage

Learner	Threshold	n labelled	Precision	Recall	F1
PL RF	0.35	34,453,432	0.977	0.410	0.578
PL XGB	0.35	34,453,432	0.980	0.414	0.582
AL RF (Initial)	0.35	1,131	0.986	0.340	0.506
AL XGB (Initial)	0.35	1,131	0.941	0.318	0.471
AL RF (Initial Rb)	0.35	10	0.244	0.391	0.292
AL XGB (Initial Rb)	0.35	10	0.165	0.378	0.215
AL RF Uncert	0.35	7,331	0.977	0.408	0.576
AL XGB Uncert	0.35	8,221	0.977	0.413	0.580
AL RF Uncert (Rb)	0.35	6,670	0.976	0.409	0.576

Continued on next page...

Learner	Threshold	n labelled	Precision	Recall	F1
AL XGB Uncert (Rb)	0.35	6,630	0.977	0.413	0.580
AL RF Random (Rb)	0.35	25,710	0.973	0.368	0.534
PL RF	0.50	6,525,036	0.983	0.338	0.503
PL XGB	0.50	6,525,036	0.986	0.340	0.506
AL RF (Initial)	0.50	334	0.988	0.298	0.458
AL XGB (Initial)	0.50	334	0.897	0.304	0.453
AL RF (Initial Rb)	0.50	10	0.367	0.334	0.340
AL XGB (Initial Rb)	0.50	10	0.315	0.320	0.297
AL RF Uncert	0.50	6,094	0.984	0.338	0.503
AL XGB Uncert	0.50	6,154	0.985	0.339	0.505
AL RF Random	0.50	26,254	0.984	0.321	0.485
AL XGB Random	0.50	27,454	0.972	0.331	0.494
AL RF Uncert (Rb)	0.50	5,870	0.984	0.338	0.503
AL XGB Uncert (Rb)	0.50	6,070	0.985	0.339	0.505

Continued on next page...

Learner	Threshold	n labelled	Precision	Recall	F1
AL RF Random (Rb)	0.50	28,690	0.986	0.321	0.485
AL XGB Random (Rb)	0.50	25,570	0.975	0.330	0.493

Table H.3.: Table comparing performance scores for assorted Learner configurations for Birth-Death Identity linkage, using the TD100k (Corrupt) dataset. Values are means over all completed runs.

The following runs were deemed computationally infeasible:

- Active learning Random Forest random sampling at threshold 0.35 - all failed
- Active learning Random Forest random sampling (rebalanced) at threshold 0.35 - all failed
- Active learning XGBoost random sampling at threshold 0.35 - all failed
- Active learning XGBoost random sampling (rebalanced) at threshold 0.35 - all failed

Sibling Bundling from Birth Records

Learner	Threshold	n labelled	Precision	Recall	F1
PL RF	0.35	9,552,870	0.997	0.484	0.652
PL XGB	0.35	9,552,870	0.996	0.484	0.651
AL RF (Initial)	0.35	179	0.942	0.457	0.615
AL XGB (Initial)	0.35	179	0.962	0.426	0.590
AL RF (Initial Rb)	0.35	10	0.728	0.463	0.541

Continued on next page...

Learner	Threshold	n labelled	Precision	Recall	F1
AL XGB (Initial Rb)	0.35	10	0.133	0.409	0.195
AL RF Uncert	0.35	6,059	0.995	0.481	0.648
AL XGB Uncert	0.35	6,719	0.993	0.483	0.650
AL RF Random	0.35	16,099	0.985	0.475	0.641
AL XGB Random	0.35	26,139	0.986	0.475	0.641
AL RF Uncert (Rb)	0.35	6,570	0.995	0.481	0.649
AL XGB Uncert (Rb)	0.35	6,470	0.994	0.483	0.650
AL RF Random (Rb)	0.35	10,079	0.984	0.471	0.637
AL XGB Random (Rb)	0.35	18,310	0.987	0.475	0.641
PL RF	0.50	2,084,830	0.998	0.372	0.542
PL XGB	0.50	2,084,830	0.998	0.372	0.542
AL RF (Initial)	0.50	40	0.851	0.332	0.476
AL XGB (Initial)	0.50	40	0.831	0.289	0.426
AL RF (Initial Rb)	0.50	10	0.765	0.331	0.458
AL XGB (Initial Rb)	0.50	10	0.445	0.284	0.335

Continued on next page...

Learner	Threshold	n labelled	Precision	Recall	F1
AL RF Uncert	0.50	5,940	0.997	0.371	0.541
AL XGB Uncert	0.50	6,220	0.997	0.371	0.541
AL RF Random	0.50	7,540	0.990	0.367	0.536
AL XGB Random	0.50	6,220	0.990	0.367	0.536
AL RF Uncert (Rb)	0.50	5,630	0.997	0.371	0.541
AL XGB Uncert (Rb)	0.50	6,290	0.997	0.372	0.541
AL RF Random (Rb)	0.50	7,550	0.990	0.367	0.536
AL XGB Random (Rb)	0.50	7,990	0.991	0.367	0.536

Table H.4.: Table comparing performance scores for assorted learner configurations for sibling bundling from birth records, using the TD100k (Corrupt) dataset. Values are means over all completed runs.

POPLINK

Birth-Death Identity Linkage

Learner	Threshold	n labelled	Precision	Recall	F1
PL RF	0.35	37,962,304	0.869	0.131	0.228
PL XGB	0.35	37,962,304	0.890	0.130	0.227

Continued on next page...

Learner	Threshold	n labelled	Precision	Recall	F1
AL RF (Initial)	0.35	3,501	0.638	0.074	0.129
AL XGB (Initial)	0.35	3,501	0.806	0.080	0.142
AL RF (Initial Rb)	0.35	10	0.009	0.203	0.017
AL XGB (Initial Rb)	0.35	10	0.007	0.189	0.013
AL RF Uncert	0.35	14,581	0.858	0.131	0.227
AL XGB Uncert	0.35	17,081	0.898	0.130	0.227
AL RF Uncert (Rb)	0.35	10,510	0.641	0.148	0.229
AL XGB Uncert (Rb)	0.35	11,950	0.898	0.130	0.227
PL RF	0.50	9,124,940	0.894	0.099	0.179
PL XGB	0.50	9,124,940	0.906	0.099	0.178
AL RF (Initial)	0.50	1,293	0.784	0.041	0.076
AL XGB (Initial)	0.50	1,418	0.885	0.038	0.072
AL RF (Initial Rb)	0.50	10	0.024	0.155	0.041
AL XGB (Initial Rb)	0.50	10	0.016	0.139	0.028
AL RF Uncert	0.50	9,313	0.881	0.099	0.177
AL XGB Uncert	0.50	9,313	0.907	0.098	0.177

Continued on next page...

Learner	Threshold	n labelled	Precision	Recall	F1
AL RF Random	0.50	53,913	0.876	0.085	0.156
AL XGB Random	0.50	110,147	0.908	0.089	0.163
AL RF Uncert (Rb)	0.50	9,710	0.597	0.115	0.180
AL XGB Uncert (Rb)	0.50	9,430	0.906	0.098	0.177
AL RF Random (Rb)	0.50	33,435	0.871	0.084	0.153
AL XGB Random (Rb)	0.50	146,210	0.841	0.095	0.170

Table H.5.: Table comparing performance scores for assorted learner configurations for birth-death identity linkage, using the POPLINK dataset. Values are means over all completed runs.

The following runs were deemed computationally infeasible:

- Active learning Random Forest random sampling at threshold 0.35 - all failed
- Active learning Random Forest random sampling (rebalanced) at threshold 0.35 - all failed
- Active learning XGBoost random sampling at threshold 0.35 - all failed
- Active learning XGBoost random sampling (rebalanced) at threshold 0.35 - all failed
- Active learning Random Forest random sampling (rebalanced) at threshold 0.5 - 1/5 failed
- Active learning XGBoost random sampling at threshold 0.5 - 3/5 failed

- Active learning XGBoost random sampling (rebalanced) at threshold 0.5 - all failed

Sibling Bundling from Birth Records

Learner	Threshold	n labelled	Precision	Recall	F1
PL RF	0.35	41,186,943	0.88	0.001	0.002
PL XGB	0.35	41,186,943	0.861	0.000	0.000
AL RF (Initial)	0.35	121	0.634	0.000	0.000
AL XGB (Initial)	0.35	121	0.463	0.000	0.000
AL RF (Initial Rb)	0.35	10	0.119	0.001	0.001
AL XGB (Initial Rb)	0.35	10	0.076	0.000	0.001
AL RF Uncert	0.35	5,021	0.880	0.000	0.001
AL XGB Uncert	0.35	5,021	0.886	0.000	0.001
AL RF Random	0.35	5,021	0.737	0.000	0.001
AL XGB Random	0.35	5,021	0.673	0.000	0.001
AL RF Uncert (Rb)	0.35	4,910	0.884	0.000	0.001
AL XGB Uncert (Rb)	0.35	4,910	0.875	0.000	0.001
AL RF Random (Rb)	0.35	4,910	0.740	0.000	0.001

Continued on next page...

Learner	Threshold	n labelled	Precision	Recall	F1
AL XGB Random (Rb)	0.35	4,910	0.660	0.000	0.001
PL RF	0.50	8,484,192	0.926	0.000	0.000
PL XGB	0.50	8,484,192	0.918	0.000	0.000
AL RF (Initial)	0.50	55	0.774	0.000	0.000
AL XGB (Initial)	0.50	55	0.603	0.000	0.000
AL RF (Initial Rb)	0.50	10	0.241	0.000	0.001
AL XGB (Initial Rb)	0.50	10	0.174	0.000	0.001
AL RF Uncert	0.50	4,955	0.911	0.000	0.001
AL XGB Uncert	0.50	4,955	0.905	0.000	0.001
AL RF Random	0.50	4,955	0.877	0.000	0.001
AL XGB Random	0.50	4,955	0.846	0.000	0.001
AL RF Uncert (Rb)	0.50	4,910	0.910	0.000	0.001
AL XGB Uncert (Rb)	0.50	4,910	0.913	0.000	0.001
AL RF Random (Rb)	0.50	4,910	0.874	0.000	0.001

Continued on next page...

Learner	Threshold	n labelled	Precision	Recall	F1
AL XGB Random (Rb)	0.50	4,910	0.841	0.000	0.001

Table H.6.: Table comparing performance scores for assorted learner configurations for sibling bundling from birth records, using the POPLINK dataset. Values are means over all completed runs.

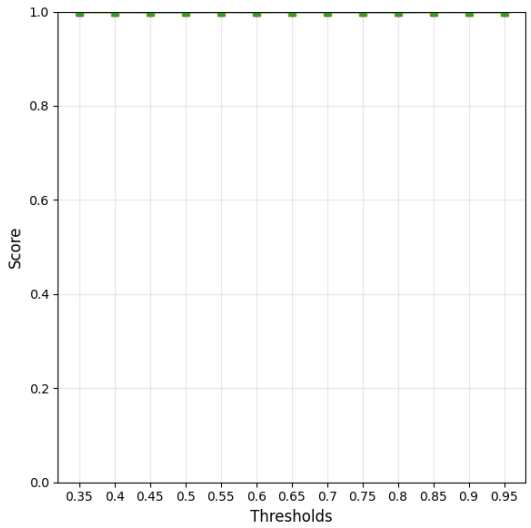
I. Passive Learner Plots

The following figures plot Precision, Recall and F1 scores for all combinations of the experiment configurations described in Section 6.4, with both classifier-only and combination evaluation methods (see Section 5.8.1).

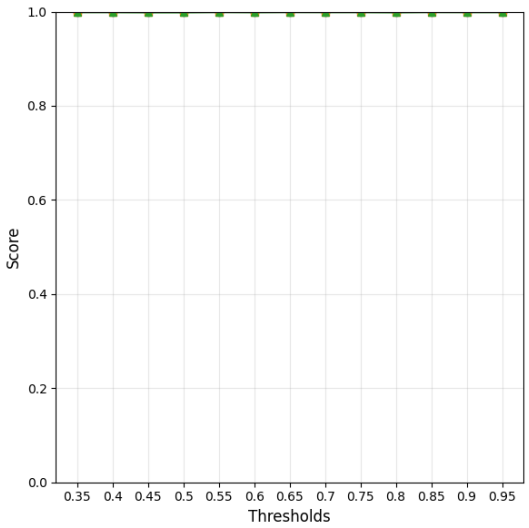


Figure I.1.: Legend for all passive learner plots in this appendix.

Birth-Death Identity Linkage (Combined)

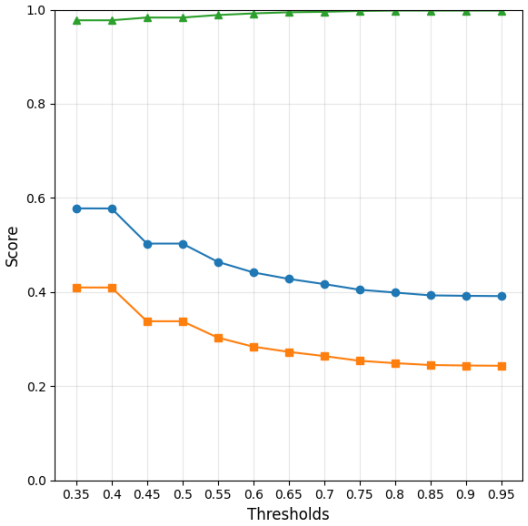


(a) TD100k (Clean) Random Forest

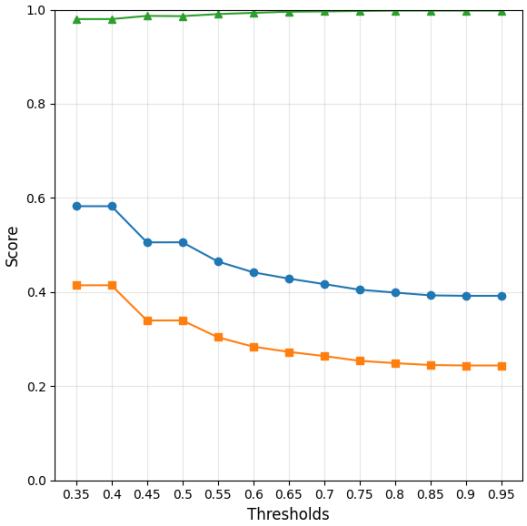


(b) TD100k (Clean) XGBoost

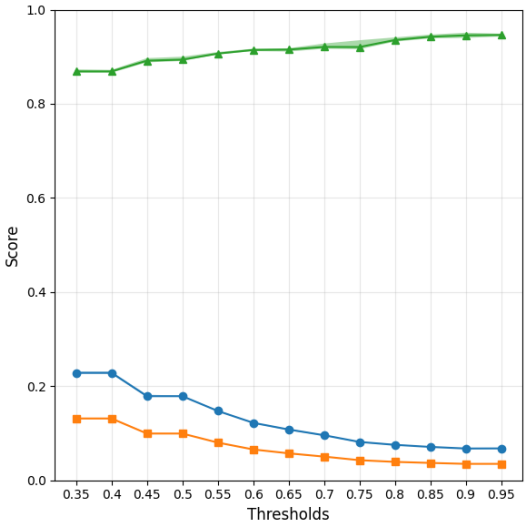
Figure I.2.: Line graphs plotting performance metrics (with combined evaluation) for a passive learner, against blocking thresholds, for birth-death identity linkage (1/2).



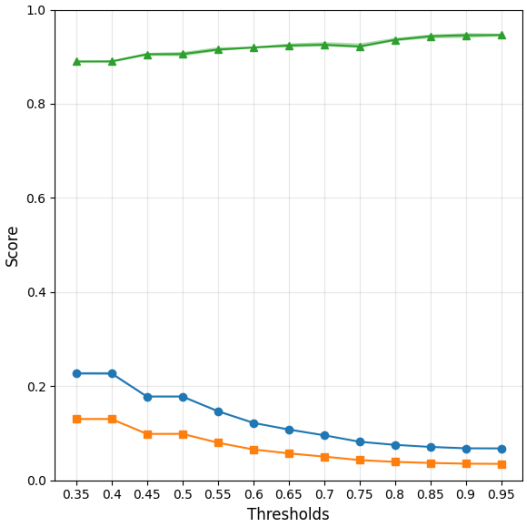
(c) TD100k (Corrupt) Random Forest



(d) TD100k (Corrupt) XGBoost



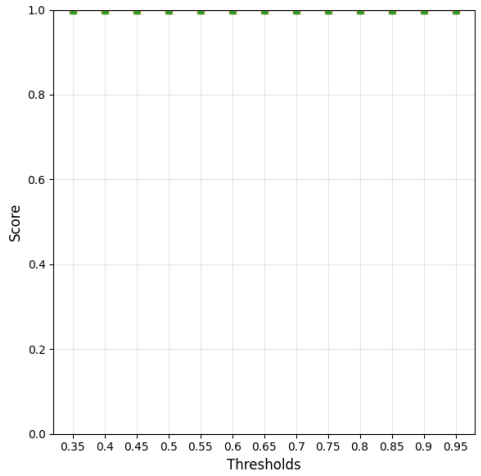
(e) POPLINK Random Forest



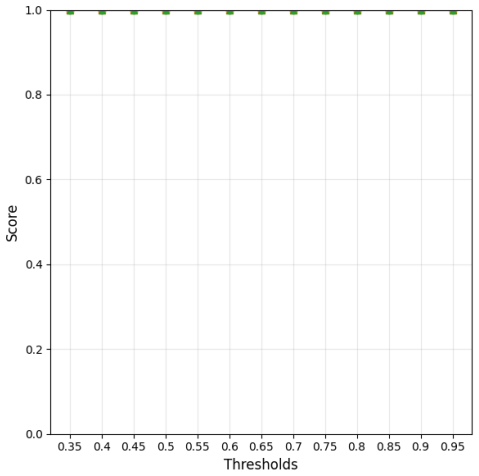
(f) POPLINK XGBoost

Figure I.2.: Line graphs plotting performance metrics (with combined evaluation) for a passive learner, against blocking thresholds, for birth-death identity (2/2).

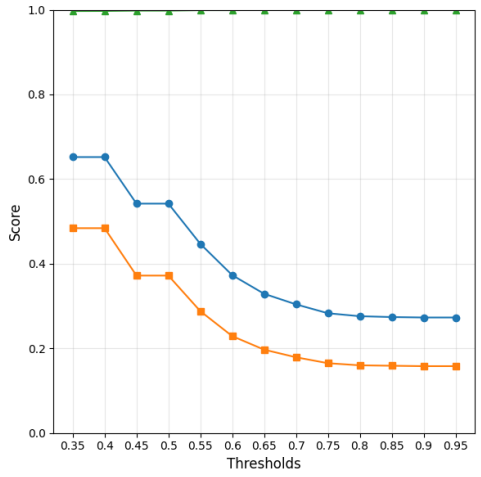
Sibling Bundling from Birth Records (Combined)



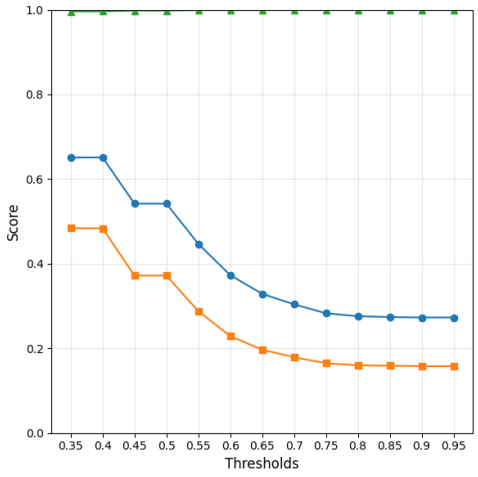
(a) TD100k (Clean) Random Forest



(b) TD100k (Clean) XGBoost

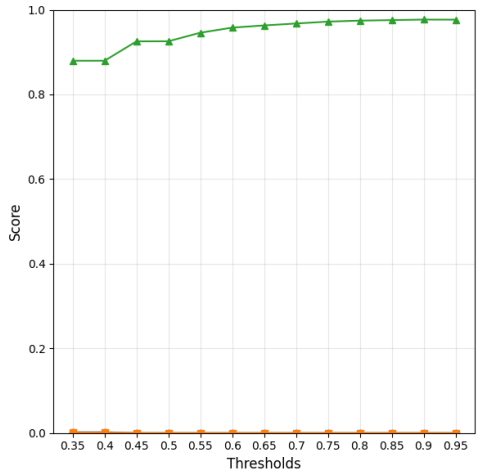


(c) TD100k (Corrupt) Random Forest

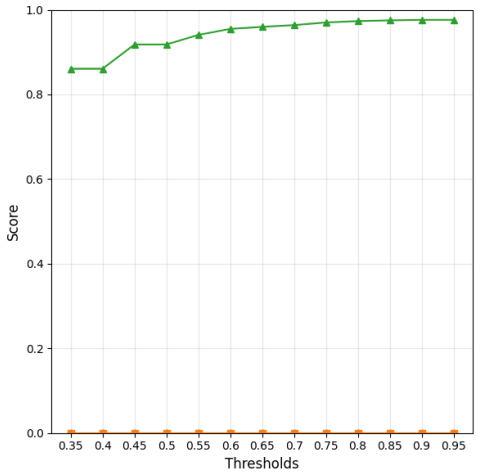


(d) TD100k (Corrupt) XGBoost

Figure I.3.: Line graphs plotting performance metrics (with combined evaluation) for a passive learner, against blocking thresholds, for sibling bundling from birth records (1/2).



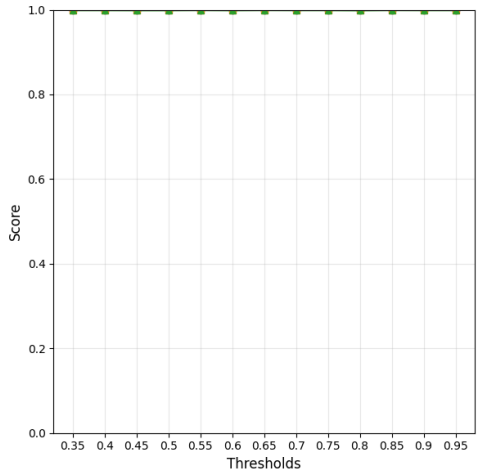
(e) POPLINK Random Forest



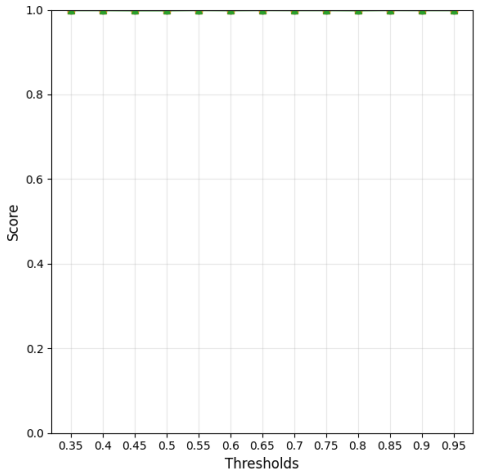
(f) POPLINK XGBoost

Figure I.3.: Line graphs plotting performance metrics (with combined evaluation) for a passive learner, against blocking thresholds, for sibling bundling from birth records (2/2).

Birth-Death Identity Linkage (Classifier-Only)

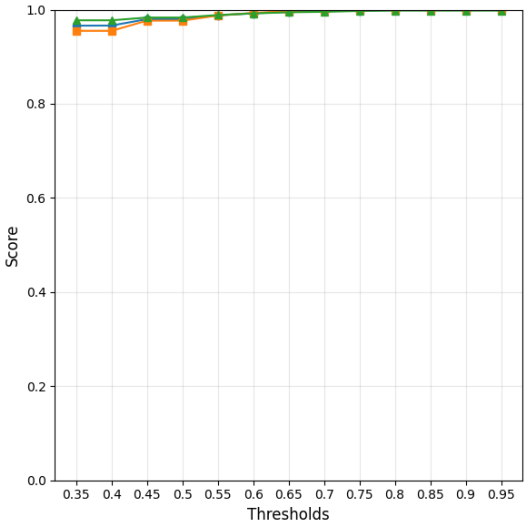


(a) TD100k (Clean) Random Forest

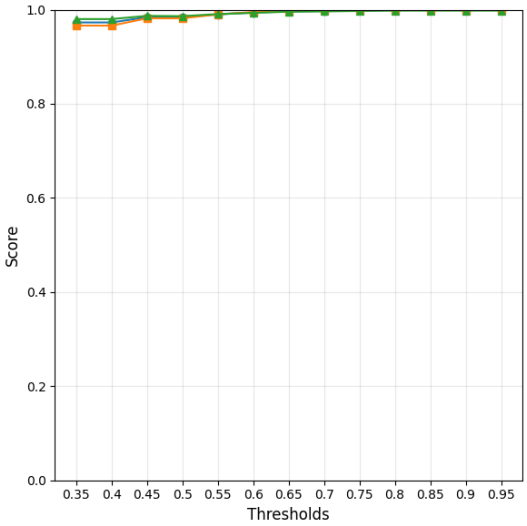


(b) TD100k (Clean) XGBoost

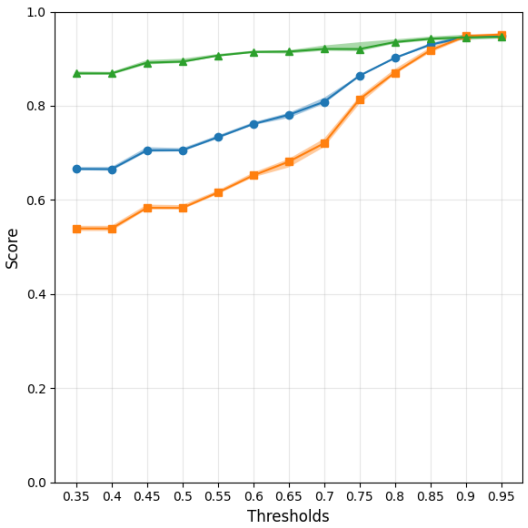
Figure I.4.: Line graphs plotting performance metrics (with classifier-only evaluation) for a passive learner, against blocking thresholds, for birth-death identity linkage (1/2).



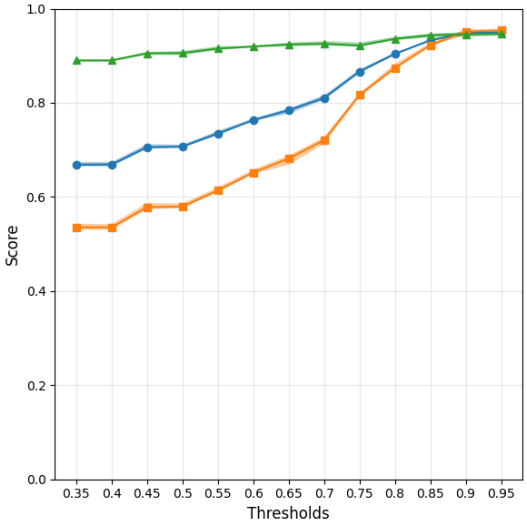
(c) TD100k (Corrupt) Random Forest



(d) TD100k (Corrupt) XGBoost



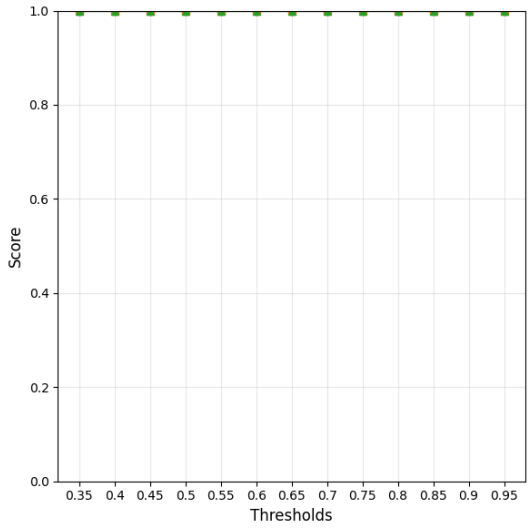
(e) POPLINK Random Forest



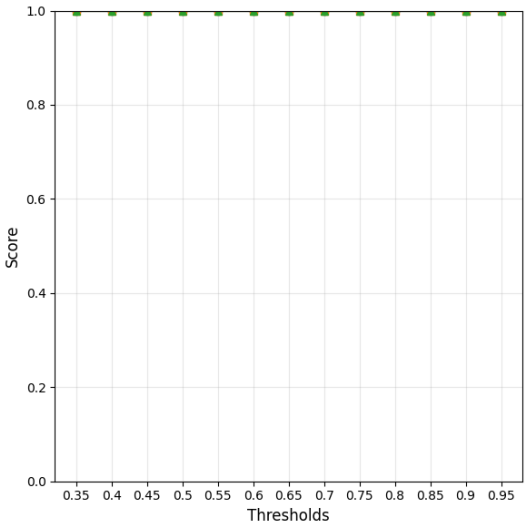
(f) POPLINK XGBoost

Figure I.4.: Line graphs plotting performance metrics (with classifier-only evaluation) for a passive learner, against blocking thresholds, for birth-death identity linkage (2/2).

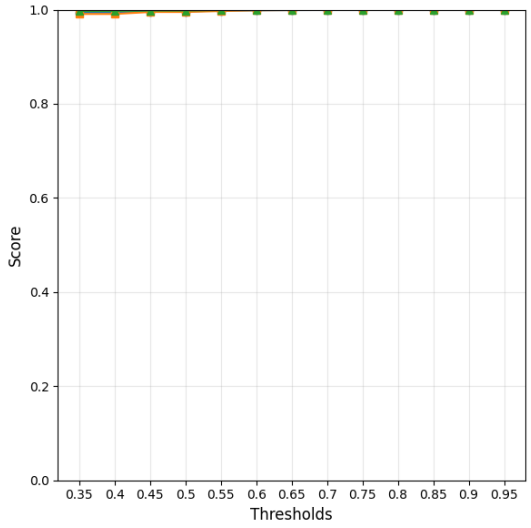
Sibling Bundling from Birth Records (Classifier-Only)



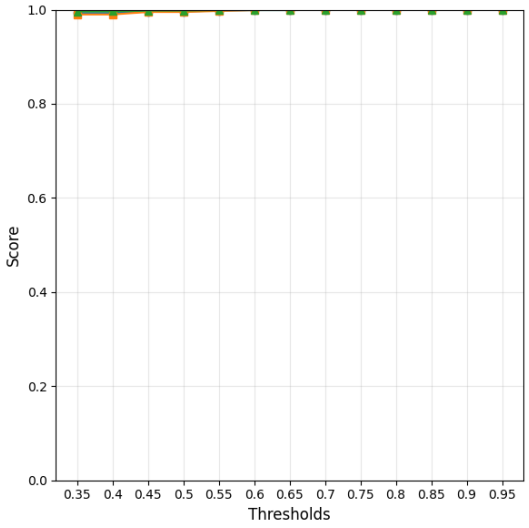
(a) TD100k (Clean) Random Forest



(b) TD100k (Clean) XGBoost

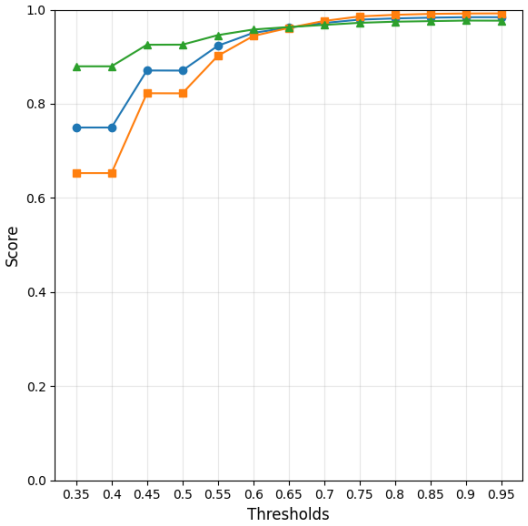


(c) TD100k (Corrupt) Random Forest

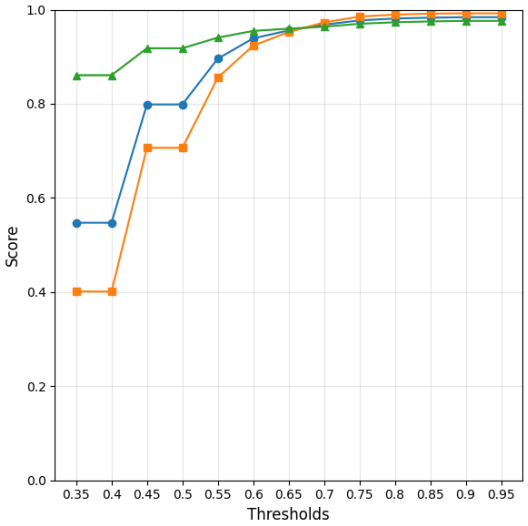


(d) TD100k (Corrupt) XGBoost

Figure I.5.: Line graphs plotting performance metrics (with classifier-only evaluation) for a passive learner, against blocking thresholds, for sibling bundling from birth records (1/2).



(e) POPLINK Random Forest



(f) POPLINK XGBoost

Figure I.5.: Line graphs plotting performance metrics (with classifier-only evaluation) for a passive learner, against blocking thresholds, for sibling bundling from birth records (2/2).

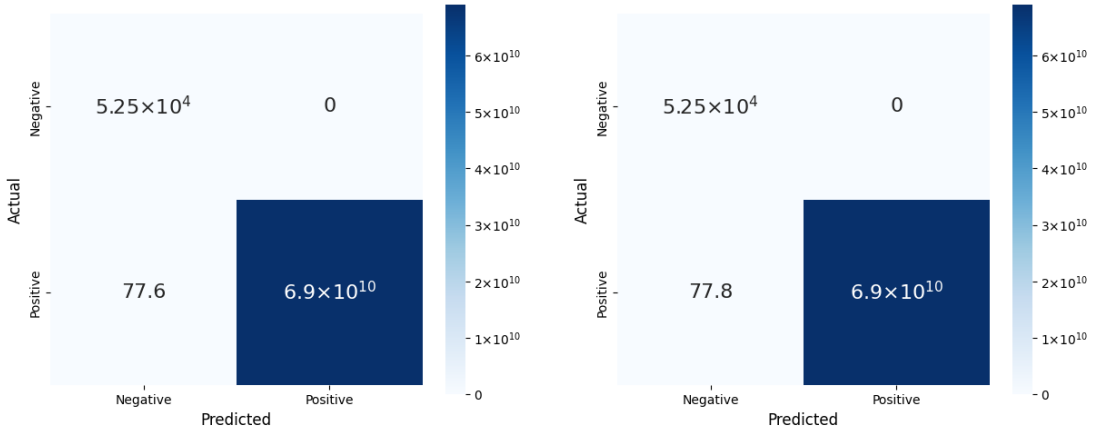
J. Passive Learner Confusion Matrixes

The following confusion matrixes plot predictions from the passive learner models described in Section 6.4.

TD100k (Clean)

Birth-Death Identity Linkage

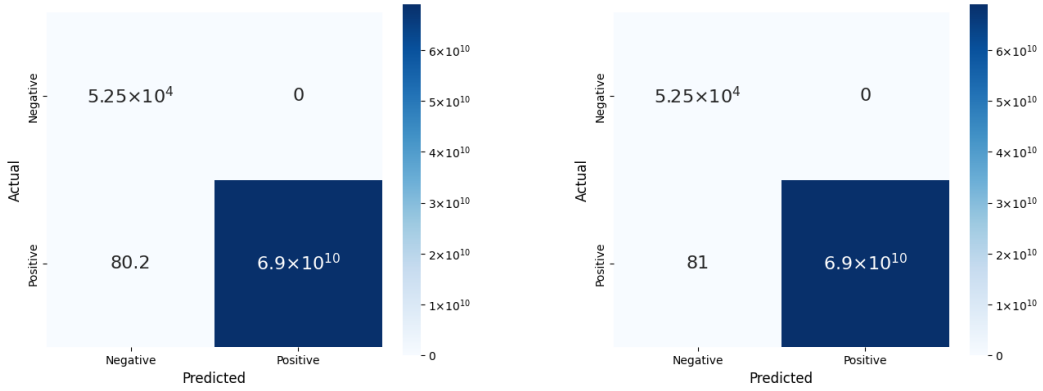
Blocking Threshold = 0.35



(a) Random Forest (b) XGBoost

Figure J.1.: Confusion matrixes for birth-death identity linkage using a passive learner on TD100k (Clean) blocked at threshold 0.35. Values are means over 5 runs.

Blocking Threshold = 0.5

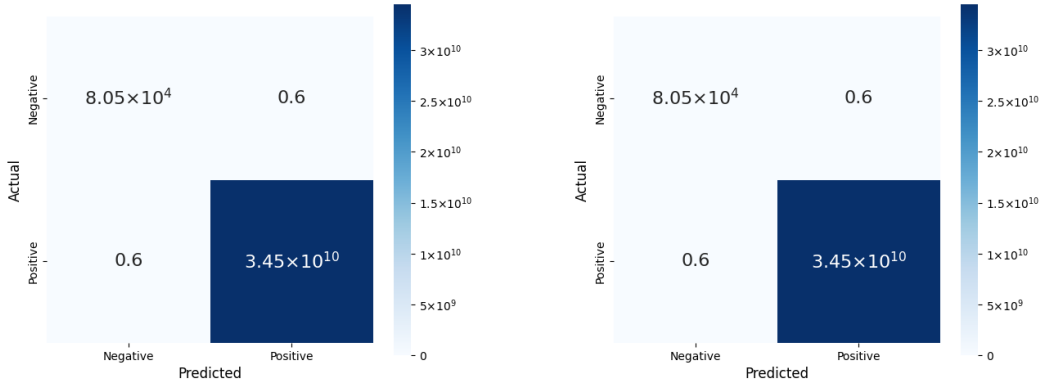


(a) Random Forest (b) XGBoost

Figure J.2.: Confusion matrixes for birth-death identity linkage using a passive learner on TD100k (Clean) blocked at threshold 0.5. Values are means over 5 runs.

Sibling Bundling from Birth Records

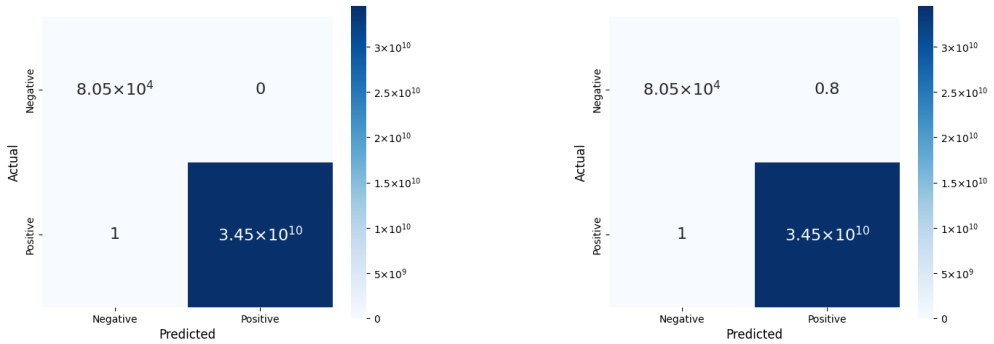
Blocking Threshold = 0.35



(a) Random Forest (b) XGBoost

Figure J.3.: Confusion matrixes for sibling bundling from birth records using a passive learner on TD100k (Clean) blocked at threshold 0.35. Values are means over 5 runs.

Blocking Threshold = 0.5



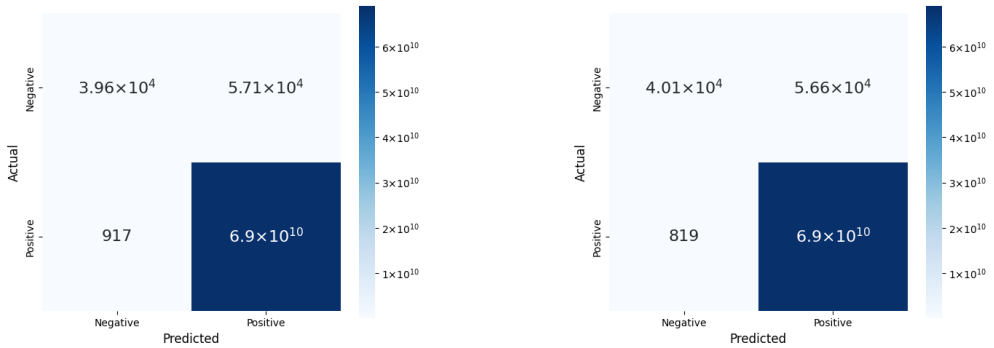
(a) Random Forest (b) XGBoost

Figure J.4.: Confusion matrixes for sibling bundling from birth records using a passive learner on TD100k (Clean) blocked at threshold 0.5. Values are means over 5 runs.

TD100k (Corrupt)

Birth-Death Identity Linkage

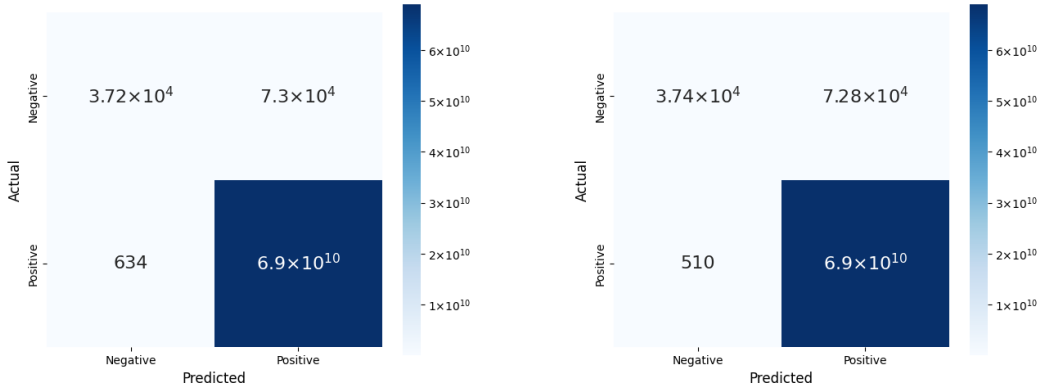
Blocking Threshold = 0.35



(a) Random Forest (b) XGBoost

Figure J.5.: Confusion matrixes for birth-death identity linkage using a passive learner on TD100k (Corrupt) blocked at threshold 0.35. Values are means over 5 runs.

Blocking Threshold = 0.5

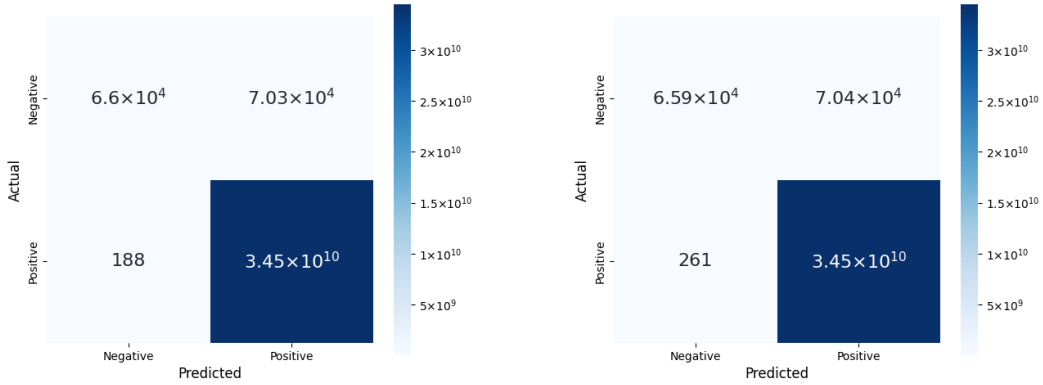


(a) Random Forest (b) XGBoost

Figure J.6.: Confusion matrixes for birth-death identity linkage using a passive learner on TD100k (Corrupt) blocked at threshold 0.5. Values are means over 5 runs.

Sibling Bundling from Birth Records

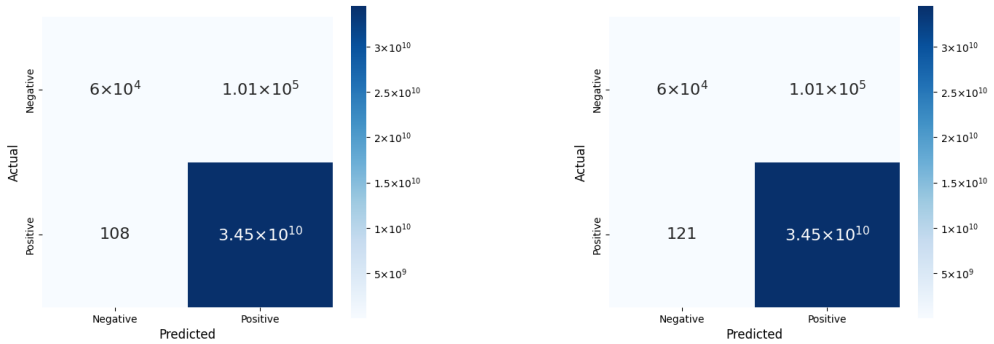
Blocking Threshold = 0.35



(a) Random Forest (b) XGBoost

Figure J.7.: Confusion matrixes for sibling bundling from birth records using a passive learner on TD100k (Corrupt) blocked at threshold 0.35. Values are means over 5 runs.

Blocking Threshold = 0.5



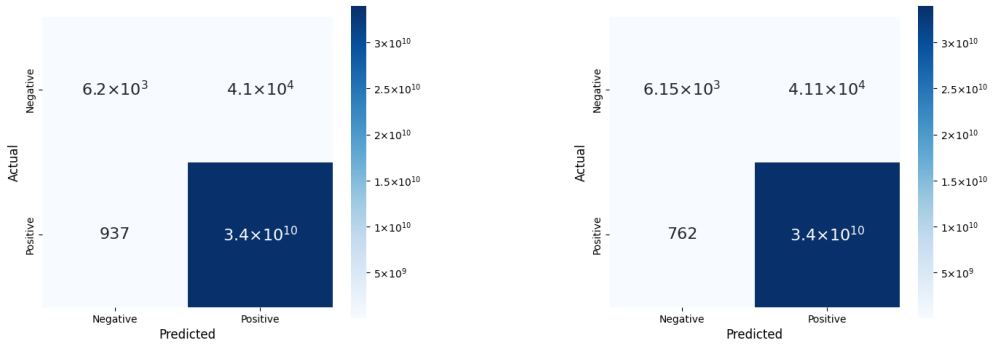
(a) Random Forest (b) XGBoost

Figure J.8.: Confusion matrixes for sibling bundling from birth records using a passive learner on TD100k (Corrupt) blocked at threshold 0.5. Values are means over 5 runs.

POPLINK

Birth-Death Identity Linkage

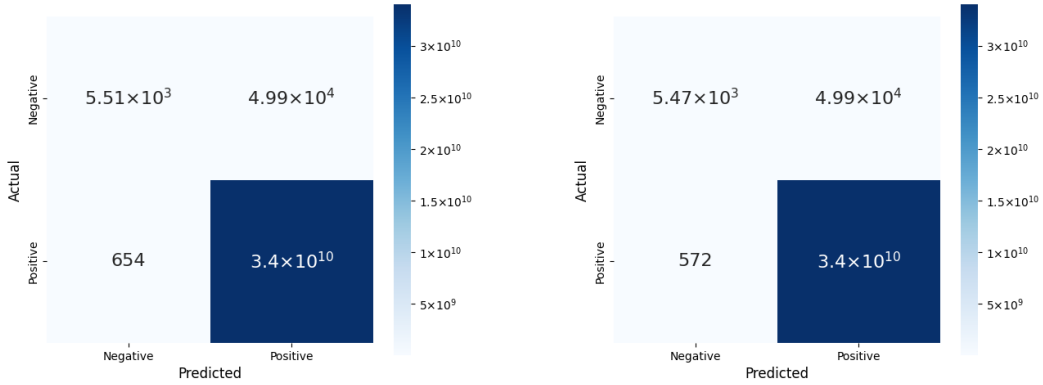
Blocking Threshold = 0.35



(a) Random Forest (b) XGBoost

Figure J.9.: Confusion matrixes for birth-death identity linkage using a passive learner on POPLINK blocked at threshold 0.35. Values are means over 5 runs.

Blocking Threshold = 0.5

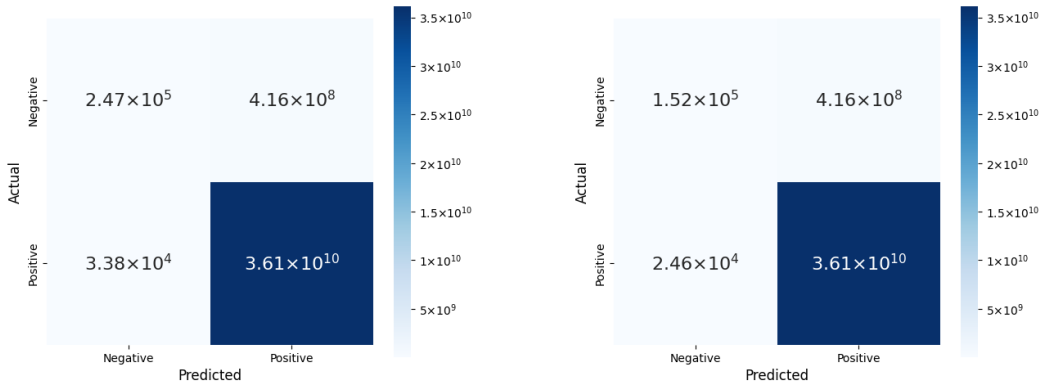


(a) Random Forest (b) XGBoost

Figure J.10.: Confusion matrixes for birth-death identity linkage using a passive learner on POPLINK blocked at threshold 0.5. Values are means over 5 runs.

Sibling Bundling from Birth Records

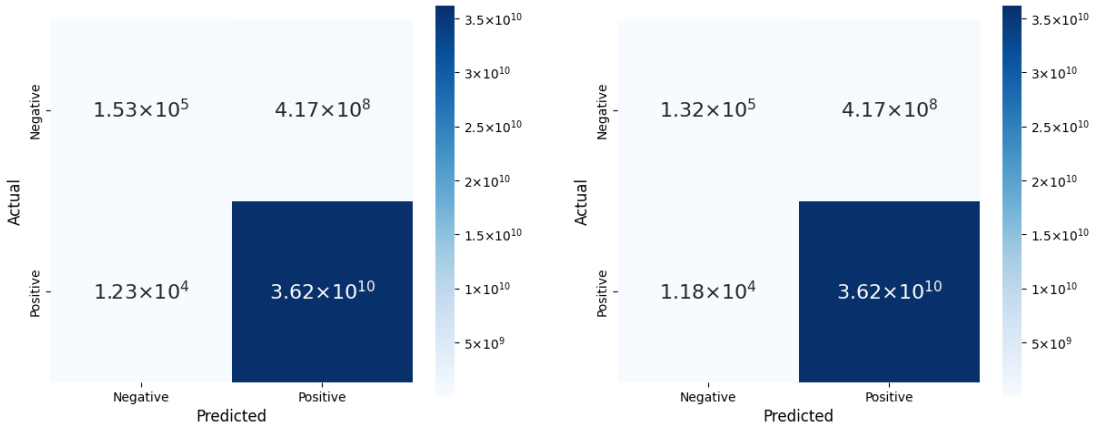
Blocking Threshold = 0.35



(a) Random Forest (b) XGBoost

Figure J.11.: Confusion matrixes for sibling bundling from birth records using a passive learner on POPLINK blocked at threshold 0.35. Values are means over 5 runs.

Blocking Threshold = 0.5



(a) Random Forest (b) XGBoost

Figure J.12.: Confusion matrixes for sibling bundling from birth records using a passive learner on POPLINK blocked at threshold 0.5. Values are means over 5 runs.

K. Active Learner Plots

The following figures plot performance metrics, with a passive learning benchmark for comparison, for all combinations of the experimental configurations described in Section 6.5.

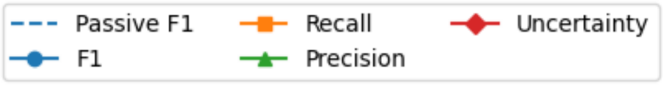
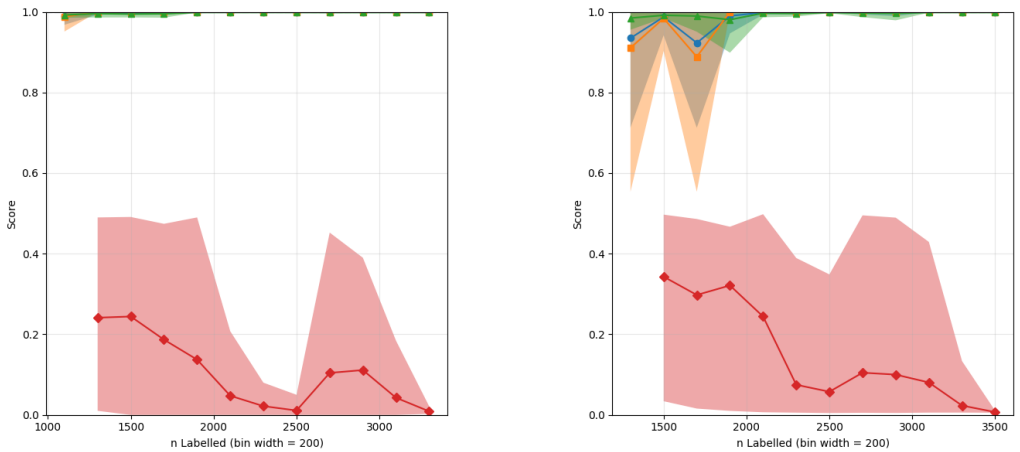


Figure K.1.: Legend for all active learner plots in this appendix.

TD100k (Clean)

Birth-Death Identity Linkage

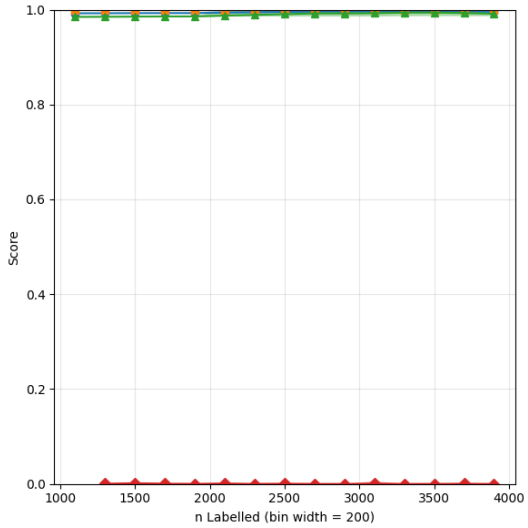
Blocking Threshold = 0.35



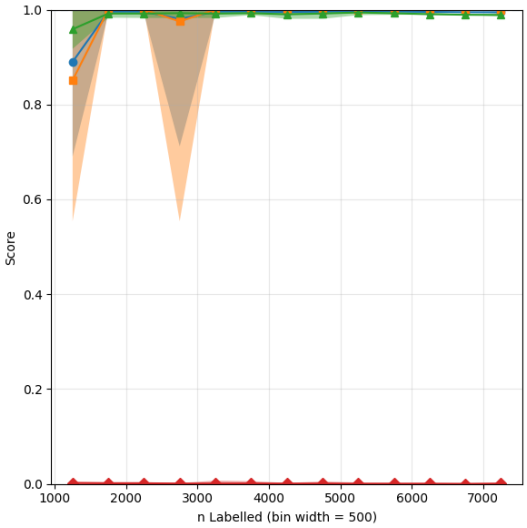
(a) Random Forest Uncertainty

(b) XGBoost Uncertainty

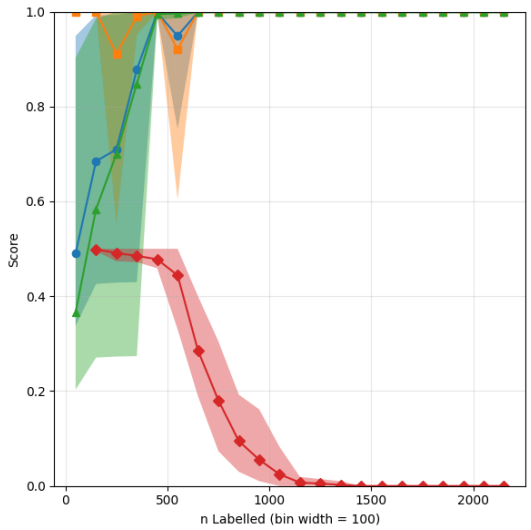
Figure K.2.: Line graphs plotting performance against number of labelled pairs for an AL system performing birth-death identity linkage on the TD100k (Clean) dataset blocked at threshold 0.35. Shaded areas represent the range of values across 5 runs (1/3).



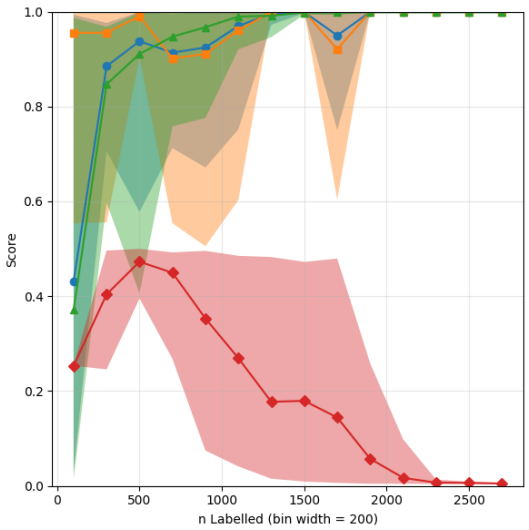
(c) Random Forest Random



(d) XGBoost Random

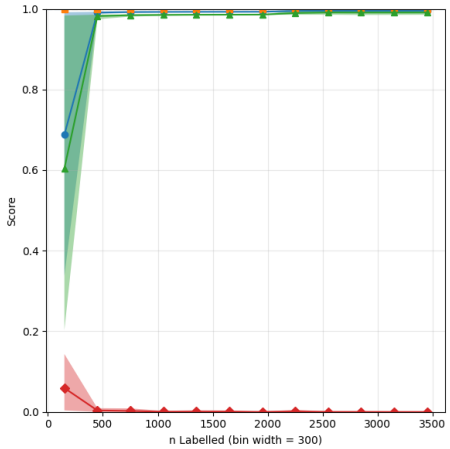


(e) Random Forest Uncertainty Rebalanced

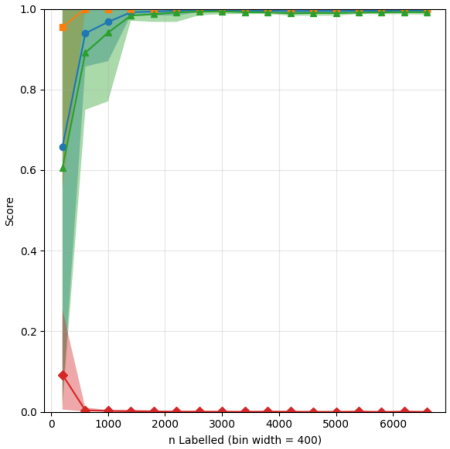


(f) XGBoost Uncertainty Rebalanced

Figure K.2.: Line graphs plotting performance against number of labelled pairs for an AL system performing birth-death identity linkage on the TD100k (Clean) dataset blocked at threshold 0.35. Horizontal dashed line represents a benchmark from passive learning and shaded areas represent the range of values across 5 runs (2/3).



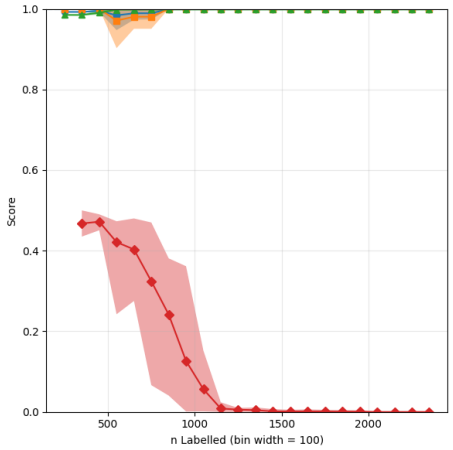
(g) Random Forest Random Rebalanced



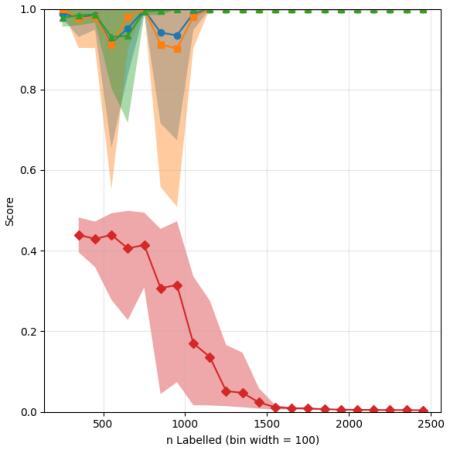
(h) XGBoost Random Rebalanced

Figure K.2.: Line graphs plotting performance against number of labelled pairs for an AL system performing birth-death identity linkage on the TD100k (Clean) dataset blocked at threshold 0.35. Shaded areas represent the range of values across 5 runs (3/3).

Blocking Threshold = 0.5

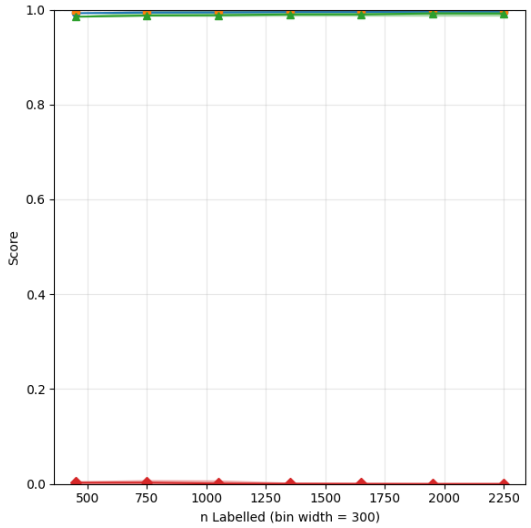


(a) Random Forest Uncertainty

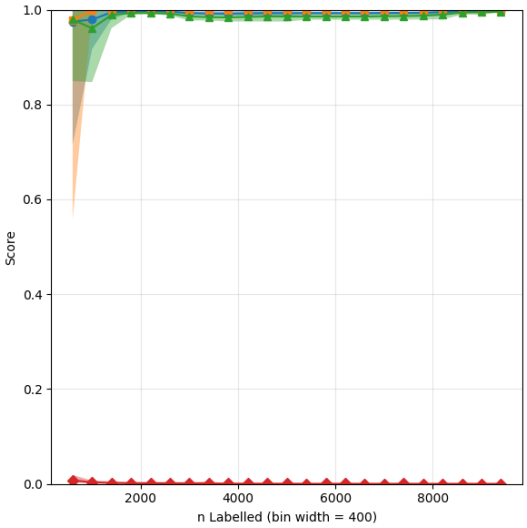


(b) XGBoost Uncertainty

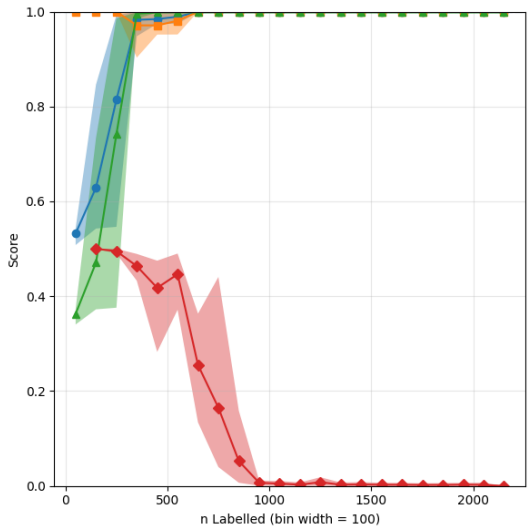
Figure K.3.: Line graphs plotting performance against number of labelled pairs for an AL system performing birth-death identity linkage on the TD100k (Clean) dataset blocked at threshold 0.5. Shaded areas represent the range of values across 5 runs (1/3).



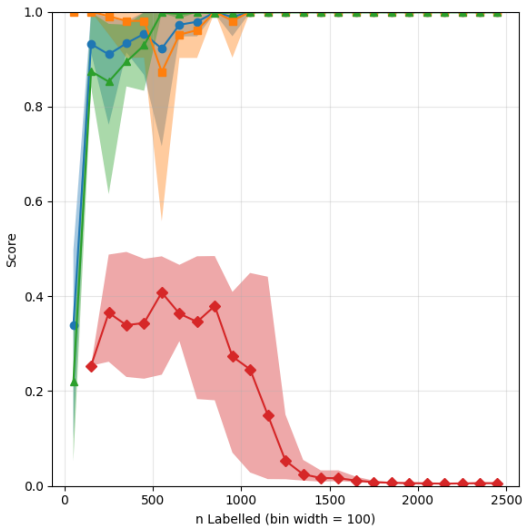
(c) Random Forest Random



(d) XGBoost Random

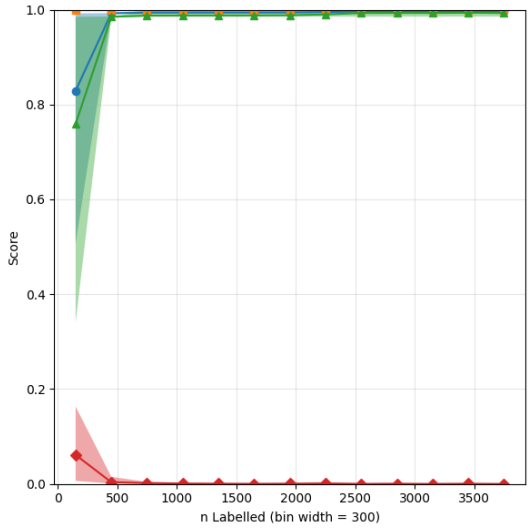


(e) Random Forest Uncertainty Rebalanced

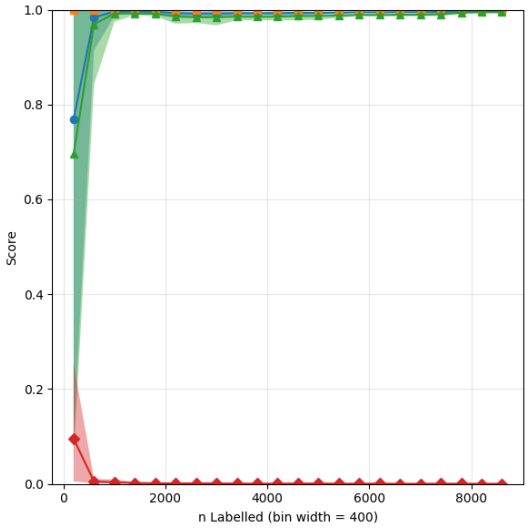


(f) XGBoost Uncertainty Rebalanced

Figure K.3.: Line graphs plotting performance against number of labelled pairs for an AL system performing birth-death identity linkage on the TD100k (Clean) dataset blocked at threshold 0.5. Horizontal dashed line represents a benchmark from passive learning and shaded areas represent the range of values across 5 runs (2/3).



(g) Random Forest Random Rebalanced

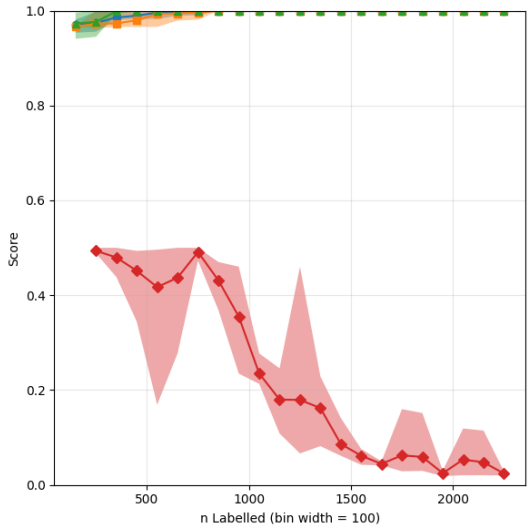


(h) XGBoost Random Rebalanced

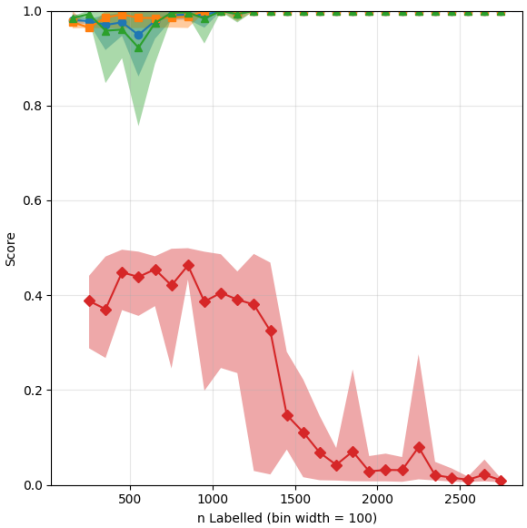
Figure K.3.: Line graphs plotting performance against number of labelled pairs for an AL system performing birth-death identity linkage on the TD100k (Clean) dataset blocked at threshold 0.5. Horizontal dashed line represents a benchmark from passive learning and shaded areas represent the range of values across 5 runs (3/3).

Sibling Bundling from Birth Records

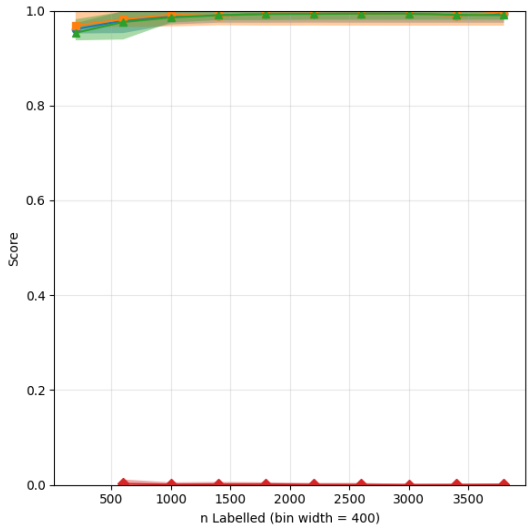
Blocking Threshold = 0.35



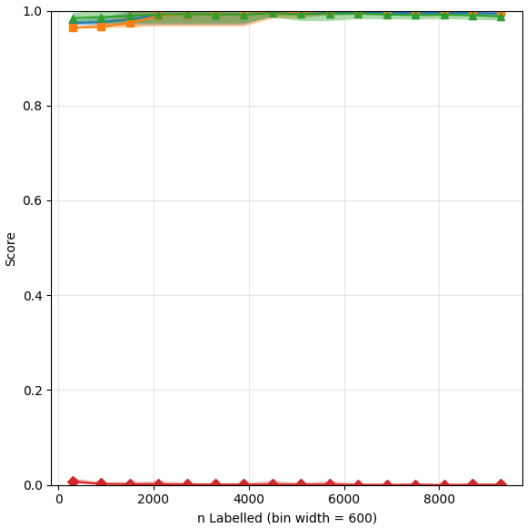
(a) Random Forest Uncertainty



(b) XGBoost Uncertainty

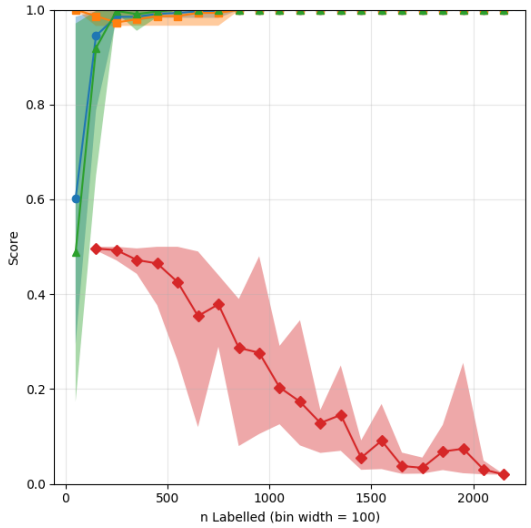


(c) Random Forest Random

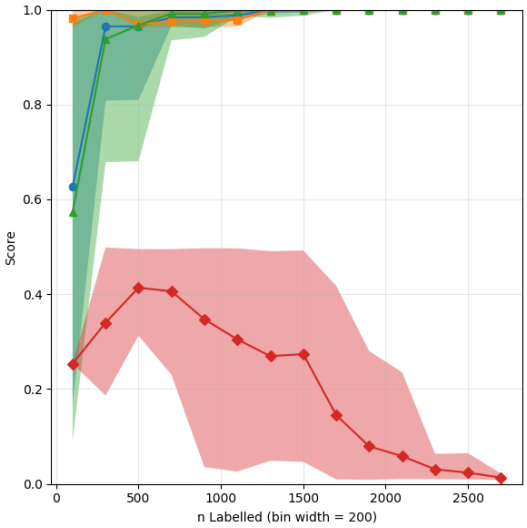


(d) XGBoost Random

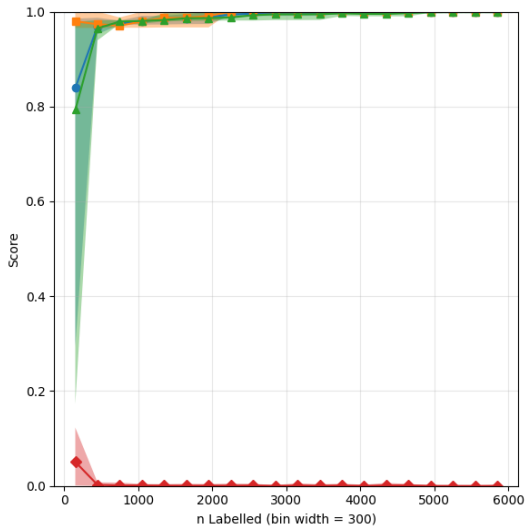
Figure K.4.: Line graphs plotting performance against number of labelled pairs for an AL system performing sibling bundling from birth records on the TD100k (Clean) dataset blocked at threshold 0.35. Horizontal dashed line represents a benchmark from passive learning and shaded areas represent the range of values across 5 runs (1/2).



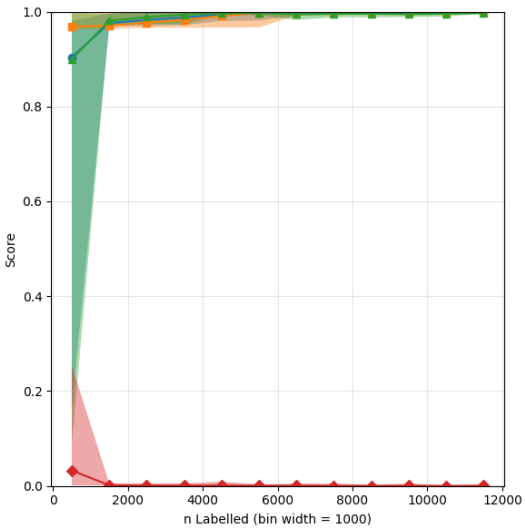
(e) Random Forest Uncertainty Rebalanced



(f) XGBoost Uncertainty Rebalanced



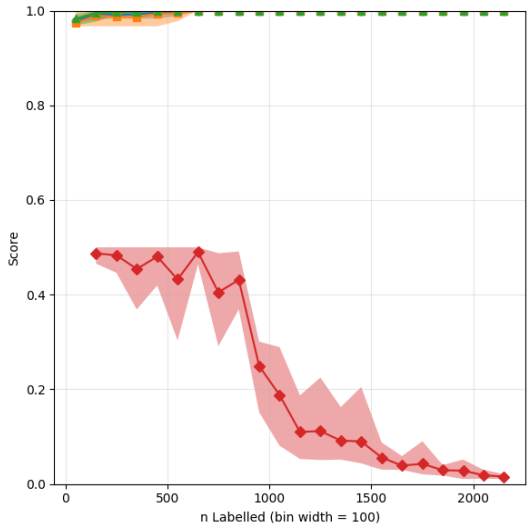
(g) Random Forest Random Rebalanced



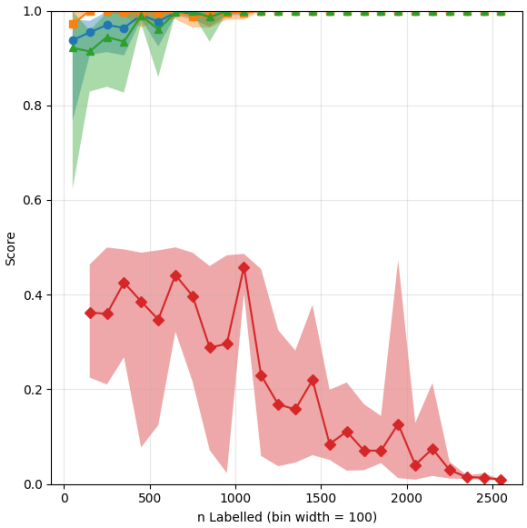
(h) XGBoost Random Rebalanced

Figure K.4.: Line graphs plotting performance against number of labelled pairs for an AL system performing sibling bundling from birth records on the TD100k (Clean) dataset blocked at threshold 0.35. Horizontal dashed line represents a benchmark from passive learning and shaded areas represent the range of values across 5 runs (2/2).

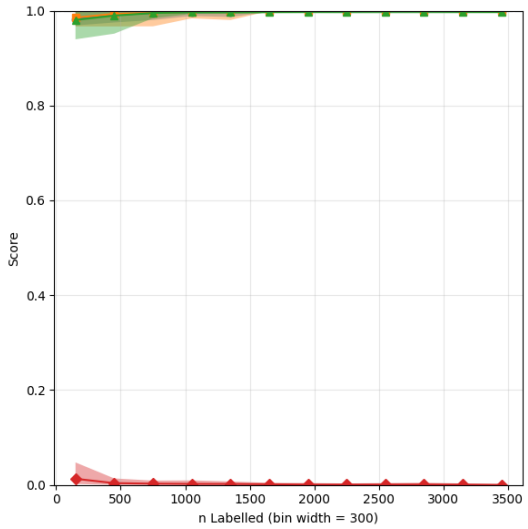
Blocking Threshold = 0.5



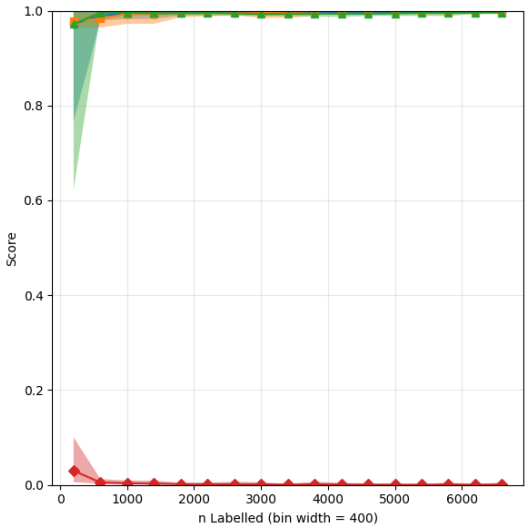
(a) Random Forest Uncertainty



(b) XGBoost Uncertainty

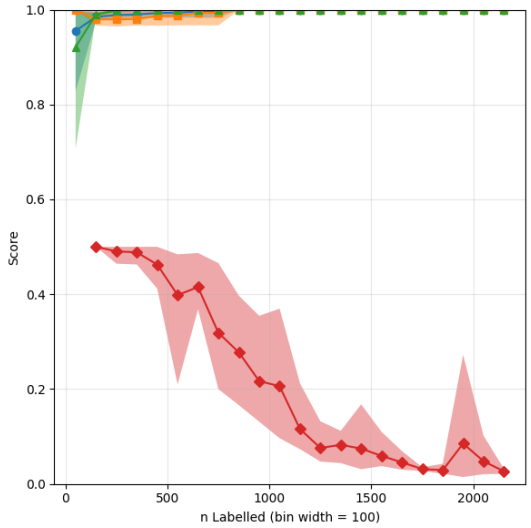


(c) Random Forest Random

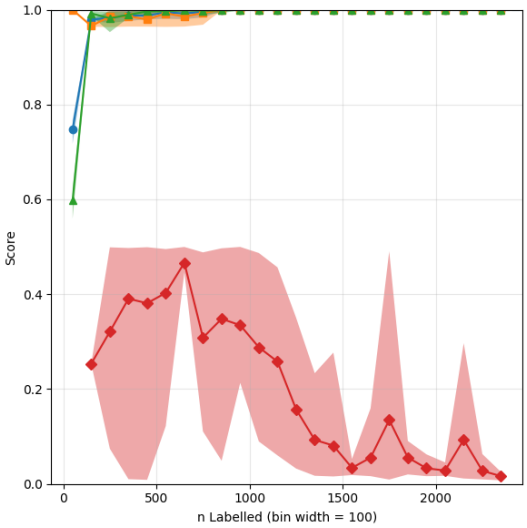


(d) XGBoost Random

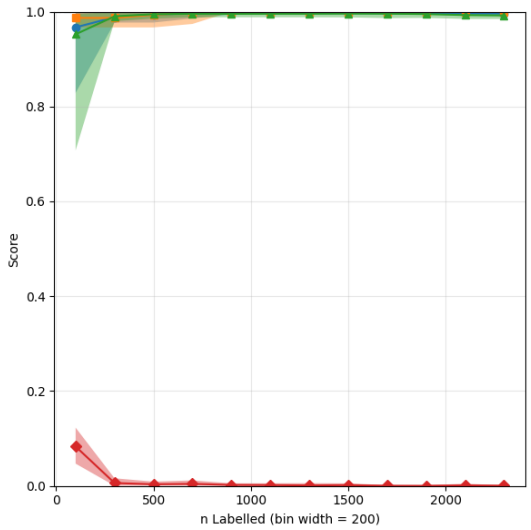
Figure K.5.: Line graphs plotting performance against number of labelled pairs for an AL system performing sibling bundling from birth records on the TD100k (Clean) dataset blocked at threshold 0.5. Horizontal dashed line represents a benchmark from passive learning and shaded areas represent the range of values across 5 runs (1/2).



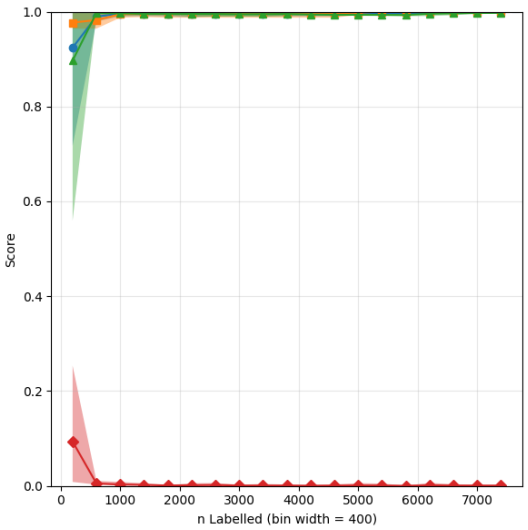
(e) Random Forest Uncertainty Rebalanced



(f) XGBoost Uncertainty Rebalanced



(g) Random Forest Random Rebalanced



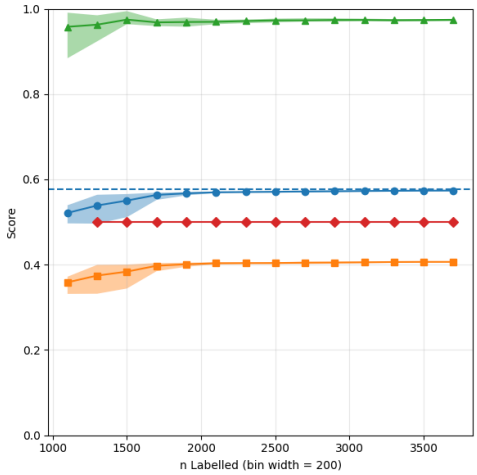
(h) XGBoost Random Rebalanced

Figure K.5.: Line graphs plotting performance against number of labelled pairs for an AL system performing sibling bundling from birth records on the TD100k (Clean) dataset blocked at threshold 0.5. Horizontal dashed line represents a benchmark from passive learning and shaded areas represent the range of values across 5 runs (2/2).

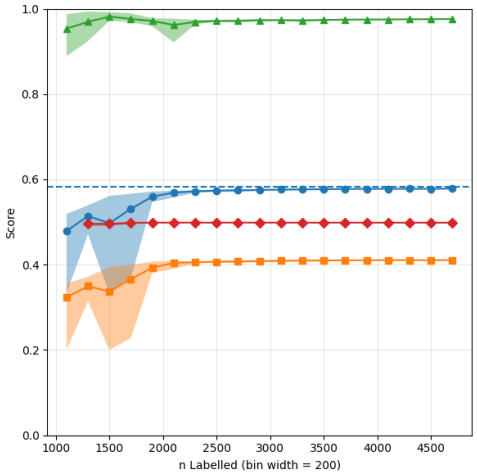
TD100k (Corrupt)

Birth-Death Identity Linkage

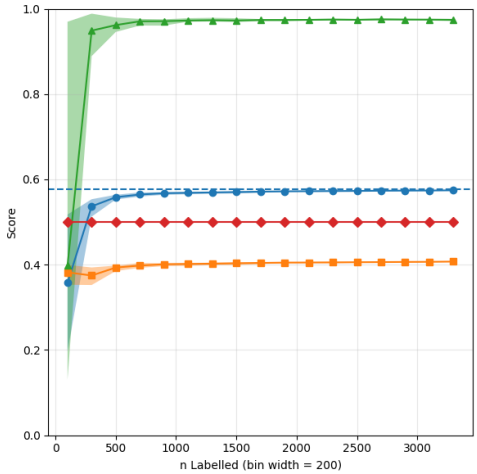
Blocking Threshold = 0.35



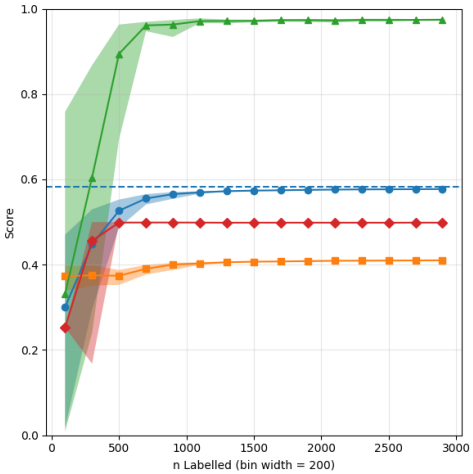
(a) Random Forest Uncertainty



(b) XGBoost Uncertainty



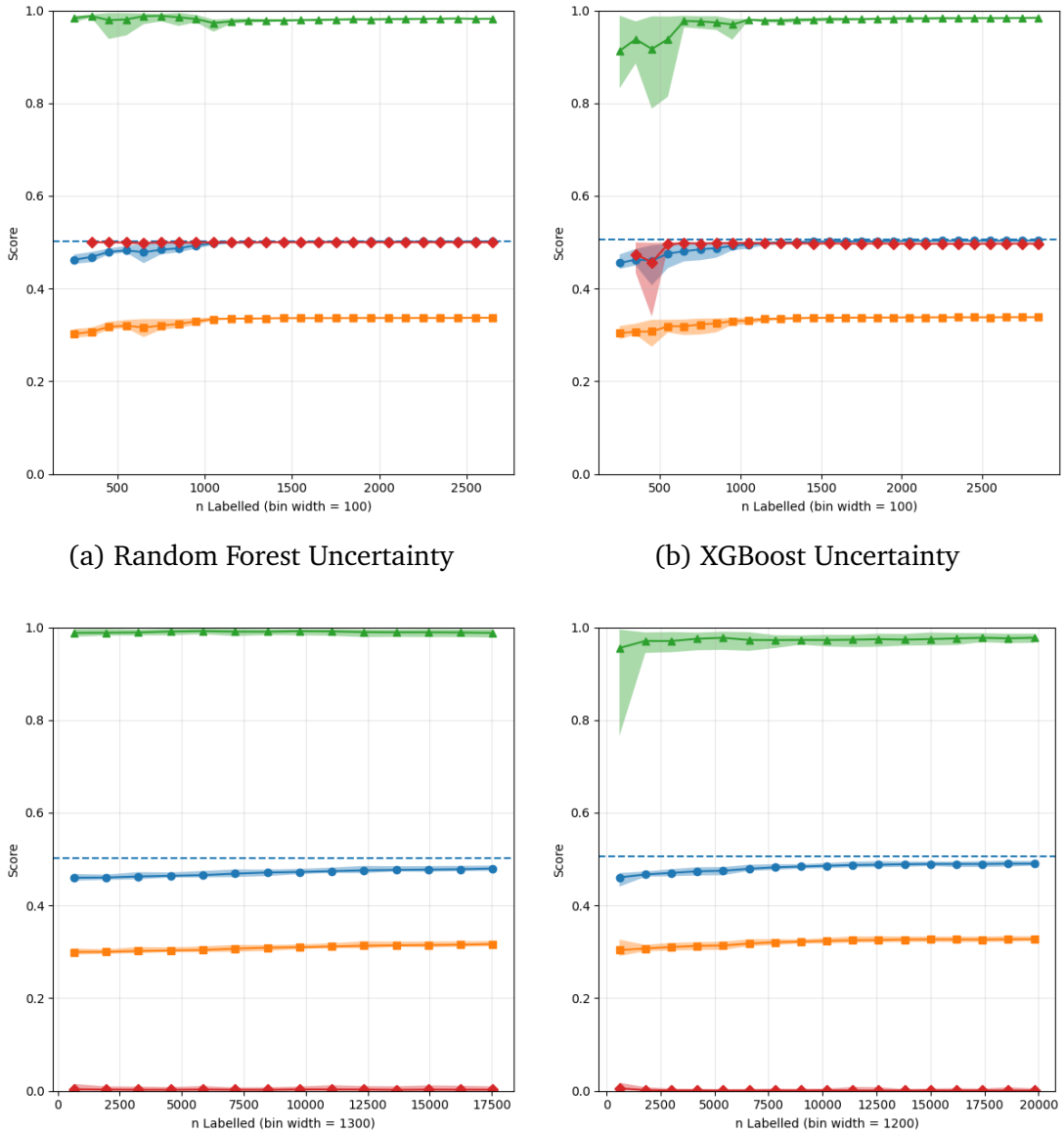
(c) Random Forest Uncertainty Rebalanced



(d) XGBoost Uncertainty Rebalanced

Figure K.6.: Line graphs plotting performance against number of labelled pairs for an AL system performing birth-death identity linkage on the TD100k (Corrupt) dataset blocked at threshold 0.35. Shaded areas represent the range of values across 5 runs.

Blocking Threshold = 0.5



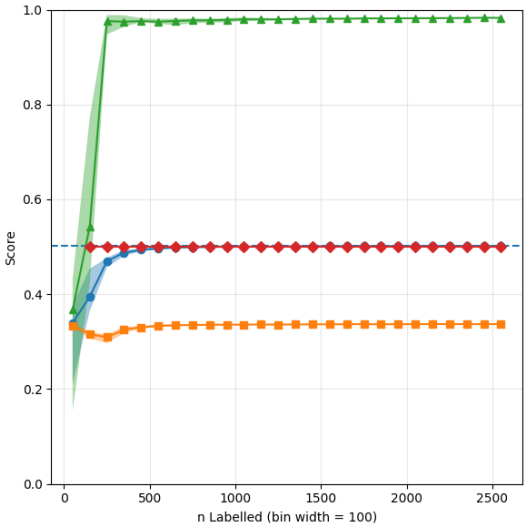
(a) Random Forest Uncertainty

(b) XGBoost Uncertainty

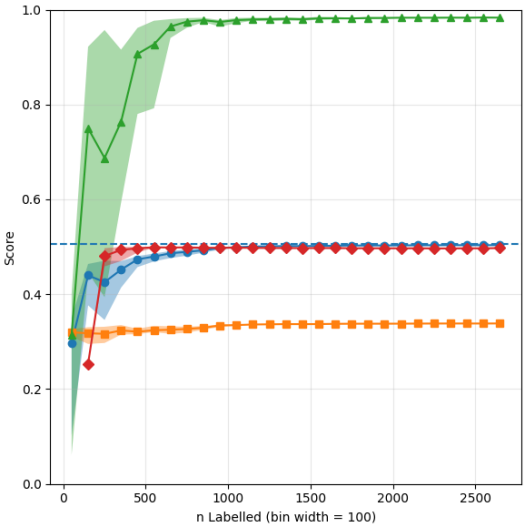
(c) Random Forest Random

(d) XGBoost Random

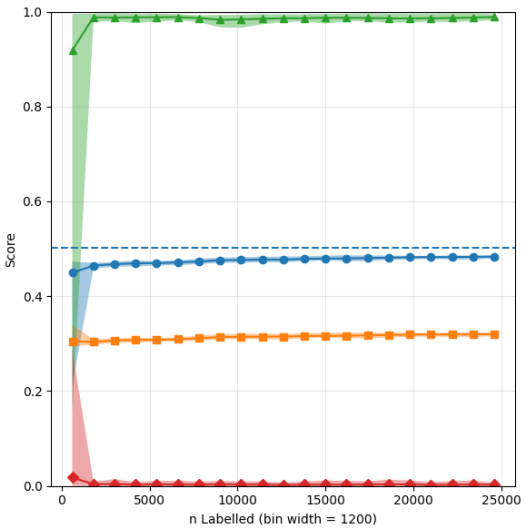
Figure K.7.: Line graphs plotting performance against number of labelled pairs for an AL system performing birth-death identity linkage on the TD100k (Corrupt) dataset blocked at threshold 0.5. Horizontal dashed line represents a benchmark from passive learning and shaded areas represent the range of values across 5 runs (1/2).



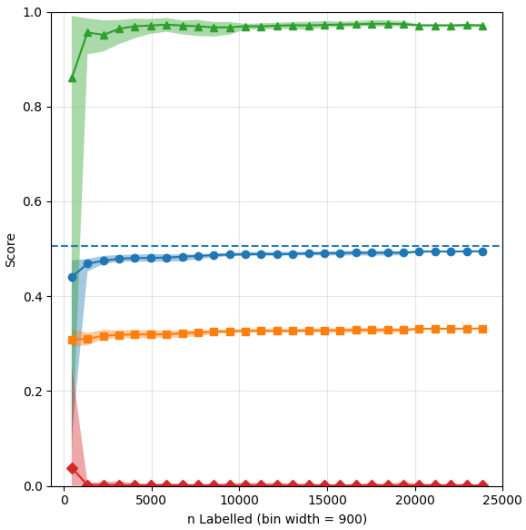
(e) Random Forest Uncertainty Rebalanced



(f) XGBoost Uncertainty Rebalanced



(g) Random Forest Random Rebalanced

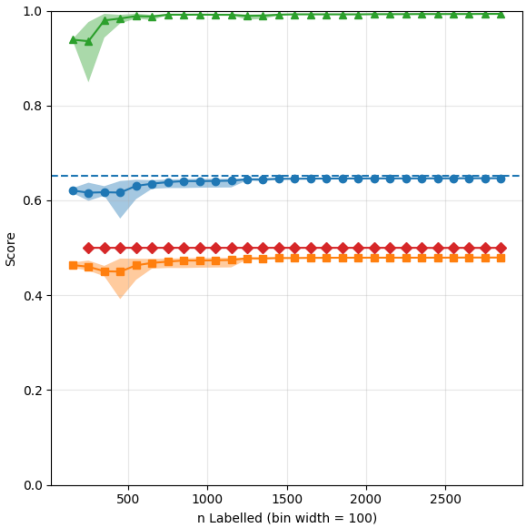


(h) XGBoost Random Rebalanced

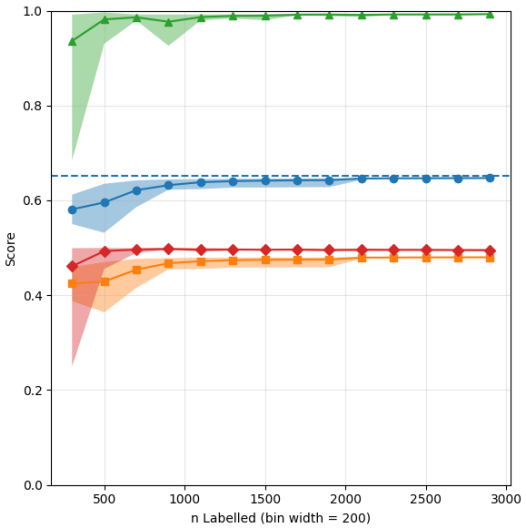
Figure K.7.: Line graphs plotting performance against number of labelled pairs for an AL system performing birth-death identity linkage on the TD100k (Corrupt) dataset blocked at threshold 0.5. Horizontal dashed line represents a benchmark from passive learning and shaded areas represent the range of values across 5 runs (2/2).

Sibling Bundling from Birth Records

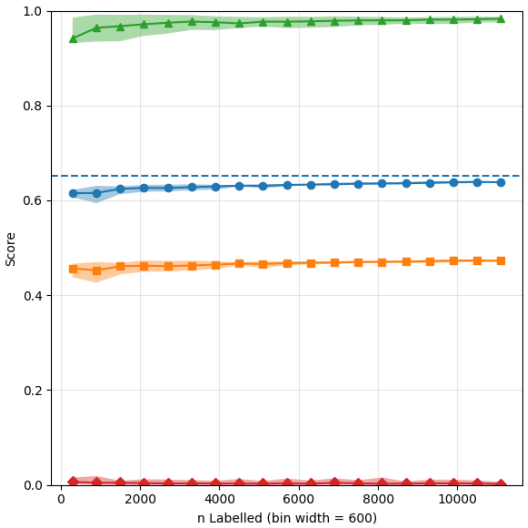
Blocking Threshold = 0.35



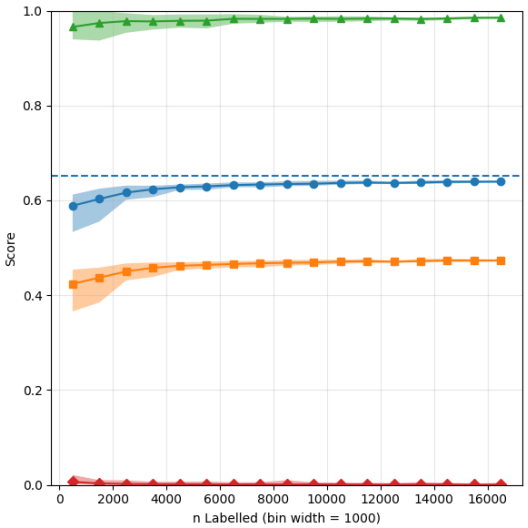
(a) Random Forest Uncertainty



(b) XGBoost Uncertainty

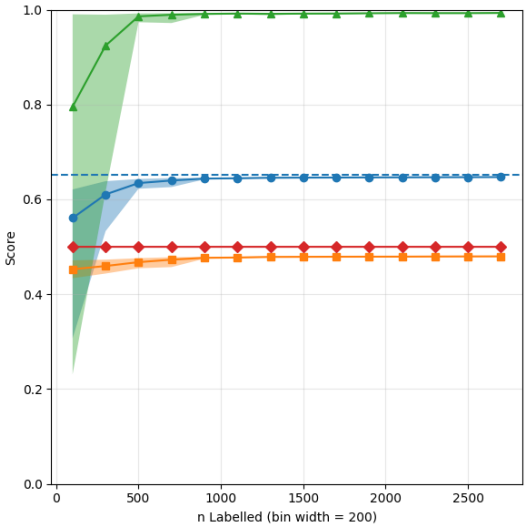


(c) Random Forest Random

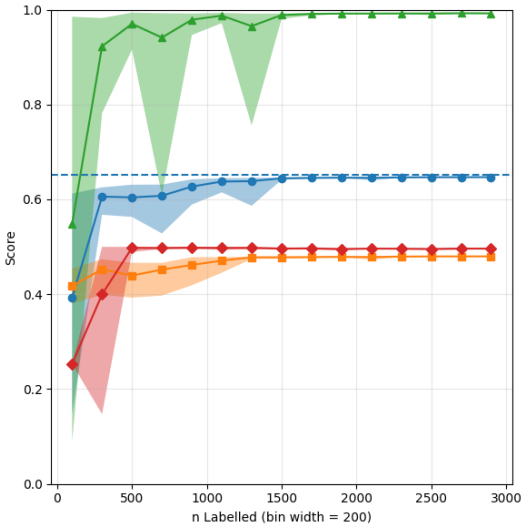


(d) XGBoost Random

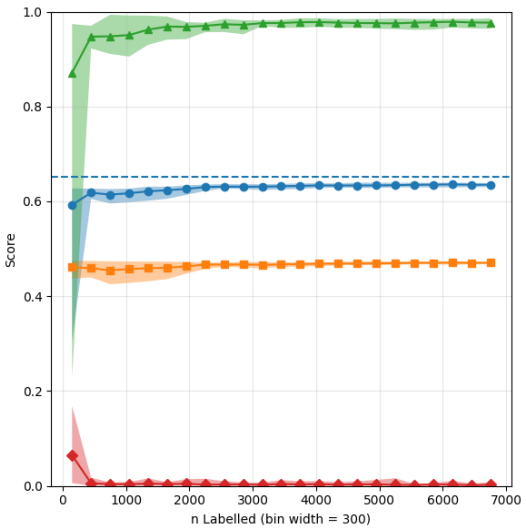
Figure K.8.: Line graphs plotting performance against number of labelled pairs for an AL system performing sibling bundling from birth records on the TD100k (Corrupt) dataset blocked at threshold 0.35. Horizontal dashed line represents a benchmark from passive learning and shaded areas represent the range of values across 5 runs (1/2).



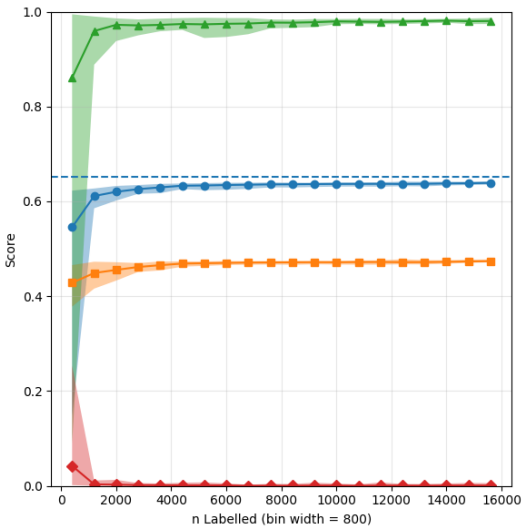
(e) Random Forest Uncertainty Rebalanced



(f) XGBoost Uncertainty Rebalanced



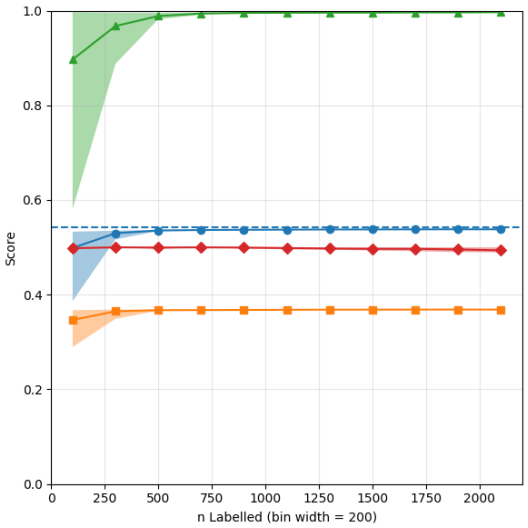
(g) Random Forest Random Rebalanced



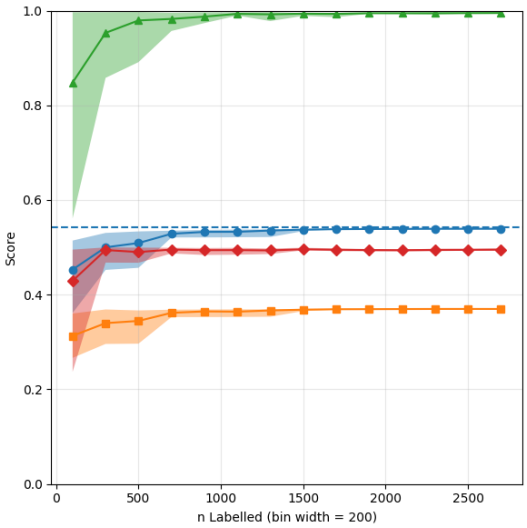
(h) XGBoost Random Rebalanced

Figure K.8.: Line graphs plotting performance against number of labelled pairs for an AL system performing sibling bundling from birth records on the TD100k (Corrupt) dataset blocked at threshold 0.35. Horizontal dashed line represents a benchmark from passive learning and shaded areas represent the range of values across 5 runs (2/2).

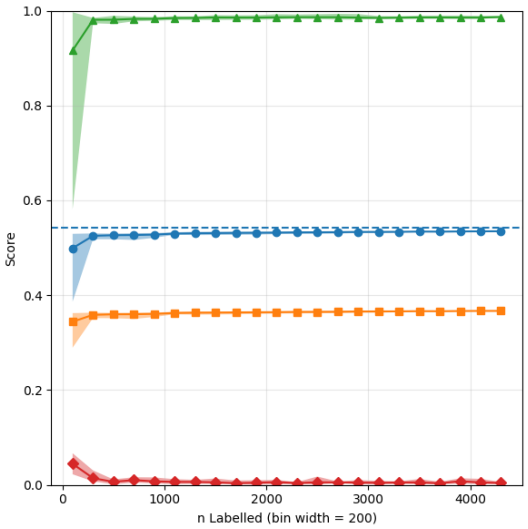
Blocking Threshold = 0.5



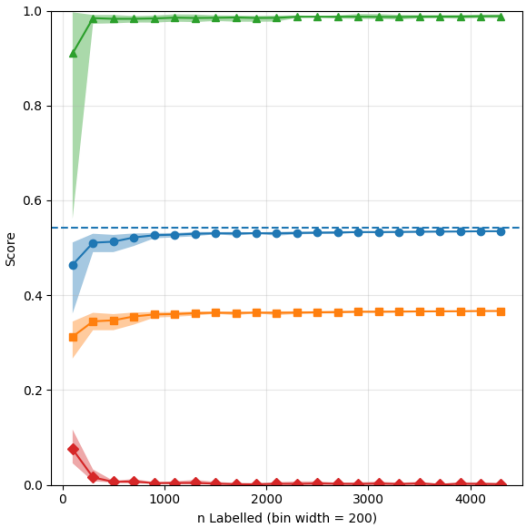
(a) Random Forest Uncertainty



(b) XGBoost Uncertainty

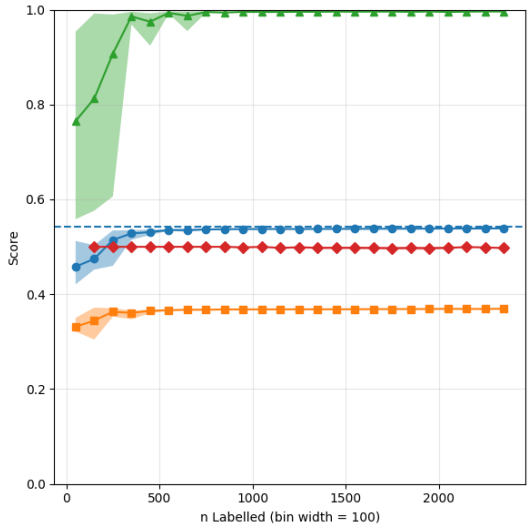


(c) Random Forest Random

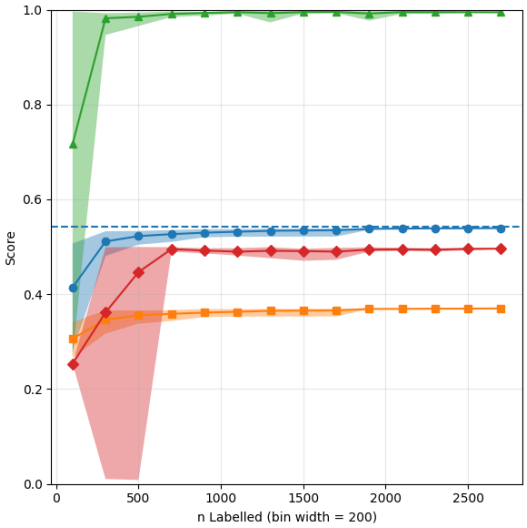


(d) XGBoost Random

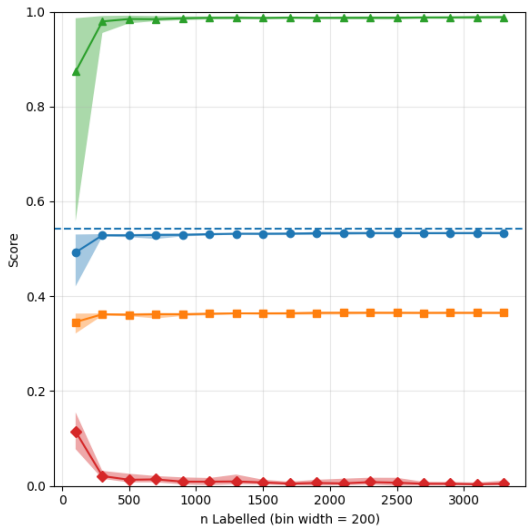
Figure K.9.: Line graphs plotting performance against number of labelled pairs for an AL system performing sibling bundling from birth records on the TD100k (Corrupt) dataset blocked at threshold 0.5. Horizontal dashed line represents a benchmark from passive learning and shaded areas represent the range of values across 5 runs (1/2).



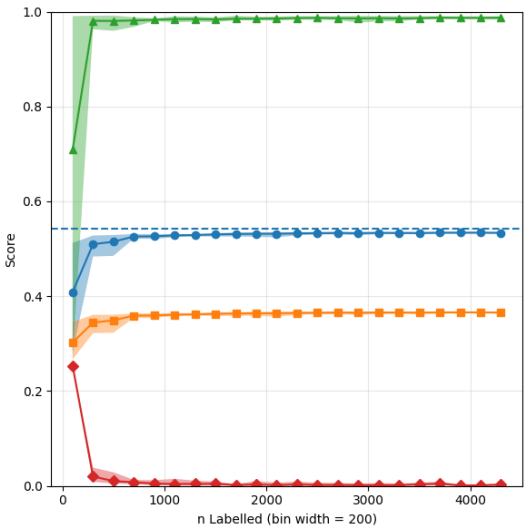
(e) Random Forest Uncertainty Rebalanced



(f) XGBoost Uncertainty Rebalanced



(g) Random Forest Random Rebalanced



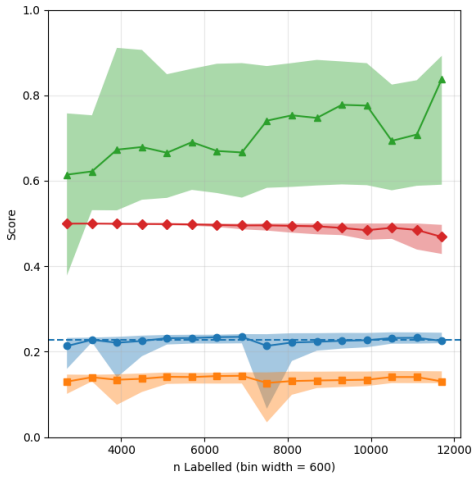
(h) XGBoost Random Rebalanced

Figure K.9.: Line graphs plotting performance against number of labelled pairs for an AL system performing sibling bundling from birth records on the TD100k (Corrupt) dataset blocked at threshold 0.5. Horizontal dashed line represents a benchmark from passive learning and shaded areas represent the range of values across 5 runs (2/2).

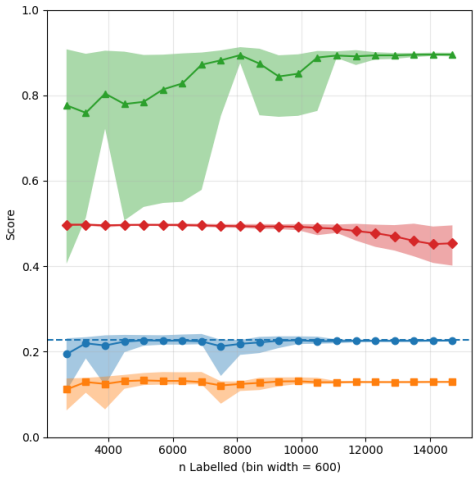
POPLINK

Birth-Death Identity Linkage

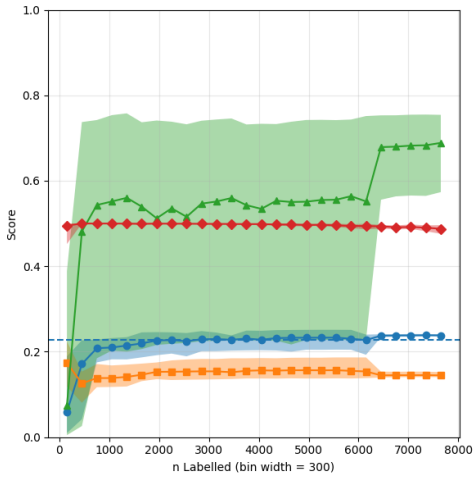
Blocking Threshold = 0.35



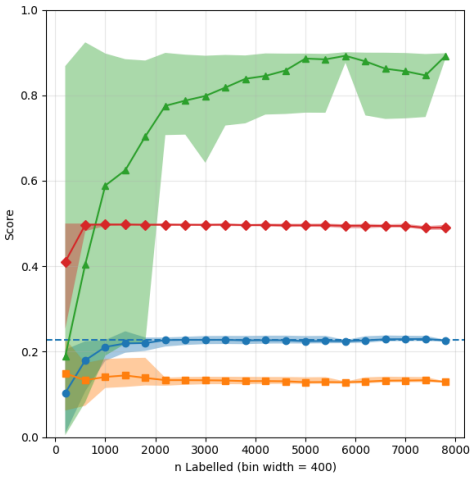
(a) Random Forest Uncertainty



(b) XGBoost Uncertainty



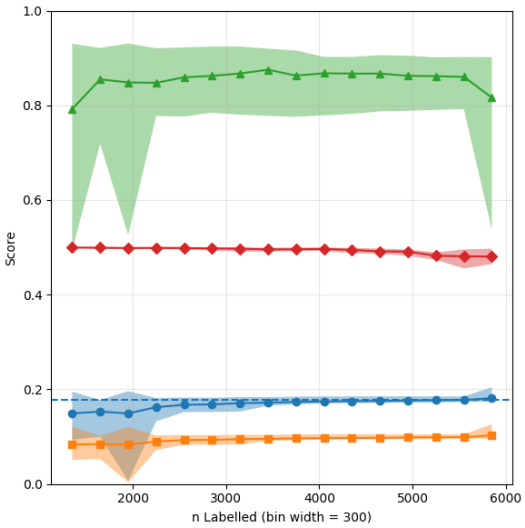
(c) Random Forest Uncertainty Rebalanced



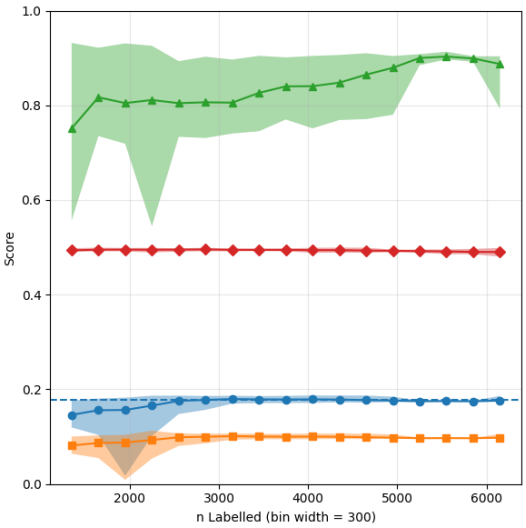
(d) XGBoost Uncertainty Rebalanced

Figure K.10.: Line graphs plotting performance against number of labelled pairs for an AL system performing birth-death identity linkage on the UMEA dataset blocked at threshold 0.35. Shaded areas represent the range of values across 5 runs.

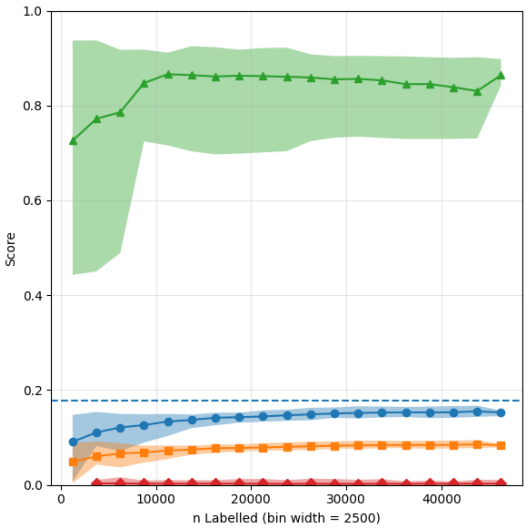
Blocking Threshold = 0.5



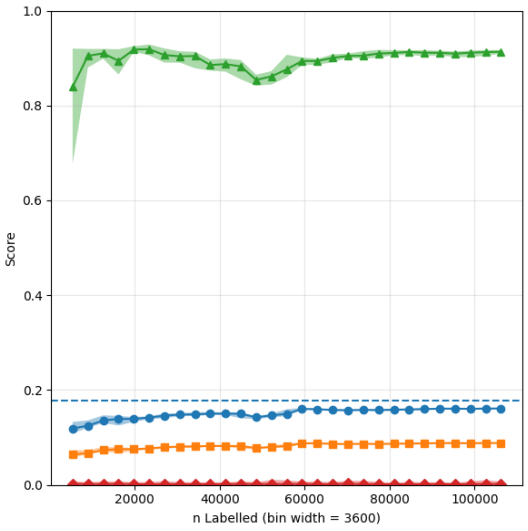
(a) Random Forest Uncertainty



(b) XGBoost Uncertainty

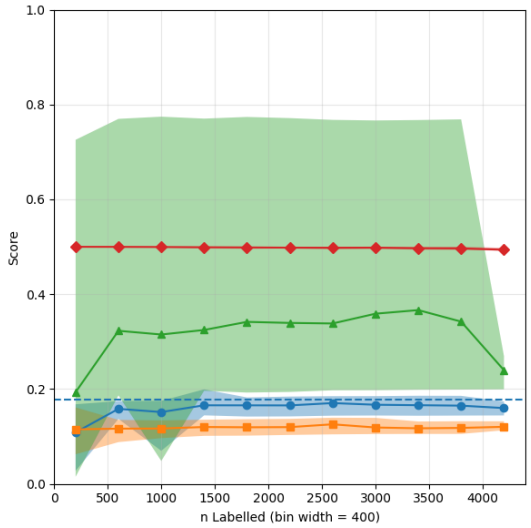


(c) Random Forest Random

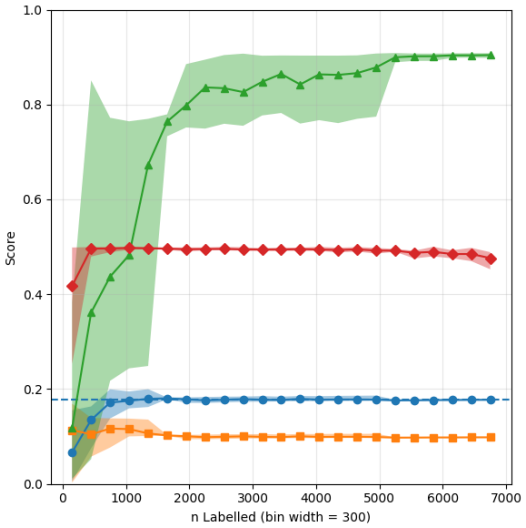


(d) XGBoost Random

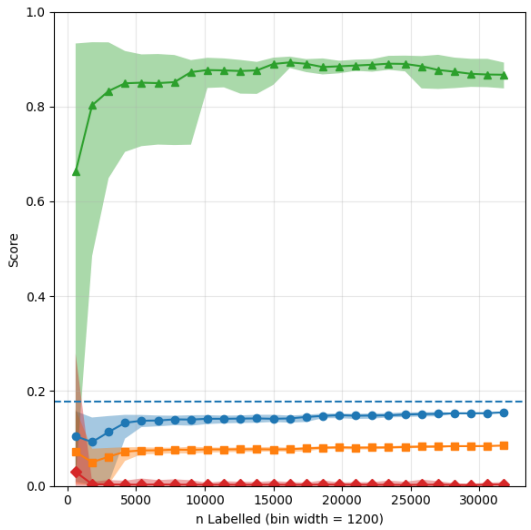
Figure K.11.: Line graphs plotting performance against number of labelled pairs for an AL system performing birth-death identity linkage on the UMEA dataset blocked at threshold 0.5. Horizontal dashed line represents a benchmark from passive learning and shaded areas represent the range of values across 5 runs (1/2).



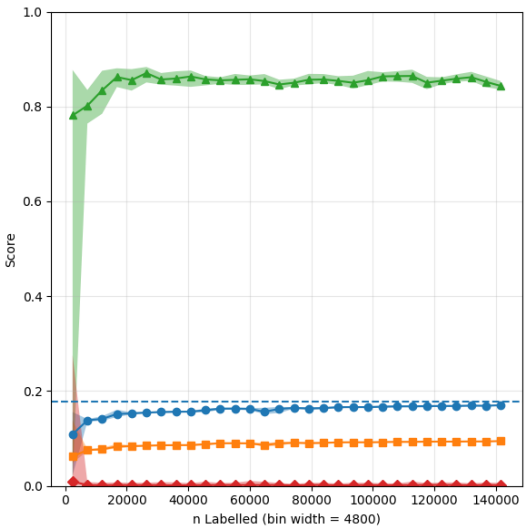
(e) Random Forest Uncertainty Rebalanced



(f) XGBoost Uncertainty Rebalanced



(g) Random Forest Random Rebalanced

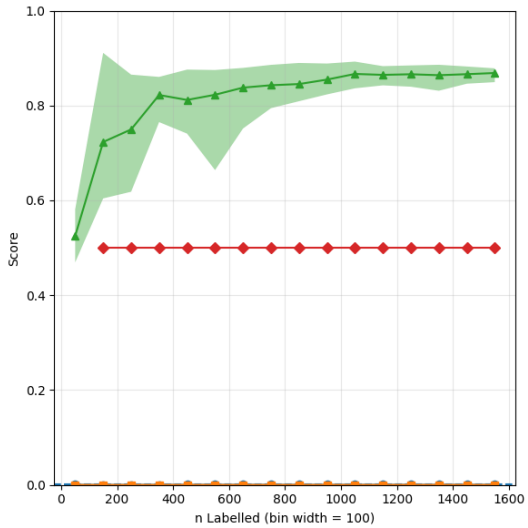


(h) XGBoost Random Rebalanced

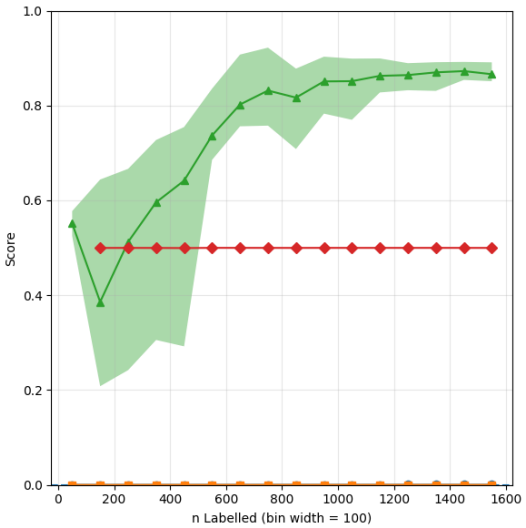
Figure K.11.: Line graphs plotting performance against number of labelled pairs for an AL system performing birth-death identity linkage on the UMEA dataset blocked at threshold 0.5. Horizontal dashed line represents a benchmark from passive learning and shaded areas represent the range of values across 5 runs (2/2).

Sibling Bundling from Birth Records

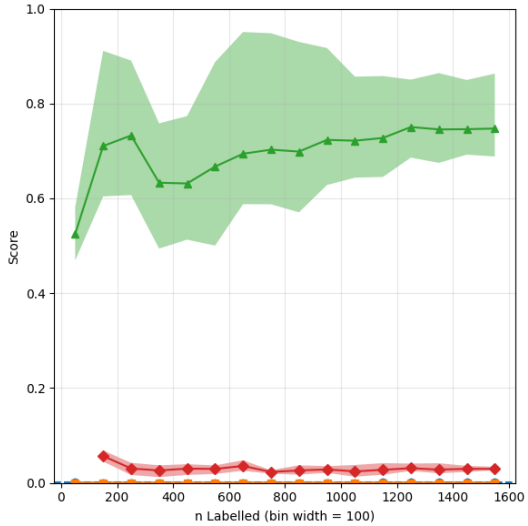
Blocking Threshold = 0.35



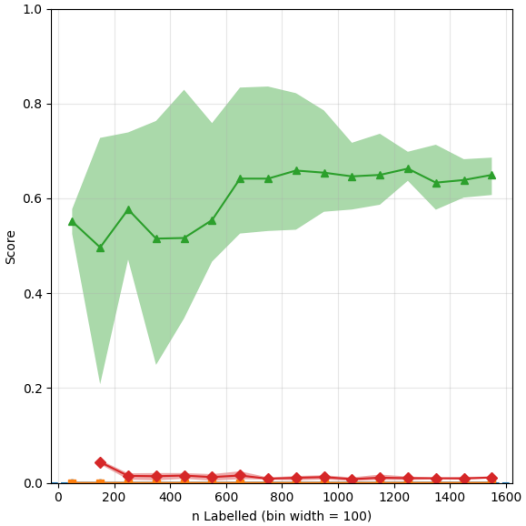
(a) Random Forest Uncertainty



(b) XGBoost Uncertainty

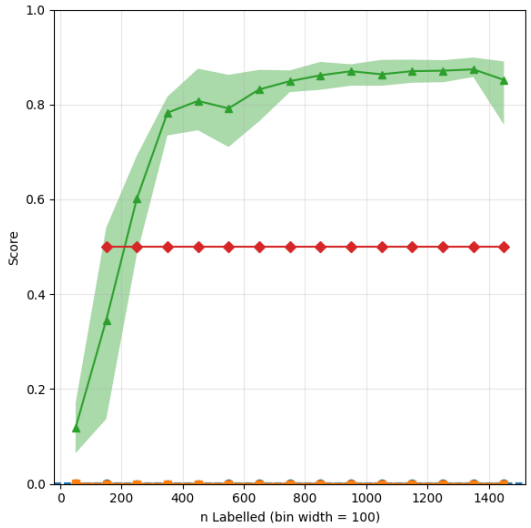


(c) Random Forest Random

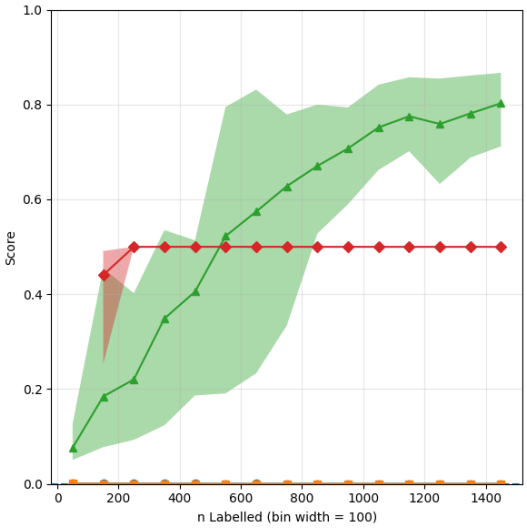


(d) XGBoost Random

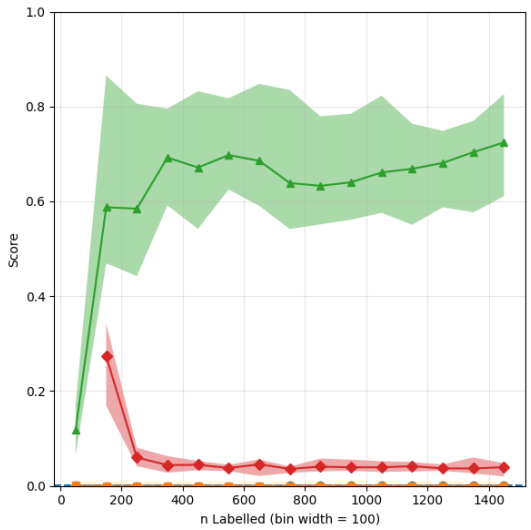
Figure K.12.: Line graphs plotting performance against number of labelled pairs for an AL system performing sibling bundling from birth records on the POPLINK dataset blocked at threshold 0.35. Horizontal dashed line represents a benchmark from passive learning and shaded areas represent the range of values across 5 runs (1/2).



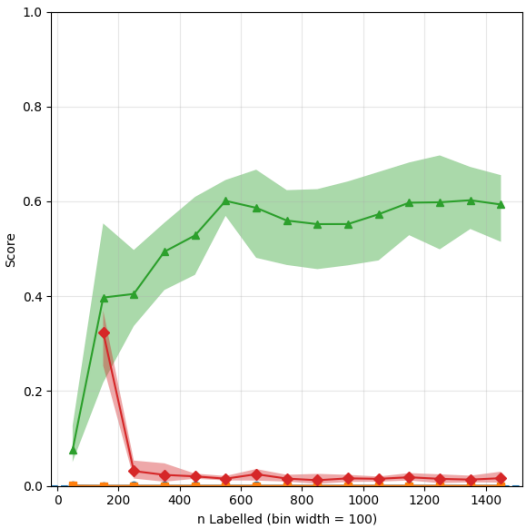
(e) Random Forest Uncertainty Rebalanced



(f) XGBoost Uncertainty Rebalanced



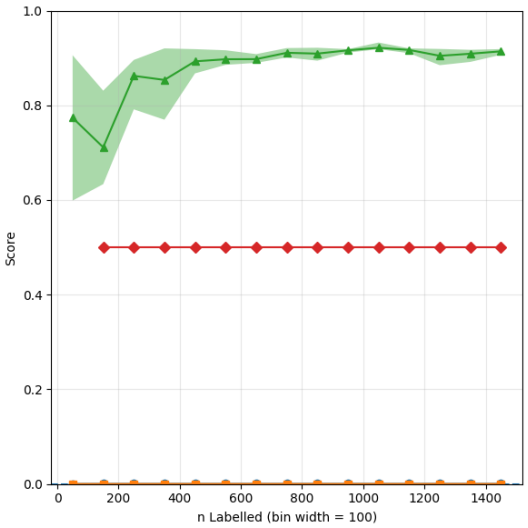
(g) Random Forest Random Rebalanced



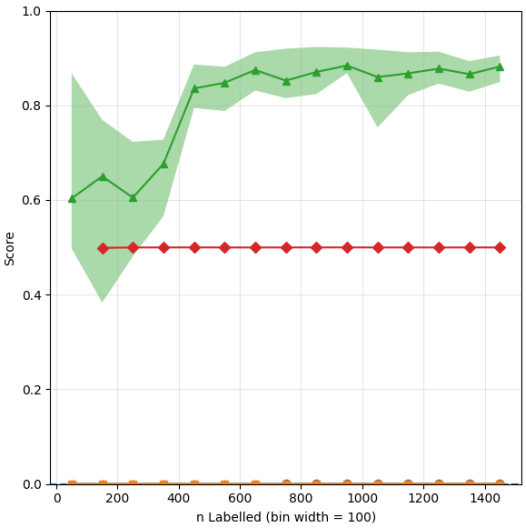
(h) XGBoost Random Rebalanced

Figure K.12.: Line graphs plotting performance against number of labelled pairs for an AL system performing sibling bundling from birth records on the POPLINK dataset blocked at threshold 0.35. Horizontal dashed line represents a benchmark from passive learning and shaded areas represent the range of values across 5 runs (2/2).

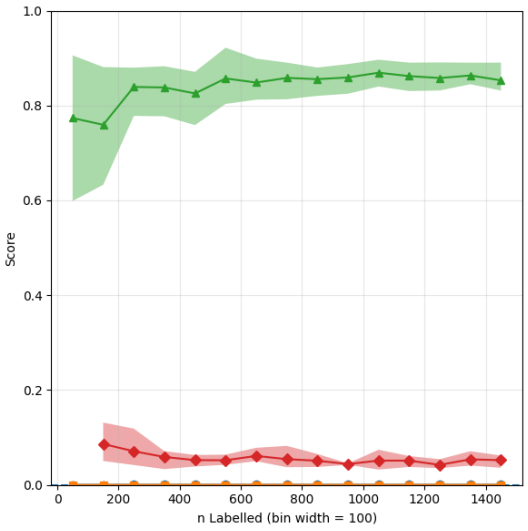
Blocking Threshold = 0.5



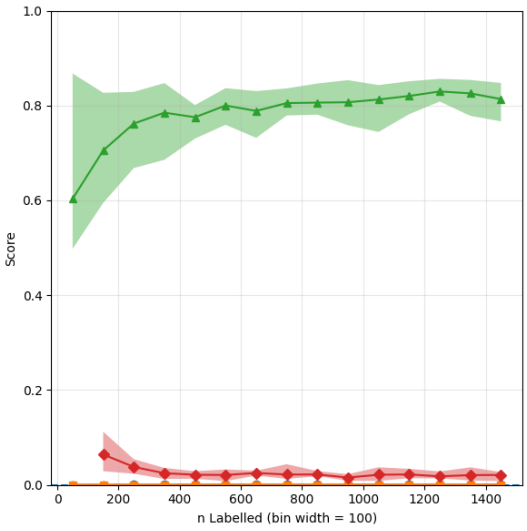
(a) Random Forest Uncertainty



(b) XGBoost Uncertainty

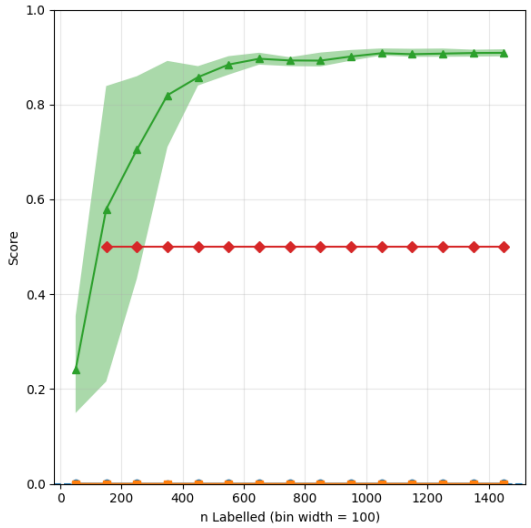


(c) Random Forest Random

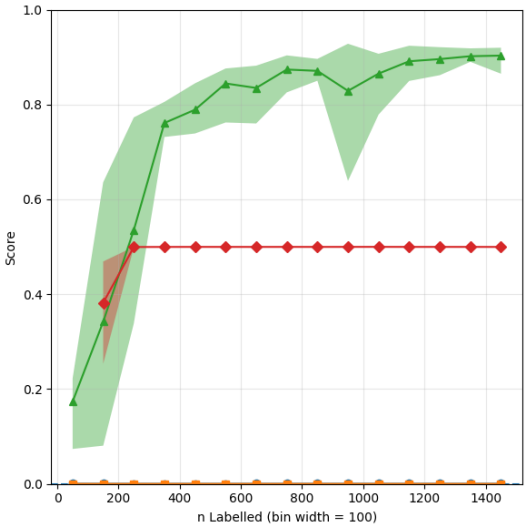


(d) XGBoost Random

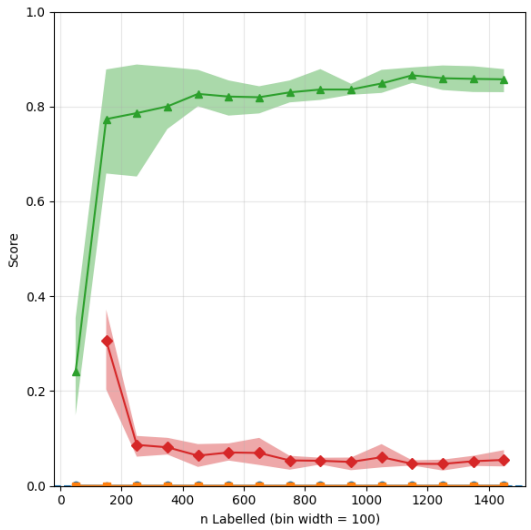
Figure K.13.: Line graphs plotting performance against number of labelled pairs for an AL system performing sibling bundling from birth records on the POPLINK dataset blocked at threshold 0.5. Horizontal dashed line represents a benchmark from passive learning and shaded areas represent the range of values across 5 runs (1/2).



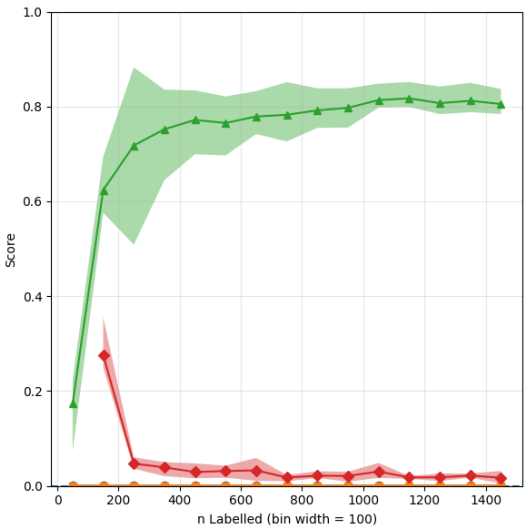
(e) Random Forest Uncertainty Rebalanced



(f) XGBoost Uncertainty Rebalanced



(g) Random Forest Random Rebalanced



(h) XGBoost Random Rebalanced

Figure K.13.: Line graphs plotting performance against number of labelled pairs for an AL system performing sibling bundling from birth records on the POPLINK dataset blocked at threshold 0.5. Horizontal dashed line represents a benchmark from passive learning and shaded areas represent the range of values across 5 runs (2/2).