BENCHMARKING THE EFFECTIVENESS AND EFFICIENCY OF MACHINE LEARNING ALGORITHMS FOR RECORD LINKAGE

A Thesis

by

GURUDEV ILANGOVAN

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

August 2019

Major Subject: Computer Science

ABSTRACT

Record linkage which refers to the identification of the same entities across several databases in the absence of an unique identifier is a crucial step for data integration. In this research, we study the effectiveness and efficiency of different machine learning algorithms (SVM, Random Forest, and neural networks) to link databases in a controlled experiment. We control for % of heterogeneity in data and size of training dataset. We evaluate the algorithms based on (1) quality of linkages such as F1 score based on a one threshold model and (2) size of uncertain regions that need manual review based on a two threshold model. We find that random forests performed very well both in terms of traditional metrics like F1 score (99.2% - 95.9%) as well as manual review set size (7.1% - 21%) for error rates from 0% to 60%. Though in terms of F1 scores, the algorithms (Random Forests, SVMs and Neural Nets) fared fairly similar, random forests outperformed the next best model by 28% on average in terms of the percentage of pairs that need manual review.

DEDICATION

To my advisor, mentor and well-wisher, Dr. Hye-Chung Kum whose patience and support made this work possible.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a thesis committee consisting of Hye Chung Kum and Dilma da Silva of the Department of Computer Science and Mark Fossett of the Department of Sociology.

All other work conducted for the thesis was completed by the student independently.

Funding Sources

This work was funded in part by Patient Centered Outcomes Research Institute (PCORI) methods program contract ME1602-34486.

NOMENCLATURE

DNN	Dense Neural Network
SVM	Support Vector Machine
RF	Random Forest
NC	North Carolina
DOB	Date of Birth
ML	Machine Learning

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Record linkage often called, entity resolution [1] or de-duplication [2], refers to identifying the same entities across several databases. It is a well known problem in data mining that has been researched for over 50 years. Whenever there is a unique field to identify the same records in the databases, the problem is trivially easy and solved by simple joins [3]. However, challenges arise when there is no unique field present (or when fields are inaccessible due to privacy concerns). In such scenarios, we need methods that can link the databases reliably so that subsequent analyses can be performed on the linked data, downstream. Record linkage is therefore, a crucial step in the data cleaning process [4].

At the risk of oversimplification, record linkage essentially is a problem of looking at a pair of records and deciding whether they refer to the same entity. There are a variety of approaches to accomplish this. Manual linkage frameworks involve a human looking at the pairs and making the decisions. While very reliable, this method is prohibitively expensive and time consuming when it comes to large databases as the man hours it would take to resolve the linkages is very high. It hence does not scale very well to large databases. Automatic methods involve using different kinds of algorithms to perform the linkage. While automatic methods may not have the high degree of reliability of manual link resolution, they are faster and much more scalable. Initially, rule based approaches and probabilistic methods were used but machine learning approaches are rapidly gaining traction in the record linkage space and proving to be the preferred automatic linkage methods.

There has been a lot of research studying machine learning in the record linage space in different contexts. For instance, a radial basis kernel SVM was used to link genealogy records from 19th century Canada and showed promising results [3]. Another study [5] compared SVM, random forest, logistic regression and other heuristic approaches on US census datasets. A lot of work involving random forests for record linkage in financial entity recognition [6] and author disambiguation [7] shows promising results about the efficiency for random forests for this task. With the recent re-emergence of neural networks, a lot of research shows potential for neural networks in entity recognition. Using structured neural networks for genealogical record linkage was shown to give very reliable results [8] [9].

These works studied the performance of different algorithms in very specific contexts. However results that hold in one domain need not hold in another domain as the databases being linked vary in a lot of ways. Two primary differences between different record linkage settings are the amount of heterogeneity between the databases and the amount of data available to train. For instance, genealogy records often tend to be very dirty as they are the results of digitizing handwritten documents, the setting being rife for typographical errors. The effort it takes to generate good training data in such settings can be huge as expert sociologists must manually resolve the linkages first. On the contrary, modern databases can be way cleaner and generating training data can be comparatively easier.

This research focuses on systematically studying the performances of three popular machine learning algorithms for record linkage - random forests, SVMs (RBF kernel) and Dense Neural Networks in the two important aforementioned data dimensions - heterogeneity rate and amount of training data. The performance is evaluated using two measures - F_1 score and the *percentage of database that require manual review*.

Typically, the F_1 score has been a standard measure of algorithm performance in the record linkage space in fully automatic frameworks. Since record linkage data is inherently skewed, the F_1 score can be a very useful measure for evaluation. Increasingly it is becoming common to perform the linkage by a hybrid two pass approach where manual resolution is employed for ambiguous pairs [10] [11]. It is based on the fact that inherently, not all of the decisions to be made have the same level of complexity. Some decisions are markedly easier than the rest. The first pass would involve the use of an automatic record linkage framework to resolve as many linkages as possible such that a quality threshold among the resolved linkages is met. Manual resolution would then be required for the pairs, the automatic algorithm is not confident about. Such a two-pass approach aims to make sure that the majority of the linkages are resolved quickly and that manual intervention is minimized as much as possible and focuses on cases that require human judgment. To evaluate the usability of the methods being studied in hybrid frameworks, the percentage of pairs that are assigned for manual review is also measured. This metric has the benefit of being very interpretable and directly usable in a real world system.

Thus the research questions that we are specifically interested in answering would be how the amount of training data affects the performance of different algorithms, how the error rate affects the performance of different algorithms and the trade-off between the quality threshold for the automatic algorithms and the percentage of manual review. We link snapshots of the North Carolina voter registry data 4 years apart as our base data sets. A flexible error generator that can generate common record linkage errors is used to introduce noise into the data systemically to conduct a controlled experiment. In the experiment, the three models are trained for each of these error rates at different training set sizes and each of these conditions is evaluated and studied.

Section 2, briefly surveys the existing literature on record linkage and using machine learning for record linkage. Section 3 delves into the methodology and section 4 describes the key findings. We briefly discuss a few considerations in section 5 and conclude the paper in section 6.

2. RELATED WORK

As the amount of data that is generated grows at an exponential rate, it becomes increasingly important to be able to integrate data from several sources to perform richer analyses and make more intelligent decisions. In error free clean databases with unique identifiers common to all the databases, integrating them can be easily accomplished with the help of simple joins. [12]

However, such identifiers are not always available due to a lot of reasons - they may be withdrawn because of privacy regulations, the source databases might be totally unrelated and such a identifier might not exist, etc. In such scenarios, the available fields common to the databases are compared and a decision as to whether the two records refer to the same entity has to be made. This process is referred to as record linkage [13]. Ironically enough, record linkage goes by a lot of alternate terms - record matching, entity resolution, data deduplication and data matching depending on the field. Record linkage is an important preprocessing step in the data analysis pipeline as the quality of linkage considerably influences the decisions made using subsequent analysis downstream. Hence, it is important to study how different factors affect the process of record linkage in order to achieve better linkage performance.

The problem of record linkage has been studied for over sixty years. In this section, we briefly survey the different approaches to record linkage. Record linkage approaches can broadly be classified into two types - manual record linkage and automatic record linkage.

Manual record linkage refers to experts in record linkage manually reviewing pairs of records from the two databases and making decisions as to whether the pair of records refers to the same entity. This process generally tends to be very accurate [14] but is inherently slow and not scalable to larger databases where the number of pairs that may need to be reviewed maybe large .

2.1 Automatic Record Linkage

Automatic record linkage refers to the usage of algorithms to automate the process of record linkage. While automatic methods may not be as accurate as manual methods, they are several orders of magnitude faster and much more scalable. Also, typically automatic record linkage methods rely on the existence of “labeled data”, data being some pairs of records and labels being whether or not the pairs refer to the same entity. Labeled data may not be easily available and it is often a challenge to acquire the data necessary for automatic algorithms.

2.2 Extracting Features for Automatic Record Linkage

Since a labeled sample comes in the form of a pair of fields, methods must be designed to extract a feature vector from them. In this section, methods to extract features from string fields are discussed. Even for date and numeric fields, the dates or the components of the dates and numbers can be considered as strings and such distance based features can be extracted [12]. Many a time, the features that are extracted depends on the problem at hand. Since most of the string fields used in this study had a single token, the Damerau-Levenshtein distance, the Jaro-Winkler distance and Soundex are discussed.

2.2.1 Damerau-Levenshtein Distance

The original edit distance (also called as the Levenshtein distance) between two strings s_1 and s_2 is the minimum number of edit operations needed to transform s_1 to s_2 (or s_2 to s_1) where the edit operations are insertion of a character, deletion of a character and replacement of a character [15].

The Damerau-Levenshtein distance is a slighly modified version of the edit distance that has an addional edit operation, transposition where two adjascent characters have their positions swapped. The Damerau-Levenshtein distance can be calculated using a program that has $O(|s_1|.|s_2|)$ time complexity where $|s|$ is the length of a string $|s|$. Equation 2.1 shows us the Damerau-Levenshtein calculation from the string “gifts” to “profit” in five steps using the four edit operations described. The Damerau-Levenshtein between the strings “gifts” to “profit” is thus 5.

gifts => pgifts (insertion of 'p') (2.1)

pgifts => prgifts (insertion of 'r')

prgifts => proifts (substitution of 'g' to 'o')

proifts => profits (transposition of 'if' to 'fi')

profits => profit (deletion of 's')

In the paper where the Damerau-Levenshtein distance was proposed, the author argued that 80% of human typographical errors can be captured and corrected by this distance [16]. This distance performs well for typographical errors but not for other types of mismatches.

2.2.2 Jaro-Winkler Similarity

The Jaro distance [17] between two strings s_1 and s_2 is given by the formula,

$sim_j = \frac{1}{3}(\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{m})$

where $|s|$ represents the length of the string s, m is the number of matching characters and t is half the number of transpositions. Two characters from s_1 and s_2 are considered matching if they are no farther than $\frac{max(|s_1|,|s_2|)}{2} - 1$.

The Jaro-Winkler similarity uses a weighting factor for a "prefix", p for a "prefix length", l. It is calculated using the formula, $sim_{jw} = sim_j + l.p.(1 - sim_j)$

In general, l is fixed to be the number of characters, from the start of the input strings to the first character mismatch between the two strings, with a maximum of four and p is fixed to be 0.1 [18].

2.2.3 Soundex

Soundex is one of the most widely used phonetic encoding schemes where phonetic codes are assigned to a string [19]. Soundex codes are shown to work very well for phonetically clustering names and particularly Caucasian surnames [20]. For example, the soundex code for “Robert” is R163, the same as that of the code for “Rupert” while “Rubin” yields R150.

The algorithm to calculate a soundex code is as follows:

1. Keep the first letter of the surname as the prefix letter and completely ignore all occurrences of W and H in other positions.
2. Assign the following codes to the remaining letters:
    - B, F, P, V - 1
    - C, G, J, K, Q, S, X, Z - 2
    - D, T - 3
    - L - 4
    - M, N - 5
    - R - 6
3. A, E, I, O, U, and Y are not coded but serve as separators.
4. Consolidate sequences of identical codes by keeping only the first occurrence of the code.
5. Drop the separators.
6. Keep the letter prefix and the three first codes, padding with zeros if there are fewer than three codes

Once the soundex codes are extracted, any of the string distance measures can be used to calculate the similarity. The soundex codes are also typically used as blocking hash functions.

2.3 Deterministic Record Linkage

In the past, automatic record linkage was primarily achieved via deterministic (rule-based) record linkage and probabilistic record linkage methods.Deterministic linkage is found to be generally accurate when done by a domain expert [21]. It involves the careful and meticulous crafting of rules for the problem at hand and is hence, problem specific and not truly scalable. Some modifications to this approach also include using a system to initially generate the rules necessary and then manually tuning those rules to get better performance. Though once, such rules were formed they functioned as automatic linkage methods, they were not truly automatic as this approach typically involved a lot of manual effort.

2.4 Probabilistic Record Linkage

Probabilistic record linkage was the preferred approach to automatic record linkage until machine learning based approaches started taking over. The seminal paper [22] formalizes the probabilistic intuition that record linkage can be viewed as a Bayesian inference problem. Once features was extracted for each of the pairs any pair would have a feature vector. The feature vector $\bar{x}$ is a random variable, the density of which would follow different distributions for the two classes. If the density distribution for each of the classes are known, the problem becomes a Bayesian inference problem. The distributions can be obtained using the expectation maximization algorithm and maximizing the likelihood. If one makes the simplistic assumption that the features in the feature vector are independent of each other, then one would be using the popular *Naive Bayes* classifier [9]. This assumption is seldom true in practice.

2.5 Learning to Link

However, these days deterministic or probabilistic methods are rarely used. When machine learning methods started gaining popularity, the problem of record linkage was framed as a simple binary classification task and supervised algorithms were fed the feature vectors and labels of all the samples and were used to arrive at the linkage decisions. After the advent of machine learning approaches, the algorithms were able to capture more sophisticated record linkage constructs and quickly started outperforming the deterministic and probabilistic methods.

There are many classification algorithms that can be used for the record linkage but three popular algorithms will be discussed in this work. Approaches like trees and random forests were fast but more complicated models like SVM were claimed to have really good performance [23]. Many linkages (especially genealogical linkages) [3][24] show that supervised learning approaches are scalable and are much more reliable than purely probabilistic techniques and achieve excellent model performance. More recently, with the re-emergence of neural networks, using deep learning to perform record linkage is beginning to gain traction. Many neural networks with few hidden layers are achieving results much better than probabilistic linkage [9]. Thus, in this research we compare the performances of neural nets, SVMs and random forests for record linkage at different levels of heterogeneity and different training set size.

2.6 Blocking

As discussed in the previous sections, record linkage can be modeled by supervised algorithms once we have labeled pairs. However, creating the labeled pairs cannot be done by comparing each record in first database with each record in the second. This is because if there are n records in both databases, there would be n^2 pairs [25] out of which there would be a maximum of n matches and a minimum of $n^2 - n$ pairs that are not matches. There are many complexities associated with such a system. One, the computational complexity of calculating the feature vector would be enormous especially if there a lot of features to be calculated. Two, the computational complexity of training the machine algorithms on a large number of samples would also be resource intensive. Three, the class imbalance in such a pairing mechanism would be extremely high giving rise to problems in properly training the algorithms. In such scenarios, corrective measures to counter class imbalance have to be taken which further raises a slew of challenges. Finally, most of the pairs that come out of such a simplistic pairing mechanism would be uninformative.

A lot of approaches are available to greatly reduce the number of pairs that are considered. Blocking has been one of the most used and most reliable pair generation approaches. Typically, in blocking a hash function is applied to a highly discriminative field, the results of which are called as blocks. The fields that have the same blocks are grouped together such that mutually exclusive blocks are created [20]. This process can however, miss a lot of true links [12]. To mitigate this, a multipass blocking approach is followed. In a multipass blocking approach, the same process is repeated multiple times on multiple different fields, thereby substantially reducing the number of missed matches [26].

2.7 Evaluation Metrics

2.7.1 F_1 Score

Record linkage systems are usually evaluated using the F_1 score [27]. Since the pairs used for training and evaluating are usually characterized by class imbalance (pairs that match are outnumbered by those that do not), accuracy is not an appropriate measure. F_1 score however, performs well even with unbalanced data sets. The F_1 score is the harmonic mean between the precision and recall and hence is very useful in evaluating the effectiveness of linkage. Consider a confusion matrix as shown in figure 2.1:

		True link status	
		Match	Non-match
Predicted link status	Match	d (true match)	b (false match)
	Non-match	c (false non-match)	a (true non-match)

Figure 2.1: A typical confusion matrix

$$\begin{aligned} F_1 &= 2.\frac{Precision.Recall}{Precision + Recall} \qquad (2.2) \\ &= \frac{2d}{c+b+2d} \end{aligned}$$

There are however problems with F_1 score. In [28], it is argued that the F_1 score can be expressed as a weighted sum of the precision and recall where the weights are dependent on the linkage algorithm, thus making them unreliable.

However, since it is a commonly used measure in record linkage literature, the F_1 score is included in our evaluation.

2.7.2 Percentage of Manual Review

Though such machine learning algorithms are primarily used in record linkage as automatic linkage algorithms, they could also be used in combination with manual linkage. This hybrid approach would involve two passes where manual resolution is used for doubtful pairs [10] [11]. The machine learning algorithm could be trained and used to give predictions in the first pass. Since the predictions would be given in terms of probabilities, we could define two thresholds between which if a prediction falls, it would be sent to the second pass. This would involve manual review of the ambiguous pairs. The two thresholds are defined such that pairs that are not sent to the second pass meet a performance standard. Such an approach hopes to achieve the sweet spot in the trade-off between scalability (the degree to which the automatic algorithm is used for linkage) and performance (the degree to which manual review is used). In such a system we could measure the performance of the automatic algorithm by the percentage of pairs that are sent to the next pass for a given performance requirement. This is called as the percentage of manual review and is discussed in detail in section 5.2.2. This measure is beginning to be used for evaluation of hybrid approaches [29].

3. RECORD LINKAGE

As discussed in the previous section, machine learning approaches rely on the existence of training data to learn from and test data to evaluate the algorithms on. In this section, the creation of pairs from two databases A and B that share a common unique identifier is discussed. The idea is to use the identifier field just to determine the label for any pair that is generated but not as a feature that goes into the algorithms. Once the pairs are generated, features would be extracted from each pair and fed into the machine learning models. Note that it might seem counter-productive to link databases that share a common unique identifier but once a model is trained and evaluated on a small subset of databases where the ID can be acquired, it can be used to link larger databases where the such an identifier is not available.

3.1 Data Description

It is assumed in this section that generic databases with personal data are being linked. In general, fields that are present in a database with personally identifiable information include an ID, first name, last name, date of birth, gender and race much like what will be used in this study (section 4.1). Note that the ID field here is only used to generate the labels.

3.2 Creation of Paired Data for Classification

One naive approach to generating the pairs is to pair every record in A with every record in B. The ID field can then be used to generate the labels (if the two records in a pair have the same ID, the pair would be labeled a match or else, a non-match). As discussed in section 2.6, this approach is computationally very expensive. If there are n_1 records in A and n_2 in B, then that would yield a total of $n_1 * n_2$ pairs out of which at the maximum, there would be $min(n_1, n_2)$ matches. The amount of non-matches, $n_1.n_2 - min(n_1, n_2)$ would be extremely high. Databases with 10,000 records each would yield 100 million possible pairs, out of which there would be a maximum of 10,000 matches. This is not ideal for many reasons. It is computationally inefficient to calculate the feature vector or to train models on prohibitively large training data. The data would be extremely unbalanced. In such a scenario, for every match, there would be 9999 non-matches. Such high degree of class imbalance leads to poor recall performance for the underrepresented class when training machine learning models. In record linkage, this is even more an issues because the main class of interest is the under represented class. Most of the pairs would be uninformative - extremely different from each other and clearly non-matches.

Therefore a pipeline to generate paired data that is a lot more selective should be built. The pipeline is an adaptation of the scheme used in a similar work [3] which is based on the concept that non-identical matches and similar non-matches are needed to create an efficient data set. Consider two pairs one for which all fields are identical and another for which all fields are completely different. It is very clear that the former pair should be predicted a match and the latter, a non-match. There is not a lot to be "learned" from such pairs. Therefore, we want the matches in our data to not be completely identical. Similarly, we do not want the non-matches to be completely different from each other. The reason for this is two fold. One, a lot of non-matches that are similar to each other are needed for the most informative training data. Close pairs are the ones for which there is uncertainty in the decision making and they are the ones that provide the maximum information gain. Two, since the data that is generated will go into building the training set and the evaluation sets, it is essential that the evaluation task be complex enough. If matches and non-matches are selected randomly, it would become extremely hard to gauge the true performance of the machine learning algorithm. The models would have inflated performance metrics but would be useless in real world scenarios.

3.2.1 Generation of Non-identical Matches

The process for generating matches is straightforward. Among the pairs that have the same identifier across A and B, the pairs that have at least one difference in one of their fields (first name, last name, date of birth, sex and gender) can be considered. In other words, the pairs that are identical in all the fields are discarded. The process flow is illustrated in figure 3.1.

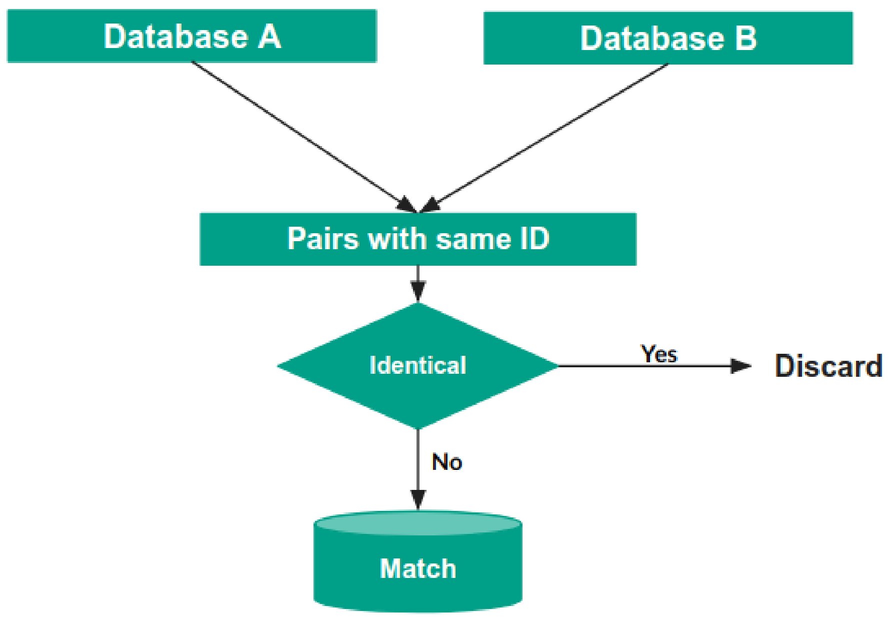


Figure 3.1: Match generation

3.2.2 Generation of Similar Non-matches

The process for generating close non-matches is a little more complex (figure 3.2). If multi-pass blocking is to be used in such a scenario, good choices for the blocking fields would be first name, last name and date of birth. The fields gender and race have very low cardinality (very few unique values) and would yield huge blocks thus defeating the very purpose of blocking. Consider a multi-pass blocking approach on the fields first name, last name and date of birth. Such an approach would narrow down the subset of the potential candidates. Since pairs that appear in a block using one blocking field might reappear again when we block on another blocking field, after the blocking process, only unique pairs should be considered. This approach may also include matches (matching IDs) which need to be removed as non-matches are being generated in this step. At the end of the blocking procedure there might still be a lot of non-matches and the set might have to be further shortlisted. However, the multi-pass blocking approach makes the problem tractable to a great degree.

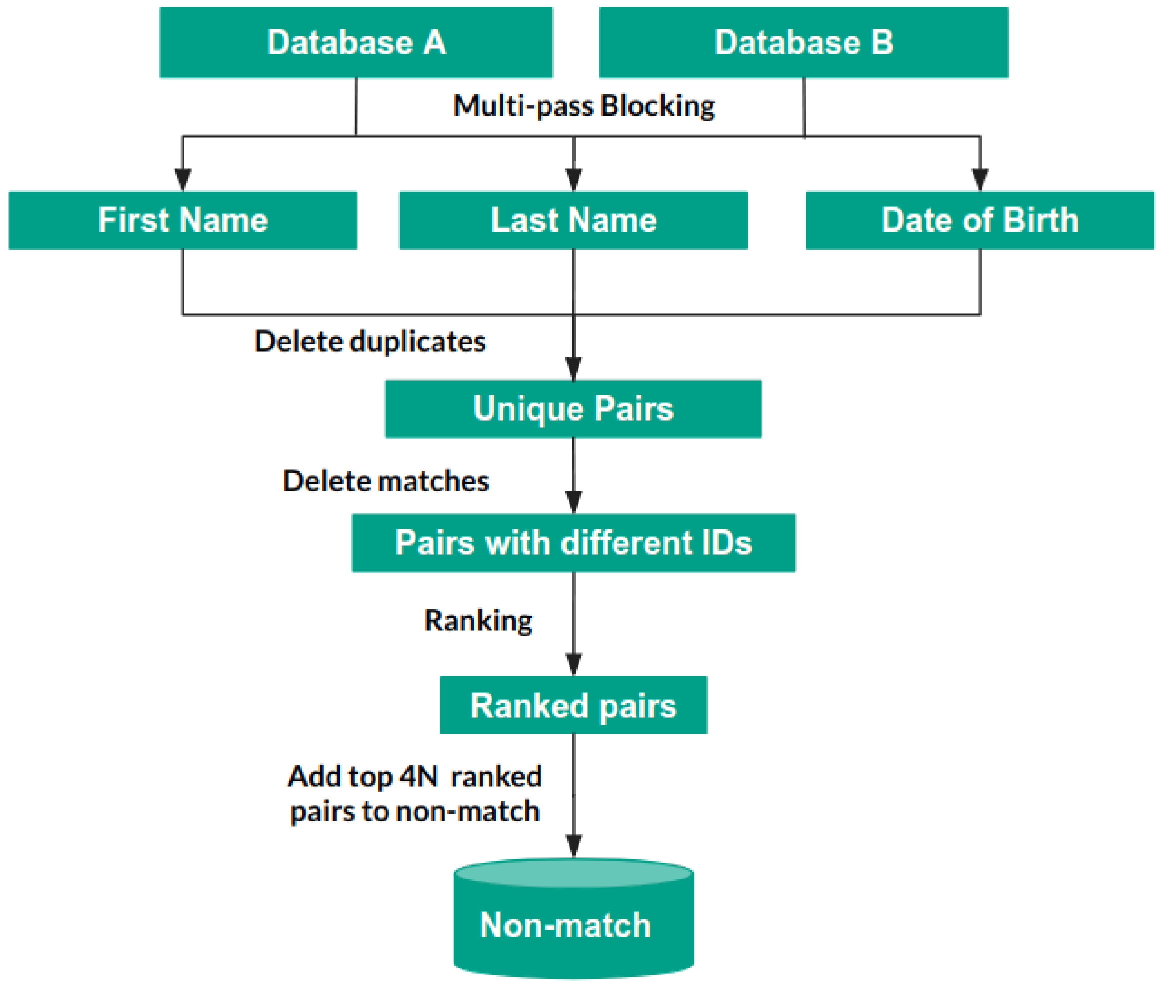


Figure 3.2: Non-match generation

To shortlist the hardest (most similar) non-matches, a ranking procedure can be employed. For each pair, a score can be calculated using this ranking procedure. For instance, the ranking procedure that was used in this study is described. If the first names were within a Jaro-Winkler distance of 0.15, 1 point was awarded. If the last names were within a Jaro-Winkler distance of 0.15, 1 point was awarded. If the day components of the dates of birth matched exactly, 1/3 points were given. If the month components of the dates of birth matched exactly, 1/3 points were given. Similarly, if the year components of the dates of birth matched exactly, 1/3 points were added. If there was a day/month swap between the dates of birth, 1/3 points were given. In a date, if there is a swap between the day and the month component, there cannot be day match or month match.

$$\begin{aligned}
I_{fname} &= \mathbb{1}(JW_{(fname_A, fname_B)} < 0.15) \qquad (3.1)\\
I_{lname} &= \mathbb{1}(JW_{(lname_A, lname_B)} < 0.15)\\
I_{dob_{day}} &= \mathbb{1}(day_{dob_A} = day_{dob_B})\\
I_{dob_{month}} &= \mathbb{1}(month_{dob_A} = month_{dob_B})\\
I_{dob_{year}} &= \mathbb{1}(year_{dob_A} = year_{dob_B})\\
I_{dob_{swap}} &= \mathbb{1}((day_{dob_A} = month_{dob_B}) \& (month_{dob_A} = day_{dob_B}))\\
I_{dob} &= \frac{I_{dob_{day}} + I_{dob_{month}} + I_{dob_{year}} + I_{dob_{swap}}}{3}\\
score_{pair} &= I_{fname} + I_{lname} + I_{dob}
\end{aligned}$$

The score for each pair can then be calculated by adding up the points. Higher the score, greater the similarity between the pairs. It should be noted that the maximum score that any pair can get in this example is 3 (first names, last names and dates of birth are identical). If the dates of birth are identical, then the swap score should be zero and then I_{dob} will be equal to 1. If the swap score is one, then the day and month components cannot match and I_{dob} will be lesser than 1. The minimum score for any pair is 1 (as in the previous step records are blocked on either first name, last name or date of birth). There are no hard and fast rules to generate these rules and much of it is derived empirically from trial and error.

Once the scoring is done, the pairs are sorted by the scores from highest to lowest. Typically, the number of non-matches to be considered are defined in terms of the number of matches as the degree of class imbalance is made more apparent. If there are n matches, the top $k.n$ non-matches with the highest scores can be taken. In our study, 4 was chosen as the value of k i.e the top $4.n$ records were selected. If there are ties among pairs, the pairs can be selected by random sampling.

Note that such a scoring and ranking scheme need not be used for matches, as the number of matches are few and are always tractable.

3.2.3 Feature Engineering

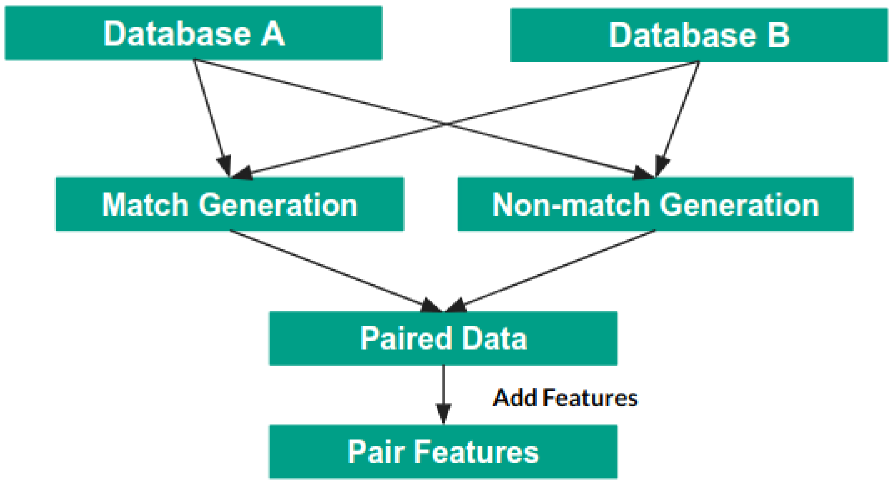


Figure 3.3: Generating the feature vector

Once the match and the non-match pairs are created, the pairs are aggregated. A feature vector needs to be built for representing the pairs to a machine learning model. This step is key as this would be where a machine learning model would learn from. The following sections discuss the features used in the record linkage algorithms.

3.2.3.1 Name Features

For each pair of first names, Damerau-Levenshtein distance, Jaro-Winkler distance and Damerau-Levenshtein distance on soundex codes were calculated, totalling 3 features. Similarly for each pair of last names, Damerau-Levenshtein distance, Jaro-Winkler distance and Damerau-Levenshtein distance on soundex codes were calculated, totalling 3 features. A boolean to detect first name and last name swap was also added. Finally, the normalized frequency of the first and last names in their respective databases was also added, totalling 4 features. To normalize a numeric vector x, $x_{norm} = \frac{x - mean(x)}{standard\text{-}deviation(x)}$

In all, first and last names had 11 features derived from the first and last names.

3.2.3.2 Date Features

For date of birth, Damerau-Levenshtein distance was calculated considering the dates as strings. The Damerau-Levenshtein distance was also calculated individually for the year, month and day components which gave 3 features. A flag to detect month-day swap was also added as shown in equation 3.2. The raw birth years were also included as features as a proxy for age. In all there were 7 features derived from dates.

$$\phi_{dob_{swap}} = \mathbb{1}((day_{dob_A} = month_{dob_B})\&(month_{dob_A} = day_{dob_B})) \tag{3.2}$$

3.2.3.3 Gender and Race Features

Two fields were created for gender - one that counted the number of records in the pair that were female in the gender column of A and B and another that counted the number of males. The values that each of those two columns can take are 0, 1 and 2. The gender column had 3 possible values (*m* for males, *f* for females, *u* for unknown). By using these 2 features, any combination of gender occurrences in two records of a pair could be represented. There was hence, 2 gender based features. For race, two fields were calculated - one field to indicate if there was a match in the races and another field to indicate if both the race values represented whites or blacks as they formed the majority of the values. The rationale was that if there was a match on infrequently occurring race values, it was more informative and hence would be captured by true in the first field but false in the second.

3.2.3.4 Custom Features

Lastly, a field to explicitly identify a possible marriage was also added. This was essentially a flag for females over eighteen changing their last name. Ideally, the models should be able to figure such features out on its own. However, such features can be added to make it easier for the model to account for constructs like marriages. In this study, marriages were the largest source of heterogeneity in our data (as mentioned in table 4.1) and hence, an additional feature was explicitly added.

3.3 Training and Tuning the Machine Learning Algorithms

Once the features are extracted, the data can be used to train the machine learning models. In this section, the training procedure that was followed for the three models studied- a dense neural net, support vector machine with a radial basis kernel and random forest given a training set, is discussed.

3.3.1 Dense Neural Net

The neural net had one input layer, two hidden layers and one output layer with dense connections between all layers. The input layer had 23 units (from the feature vector discussed). The two hidden layers had 64 units each with *relu* activations. After the first layer, there was batch normalization and 0.1% dropout. There was a batch normalization after the second layer but no dropout as is the standard practice for layers preceding the output layer. The output layer had 1 unit with a *sigmoid* activation that returned the probability of a match. An RMSprop optimizer with a learning rate of 0.001 was used with *binary cross-entropy* as the loss function. The batch-size was maintained at the default 32 and the network was trained for 50 epochs. 20% of the data provided was used as the validation set for monitoring the validation performance. The performance on both the training and the validation sets tended to peak and plateau after about 20 epochs and sometimes, the performance went down marginally after that. To solve this two callbacks were used:

1. Model Checkpointing: The validation accuracy was monitored over the epochs and the model with the best validation performance until that point was saved.

2. Learning Rate Reduction on Plateau: When the performance of the validation loss plateaued, the learning rate was reduced by a factor of 80% such that $lr_{new} = 0.8.lr_{old}$

This architecture and training procedure was followed for all combinations of training data set sizes and heterogeneity rates to get the best set of weights for each training set size - heterogeneity rate combination.

3.3.2 Support Vector Machine

For each of the training set sizes and heterogeneity rates, there was a training set. For each of those training sets, a support vector machine with a radial basis function was built. The two key parameters for a radial basis kernel SVM were the penalty parameter, C and the kernel coefficient, sigma. Those two parameters were tuned using 10 fold cross validation and grid search on the training data. The grid was validated for all combinations of C (0.1, 0.5, 1, 10) and sigma (0.03, 0.5, 0.9). The model was retrained on all of the training data once the hyper parameters were fixed.

3.3.3 Random Forest

Similar to the SVM, a grid search with 10-fold cross validation was used on all the training sets to tune the *maximum number of features at each split* hyperparameter. The values that parameter was tested for were 3, 5, 7, 9, 11, 13 and 15. As the number of estimators goes up, the performance typically goes up initially and plateaus after a point. It was observed that the performance of the random forest started plateauing at about 250 estimators and so a margin of 100 was given just to ensure optimal performance and the number of estimators was fixed to be 350. Once the best performing hyperparameter was identified, the random forest model was rebuilt on all of the training data using the same hyperparameter.

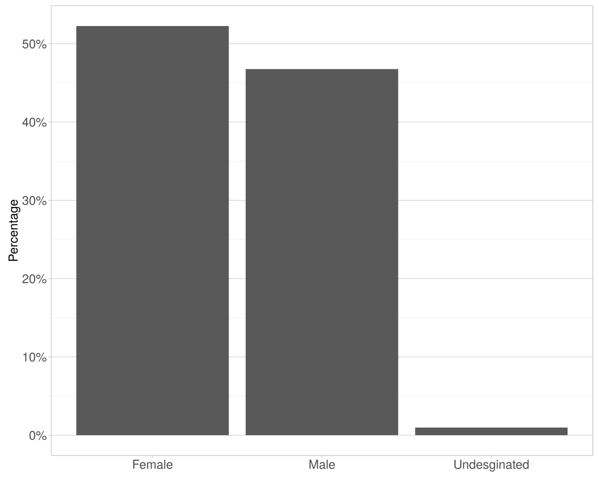
4. DATA GENERATION

4.1 Data

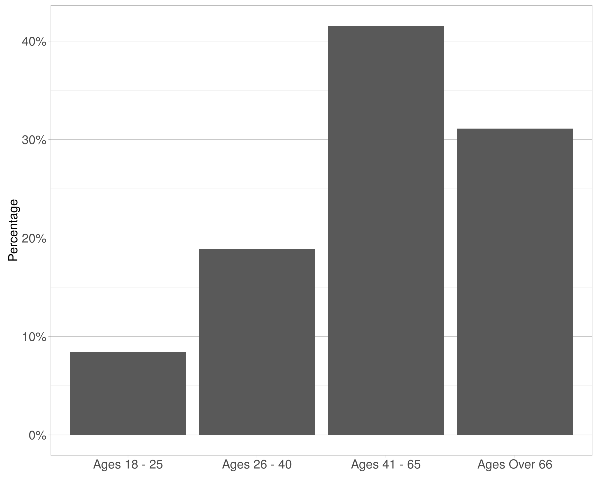
One of the major challenges for benchmarking and studying record linkage frameworks is the lack of real world data sets that can be used as yard sticks for performance. In this digital age, a large amount of personal data which changes over time is collected but remains inaccessible because of privacy regulations. Voter registry databases in many countries are an exception to this scenario and are often made publicly available. However many voter registry databases have their own challenges in being used for record linkage - some are not electronically available, some have to be purchased on record-by-record basis and some can be accessed only in person from the electoral offices.

The Board of Elections of the US State of North Carolina curates a large amount of data on voter registration and state elections and makes the data publicly and electronically available in FTP servers. Fields like social security number and driving license numbers are withdrawn in the publicly available version but many useful fields that help evaluate record linkage procedures like names, ids, gender, race, etc are available. Snapshots of the database have been released on a monthly basis for the past decade, thus creating an excellent temporal database that could be used for studying record linkage [30]. A subset of counties from the publicly available North Carolina voter registry database were chosen to be linked in the study. The counties that were used were Yancey, Hyde, Graham, Alleghany, Pamlico, Swain, Chowan, Avery, Warren and Yadkin. Two snapshots of the data on April 2013 and March 2017 (4 years apart) for these counties were chosen to be linked. These two snapshots will be referred to as A and B for brevity.

The data had a lot of fields like names, birth place, age, address, party affiliation, etc. The fields that were used directly for the linkage process were first name, last name, sex and race. The age and voter registry number fields were used to generate the date of birth and labels respectively, as described in detail in the sections 4.1.1 and 4.1.3. The voter registry number essentially functions as the identifier (ID field) described in section 3.2. Since the study had to be kept as generic as possible, all the fields available in the data were not used and only the most common fields were utilized in the linkage process.



(a) Gender distribution



(b) Age distribution

Figure 4.1: Distribution by gender and age

As per figure 4.1a, 52% of the records were female, 47% male and the remaining was 1% was undesignated. Since the data was from a voter registry, the minimum age as of March 2017 was 18 years. Middle aged people (Ages 41 - 60) formed the biggest chunk of the records followed by older populations (figure 4.1b).

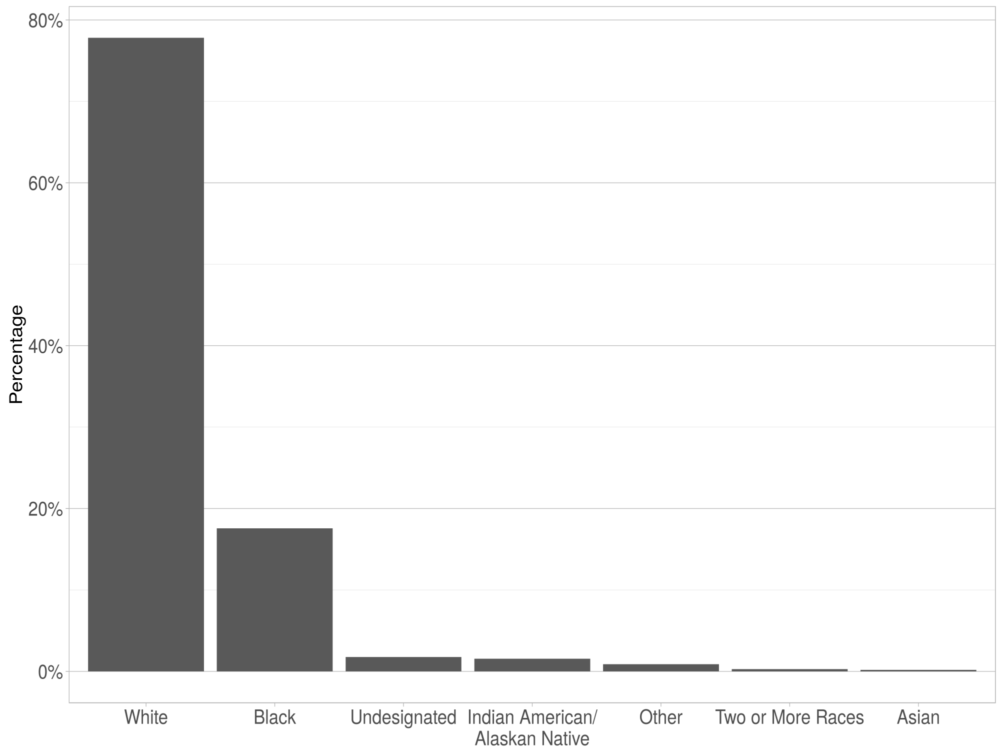


Figure 4.2: Race distribution

As shown in figure 4.2, the two primary races in the database were white and black. About 79% of the people in the data were white and 17.6% were black. The other races together constituted 4.6% of the data.

4.1.1 Date of Birth Generation

One issue with using the NC voter registry data is that it only has the age information. Dates and in particular, dates of birth are important fields in record linkage. Thus, the original data was perturbed by generating random date of birth (DOB) based on the age field. When generating the dates of birth there were a few important considerations that we had to pay close attention to maintain integrity of the database, so that the linkage problem is as close to a realistic dataset as possible. First, the same entities in the two time points must be assigned identical dates of birth. This can be done easily using the voter registry number as discussed in section 4.1.3. Second, the population being linked should contain some number of twins who are born on the same day because twins represent one of the most difficult issues in linkage performance. This is discussed in detail in section 4.1.2. This is how the age field and voter registry number were used in tandem as described above to generate the date of birth information in the database.

4.1.2 Twin Identification

Record linkage datasets typically contain some number of twins. If dates of birth are assigned totally at random, the real twins would be assigned different dates of birth as described in section 4.1.1 and the resulting linkage database will no longer have the comparatively difficult cases. Also, since the study is about studying record linkage performance in controlled conditions, there had to be a flexible mechanism to introduce a controllable number of twins. This problem was alleviated by identifying "potential" twins in the databases and created "potential twin birthdays" for them. The mechanism to identify twins (and triplets, quadruplets, etc.) was simple - people living in the same address, having the same last name and the same age, were considered "potential twins" and one birthday was assigned as the potential twin birthday for all those people. This meant that potentially up to 4 records, 2 records representing a twin in each of the two time points, will be assigned the same DOB. These twin birthdays were stored separately in addition to the birthdays generated in section 4.1.1. This enables easy controlling of the twin rate using the heterogeneity generator as described in section 4.2.

4.1.3 Labeling Procedure

Gold standard data was necessary to train and evaluate approaches for record linkage data. The "voter registration number" is the unique identifier (ID field) common to both the databases and it is used to generate the labels, as described in section 3.2. That is, if two records had the same voter registration number, they referred to the same person and were labeled a "match" and two records having different voter registration numbers were labeled "non-matches". Since the data was generated for each county, only pairs within a county were considered. A had a total of 212,135 records and B had 212,232 records. There were 175,487 (approximately 83%) matches between A and B. The 36648 records found only in A but not in B represents the people who were no longer eligible to vote in 2017 due to moving out or death. Similarly the 36745 records found only in B but not in A represent people who were not eligible to vote in 2014, but became eligible in 2017 due to moving in or coming of age.

4.2 Heterogeneity Generation

Once the data was acquired and perturbed, heterogeneity generation was the next and the most crucial part of the pipeline. One of the main factors that were studied was the heterogeneity rate. To study the heterogeneity rate, different amounts of heterogeneity were introduced into A to get different versions of $A^{'}_{e\%}$, each of which would then be linked with B and the performance evaluated. e% of heterogeneity introduced into A meant that e% of the records in $A^{'}_{e\%}$ would be perturbed in some way, some more than once. To generate these heterogeneities at different rates two things were needed - a flexible mechanism that allowed for control over how much heterogeneity would get introduced and a way to control the proportion of different kinds of heterogeneities.

4.2.1 A Mechanism to Generate Heterogeneities

Typically, the problem of record linkage occurs because of a lack of perfect consistency between datasets. Some common reasons that make record linkage a challenge are typographical errors in names and dates of birth, marital last name changes, duplicate records in a database, etc. Complexity because of cases like marriages, twins and duplicates are not exactly "errors". Calling them errors is technically a misnomer and so the term "heterogeneity" is used to refer to them.

To introduce heterogeneities at different rates and distributions, an heterogeneity generator needs to be built. This heterogeneity generator take in the heterogeneity distribution and the overall heterogeneity rate e% as inputs and generate heterogeneities according to the distribution until e% of the records. This heterogeneity generator hence had to be created such that it could be used to emulate the heterogeneities found in real record linkage scenarios. A variety of heterogeneities found in real word linkage scenarios were identified and modeled. The kinds of heterogeneities that could be introduced are discussed below.

4.2.1.1 Name Heterogeneities

1. Typographical Errors - These errors are really common in real world scenarios and occur most commonly as a result of incorrect data entry.

    - *indel* - This is when a character is randomly added into or deleted from a field.

      Example: Johnathan vs Jonathan

    - *replace* - This is when a character is randomly substituted by another.

      Example: Cristen vs Kristen

    - *transpose* - This is when two adjacent characters switch places.

      Example: Johnathan vs Johanthan

2. Nicknames - For nickname based errors, a nickname - real name lookup table from many public repositories was aggregated and used it to randomly switch first names (real names with their corresponding nicknames and vice versa).

   Example: Elizabeth vs Beth/Elize

3. Suffix Additions - Many last names have suffixes like "JR", "SR", "I", "II", etc. So random males were chosen and a suffix was added to their last names.

   Example: Mark vs Mark II

4. Marital Name Changes - Females over 20 were identified and their last names were randomly changed with last names from a last name table.

   Example: Emily Robinson vs Emily Smith

5. Name Swaps - This is when first and last names are swapped. In real life occurs most commonly due to data entry errors.

   Example: John Smith vs Smith John

4.2.1.2 Date Errors

When errors are introduced into dates, it must be made sure that the new dates with errors are valid. A valid date is a date that is actually possible (exists on a calendar). This is because most of the cases where data is ingested, there is a check on computer systems to make sure the entered date is valid.

1. Day - Month Swaps - The day and month of a date are swapped with the caveat that the swapped dates are valid.

   Example: 10/03/1992 vs 03/10/1992 (valid), 10/13/1992 (invalid)

2. Month Change - A month is randomly replaced with another month with the caveat that the changed date is valid. This meant that the month selected to replace an existing month, had to be sampled from a list of applicable months.

   Example: 10/31/1992 vs 03/31/1992 (valid), 10/31/1992 vs 02/31/1992 (invalid)

3. Day Change - A day is randomly replaced with another day with the caveat that the changed date is valid. This meant that the new day picked to replace the old day had to be sampled from a list of applicable days.

   Example: 02/28/1992 vs 02/15/1992 (valid), 02/28/1992 vs 02/31/1992 (invalid)

4. Day Transpose - The day component is randomly transposed with the caveat that the changed date is valid. This meant that we had to select the day to replace from the applicable days.

   Example: 02/01/1992 vs 02/10/1992 (valid), 02/13/1992 vs 02/31/1992 (invalid), 01/18/1992 vs 01/81/1992 (invalid)

5. Year Change - The year part of the date is changed by randomly replacing the third or fourth character of a date.

   Example: 03/10/1992 vs 03/10/1982, 03/10/1992 vs 03/10/1997

6. Year Transpose - This is when the third and fourth digits are randomly swapped with the caveat that the changed date is valid.

   Example: 03/10/1958 vs 03/10/1985

4.2.1.3 Missing Data Errors

A field for a record is removed. If a few first names are to be made missing, a few records are selected at random and their first names are erased. The same goes with dates and last names fields.

4.2.1.4 Record Level Heterogeneities

1. Duplicates - If the probability of duplication is d%, when each of the other errors are introduced, the original record would be retained with a probability of d%

2. Twins - As discussed in section 4.1.2, records that could potentially be twins (or triplets) based on the address, last name and age information were flagged beforehand and had potential twin dates of birth created in advance. To make a pair "twins", a few pairs from the list of potential twins were sampled and their dates of birth were switched for the potential twin date of birth.

4.2.2 Distribution of Heterogeneities

Once the heterogeneity generator was built, the next step was to determine the distribution of different types of heterogeneities. We estimated the distribution using the naturally occurring heterogeneity distribution in the two datasets. The process for extracting the heterogeneity distribution was straightforward. The same entities in A and B could easily be identified using the voter ids as described in 4.1.3. Among the matches, the pairs of records which were not identical and had differences between their fields were identified. These naturally prevalent differences were the baseline "heterogeneities" and their distribution was analyzed. There were some heterogeneity types that were not found in the current dataset. In these cases, we estimated a rate. The final heterogeneity distribution can be found in the table 4.1. We note that marital last name change

Field	**Heterogeneity**	**Heterogeneity Probability**
First Name	Typos (indel/tranpose/replace)	14%
	Nick name/Real name	5%
Last Name	Typos (indel/tranpose/replace)	14%
	Marital Name Change	24%
	Suffix	5%
First and Last Name	Swaps	5%
Date of Birth (DOB)	Swaps	5%
	Month Typos (replace)	3%
	Day Typos (tranpose/replace)	6%
	Year Typos (transpose/replace)	6%
First/Last Name, DOB	Missing	5%
Record Level	Duplicates	3%
	Twins	5%
	Total	**100%**

Table 4.1: Heterogeneity distribution

was the highest problem found in the real data at 30%. We reduced this to 24% to account for heterogeneities that did not occur in this clean dataset.

5.1 Experiments

Using the heterogeneity distribution and any given heterogeneity rate $e\%$, the heterogeneity generation mechanism could be used to introduce heterogeneity into a database. In this study, the heterogeneity generator is used to introduce heterogeneity into the database A at different rates of heterogeneity e% (from 0% to 60% in steps of 5%) to get $A^{'}_{e\%}$. As stated $A^{'}_{e\%}$ means that at least $e\%$ of the records in $A^{'}_{e\%}$ have heterogeneities in them. This means that some records might have many heterogeneities in them. The distribution of the number of heterogeneities is given in figure 5.1

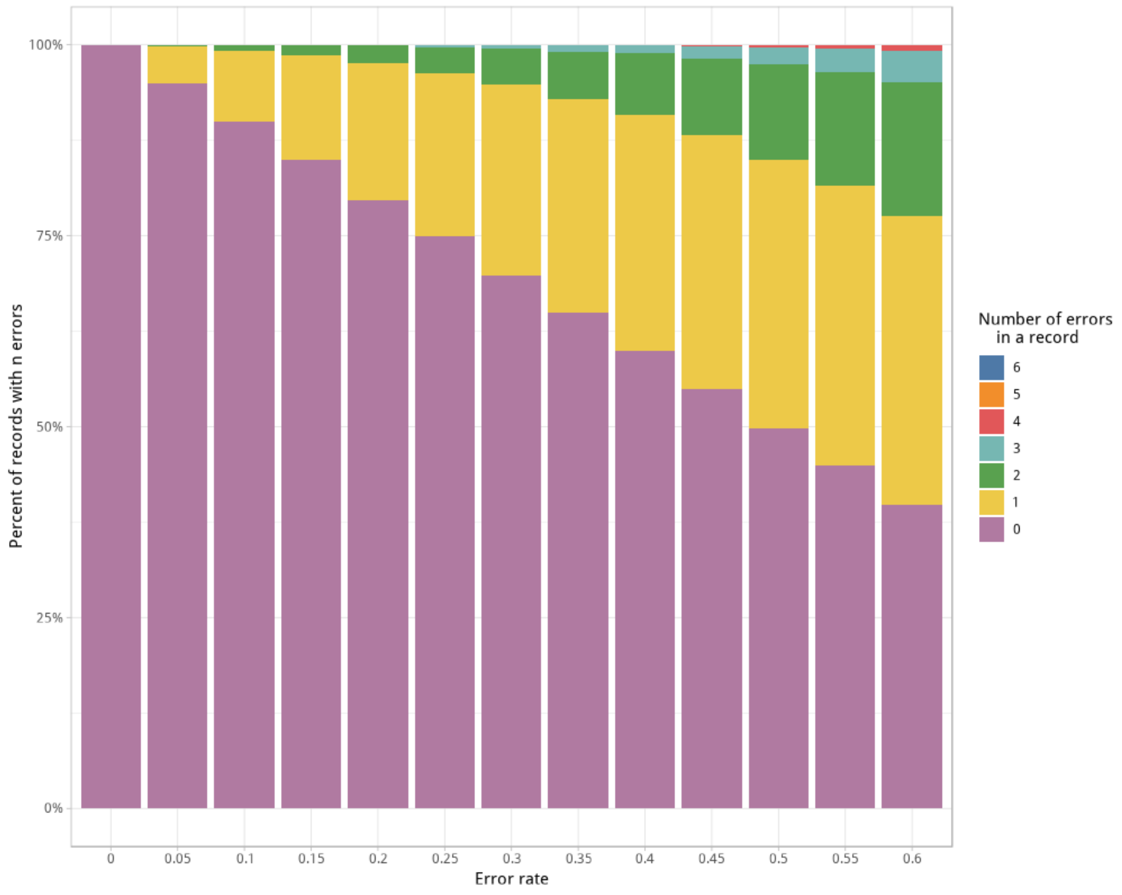


Figure 5.1: Distribution of the number of heterogeneities

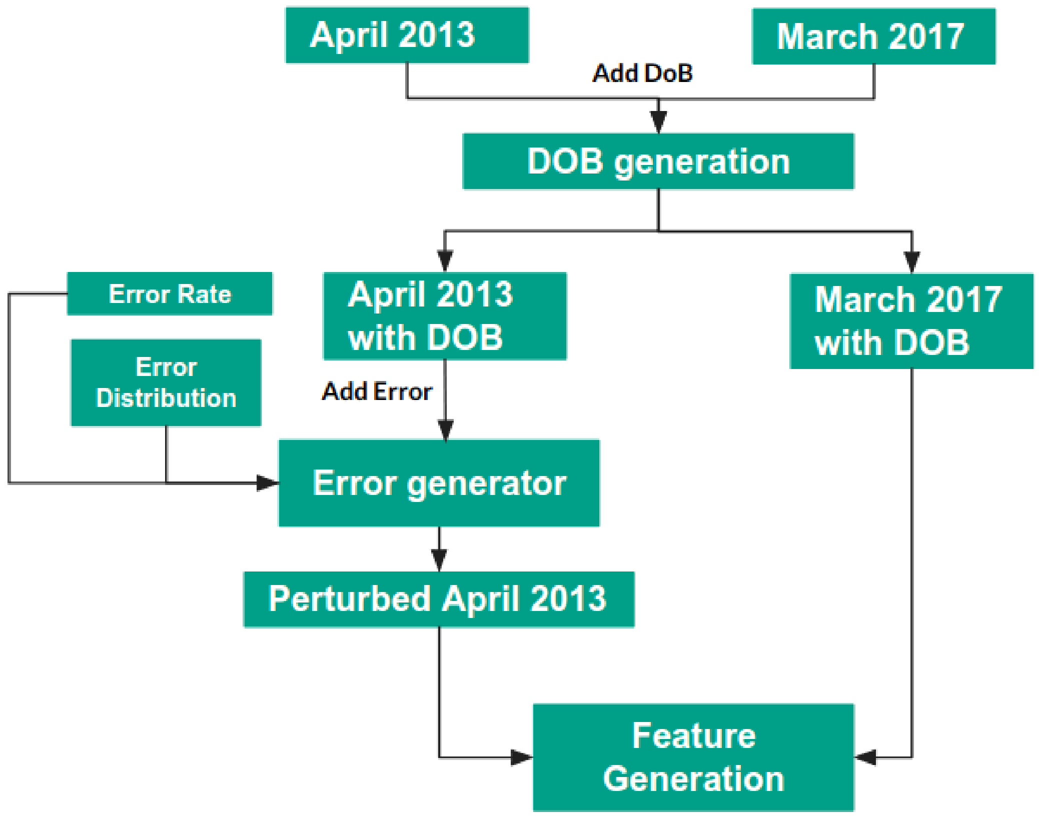


Figure 5.2: Heterogeneities generation to feature generation

The datasets $A'_{e\%}$ and B are then sent to the feature generation pipeline discussed in the record linkage section as shown in figure 3.3. As discussed in the labeled data generation section (section 4.1.3), the voter registration number field is supplied as the ID field to the pair feature generation pipeline. All the steps involved in generating the features at different heterogeneity rates e% from the two databases A and B are shown in figure 5.2

This procedure yielded 27000 pairs for each heterogeneity rate. There were totally twenty three features to represent each pair of records. So for each heterogeneity rate from 0% to 60% in increments of 5%, 27,000 pairs of records were created each of which had 23 features. 17000 of these records were sampled to be the test set and kept apart. The 17,000 records were randomly split up into 10 mutually exclusive test subsets, each of which had 1700 pairs. Among the remaining 10,000 records, 4 training set versions with 1000, 4000, 7000 and all of the 10,000 records were created. Every smaller training set was a subset of every bigger training set. Since different versions of training data with different training set sizes were present, the effect of training set size on model performance could be studied. The split is shown in figure 5.3

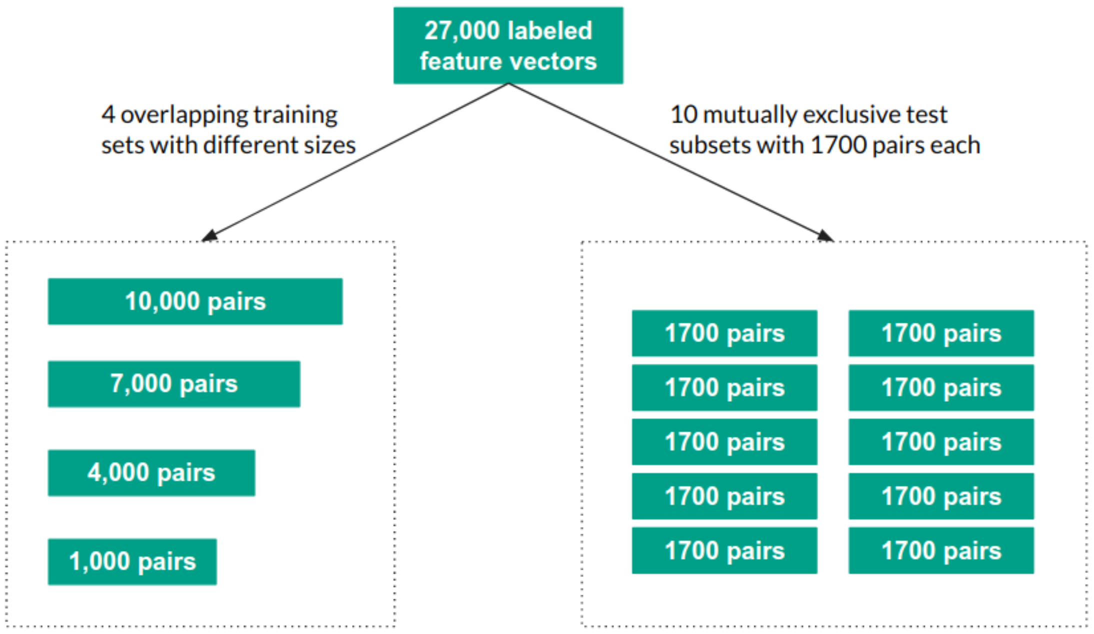

27,000 labeled feature vectors
4 overlapping training sets with different sizes
10 mutually exclusive test subsets with 1700 pairs each
10,000 pairs
7,000 pairs
4,000 pairs
1,000 pairs
1700 pairs

Figure 5.3: Splitting training and test sets

On each of these training sets, the three machine learning models are trained as described in section 3.3. At the end of the training procedure, the 3 models were trained for all combinations of 13 heterogeneity rates and 4 training set sizes. There were hence a total of 156 models. Each of these 156 models, were evaluated on the 10 test subsets for greater reliability using the metrics described below.

5.2 Evaluation Metrics

5.2.1 Effectiveness of Record Linkage - F_1 score

A common measure of linkage quality is the F_1 score. F_1 score is commonly used to measure model performance in classification tasks especially in unbalanced scenarios where accuracy is a naive metric. Since record linkage as a classification problem is essentially extremely unbalanced where the number of non-matches outnumber the number of matches, the F_1 score has been widely adopted in the record linkage community where it is commonly used to report the effectiveness of a system. The F_1 score is the harmonic mean between the precision and recall and hence is very useful in evaluating the effectiveness of linkage.

$$F_1 = 2.\frac{Precision.Recall}{Precision + Recall} \tag{5.1}$$

As discussed, F_1 score is not without its flaws. In [28], the authors argue that the F_1 score can be expressed as a weighted sum of the precision and recall where the weights are dependent on the linkage algorithm, thus making them unreliable. However, since its a commonly used measure in record linkage literature, the F_1 score is included in our evaluation. In contrast, a more practical metric, the percentage of pairs that need manual review after running an algorithm is also included in section 5.2.2.

5.2.2 Efficiency of Record Linkage - Percentage of Manual Review

As stated in the literature review, hybrid approaches for record linkage involve a first pass performed by an automatic algorithm. After the first pass, the pairs that the algorithm is most uncertain about, is reviewed manually by experts. This process however requires the selection of pairs that the algorithm is uncertain about. The advantage of such a hybrid approach is that the quality of linkage requirements can be set beforehand and automatic linkage can be used only to the extent that those requirements are met. In this study, the requirements are defined in terms of Positive Predictive Value (PPV) and Negative Predictive Value (NPV). The Positive Predictive Value is the ratio of the number of true positives to the number of predicted positives. Similarly the

Negative Predictive Value is the ratio of the number of true negatives to the number of predicted negatives. When both are 100% that implies that all predictions must be correct. When the PPV and NPV to be less than 100%, it implies that is some allowance for a few errors.

$$PPV = \frac{\textit{number of true positives}}{(\textit{number of true positives} + \textit{number of false positives})} \tag{5.2}$$

$$= \frac{\textit{number of true positives}}{\textit{number of positive calls}} \tag{5.3}$$

$$NPV = \frac{\textit{number of true negatives}}{(\textit{number of true negatives} + \textit{number of false negatives})} \tag{5.4}$$

$$= \frac{\textit{number of true negatives}}{\textit{number of negative calls}} \tag{5.5}$$

Since the predictions of algorithms are given in terms of probabilities, selecting the uncertain pairs can be achieved by selecting two thresholds T_1 and T_2 between which predictions fall into the uncertain category. The two thresholds T_1 and T_2 are chosen such that the predictions with predicted probabilities greater than T_1 or lesser than T_2 have to meet the performance requirements set.

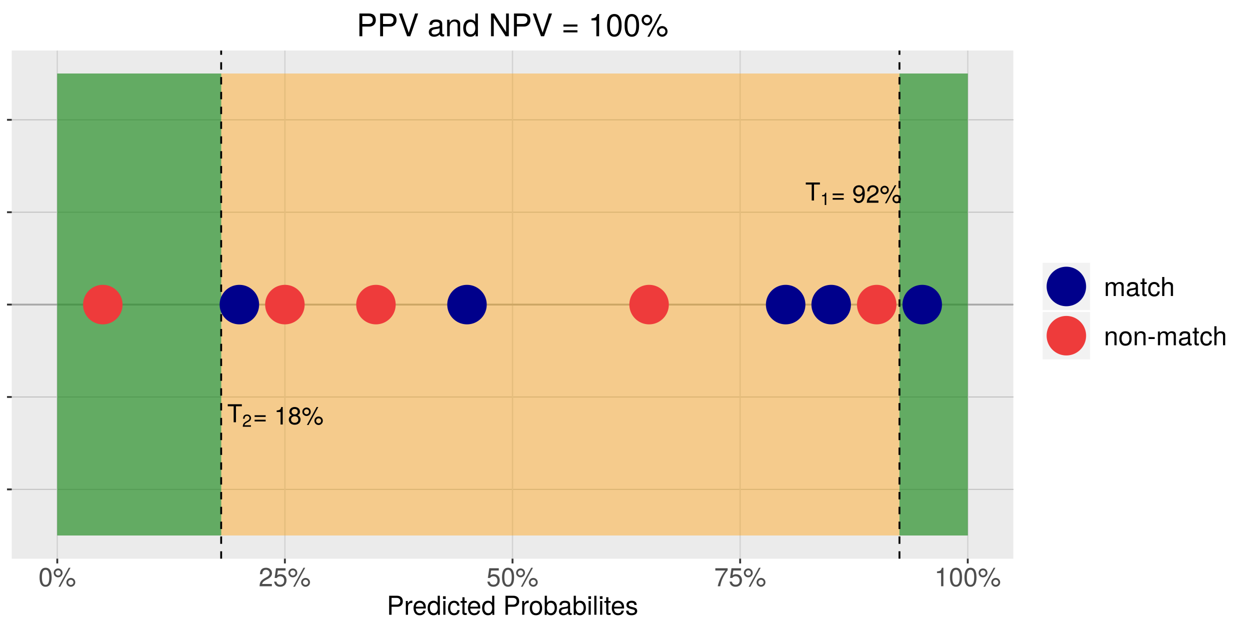


Figure 5.4: Setting a strict threshold increases an uncertain region

Depending on how stringent the requirements are, the number of “uncertain” pairs can vary - a very low bar means very few pairs need manual review and a high bar means a lot of manual review. Figures 5.4 and 5.5 show 10 observations with the same predicted probabilities but different thresholds. Matches are indicated by blue dots and non-matches by red. The area in orange refers to the region of uncertainty and pairs falling in that region are sent for manual review.

Figure 5.4 shows a very strict PPV and NPV requirement of 100%. Since there is absolutely no margin for error, T_1 and T_2 are selected such that among all predictions with probabilities above and below them respectively, the predictions are perfect. However, this comes at a cost - a large number of pairs for manual review.

There is just 1 pair with probabilities greater that 92% and it is a match (positve). According to the equation,

$$PPV = \frac{\textit{number of true positives}}{\textit{number of positive calls}} \tag{5.6}$$

$$= \frac{1}{1} \tag{5.7}$$

$$= 100\% \tag{5.8}$$

Similarly, there is just 1 pair with probabilities lesser that 18% that is a non-match (negative). According to the equation,

$$NPV = \frac{\textit{number of true negatives}}{\textit{number of negative calls}} \tag{5.9}$$

$$= \frac{1}{1} \tag{5.10}$$

$$= 100\% \tag{5.11}$$

The number of uncertain pairs, expressed as a percentage of the total number of pairs is the *metric, percentage of manual review*. In this case (figure 5.4), since 8 out of the 10 pairs fall between the thresholds described, the percentage of manual review is 80% meaning 80% of the records are sent to the second pass.

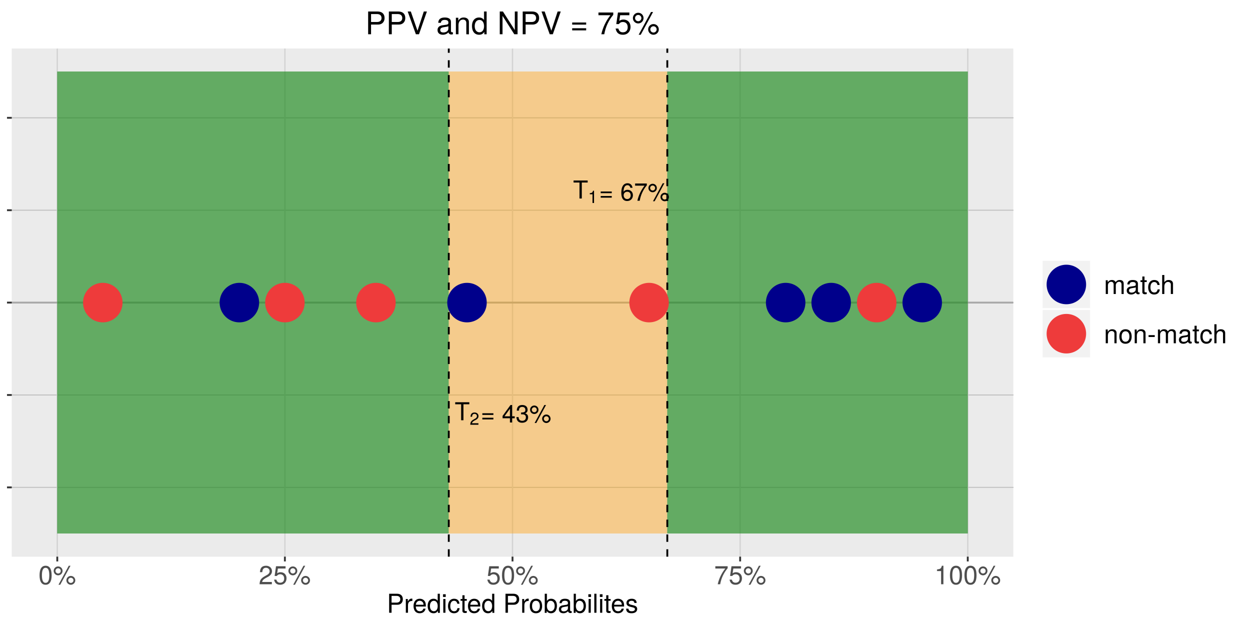


Figure 5.5: Setting a slightly lenient threshold decreases uncertain region

Figure 5.5 shows the smaller uncertain region that comes with setting a lower NPV and PPV requirement. The PPV and NPV are set to be 75% in this case.

There are 4 pairs with probabilities greater that 67% all of which will be predicted "matches" (positive) but only 3 of which are actually matches. According to the equation,

$$PPV = \frac{\textit{number of true positives}}{\textit{number of positive calls}} \tag{5.12}$$

$$= \frac{3}{4} \tag{5.13}$$

$$= 75\% \tag{5.14}$$

Similarly, there are 4 pairs with probabilities lesser that 43% all of which will be predicted "non-matches" (negatives) but only 3 of which are actually non-matches. According to the equation,

$$NPV = \frac{\textit{number of true negatives}}{\textit{number of negative calls}} \tag{5.15}$$

$$= \frac{3}{4} \tag{5.16}$$

$$= 75\% \tag{5.17}$$

In the second scenario (figure 5.5), since the thresholds are a lot more lenient, only two out of the ten pairs fall between them. There is one error on each side but the percentage of manual review drops drastically to 20%.

It is a common practice to set the NPV and PPV to be 100% and study the review percent [29]. However the effects of slightly more lenient performance requirements (PPV and NPV values of 99%, 95% and 90%) are compared to that of a flawless (100%) requirement. The trade-off between PPV/NPV and manual review percentage is studied.

6.1 Trade-off between NPV/PPV Level and Percentage of Manual Review

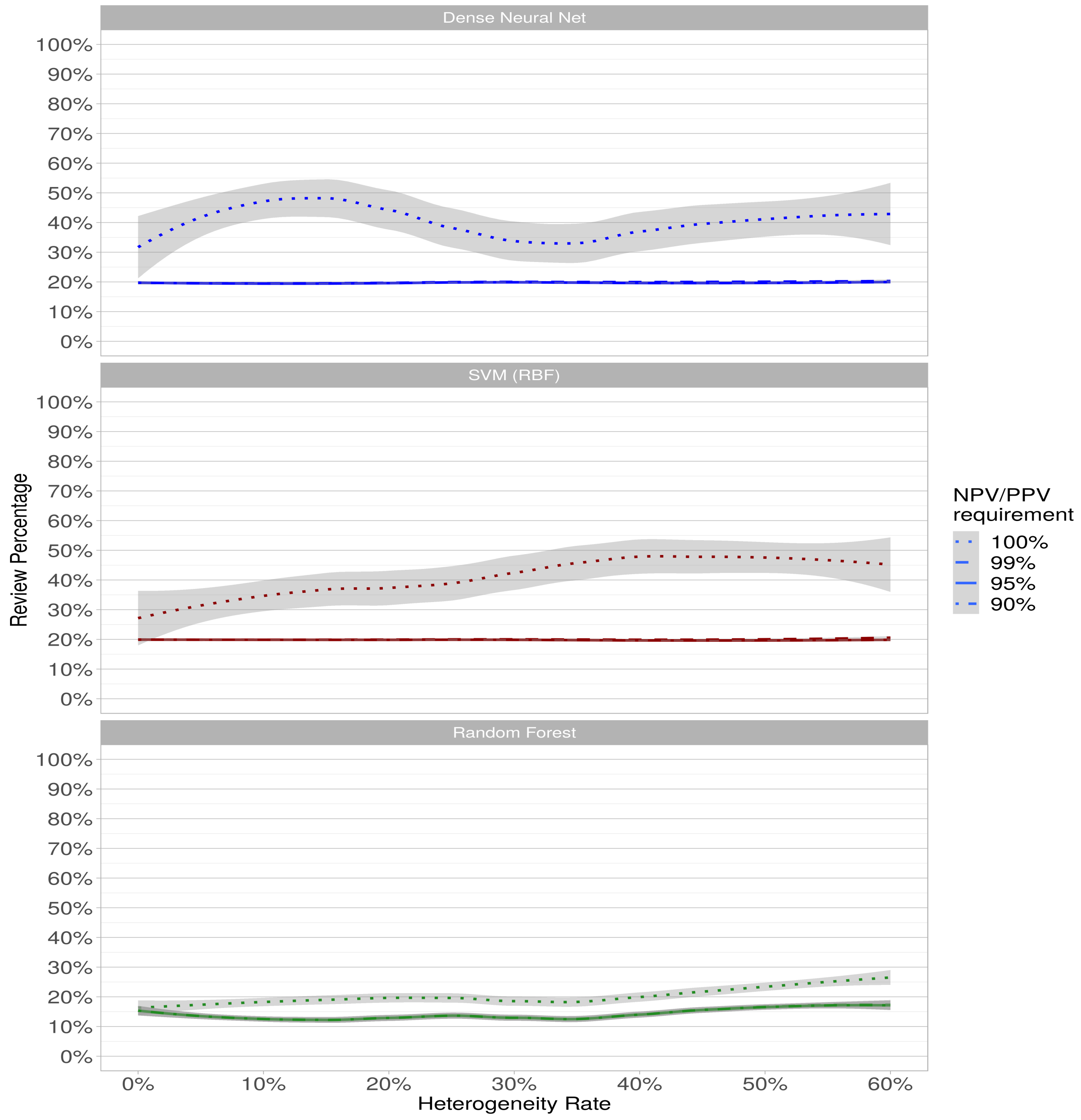


Figure 6.1: Setting the right level of leeway for measuring the review percentage

The figure 6.1 shows the percentage of records that need to be manually reviewed to have a Positive Predictive Value and a Negative Predictive Value of $p\%$ at p = 100, 99, 95 and 90 for each of the algorithms, as the heterogeneity rate varies. The training set size is $n = 7000$ as discussed in section 6.2. It should be noted that the performances at 99%, 95% and 90% are nearly identical and hence the trend lines overlap. The grey bands in all the plots represent the 95% confidence interval of the estimated values.

It is evident that there is a price to be paid in terms of the manual review size if only the most confident predictions such that the model performance is perfect are to be considered. This effect is discussed in detail in section 5.2.2. Even one confident mistake can throw the percentage of manual review off because at 100% PPV and NPV, no mistakes are allowed. The instability of the measure at 100% is also reflected by the variance of the measure, lack of any discernible trend in the review percent as the heterogeneity rate increases and the uncertainty of the confidence intervals (for SVM and DNN).

For Dense Neural Networks, the average manual review percent at 100% NPV/PPV varied from 17.8% to 95.8% depending on the set that it was evaluated on and the heterogeneity rate. The average review percent was 40.8%. SVMs also showed comparable performance - the review ranged from 17.8% to 89.8% and had a mean of 40.8%. Random forests also showed great variation at 100% NPV/PPV but their performance was much better. The percentage of manual review varied from 7.9% to 47.1% but their mean was 20.3%.

When even a little leeway is given to the algorithms by making the minimum NPV and PPV 99%, the percentage of manual review not only drops but also stabilizes. However, if the PPV and NPV requirements are decreased further (95% and 90%), similar drops are not seen because the models achieve their near best performances at 99% and saturate there. This is why the lines for 99%, 95% and 90% are almost perfectly congruent. Based on this behavior, the NPV and PPV requirement is chosen to be 99% when calculating the percentage of pairs to be manually reviewed in all further analyses.

At 99% NPV and PPV, the percentage of manual review varied from 16.9% to 23.1% for DNNs, 16.9% to 23.2% for SVMs and 7.1% to 20.1% for Random Forests. The mean review percentages for Neural Nets, SVMs and Random Forests were 19.9%, 20% and 14.3% respectively.

6.2 Effect of Training Set Size on Model performance

Data with no artificially induced heterogeneity (0% heterogeneity) is used to study the effect of the size of the training data on model performance. The two evaluation metrics that were introduced - F_1 score and percentage of manual review are used to study the performance characteristics. The three models are built on training data sets of size 1000, 4000, 7000 and 10,000. Once the models are built, they are evaluated on 10 evaluation sets for better reliability. The means of the metrics are analyzed further.

6.2.1 F1 Score

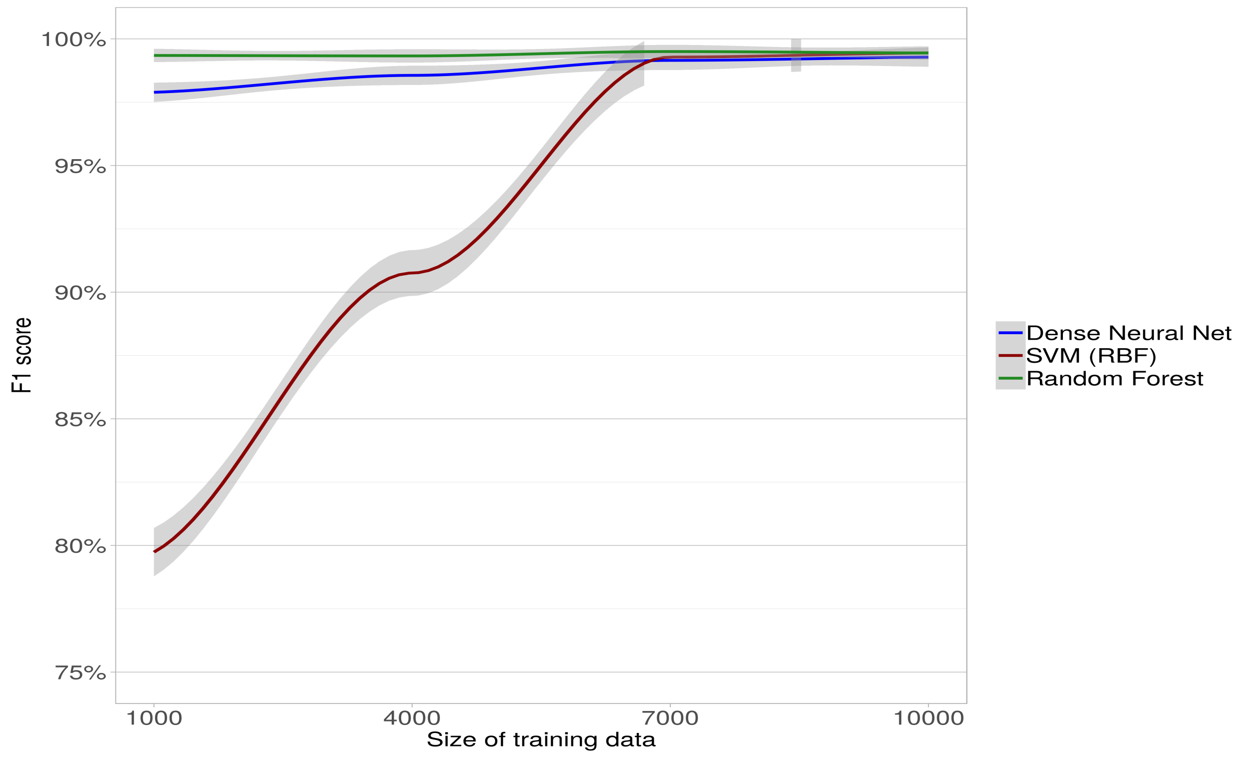


Figure 6.2: F1 performance by number of observations in training data

As we increase the training set size, the F_1 score is expected to go up. This effect is very evident for the Support Vector Machine which has extremely poor performance at $n = 1000$. The F_1 score at $n = 1000$ for SVM has a mean of 80% but quickly improves to 91% at $n = 4000$ and 99.3% at $n = 7000$. It plateaus at $n = 7000$ and improves little beyond that. The Dense Neural Net shows a minor but steady improvement from 97.9% to 99.2% from $n = 1000$ to $n = 7000$ but plateaus like SVM at that point. The Random Forest model performs very well at $n = 1000$ and there is little scope for improvement beyond that. It however improves from 99.3% marginally to 99.5%. In fact, the random forest achieves its best performance at $n = 7000$.

At $n = 7000$, the F_1 score plateaus for all the algorithms and their performances are comparable to each other. There is no discernible increase in performance as n is increased further. It is worth noting that SVM models seems to require much more training data compared to the other models with Random Forrest, requiring the least amount of training data.

6.2.2 Size of Manual Review

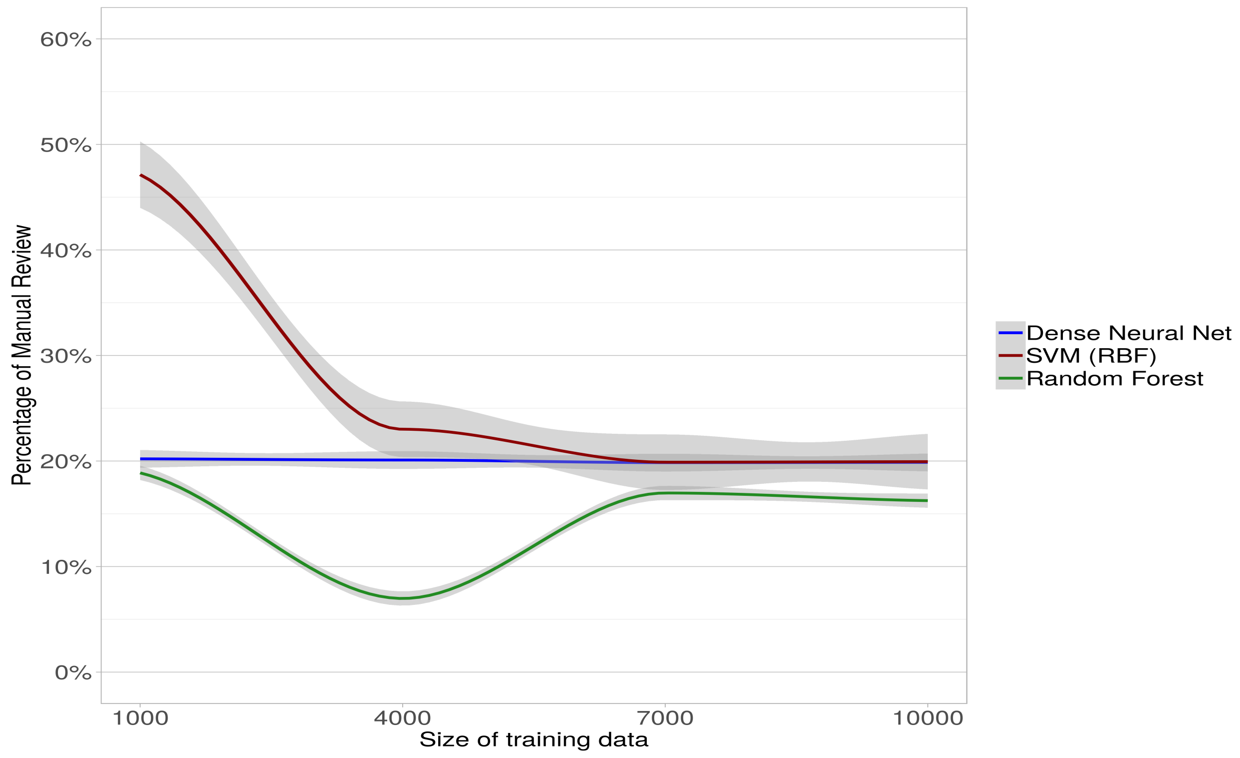


Figure 6.3: Percentage of manual review by number of observations in training data

As discussed in the section 6.1, the percentage of manual review at PPV/NPV = 99% is chosen as the metric.

The performance of support vector machines in terms of review percentage mirrors their performance measured by F_1 score as the amount of training data is varied. They perform poorly at $n = 1000$ needing manual review for 56% of the pairs on average but in some cases needing as much as 98% manual review! However, as with the F_1 score, the review percent drops to 23% at $n = 4000$ and 19.9% at $n = 7000$. The review percentage performance saturates at $n = 7000$. The Dense Neural Net performance is more or less constant and any improvement seen is very minimal (20.2% at $n = 1000$ to 19.9% at $n = 10000$). The model achieves its best performance of 19.8%

at $n = 7000$. The review percentage performance for random forest is erratic before $n = 7000$. It starts out at 18.9% at $n = 1000$, achieves an exceptional 6.9% at $n = 4000$ and stabilizes at $n = 7000$ with 17%.

6.2.3 Choosing the Right Training Set Size

Based on the F_1 and manual review percentage performance, a training set size of $n = 7000$ is selected for further analysis of heterogeneity characteristics. At $n = 7000$, all algorithms seem to achieve their best or near-best performances. Hence, further comparisons will made at $n = 7000$.

6.3 Effect of Heterogeneity Rate on Model Performance

In this section, the effects of increasing the heterogeneity rate for the three algorithms are studied in terms of F_1 and review percentage at 99% NPV and PPV as per section 6.1. The training set size is chosen to be $n = 7000$ as discussed in the previous section as per section 6.2.

6.3.1 F1 Score

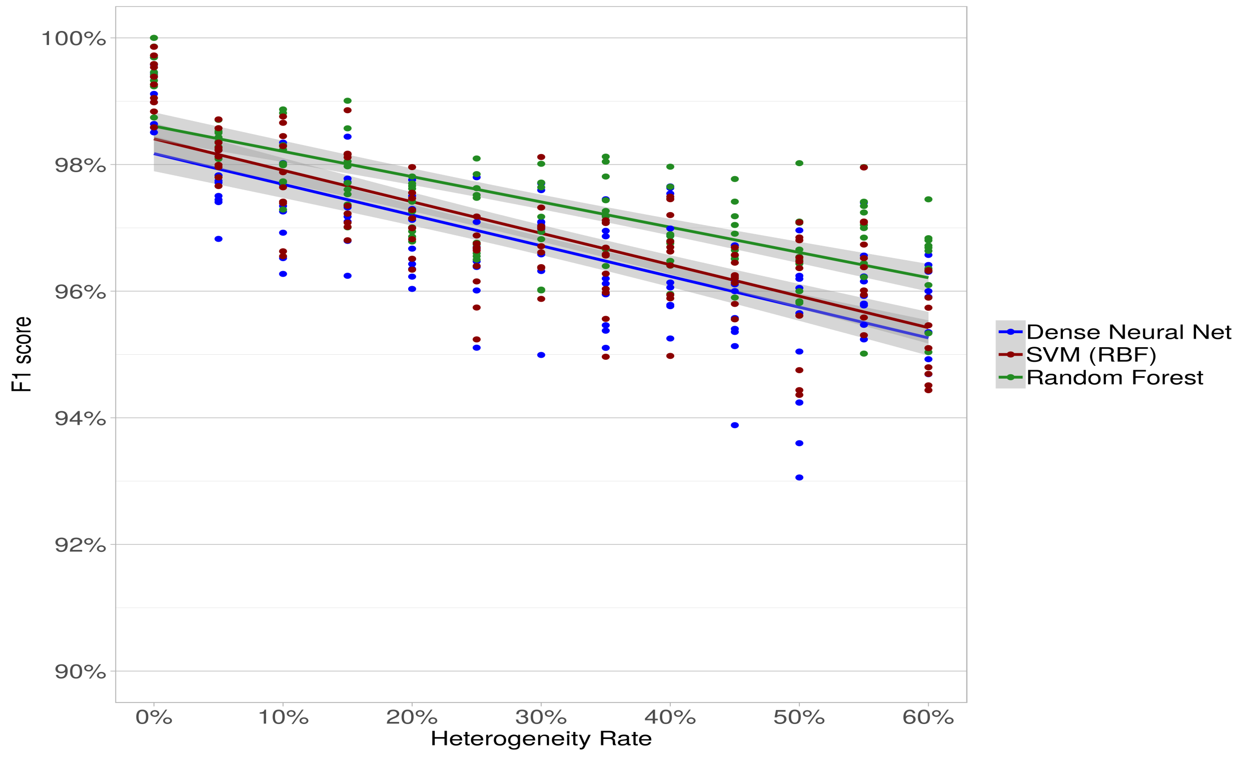


Figure 6.4: F1 score by number of observations in training data

As shown in figure 6.4, as the heterogeneity rate increases, the F_1 performance of all the models goes down. SVMs start at 99.3% at 0% heterogeneity rate and the performance steadily drops to 95.3% at 60% heterogeneity rate. The F_1 performance of Neural Nets are also largely similar to the SVMs - the performance drops from 99.2% to 95.7%. Random Forests also show a similar trend but in general had much better scores with the scores dropping from 99.5% to 96.4%. It should be noted that performance of the random forests drops at a slower rate than the other models. At all the heterogeneity rates the mean F_1 score of random forests are more than that of SVMs and neural nets though not by much, especially at lower heterogeneity rates.

6.3.2 Size of Manual Review

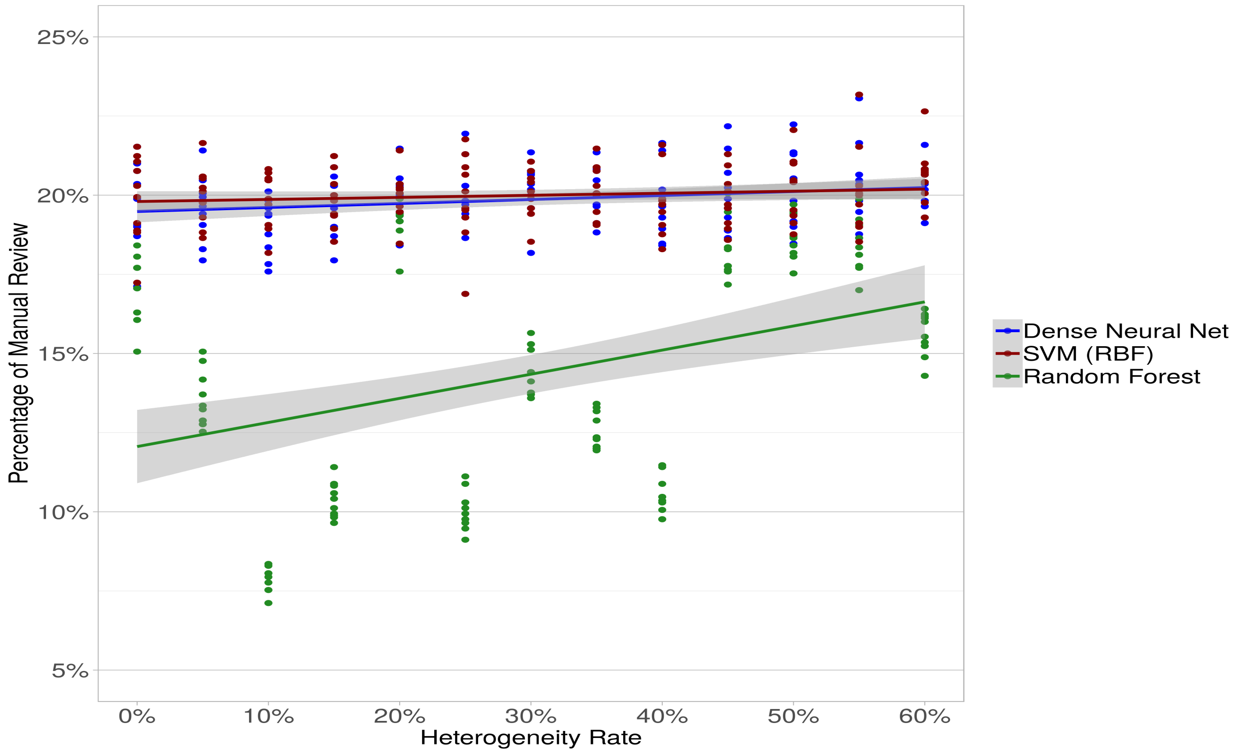


Figure 6.5: Percentage of manual review by heterogeneity rate

The performance of SVM measured by percentage of pairs needing manual review, remains largely constant from 19.9% at 0% heterogeneity rate to 20.6% at 60% heterogeneity rate. Dense Neural Nets also follow the same pattern. Their performance varies from 19.8% to 20.2% at 0% and 60% heterogeneity rates respectively. Random Forests clearly outperform the other two algorithms in light of the review percentage measure. There is however a lot of variation in the mean review percent value between different heterogeneity rates. For instance, the range of the review percents is 11.3% (7.8% at heterogeneity rate 10% to 19.1% at heterogeneity rate 20%).

As shown in figure 6.4, as the heterogeneity rate increases, the percentage of pairs needing manual review goes up in line with the expectations. As with the F_1 performance, random forests outperform the other algorithms at all the heterogeneity rates. However, the difference between the algorithms in terms of manual review percentage is much more substantial. Interestingly, as opposed to the F_1 performance, the random forests suffer the greatest loss in terms of review percentage when compared to the other algorithms. One possible reason for this could be that SVMs and Neural Nets perform very poorly in terms of the review percentage to begin with (at 0%) heterogeneity and have very little to lose. The random forests however shine at lower rates and in contrast performs about 40 - 50% better than SVMs and DNNs at lower heterogeneity rates. There is a lot of scope for performance degradation with them, which is what happens.

7. DISCUSSION

This research sought to systematically study the effects of size of training data and heterogeneity rate on Neural Nets, Support Vector Machines and Random Forests in terms of F_1 score and manual review percentage. The trade-off between the performance requirement for the algorithms and the review percentage was also discussed.

It is very clear that in a hybrid system the price for perfect performance in the first pass has to be paid by a lot of manual review in the second pass. Clearly, the performance requirements for the algorithms are affected by the importance of the linkage task being performed. When medical databases are to be linked [29], the process is often critical and very stringent thresholds are set for the algorithms. However, when genealogy records are linked, small error rates are often acceptable. In situations where a small amount of errors in predictions are allowable, the amount of manual effort that can be saved by slightly lowering the performance requirements is immense. In this study, it was observed that when the PPV and NPV requirements are dropped by as little as 1%, the manual review percentage dropped by as much as 50% as shown in figure 6.1. We therefore suggest that the trade-off properties initially be studied on a small subset of the labeled data. Depending on the performance characteristics and the importance of the task at hand, the performance requirements for the automatic linkage can be defined. This can potentially save a lot of time and effort on manual linkage.

The effects of training data size on model performance was also studied. The greatest gains of increasing the amount of training data is when the training data is scarce. The performance quickly saturates as more training data is added. Beyond a point, adding more data has little effect on the performance as per figure 6.2 and figure 6.3. This is consistent with the traditional machine learning curve of training data size and model performance. Acquiring quality labeled data can be expensive and is often resource intensive. Based on the methods studied, it is recommended that only as much training data as required be acquired as simply adding more data has no significant effect. It should also be noted that random forests outperform the other models to a great degree at scarcer training data conditions.

The experiments also show that as heterogeneity rate increases, the model performance goes down in terms of F_1 score. In terms of manual review percentage, SVMs and Neural Nets remain largely constant as heterogeneity rate increases whereas Random Forests suffer. As discussed in section 6.3.2, one plausible explanation for this behavior might be that SVMs and Neural Nets have bad review percentage performance even at lower heterogeneity rates, that they do not get worse after that. Random forests on the other hand perform exceptionally well at lower heterogeneity rates and there is a lot of room for performance to deteriorate.

In almost all scenarios with different heterogeneity rates and training set sizes Random Forests outperform SVMs and Neural Nets both in terms of F_1 score and review percentage. Other metrics like area under ROC curve (AUC), Brier Score, etc. were also measured and they showed similar trends as well. Traditionally, F_1 score has been the primary measure to benchmark algorithm performance in record linkage. Though Random Forests have a better F_1 performance at all heterogeneity rates than SVMs (0.2% better at 0% heterogeneity rate and 1.2% better at 60%) and Neural Nets (0.2% better at 0% heterogeneity rate and 0.7% better at 60%), the effect seems inconsequentially minor. However, the F_1 score seems to tell only part of the story.

The manual review percentage at 99% PPV and NPV paints a new picture. SVMs and Neural Networks have a mean review percent of 20% and Random Forests have a mean review percentage of 14.3% across all rates. Random Forests outperform the two methods by 28% on average which can be a significant improvement in real world linkage projects as the manual review needed is minimized. With more stringent requirements (PPV/NPV = 100%), this effect is accentuated - random forests outperforms the models by 48% on average across the heterogeneity rates.

8. CONCLUSIONS

As the amount of data gathered in the digital age increases and privacy becomes more and more important, data integration becomes a critical problem that is tough to solve. Record linkage is an important data integration step in the data analysis pipeline as the speed and quality of linkage have a considerable impact on the validity of the decisions made downstream and the completion rate of the data analysis project. In a bid to study and understand how different factors affect the process of record linkage in order to achieve better linkage performance, the performances of three popular machine learning algorithms in record linkage - random forests, support vector machines and neural nets were studied under two controlled dimensions - heterogeneity rates and training set sizes.

Firstly, it was observed that even a small compromise with respect to the quality of linkage could have a considerable impact on the percentage of pairs manually reviewed in a hybrid system. Hence, the quality requirements of the automatic linkage step needs to be carefully studied and defined. Secondly, the amount of training data added has diminishing returns. Beyond a certain amount of training data, the performance does not show a commensurate improvement in performance. Hence, resources should not be wasted in acquiring more data after performance plateaus. Finally, In our experiments, random forests show a better performance than the SVM and Dense Neural Net model with both the metrics at all the heterogeneity rates and training set sizes. Though the difference in performance seems marginal with respect to the F_1 score, in terms of the percentage of manual review, they perform substantially better. Hence, random forests should definitely be used at least as benchmarking models in record linkage problems.

REFERENCES

[1] H. Kang, L. Getoor, B. Shneiderman, M. Bilgic, and L. Licamele, “Interactive entity resolution in relational data: A visual analytic tool and its evaluation,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 5, pp. 999–1014, 2008.

[2] M. Bilgic, L. Licamele, L. Getoor, and B. Shneiderman, “D-dupe: An interactive tool for entity resolution in social networks,” in *2006 IEEE Symposium on Visual Analytics Science and Technology*, pp. 43–50, IEEE, 2006.

[3] L. Antonie, K. Inwood, D. J. Lizotte, and J. A. Ross, “Tracking people over time in 19th century canada for longitudinal analysis,” *Machine Learning*, vol. 95, no. 1, pp. 129–146, 2014.

[4] S. Sarawagi and A. Bhamidipaty, “Interactive deduplication using active learning,” in *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 269–278, ACM, 2002.

[5] J. J. Feigenbaum, “Automated census record linking: A machine learning approach,” 2016.

[6] K. Kim and C. L. Giles, “Financial entity record linkage with random forests,” in *Proceedings of the Second International Workshop on Data Science for Macro-Modeling*, p. 13, ACM, 2016.

[7] P. Treeratpituk and C. L. Giles, “Disambiguating authors in academic publications using random forests,” in *Proceedings of the 9th ACM/IEEE-CS Joint Conference on Digital Libraries*, pp. 39–48, ACM, 2009.

[8] B. Pixton and C. Giraud-Carrier, “Using structured neural networks for record linkage,” in *Proceedings of the Sixth Annual Workshop on Technology for Family History and Genealogical Research*, 2006.

[9] D. R. Wilson, “Beyond probabilistic record linkage: Using neural networks and complex features to improve genealogical record linkage,” in *The 2011 International Joint Conference on Neural Networks*, pp. 9–14, IEEE, 2011.

[10] H. Köpcke, A. Thor, and E. Rahm, “Evaluation of entity resolution approaches on real-world match problems,” *Proceedings of the VLDB Endowment*, vol. 3, no. 1-2, pp. 484–493, 2010.

[11] G. Darroch, “Semi-automated record linkage with surname samples: a regional study of âĂŸcase lawâĂŹlinkage, ontario 1861–1871,” *History and Computing*, vol. 14, no. 1-2, pp. 153–183, 2002.

[12] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios, “Duplicate record detection: A survey,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 1, pp. 1–16, 2007.

[13] B. J. Tepping, “A model for optimum linkage of records,” *Journal of the American Statistical Association*, vol. 63, no. 324, pp. 1321–1332, 1968.

[14] E. D. Ragan, H.-C. Kum, G. Ilangovan, and H. Wang, “Balancing privacy and information disclosure in interactive record linkage with visual masking,” in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, p. 326, ACM, 2018.

[15] V. I. Levenshtein, “Binary codes capable of correcting deletions, insertions, and reversals,” in *Soviet Physics Doklady*, vol. 10, pp. 707–710, 1966.

[16] F. J. Damerau, “A technique for computer detection and correction of spelling errors,” *Communications of the ACM*, vol. 7, no. 3, pp. 171–176, 1964.

[17] M. A. Jaro, *UNIMATCH, a record linkage system: users manual*. Bureau of the Census, 1980.

[18] W. E. Winkler and Y. Thibaudeau, *An application of the Fellegi-Sunter model of record linkage to the 1990 US decennial census*. Citeseer, 1991.

[19] J. Mortimer and J. Salathiel, "'Soundex'codes of surnames provide confidentiality and accuracy in a national hiv database.," *Communicable Disease Report. CDR Review*, vol. 5, no. 12, pp. R183–6, 1995.

[20] H. B. Newcombe, "Record linking: the design of efficient systems for linking records into individual and family histories," *American Journal of Human Genetics*, vol. 19, no. 3 Pt 1, p. 335, 1967.

[21] J. Wang and S. E. Madnick, "The inter-database instance identification problem in integrating autonomous systems," in *[1989] Proceedings. Fifth International Conference on Data Engineering*, pp. 46–55, IEEE, 1989.

[22] I. P. Fellegi and A. B. Sunter, "A theory for record linkage," *Journal of the American Statistical Association*, vol. 64, no. 328, pp. 1183–1210, 1969.

[23] M. Bilenko and R. J. Mooney, "Adaptive duplicate detection using learnable string similarity measures," in *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 39–48, ACM, 2003.

[24] Y. Siegert, X. Jiang, V. Krieg, and S. Bartholomäus, "Classification-based record linkage with pseudonymized data for epidemiological cancer registries," *IEEE Transactions on Multimedia*, vol. 18, no. 10, pp. 1929–1941, 2016.

[25] M. Michelson and C. A. Knoblock, "Learning blocking schemes for record linkage," in *AAAI*, vol. 6, pp. 440–445, 2006.

[26] M. A. Hernández and S. J. Stolfo, "Real-world data is dirty: Data cleansing and the merge/purge problem," *Data Mining and Knowledge Discovery*, vol. 2, no. 1, pp. 9–37, 1998.

[27] P. Christen and K. Goiser, "Quality and complexity measures for data linkage and deduplication," in *Quality Measures in Data Mining*, pp. 127–151, Springer, 2007.

[28] D. Hand and P. Christen, "A note on using the f-measure for evaluating data linkage algorithms," no. Pre-print NI16047, 2016.

[29] E. Joffe, M. J. Byrne, P. Reeder, J. R. Herskovic, C. W. Johnson, A. B. McCoy, D. F. Sittig, and E. V. Bernstam, “A benchmark comparison of deterministic and probabilistic methods for defining manual review datasets in duplicate records reconciliation,” *Journal of the American Medical Informatics Association*, vol. 21, no. 1, pp. 97–104, 2013.

[30] P. Christen, “Preparation of a real temporal voter data set for record linkage and duplicate detection research,” 2013.

APPENDIX A

DETAILED RESULTS

In this chapter, all the results are displayed in detail. The metrics are

1. F_1 score
2. Recall
3. Precision
4. Review Percent at 100% PPV and NPV
5. Review Percent at 99% PPV and NPV
6. Review Percent at 95% PPV and NPV
7. Review Percent at 90% PPV and NPV

At all combinations of training set sizes and heterogeneity rates, these metrics are calculated for the three models and displayed.

A.1 F_1 Results

Training size	Heterogeneity (%)	Dense Neural Net (%)	SVM RBF (%)	Random Forest (%)
1000	0	97.89 +/- 0.86	79.24 +/- 2.67	99.34 +/- 0.39
	5	95.68 +/- 0.86	96.73 +/- 0.68	96.93 +/- 0.54
	10	95.89 +/- 0.70	95.65 +/- 0.68	96.53 +/- 0.55
	15	94.82 +/- 0.58	96.67 +/- 0.61	96.45 +/- 0.72
	20	94.40 +/- 0.89	95.83 +/- 1.02	95.66 +/- 0.93
	25	92.46 +/- 0.92	94.59 +/- 0.63	93.99 +/- 0.83
	30	93.76 +/- 0.76	94.54 +/- 0.52	95.85 +/- 0.63
	35	94.34 +/- 0.92	94.53 +/- 0.84	95.25 +/- 0.90
	40	93.54 +/- 0.83	93.81 +/- 1.06	95.52 +/- 0.86
	45	92.06 +/- 0.53	93.57 +/- 0.80	94.05 +/- 0.81
	50	92.85 +/- 1.17	93.87 +/- 0.87	94.60 +/- 0.83
	55	92.79 +/- 1.41	93.49 +/- 1.02	95.11 +/- 0.88
	60	92.91 +/- 0.82	92.90 +/- 0.75	94.56 +/- 0.62
4000	0	98.56 +/- 0.54	90.75 +/- 1.63	99.33 +/- 0.41
	5	97.56 +/- 0.69	97.79 +/- 0.21	98.27 +/- 0.23
	10	96.96 +/- 0.52	97.16 +/- 0.54	97.39 +/- 0.65
	15	96.72 +/- 0.67	97.25 +/- 0.38	97.71 +/- 0.66
	20	96.48 +/- 0.71	96.58 +/- 0.61	96.84 +/- 0.37
	25	95.94 +/- 0.67	95.58 +/- 0.75	96.61 +/- 0.59
	30	96.35 +/- 0.71	96.43 +/- 0.60	96.76 +/- 0.69
	35	94.13 +/- 0.78	95.86 +/- 0.70	96.99 +/- 0.70
	40	95.42 +/- 1.34	96.07 +/- 0.67	96.79 +/- 0.68
	45	95.42 +/- 0.51	95.50 +/- 0.45	96.24 +/- 0.73
	50	94.94 +/- 1.18	95.43 +/- 1.06	96.02 +/- 0.94
	55	93.85 +/- 1.24	95.59 +/- 1.17	96.51 +/- 0.88
	60	94.70 +/- 0.90	95.07 +/- 0.54	95.96 +/- 0.65

Table A.1: F_1 scores at training set sizes of 1000 and 4000

Training size	**Heterogeneity (%)**	**Dense Neural Net (%)**	**SVM (RBF) (%)**	**Random Forest (%)**
7000	0	99.15 +/- 0.45	99.28 +/- 0.41	99.50 +/- 0.40
	5	97.66 +/- 0.40	98.17 +/- 0.33	98.41 +/- 0.18
	10	97.47 +/- 0.73	97.77 +/- 0.79	98.13 +/- 0.49
	15	97.41 +/- 0.65	97.60 +/- 0.67	97.90 +/- 0.59
	20	96.79 +/- 0.59	97.11 +/- 0.49	97.31 +/- 0.39
	25	96.55 +/- 0.70	96.43 +/- 0.58	97.17 +/- 0.60
	30	96.91 +/- 0.81	96.84 +/- 0.61	96.97 +/- 0.69
	35	96.26 +/- 0.80	96.28 +/- 0.67	97.32 +/- 0.56
	40	96.57 +/- 0.88	96.55 +/- 0.78	96.92 +/- 0.61
	45	95.70 +/- 0.84	96.14 +/- 0.40	96.84 +/- 0.54
	50	95.13 +/- 1.29	95.93 +/- 1.05	96.60 +/- 0.68
	55	96.03 +/- 0.61	96.46 +/- 0.80	96.85 +/- 0.81
	60	95.69 +/- 0.65	95.33 +/- 0.72	96.40 +/- 0.73
10000	0	99.28 +/- 0.41	99.45 +/- 0.38	99.44 +/- 0.44
	5	98.33 +/- 0.27	98.18 +/- 0.29	98.74 +/- 0.30
	10	97.99 +/- 0.42	97.87 +/- 0.68	98.33 +/- 0.50
	15	97.43 +/- 0.56	97.79 +/- 0.61	98.12 +/- 0.70
	20	97.02 +/- 0.39	97.18 +/- 0.50	97.60 +/- 0.50
	25	96.21 +/- 0.77	96.64 +/- 0.41	97.47 +/- 0.65
	30	96.42 +/- 0.80	96.97 +/- 0.57	97.17 +/- 0.53
	35	96.11 +/- 0.57	96.62 +/- 0.53	97.49 +/- 0.49
	40	96.36 +/- 0.79	96.61 +/- 0.93	97.15 +/- 0.47
	45	96.35 +/- 0.71	96.44 +/- 0.40	96.98 +/- 0.55
	50	96.06 +/- 1.07	96.35 +/- 0.86	96.94 +/- 0.70
	55	96.21 +/- 0.91	96.65 +/- 0.78	97.09 +/- 0.59
	60	95.20 +/- 0.83	95.68 +/- 0.56	96.59 +/- 0.60

Table A.2: F_1 scores at training set sizes of 7000 and 10,000

A.2 Recall Results

Training size	Heterogeneity (%)	Dense Neural Net (%)	SVM (RBF) (%)	Random Forest (%)
1000	0	98.76 +/- 0.68	70.48 +/- 3.01	99.56 +/- 0.35
	5	94.97 +/- 1.24	96.11 +/- 1.18	97.38 +/- 0.85
	10	95.48 +/- 0.90	96.46 +/- 0.75	97.72 +/- 0.79
	15	93.57 +/- 0.54	95.24 +/- 0.88	96.46 +/- 0.82
	20	95.47 +/- 0.86	96.27 +/- 0.96	95.46 +/- 1.13
	25	94.21 +/- 0.71	93.97 +/- 0.81	93.69 +/- 0.95
	30	94.58 +/- 0.78	96.00 +/- 0.91	95.84 +/- 0.85
	35	95.16 +/- 1.07	94.61 +/- 1.37	94.06 +/- 1.08
	40	92.47 +/- 0.99	93.36 +/- 1.28	94.97 +/- 1.56
	45	95.68 +/- 0.45	92.95 +/- 1.03	92.67 +/- 0.99
	50	90.90 +/- 1.59	93.65 +/- 1.28	93.52 +/- 1.21
	55	93.25 +/- 1.45	93.78 +/- 1.20	96.19 +/- 1.26
	60	91.46 +/- 0.95	92.97 +/- 0.65	93.09 +/- 1.18
4000	0	99.12 +/- 0.54	87.51 +/- 1.80	99.59 +/- 0.24
	5	96.69 +/- 1.19	97.63 +/- 0.84	98.27 +/- 0.56
	10	96.39 +/- 0.71	96.19 +/- 0.94	97.16 +/- 0.89
	15	96.87 +/- 0.98	97.50 +/- 0.62	97.14 +/- 0.75
	20	96.37 +/- 0.46	95.67 +/- 0.78	96.29 +/- 0.79
	25	94.63 +/- 1.05	94.56 +/- 1.04	95.64 +/- 0.60
	30	96.85 +/- 0.73	96.34 +/- 0.59	96.50 +/- 0.56
	35	94.70 +/- 1.12	94.92 +/- 0.99	96.24 +/- 0.99
	40	93.52 +/- 2.03	95.40 +/- 1.27	96.26 +/- 1.24
	45	95.63 +/- 0.82	95.35 +/- 0.47	96.14 +/- 1.10
	50	95.15 +/- 1.26	94.90 +/- 1.23	95.80 +/- 0.86
	55	92.17 +/- 1.86	95.07 +/- 1.47	95.91 +/- 1.42
	60	94.07 +/- 1.69	94.14 +/- 1.15	94.94 +/- 1.17

Table A.3: Recall scores at training set sizes of 1000 and 4000

Training size	Heterogeneity (%)	Dense Neural Net (%)	SVM (RBF) (%)	Random Forest (%)
7000	0	99.08 +/- 0.65	99.32 +/- 0.40	99.67 +/- 0.26
	5	97.22 +/- 0.95	98.01 +/- 0.82	98.44 +/- 0.57
	10	97.06 +/- 1.24	97.69 +/- 1.08	98.28 +/- 0.84
	15	97.84 +/- 0.89	97.76 +/- 0.78	97.81 +/- 0.97
	20	96.21 +/- 0.78	96.80 +/- 0.79	96.98 +/- 0.75
	25	95.85 +/- 1.16	95.82 +/- 0.85	96.84 +/- 0.88
	30	97.19 +/- 0.58	96.71 +/- 0.38	96.94 +/- 0.60
	35	95.76 +/- 1.23	95.97 +/- 1.40	96.92 +/- 0.92
	40	95.89 +/- 1.41	95.83 +/- 1.34	96.71 +/- 0.89
	45	95.28 +/- 1.39	95.52 +/- 0.61	96.72 +/- 0.82
	50	94.43 +/- 1.29	95.60 +/- 1.34	96.04 +/- 0.92
	55	96.17 +/- 1.02	95.81 +/- 1.14	96.59 +/- 1.32
	60	95.76 +/- 1.00	94.35 +/- 1.10	96.07 +/- 1.11
10000	0	99.25 +/- 0.43	99.61 +/- 0.30	99.64 +/- 0.30
	5	98.83 +/- 0.51	97.89 +/- 0.71	98.80 +/- 0.57
	10	97.99 +/- 0.76	98.04 +/- 0.87	98.61 +/- 0.91
	15	97.30 +/- 0.87	97.80 +/- 0.64	98.11 +/- 0.89
	20	97.62 +/- 0.70	96.92 +/- 0.68	97.41 +/- 0.60
	25	97.01 +/- 1.16	96.21 +/- 1.06	97.42 +/- 0.81
	30	96.22 +/- 0.98	96.88 +/- 0.46	96.88 +/- 0.49
	35	95.76 +/- 1.28	96.35 +/- 1.14	97.36 +/- 0.68
	40	95.91 +/- 1.39	96.18 +/- 1.52	97.00 +/- 0.96
	45	96.48 +/- 0.83	95.88 +/- 0.52	97.30 +/- 0.59
	50	96.16 +/- 1.46	96.04 +/- 1.18	96.77 +/- 0.94
	55	95.41 +/- 1.09	96.03 +/- 1.04	96.72 +/- 1.21
	60	95.29 +/- 1.34	95.07 +/- 0.96	96.28 +/- 1.05

Table A.4: Recall scores at training set sizes of 7000 and 10,000

A.3 Precision Results

Training size	**Heterogeneity (%)**	**Dense Neural Net (%)**	**SVM (RBF) (%)**	**Random Forest (%)**
1000	0	97.05 +/- 1.49	90.52 +/- 2.47	99.13 +/- 0.59
	5	96.41 +/- 1.16	97.39 +/- 1.23	96.50 +/- 1.05
	10	96.30 +/- 0.64	94.86 +/- 0.92	95.36 +/- 0.48
	15	96.11 +/- 1.07	98.15 +/- 0.68	96.45 +/- 1.18
	20	93.37 +/- 1.35	95.40 +/- 1.38	95.88 +/- 1.22
	25	90.79 +/- 1.69	95.22 +/- 0.81	94.29 +/- 1.16
	30	92.95 +/- 1.31	93.14 +/- 1.13	95.87 +/- 1.30
	35	93.54 +/- 1.10	94.47 +/- 0.96	96.48 +/- 1.38
	40	94.66 +/- 1.46	94.29 +/- 1.60	96.10 +/- 0.85
	45	88.70 +/- 0.76	94.22 +/- 1.35	95.48 +/- 0.83
	50	94.92 +/- 1.48	94.10 +/- 1.16	95.71 +/- 1.07
	55	92.35 +/- 1.96	93.22 +/- 1.34	94.05 +/- 0.88
	60	94.42 +/- 1.19	92.85 +/- 1.71	96.09 +/- 1.07
4000	0	98.01 +/- 0.89	94.27 +/- 2.09	99.07 +/- 0.66
	5	98.45 +/- 0.52	97.97 +/- 0.70	98.27 +/- 0.56
	10	97.55 +/- 0.70	98.16 +/- 0.64	97.62 +/- 0.68
	15	96.58 +/- 1.22	97.01 +/- 0.88	98.30 +/- 0.84
	20	96.60 +/- 1.21	97.51 +/- 0.75	97.41 +/- 0.65
	25	97.29 +/- 0.72	96.63 +/- 0.98	97.60 +/- 0.95
	30	95.86 +/- 1.07	96.54 +/- 1.11	97.02 +/- 1.16
	35	93.58 +/- 1.71	96.83 +/- 1.14	97.77 +/- 1.09
	40	97.41 +/- 1.12	96.77 +/- 1.11	97.34 +/- 0.68
	45	95.22 +/- 0.89	95.67 +/- 1.12	96.34 +/- 0.56
	50	94.73 +/- 1.39	95.99 +/- 1.49	96.26 +/- 1.39
	55	95.61 +/- 1.08	96.12 +/- 1.13	97.13 +/- 0.59
	60	95.36 +/- 0.89	96.04 +/- 0.72	97.01 +/- 0.39

Table A.5: Precision scores at training set sizes of 1000 and 4000

Training size	**Heterogeneity (%)**	**Dense Neural Net (%)**	**SVM (RBF) (%)**	**Random Forest (%)**
7000	0	99.22 +/- 0.64	99.25 +/- 0.69	99.33 +/- 0.58
	5	98.12 +/- 0.71	98.33 +/- 0.67	98.39 +/- 0.56
	10	97.89 +/- 0.79	97.85 +/- 0.83	97.98 +/- 0.64
	15	96.99 +/- 0.86	97.45 +/- 1.15	97.99 +/- 0.73
	20	97.38 +/- 0.63	97.42 +/- 0.71	97.64 +/- 0.85
	25	97.28 +/- 1.09	97.07 +/- 0.94	97.51 +/- 0.93
	30	96.64 +/- 1.39	96.98 +/- 1.04	97.01 +/- 0.97
	35	96.77 +/- 1.07	96.63 +/- 1.40	97.73 +/- 0.95
	40	97.27 +/- 1.03	97.30 +/- 1.32	97.15 +/- 0.79
	45	96.15 +/- 0.74	96.78 +/- 0.88	96.97 +/- 0.71
	50	95.84 +/- 1.55	96.26 +/- 1.20	97.17 +/- 0.92
	55	95.90 +/- 0.68	97.13 +/- 0.78	97.13 +/- 0.70
	60	95.62 +/- 0.68	96.34 +/- 1.01	96.73 +/- 0.59
10000	0	99.30 +/- 0.55	99.28 +/- 0.66	99.24 +/- 0.64
	5	97.83 +/- 0.48	98.47 +/- 0.47	98.68 +/- 0.59
	10	98.00 +/- 0.53	97.72 +/- 0.83	98.07 +/- 0.56
	15	97.57 +/- 0.85	97.79 +/- 1.00	98.14 +/- 0.91
	20	96.44 +/- 0.89	97.45 +/- 0.77	97.80 +/- 0.94
	25	95.44 +/- 1.23	97.08 +/- 0.79	97.53 +/- 0.88
	30	96.64 +/- 1.34	97.08 +/- 1.04	97.46 +/- 0.96
	35	96.48 +/- 1.04	96.91 +/- 1.19	97.64 +/- 0.93
	40	96.82 +/- 0.90	97.05 +/- 1.11	97.32 +/- 0.70
	45	96.22 +/- 1.02	97.00 +/- 0.71	96.68 +/- 0.76
	50	95.97 +/- 1.22	96.67 +/- 0.94	97.13 +/- 1.06
	55	97.02 +/- 0.91	97.29 +/- 0.96	97.48 +/- 0.53
	60	95.11 +/- 1.00	96.31 +/- 0.83	96.90 +/- 0.60

Table A.6: Precision scores at training set sizes of 7000 and 10,000

A.4 Percentage of Manual Review at 100% PPV and NPV Results

Training size	Heterogeneity (%)	Dense Neural Net (%)	SVM (RBF) (%)	Random Forest (%)
1000	0	23.56 +/- 04.20	91.41 +/- 08.54	19.08 +/- 1.22
	5	73.01 +/- 20.93	54.26 +/- 30.61	35.20 +/- 11.21
	10	57.83 +/- 21.02	38.59 +/- 17.38	26.86 +/- 5.86
	15	56.14 +/- 26.71	49.76 +/- 24.33	32.11 +/- 6.35
	20	60.22 +/- 26.01	47.25 +/- 20.99	30.65 +/- 9.36
	25	92.14 +/- 08.75	58.78 +/- 26.84	26.98 +/- 8.30
	30	46.61 +/- 15.48	49.80 +/- 11.45	28.33 +/- 4.13
	35	51.06 +/- 21.10	51.00 +/- 23.34	36.75 +/- 10.84
	40	95.55 +/- 04.51	70.87 +/- 30.83	37.42 +/- 5.54
	45	85.44 +/- 18.32	72.18 +/- 24.52	34.82 +/- 3.72
	50	80.23 +/- 15.28	62.94 +/- 18.74	37.16 +/- 11.41
	55	78.19 +/- 26.26	47.94 +/- 13.27	28.21 +/- 3.18
	60	61.07 +/- 14.86	63.31 +/- 17.51	41.39 +/- 10.59
4000	0	25.28 +/- 08.66	68.33 +/- 17.59	7.16 +/- 0.76
	5	56.02 +/- 30.01	48.93 +/- 31.49	12.28 +/- 1.91
	10	38.39 +/- 21.86	42.51 +/- 25.25	21.36 +/- 6.54
	15	58.85 +/- 26.44	39.99 +/- 21.29	26.46 +/- 6.24
	20	33.69 +/- 10.53	43.15 +/- 21.03	21.79 +/- 8.05
	25	59.31 +/- 22.20	53.95 +/- 23.08	17.96 +/- 7.10
	30	35.14 +/- 15.95	46.25 +/- 13.80	23.14 +/- 2.75
	35	64.77 +/- 31.37	48.15 +/- 17.16	22.93 +/- 3.06
	40	70.18 +/- 25.77	62.66 +/- 22.79	31.86 +/- 5.91
	45	37.55 +/- 13.93	48.65 +/- 16.62	21.34 +/- 3.01
	50	47.68 +/- 18.68	56.09 +/- 24.77	27.85 +/- 6.74
	55	35.22 +/- 16.10	40.21 +/- 08.39	27.05 +/- 2.68
	60	44.81 +/- 17.63	52.50 +/- 14.57	32.22 +/- 10.36

Table A.7: Review percent (PPV/NPV - 100%) scores at training set sizes of 1000 and 4000

Training size	Heterogeneity (%)	Dense Neural Net (%)	SVM (RBF) (%)	Random Forest (%)
7000	0	25.12 +/- 12.16	20.69 +/- 01.48	17.17 +/- 1.09
	5	47.78 +/- 26.35	44.55 +/- 28.87	16.86 +/- 2.06
	10	54.26 +/- 29.34	29.49 +/- 10.04	16.71 +/- 4.93
	15	50.84 +/- 29.17	37.16 +/- 19.78	17.55 +/- 5.87
	20	46.39 +/- 19.04	36.10 +/- 12.29	24.09 +/- 2.78
	25	32.42 +/- 12.33	44.52 +/- 15.80	16.36 +/- 6.26
	30	30.76 +/- 10.41	36.05 +/- 10.83	20.60 +/- 4.20
	35	40.96 +/- 14.22	43.22 +/- 17.50	17.81 +/- 2.29
	40	31.87 +/- 08.96	56.05 +/- 22.19	16.58 +/- 3.49
	45	33.02 +/- 12.98	45.85 +/- 19.44	23.86 +/- 1.85
	50	58.89 +/- 20.32	48.76 +/- 19.42	26.70 +/- 2.97
	55	34.34 +/- 15.00	37.73 +/- 09.46	24.20 +/- 1.67
	60	43.03 +/- 20.48	50.32 +/- 11.96	25.76 +/- 8.58
10000	0	20.96 +/- 01.60	21.46 +/- 2.28	16.44 +/- 1.24
	5	32.27 +/- 16.90	44.02 +/- 29.19	12.17 +/- 3.32
	10	37.88 +/- 19.53	29.58 +/- 11.25	18.45 +/- 4.63
	15	29.22 +/- 08.79	38.43 +/- 20.55	17.41 +/- 5.00
	20	30.20 +/- 08.46	33.56 +/- 09.90	21.71 +/- 2.08
	25	30.88 +/- 15.00	36.08 +/- 10.17	15.96 +/- 3.99
	30	26.61 +/- 03.77	35.99 +/- 10.33	22.55 +/- 2.23
	35	36.23 +/- 14.11	44.66 +/- 16.94	16.85 +/- 4.94
	40	46.27 +/- 19.92	49.03 +/- 17.00	24.72 +/- 4.36
	45	28.32 +/- 05.60	45.57 +/- 19.58	18.51 +/- 4.36
	50	42.19 +/- 23.67	45.70 +/- 13.95	22.06 +/- 6.37
	55	32.46 +/- 09.04	36.01 +/- 08.48	20.78 +/- 3.83
	60	38.94 +/- 07.31	46.15 +/- 13.03	24.04 +/- 7.79

Table A.8: Review percent (PPV/NPV - 100%) scores at training set sizes of 7000 and 10,000

A.5 Percentage of Manual Review at 99% PPV and NPV Results

Training size	**Heterogeneity (%)**	**Dense Neural Net (%)**	**SVM (RBF) (%)**	**Random Forest (%)**
1000	0	20.21 +/- 1.23	55.61 +/-18.15	18.88 +/- 1.16
	5	20.35 +/- 0.86	19.92 +/- 0.90	19.34 +/- 1.01
	10	19.71 +/- 0.78	20.30 +/- 0.82	19.76 +/- 0.82
	15	20.50 +/- 1.08	19.68 +/- 0.88	18.17 +/- 0.75
	20	20.22 +/- 0.84	20.54 +/- 0.95	20.22 +/- 0.76
	25	22.11 +/- 1.66	21.37 +/- 1.97	14.86 +/- 1.19
	30	21.37 +/- 0.98	21.05 +/- 0.79	19.05 +/- 0.81
	35	20.04 +/- 1.00	21.14 +/- 1.01	20.19 +/- 0.56
	40	21.69 +/- 2.63	21.36 +/- 1.46	18.55 +/- 1.10
	45	21.69 +/- 1.21	21.74 +/- 1.40	20.62 +/- 0.87
	50	21.52 +/- 1.14	21.45 +/- 1.25	20.59 +/- 0.92
	55	22.06 +/- 1.18	22.16 +/- 1.47	20.60 +/- 1.36
	60	22.09 +/- 1.27	22.25 +/- 1.17	20.89 +/- 1.02
4000	0	20.09 +/- 1.27	23.01 +/- 1.48	06.98 +/- 0.71
	5	19.31 +/- 0.91	19.94 +/- 0.89	09.02 +/- 0.56
	10	19.71 +/- 0.77	19.64 +/- 0.74	13.08 +/- 0.79
	15	19.79 +/- 0.80	19.78 +/- 0.88	19.26 +/- 0.85
	20	20.12 +/- 0.79	20.18 +/- 0.96	9.88 +/- 0.64
	25	20.06 +/- 1.58	20.65 +/- 1.75	11.68 +/- 0.81
	30	20.30 +/- 0.74	20.14 +/- 0.69	15.58 +/- 0.79
	35	20.86 +/- 0.80	20.16 +/- 0.88	16.27 +/- 0.80
	40	19.97 +/- 1.22	19.96 +/- 1.29	17.92 +/- 0.97
	45	20.03 +/- 0.99	20.04 +/- 1.02	14.46 +/- 1.27
	50	20.29 +/- 1.01	20.30 +/- 1.02	16.53 +/- 0.87
	55	21.35 +/- 1.52	20.52 +/- 1.21	19.10 +/- 1.25
	60	20.95 +/- 0.90	20.85 +/- 0.88	18.95 +/- 0.66

Table A.9: Review percent (PPV/NPV - 99%) scores at training set sizes of 1000 and 4000

Training size	**Heterogeneity (%)**	**Dense Neural Net (%)**	**SVM (RBF) (%)**	**Random Forest (%)**
7000	0	19.85 +/- 1.37	19.89 +/- 1.36	16.97 +/- 1.04
	5	19.60 +/- 1.01	19.94 +/- 0.89	13.54 +/- 0.87
	10	19.05 +/- 0.87	19.88 +/- 0.89	07.78 +/- 0.44
	15	19.38 +/- 0.95	19.75 +/- 0.88	10.35 +/- 0.57
	20	19.97 +/- 0.81	20.04 +/- 0.75	19.14 +/- 0.71
	25	19.79 +/- 1.39	19.88 +/- 1.41	10.06 +/- 0.62
	30	20.22 +/- 0.86	20.08 +/- 0.75	14.54 +/- 0.75
	35	20.01 +/- 0.75	20.11 +/- 0.78	12.54 +/- 0.59
	40	19.76 +/- 1.10	19.79 +/- 1.03	10.49 +/- 0.60
	45	19.95 +/- 1.21	19.72 +/- 0.95	18.31 +/- 0.92
	50	20.15 +/- 1.22	20.09 +/- 1.07	18.56 +/- 0.78
	55	20.31 +/- 1.32	20.16 +/- 1.35	18.58 +/- 0.98
	60	20.19 +/- 0.75	20.61 +/- 0.89	15.62 +/- 0.69
10000	0	19.86 +/- 1.36	19.94 +/- 1.37	16.25 +/- 1.21
	5	19.83 +/- 0.82	19.89 +/- 0.89	8.43 +/- 0.52
	10	19.16 +/- 1.00	19.96 +/- 0.96	12.44 +/- 0.68
	15	19.60 +/- 0.78	19.68 +/- 0.82	10.71 +/- 0.62
	20	20.24 +/- 0.73	20.05 +/- 0.72	16.72 +/- 0.49
	25	20.22 +/- 1.36	19.84 +/- 1.31	11.15 +/- 0.74
	30	20.17 +/- 0.82	20.10 +/- 0.80	16.86 +/- 0.73
	35	20.11 +/- 0.68	20.06 +/- 0.79	10.70 +/- 0.56
	40	19.83 +/- 0.93	19.81 +/- 0.99	17.77 +/- 0.68
	45	19.74 +/- 1.07	19.59 +/- 1.00	11.44 +/- 0.85
	50	20.03 +/- 0.94	19.95 +/- 0.88	14.88 +/- 0.91
	55	20.07 +/- 1.05	20.28 +/- 1.75	15.34 +/- 1.06
	60	20.70 +/- 0.75	20.44 +/- 0.76	15.34 +/- 0.55

Table A.10: Review percent (PPV/NPV - 99%) scores at training set sizes of 7000 and 10,000

A.6 Percentage of Manual Review at 95% PPV and NPV Results

Training size	**Heterogeneity (%)**	**Dense Neural Net (%)**	**SVM (RBF) (%)**	**Random Forest (%)**
1000	0	20.21 +/- 1.23	18.46 +/- 1.42	18.88 +/- 1.16
	5	19.70 +/- 1.00	19.75 +/- 1.00	19.33 +/- 1.01
	10	19.27 +/- 0.86	20.22 +/- 0.82	19.76 +/- 0.82
	15	19.13 +/- 0.76	19.10 +/- 0.94	18.04 +/- 0.65
	20	19.75 +/- 0.73	20.33 +/- 0.82	19.89 +/- 0.77
	25	20.51 +/- 1.22	19.58 +/- 1.18	13.54 +/- 0.58
	30	20.49 +/- 0.73	20.76 +/- 0.91	18.73 +/- 0.79
	35	19.51 +/- 0.79	20.04 +/- 0.81	19.36 +/- 0.78
	40	18.52 +/- 0.91	19.61 +/- 0.98	17.75 +/- 0.83
	45	21.25 +/- 1.09	19.42 +/- 0.79	18.99 +/- 0.81
	50	18.99 +/- 0.84	19.75 +/- 0.82	19.28 +/- 0.83
	55	20.18 +/- 0.90	20.19 +/- 0.89	20.28 +/- 0.91
	60	19.52 +/- 0.66	20.25 +/- 0.90	19.41 +/- 0.79
4000	0	20.09 +/- 1.27	18.44 +/- 1.21	06.98 +/- 0.71
	5	19.22 +/- 0.95	19.94 +/- 0.90	09.02 +/- 0.56
	10	19.65 +/- 0.83	19.49 +/- 0.86	13.08 +/- 0.79
	15	19.74 +/- 0.81	19.78 +/- 0.88	19.25 +/- 0.85
	20	20.08 +/- 0.76	19.76 +/- 0.72	09.76 +/- 0.50
	25	19.17 +/- 1.23	19.41 +/- 1.08	11.51 +/- 0.65
	30	20.26 +/- 0.72	20.09 +/- 0.64	15.55 +/- 0.81
	35	20.25 +/- 0.74	19.61 +/- 0.78	16.14 +/- 0.76
	40	19.00 +/- 0.84	19.52 +/- 0.96	17.77 +/- 0.90
	45	19.76 +/- 0.88	19.64 +/- 1.04	14.29 +/- 1.27
	50	19.86 +/- 0.78	19.62 +/- 0.82	16.32 +/- 0.90
	55	19.35 +/- 0.89	19.85 +/- 0.87	18.79 +/- 0.98
	60	19.82 +/- 0.88	19.82 +/- 0.74	18.42 +/- 0.70

Table A.11: Review percent (PPV/NPV - 95%) scores at training set sizes of 1000 and 4000

Training size	**Heterogeneity (%)**	**Dense Neural Net (%)**	**SVM (RBF) (%)**	**Random Forest (%)**
7000	0	19.85 +/- 1.37	19.89 +/- 1.36	16.97 +/- 1.04
	5	19.58 +/- 1.04	19.94 +/- 0.89	13.54 +/- 0.87
	10	19.01 +/- 0.96	19.86 +/- 0.93	07.78 +/- 0.44
	15	19.38 +/- 0.95	19.75 +/- 0.88	10.35 +/- 0.57
	20	19.84 +/- 0.76	20.01 +/- 0.74	19.11 +/- 0.70
	25	19.53 +/- 1.21	19.59 +/- 1.21	10.02 +/- 0.60
	30	20.22 +/- 0.86	20.08 +/- 0.75	14.53 +/- 0.76
	35	19.77 +/- 0.74	19.87 +/- 0.83	12.50 +/- 0.64
	40	19.50 +/- 0.99	19.50 +/- 0.83	10.44 +/- 0.59
	45	19.50 +/- 1.02	19.44 +/- 0.99	18.29 +/- 0.93
	50	19.50 +/- 0.75	19.71 +/- 0.81	18.42 +/- 0.75
	55	20.07 +/- 1.00	19.80 +/- 0.92	18.42 +/- 0.78
	60	19.86 +/- 0.71	19.80 +/- 0.77	15.45 +/- 0.64
10000	0	19.86 +/- 1.36	19.94 +/- 1.37	16.25 +/- 1.21
	5	19.83 +/- 0.82	19.89 +/- 0.89	8.43 +/- 0.52
	10	19.16 +/- 1.00	19.96 +/- 0.96	12.44 +/- 0.68
	15	19.59 +/- 0.78	19.68 +/- 0.82	10.71 +/- 0.62
	20	20.24 +/- 0.73	20.03 +/- 0.71	16.72 +/- 0.49
	25	20.18 +/- 1.31	19.66 +/- 1.21	11.15 +/- 0.74
	30	20.06 +/- 0.88	20.10 +/- 0.80	16.84 +/- 0.74
	35	19.85 +/- 0.67	19.89 +/- 0.72	10.69 +/- 0.57
	40	19.58 +/- 0.79	19.62 +/- 0.87	17.75 +/- 0.66
	45	19.65 +/- 1.08	19.47 +/- 1.01	11.44 +/- 0.85
	50	19.85 +/- 0.88	19.71 +/- 0.80	14.82 +/- 0.94
	55	19.74 +/- 0.95	19.82 +/- 1.02	15.22 +/- 0.87
	60	20.25 +/- 0.80	19.96 +/- 0.78	15.18 +/- 0.64

Table A.12: Review percent (PPV/NPV - 95%) scores at training set sizes of 7000 and 10,000

A.7 Percentage of Manual Review at 90% PPV and NPV Results

Training size	Heterogeneity (%)	Dense Neural Net (%)	SVM (RBF) (%)	Random Forest (%)
1000	0	20.21 +/- 1.23	15.46 +/- 1.15	18.88 +/- 1.16
	5	19.70 +/- 1.00	19.75 +/- 1.00	19.33 +/- 1.01
	10	19.27 +/- 0.86	20.22 +/- 0.82	19.76 +/- 0.82
	15	19.13 +/- 0.76	19.10 +/- 0.94	18.04 +/- 0.65
	20	19.75 +/- 0.73	20.33 +/- 0.82	19.89 +/- 0.77
	25	20.51 +/- 1.22	19.58 +/- 1.18	13.54 +/- 0.58
	30	20.49 +/- 0.73	20.76 +/- 0.91	18.73 +/- 0.79
	35	19.51 +/- 0.79	20.04 +/- 0.81	19.36 +/- 0.78
	40	18.52 +/- 0.91	19.61 +/- 0.98	17.75 +/- 0.83
	45	21.25 +/- 1.09	19.42 +/- 0.79	18.99 +/- 0.81
	50	18.99 +/- 0.84	19.75 +/- 0.82	19.28 +/- 0.83
	55	20.18 +/- 0.90	20.19 +/- 0.89	20.28 +/- 0.91
	60	19.52 +/- 0.66	20.25 +/- 0.90	19.41 +/- 0.79
4000	0	20.09 +/- 1.27	18.44 +/- 1.21	06.98 +/- 0.71
	5	19.22 +/- 0.95	19.94 +/- 0.90	09.02 +/- 0.56
	10	19.65 +/- 0.83	19.49 +/- 0.86	13.08 +/- 0.79
	15	19.74 +/- 0.81	19.78 +/- 0.88	19.25 +/- 0.85
	20	20.08 +/- 0.76	19.76 +/- 0.72	09.76 +/- 0.50
	25	19.17 +/- 1.23	19.41 +/- 1.08	11.51 +/- 0.65
	30	20.26 +/- 0.72	20.09 +/- 0.64	15.55 +/- 0.81
	35	20.25 +/- 0.74	19.61 +/- 0.78	16.14 +/- 0.76
	40	19.00 +/- 0.84	19.52 +/- 0.96	17.77 +/- 0.90
	45	19.76 +/- 0.88	19.64 +/- 1.04	14.29 +/- 1.27
	50	19.86 +/- 0.78	19.62 +/- 0.82	16.32 +/- 0.90
	55	19.35 +/- 0.89	19.85 +/- 0.87	18.79 +/- 0.98
	60	19.82 +/- 0.88	19.82 +/- 0.74	18.42 +/- 0.70

Table A.13: Review percent (PPV/NPV - 90%) scores at training set sizes of 1000 and 4000

Training size	**Heterogeneity (%)**	**Dense Neural Net (%)**	**SVM (RBF) (%)**	**Random Forest (%)**
7000	0	19.85 +/- 1.37	19.89 +/- 1.36	16.97 +/- 1.04
	5	19.58 +/- 1.04	19.94 +/- 0.89	13.54 +/- 0.87
	10	19.01 +/- 0.96	19.86 +/- 0.93	07.78 +/- 0.44
	15	19.38 +/- 0.95	19.75 +/- 0.88	10.35 +/- 0.57
	20	19.84 +/- 0.76	20.01 +/- 0.74	19.11 +/- 0.70
	25	19.53 +/- 1.21	19.59 +/- 1.21	10.02 +/- 0.60
	30	20.22 +/- 0.86	20.08 +/- 0.75	14.53 +/- 0.76
	35	19.77 +/- 0.74	19.87 +/- 0.83	12.50 +/- 0.64
	40	19.50 +/- 0.99	19.50 +/- 0.83	10.44 +/- 0.59
	45	19.50 +/- 1.02	19.44 +/- 0.99	18.29 +/- 0.93
	50	19.50 +/- 0.75	19.71 +/- 0.81	18.42 +/- 0.75
	55	20.07 +/- 1.00	19.80 +/- 0.92	18.42 +/- 0.78
	60	19.86 +/- 0.71	19.80 +/- 0.77	15.45 +/- 0.64
10000	0	19.86 +/- 1.36	19.94 +/- 1.37	16.25 +/- 1.21
	5	19.83 +/- 0.82	19.89 +/- 0.89	08.43 +/- 0.52
	10	19.16 +/- 1.00	19.96 +/- 0.96	12.44 +/- 0.68
	15	19.59 +/- 0.78	19.68 +/- 0.82	10.71 +/- 0.62
	20	20.24 +/- 0.73	20.03 +/- 0.71	16.72 +/- 0.49
	25	20.18 +/- 1.31	19.66 +/- 1.21	11.15 +/- 0.74
	30	20.06 +/- 0.88	20.10 +/- 0.80	16.84 +/- 0.74
	35	19.85 +/- 0.67	19.89 +/- 0.72	10.69 +/- 0.57
	40	19.58 +/- 0.79	19.62 +/- 0.87	17.75 +/- 0.66
	45	19.65 +/- 1.08	19.47 +/- 1.01	11.44 +/- 0.85
	50	19.85 +/- 0.88	19.71 +/- 0.80	14.82 +/- 0.94
	55	19.74 +/- 0.95	19.82 +/- 1.02	15.22 +/- 0.87
	60	20.25 +/- 0.80	19.96 +/- 0.78	15.18 +/- 0.64

Table A.14: Review percent (PPV/NPV - 90%) scores at training set sizes of 7000 and 10,000