## 2. Background Information

### 2.1 Maternal Mortality

In 2015, the United Nations committed to achieving 17 Sustainable Development Goals (SDGs) by 2030 to fuel progress toward eliminating global poverty and protecting the planet [1]. Specific SDGs outline important targets that would improve global health and environmental outcomes as well as reduce inequality and conflict [1]. Progress towards the SDGs is monitored by a panel of independent scientists [1]. In 2023, this panel issued warnings that the international community would fail to meet many of the SDGs given the current level of progress and the effect of crises like the COVID-19 pandemic and environmental disasters that stall progress and/or wipe away years of improvement [1].

The report emphasised the lack of progress toward maternal and child mortality goals [1]. In response, in 2024 the Seventy-seventh World Health Assembly passed an additional resolution to increase progress toward reducing maternal and child mortality [3]. This resolution targeted SDG 3.1, which aims to reduce the global maternal mortality ratio (MMR) to below 70 deaths per 100,000 live births by 2030, with no single country having an MMR of greater than 140 [21]. The World Health Organisation defines the MMR as [3]:

“the number of maternal deaths in a given time period per 100 000 live births during the same time period”

The latest version of the International statistical classification of diseases and related health problems (ICD) defines a maternal death as [2]:

“the death of a woman while pregnant or within 42 days of termination of pregnancy, irrespective of the duration and site of the pregnancy, from any cause related to or aggravated by the pregnancy or its management, but not from accidental or incidental causes.”

The concerns about insufficient progress were driven by recent MMR estimates [3]. More specifically, the global MMR was 197 deaths per 100,000 live births (uncertainty interval 174 to 234) in 2023, notably higher than the SDG’s target [3]. Concerningly, this elevated global estimate hides even greater country-specific rates due to substantial country-level inequity, as approximately 95% of maternal deaths occur in low and lower middle-income countries and fragile settings [4]. For example, Nigeria had an MMR of 993 in 2023 (uncertainty interval 718 to 1540) while Australia had an MMR of 2 (uncertainty interval 2 to 4) [3]. As a result, only a small subset of countries is projected to meet SDG 3.1 [4].

The leading global cause of maternal deaths between 2009 and 2020 was haemorrhage [1]. Studies estimate it caused 27% of maternal deaths globally, with a disproportionate incidence in lower income countries [4]. Effective haemorrhage treatments exist, meaning many of these deaths were preventable [4]. Indirect obstetric deaths, or deaths due to a condition tangential to pregnancy that was aggravated by the pregnancy, caused 23% of global maternal deaths between 2009 and 2020 [4]. The next most common causes of death during this time period were hypertensive disorders (16% of deaths), abortion (8%) and pregnancy-related infection (7%) [4]. Experts predict that, over time, MMRs will decrease and the majority of maternal deaths will be caused by indirect, non-communicable conditions instead of direct complications of pregnancy and childbirth [23]. A country’s position within this ‘obstetric transition’ has important implications for the choice of strategies used to reduce its MMR [23].

### 2.2 Monitoring Maternal Mortality

Reports published by both the World Health Organisation (WHO) and academic researchers highlight how lack of access to accurate, complete data about maternal mortality hinders effective interventions [3, 4, 21]. For example, in 2015 the WHO highlighted the need to improve measurement of maternal mortality in its Strategies toward Ending Maternal Mortality report [4, 21]. Maternal mortality data helps policymakers identify high-risk regions as well as possible region-specific causes of maternal mortality [3, 21]. This allows them to implement timely, targeted, and useful programs to reduce maternal mortality [3, 21].

MMRs are estimated from one or more of a diverse range of data sources, with a large sample size and/or complete records needed for stable MMR estimates given the relative rarity of maternal deaths [22]. Where possible, MMR estimates are informed by civil registration and vital statistics (CRVS) systems, which are national data collection systems that continuously record births and medically certified deaths [3, 22]. Cause of death is recorded in line with the International statistical classification of diseases and related health problems (ICD) [3, 22]. Thus, CRVS systems generate vital information for mortality monitoring and policy development, as in a perfect world they record all deaths in a country with their associated causes [10, 11]. However, in 2017, less than 40% of countries had CRVS systems that enabled continuous and accurate maternal mortality monitoring [10]. Unfortunately, this prevents monitoring of trends in maternal mortality, especially in the lowest income countries that have the highest MMR burdens, as they tend to have the most missing data [3, 4, 22]. For example, in 2017, only 2 of the 49 least developed countries had greater than 50% death registration coverage [10].

Even when CRVS systems are in place, they are limited by their national coverage and can be subject to a myriad of underreporting and misclassification errors, reducing the quality of the reported data [10, 21, 22, 4]. More specifically, underreporting occurs when a maternal death is not registered, while misclassification occurs when the incorrect cause of death is recorded [11]. While maternal mortality is underreported at all stages of pregnancy, it is more frequent at the earliest phases when signs of pregnancy may be missed [21, 22]. Underreporting also increases when the maternal death occurs at home or when it occurs as a result of abortion or extramarital pregnancy due to social stigma or legal barriers [21]. Maternal mortality is also often misclassified due to the complexity of isolating the exact cause of death, especially when the death is caused by an underlying health condition [21, 22]. Due to misclassification and underreporting, studies predict that maternal mortality is underestimated by at least 40%, with large differences between countries [21]. Thus, reliability of CRVS data must be confirmed before use [3].

MMR estimates can also be informed by specialised studies, which determine the MMR within a specific geographic region using police and medical records, national registries, administrative reviews, medical autopsies, and censuses [3]. They are often considered the gold-standard [3].

In addition to CRVS systems and specialised studies, MMR estimates are informed by broader national and household surveys, censuses, national surveillance data, and data collected from health providers [3, 10]. These sources are particularly useful in low and middle-income countries that lack CRVS systems [21]. Unfortunately, surveys may not provide adequate coverage, especially of rural areas that are difficult and/or expensive to reach [21, 3]. Additionally, the relative rarity of maternal mortality means these surveys require a large sample size to be statistically significant, which can make them prohibitively expensive to conduct [3]. Alternatively, maternal deaths can be monitored using surveys based on the sisterhood method, where adult respondents detail how many of their sisters have died from a pregnancy-related cause [3]. This is the WHO recommended method for countries without other reliable sources of data, as asking respondents about the health of others immediately increases sample size [3]. However, the survey does not provide current data for monitoring purposes [3].

As a result of these limitations, maternal mortality data can be sparse and low-quality, motivating use of modelling techniques to fill in the gaps.

### 2.3 Machine Learning

Emerging technologies have allowed large quantities of data to be collected and transported at scale [6]. For the first time, researchers can analyse massive datasets from a wide variety of sources, such as social media and health records like those discussed above [6]. As a result, researchers have the opportunity to identify complex, insightful, data-driven patterns [6, 12]. Increasingly, researchers are analysing these patterns using machine learning, where they train models to detect and learn relationships within the data [6, 5]. This approach differs from the traditional strategy of designing the model using hand-crafted rules informed by prior knowledge of the data’s domain [6, 5]. Machine learning (ML) is particularly useful when applied to datasets with many datapoints and/or variables, as the technique can find hidden patterns that may be missed by humans [5]. ML models can then take these patterns and use them to make predictions in the absence of empirical data [5]. Thus, it could be a useful technique to employ when working with missing epidemiological data [24].

Conventionally, the input dataset to an ML model consists of a number of samples/observations, where each sample is referred to as a ‘datapoint’ [5]. Each datapoint is defined by a certain number of variables, which are referred to as ‘features’. Features with discrete values are called ‘categorical’ and features with continuous numerical values are called ‘continuous’ [5]. For example, if a feature describes ‘risk’ and its values were ‘high’, ‘medium’, or ‘low’, it would be considered categorical. In contrast, if its values were a risk score between 0 and 5, it would be continuous. Generally, each row of the input dataset corresponds to a datapoint, and each feature corresponds to a column. ML models can be broadly classified as supervised or unsupervised methods depending on whether the input rows are associated with an output value [5].

#### 2.31 Unsupervised Learning

Unsupervised learning models act on input datasets whose datapoints are not associated with a specific categorical or continuous output value [5]. For example, datapoints in unsupervised learning may consist of a series of observations about feature variables ‘temperature’, ‘day of the week’, and ‘location’. However, the observations would not be associated with an output variable, like ‘quantity of ice cream sold’. The aim of unsupervised learning is to uncover hidden patterns and learn the data’s structure [5]. By not providing output values, the model is not explicitly guided toward learning a specific type of pattern in the data. A common application of unsupervised learning is dimensionality reduction, which transforms a dataset with many variables into a dataset with fewer variables while retaining as much of the data’s original variation as possible [5]. The transformed dataset may contain linear and non-linear transformations of the original variables. One widely used dimensionality technique is called principal component analysis and is often applied to be able to represent a dataset with many variables using only two variables, making it easier to visualise patterns in the data [5].

#### 2.32 Supervised Learning

In contrast, supervised learning occurs when a model is fit to a labelled dataset, where each input datapoint is associated with one or more output categories or values [5]. The true values of the output variables are referred to as the ‘ground truth’, which the model is trained to predict using patterns in the data [5]. Supervised learning can be applied to classification problems, where the ground truth is two or more specific categories, or regression problems, where the ground truth is a continuous numerical output [5]. This thesis will focus on supervised machine learning for regression analysis, as models can be trained on data labelled with ground truth MMR values, which are continuous.

More formally, a dataset containing *n* samples is denoted as , where refers to a *d*-dimensional input feature vector and refers to the corresponding continuous, numeric output value [7]. When solving a regression problem, the model’s goal is to learn a mapping, *f(x)*, between the input data and the associated ground truth [6]. For a new input datapoint, *x\**, the model can use the mapping *f(x\*)* to correctly predict the associated ground truth *y\** [6]. The type of mapping used defines the ML model being implemented [6]. At its core, this mapping is a mathematical function defined by a series of parameters, where the function takes in the input dataset and generates its prediction of the output [5]. To produce accurate predictions, the mapping must approximate the true, underlying relationships between the feature variables in the data [5].

##### 2.321 Model Development

Model performance depends on whether the model’s parameters are well-suited to the model’s purpose and dataset [5]. Model performance is defined by a loss function, which quantifies the difference between the model’s predictions and the ground truth [5]. The process of optimising the model’s parameters involves minimising this loss function, which commonly involves a technique called gradient descent [7, 25]. Intuitively, gradient descent takes advantage of the observation that the gradient quantifies the direction of greatest increase. Thus, taking the negative gradient of the loss function with respect to each parameter gives the direction that the parameter’s value would need to move to produce the greatest decrease in loss [25]. As a result, to minimise the loss function, the gradient of the loss with respect to a specific parameter, , can be subtracted from the parameter’s current value, as shown in Equation 1 below [26]. The symbol is the learning rate, which determines the degree to which the negative gradient is used to adjust the parameters’ value [26]. This gradient descent algorithm is applied to all model parameters to minimise the model’s loss function through optimising its parameter values. There are many different implementations of gradient descent, such as sample gradient descent, which calculates the gradient using a subset of the dataset to reduce computational complexity [25].

Model development must be done with care, as the model’s parameters are optimised with respect to a specific input dataset [5]. This can produce overfitting, where the model has high performance on the input dataset but low performance on out-of-sample data [5]. Overfitting can occur due to noise in the input dataset, where the model learns the noise as a true pattern in the data [5]. This prevents the model from learning the true, underlying patterns in the data that would allow it to extrapolate to out-of-sample data, which may have a different noise pattern [5]. Generally, more complex models have a higher risk of overfitting, as they have more parameters that can be configured to the exact, noisy patterns in the input dataset [5]. The risk of overfitting must be balanced with the risk of underfitting, which occurs when the model is too simple to accurately capture the underlying relationships in the data [5]. Overfitting and underfitting are related to the bias-variance trade-off, where bias refers to errors in the model’s predictions while variance refers to change in the model’s predictions based on the training data used [5]. The goal of model development is to produce a model with low bias and low variance. However, to reduce bias, the model generally must become more complex, which can cause overfitting and increase variance, necessitating a trade-off [5].

To balance the goal of low bias while avoiding overfitting, the dataset is split into non-overlapping training and testing subsets, generally in a ratio between 72:25 and 90:10 [5]. The model’s parameters are fit to the training dataset through minimising the loss function [5]. Then, the model’s performance is evaluated on the previously unseen test data to determine whether the model is generalisable or is overfit to the training data [5].

However, the model should not be adjusted based on its test performance to prevent overfitting to the test data, which would prevent the test set from being able to measure out-of-sample performance [5]. This is a problem when using the test set to compare the performance of different hyperparameter specifications, where hyperparameters govern the architecture of a model and the training process, but are not themselves fine-tuned during training [5]. They define the structure of the mapping used by the model, not the mapping itself. For example, the learning rate , or the rate at which parameter values are changed during training, is a hyperparameter [5]. To address this problem, the training data can be further split into non-overlapping training, validation subsets [5]. Model parameters are fit using the training data, and different model architectures and hyperparameter specifications are tested on the previously unseen validation set [5]. The ability of the best performing model to generalise to out-of-sample data is then evaluated using the unseen test set [5]. Thus, the model’s performance on the test set is often considered a measure of its real-world performance [5]. As a result, the test set should only be used once [5].

Training data is often split into training, validation subsets through a process called K-fold cross-validation [5]. In this process, the training data is split into K-1 equally sized, non-overlapping subsets. For each of K iterations, the training data consists of K-1 folds while the validation data consists of the single, remaining fold [5]. One version of the model is trained per iteration on the K-1 training folds, with its performance tested on the validation fold. By having K iterations, each individual fold has a turn to be the validation fold, testing the model’s ability to generalise on all parts of the training-validation set. The performance of the K models (one per iteration) is then compared and/or combined [5].

###### 2.3211 Loss Functions and Parameter Tuning

The model’s performance is measured using a loss function, as described above [7]. There are a variety of possible loss functions that can be used for regression problems, with one of the most common being the mean squared error (MSE), or L2 loss [7]. The MSE is the averaged squared difference between the ground truth output, , and the model’s predicted output, , across *n* datapoints [7]. The MSE is defined in Equation 2, below. A limitation of the MSE is its sensitivity to outliers, as squaring the difference between the true and predicted outputs places high importance on large errors [7].

A widely used variation of MSE is mean absolute error (MAE), also referred to as the L1 loss [7]. MAE measures the average absolute difference between the true and predicted outputs, and is defined more formally in Equation 3 below [7]. Taking the absolute difference instead of the squared difference means MAE is less affected by outliers than MSE [7]. However, unlike MSE, MAE is not differentiable everywhere due to the absolute value, presenting difficulties when using gradient based optimisation techniques [7].

Another common variation of the MSE is root mean square error (RMSE), or the square root of the MSE [7]. The RMSE is defined formally in Equation 4, below [7]. Like MSE, the squared function in RMSE heavily penalises outliers [7]. However, unlike MSE but similar to MAE, RMSE is in the scale of the original data, making it easier to interpret [7].

An alternative, widely used metric is the mean absolute percentage error (MAPE), which calculates the average prediction error as a percentage of the ground truth value [7]. It is defined in Equation 5, below [7]. MAPE is criticised for being asymmetrical, as always dividing by the true output, , can produce different errors depending on whether the predicted value underestimates or overestimates the true value [7]. For example, predicting a value of 50 if the true value is 100 gives a MAPE of 50% while predicting a value of 100 if the true value is 50 gives a MAPE of 100%. Thus, the same absolute error produces different MAPE scores depending on whether the under- or over-estimate is used as the denominator. Another limitation of using the MAPE is that it can become very large or undefined if is close to zero [7]. Additionally, MAPE is strongly affected by if is small, as large errors divided by a small number produce large relative error [7].

The coefficient of determination, also called the R2 score, is another commonly used performance metric [7]. It determines the proportion of variation in the output variable explained by the model [7]. R2 is defined in Equation 6, below, where is the mean true value [7]. R2 is equal to 1 if the model explains all variation in the output [7]. However, a high R2 score can sometimes reflect overfitting in the model. R2 is negative if the model performs more poorly than if it simply predicted [7]. Unfortuantely, the R2 is known to be sensitive to bias and can arbitrarily increase with the number of features [7].

#### 2.322 Linear Regression

One of the most well-known, basic machine learning models is linear regression, which is often described as a ‘line of best fit’ through the data [8]. Model development focuses on minimising the distance between the true values and the line produced by the model’s predictions [5]. More formally, linear regression is often used to predict output using a linear combination of *d*-dimensional input feature vectors, [8]. The model is described in Equation 7, below, with the *d*-dimensional weights denoted by [8].

One of the symptoms of overfitting is large parameter weights on feature dimensions, as this signals that the model has found a complex pattern in the dataset, which is more likely to be noise and thus less generalisable [8]. As a result, many linear regression implementations incorporate a regularisation term, which is added to the loss function to penalise model complexity. More specifically, the regulariser increases the loss by some function of the model’s parameters [5]. To minimise loss, training generally involves actions to reduce the regularisation term and thus prevent the parameter values from becoming too large [5]. The L1 norm, or the sum of the parameters’ absolute values, is a commonly used regularisation function [8]. By penalising parameters’ absolute values, it encourages the model to use zero feature weights, thus performing automatic feature selection [8]. Another widely used regularisation function is the L2 norm, which is the sum of the squared parameter values, and thus severely penalises large parameter values [8]. Elastic Net is a special version of the linear regression model that combines the L1 and L2 norms [8]. Elastic Net model’s regularisation term is defined in Equation 9, where is a hyperparameter that controls the influence of the L1 versus L2 norm [8]:

Linear regression is solely linear in the parameters, meaning the feature variables do not need to be linear [5]. However, the model can still underfit if the relationship between feature variables is non-linear [5]. Thus, more complex models have been developed.

#### 2.323 Support Vector Machines

Support vector regression fits a model based on the most informative data points [9]. More specifically, only predictions that were incorrect by at least epsilon contribute to the model’s loss during training [9]. Epsilon is a hyperparameter that defines the model’s error tolerance [9]. Data points associated with a predictive error of at least epsilon are referred to as “support vectors” [9]. This procedure allows the model to focus on correcting larger errors. Model predictions are generated from a linear combination of support vectors to be able to capture the most complex relationships in the data [9]. Often, input data is transformed into a higher dimensionality feature space to more effectively model non-linear relationships [9].

#### 3.324 Decision Tree Based Methods

Since their original proposal in the 1960s, decision trees have become an important part of the most widely used ML models [12]. Intuitively, decision tree models function like flowcharts [12]. A regression decision tree is visualised below, with the tree’s internal nodes given by circles and its terminal nodes given by squares. When predicting the output value for a specific datapoint, the model starts at the root node and applies a logical test to the values of one or more feature dimensions. For regression, this test is usually in the form *feature <= value*, and defines a split [12]. Based on the test’s Boolean result, the model moves to the right or left child node. This process repeats until the model reaches a terminal leaf node, which is a node with no children. The terminal node’s value determines the model’s prediction [12]. An alternative way to conceptualise decision trees is as a specific partitioning of the input space, where each node partitions the feature space and each new partition is passed down to the node’s children. The tree’s prediction then corresponds to a specific area of the feature input space [12].

age <= 20

age <= 65

age <= 15

False

True

0

5

0

37

**Figure 1:** Regression decision tree visualisation, where splits are defined in terms of the feature ‘age’ and the model is trying to predict the number of hours worked per week. If the result of the test is True, the model moves to the left child node, but if it is False, it moves to the right child node. The values inside the terminal nodes are the predictions.

The decision tree’s structure is developed during training, where the logical tests that best predict the outcome variable are chosen [12]. For regression problems, the logical test at each node is determined through finding the split that minimises the mean squared error in the associated child nodes [19]. Traditionally, this is done in greedily, where specific splits are evaluated solely by their effect on their children’s error [12].

One of the primary advantages of decision trees is their ability to work with data that has missing values, with specific implementations having different methods [12, 19]. CART (Classification and Regression Trees) is one of the classic decision tree implementations and uses ‘surrogate’ splits to deal with missing data [12, 19]. When a datapoint is missing a value in a specific feature dimension, nodes that partition the input space using that feature instead split using an alternate, related variable [19].

Another advantage of decision trees is their interpretability, which is due to their flow-chart-like structure [12]. As a result, they are valued in disciplines that place more emphasis on understanding why ML models have made a specific prediction, such as in drug development [12]. A further benefit of decision trees is their relatively low computation cost when compared to other ML models [12].

However, a major limitation of decision trees is their propensity to overfit, where the input space is partitioned by overly complex rules based on the specific training examples and noise in the training data [12]. As a result, shallower trees with fewer partitions tend to generalise better, but may have lower performance due to their lower complexity [12].

There are many variations of decision trees. For example, to better represent more complex functions, studies have explored basing the splits in internal nodes on multiple feature variables and/or having predictive models in the terminal nodes instead of a constant [12]. Additionally, research has explored replacing the greedy approach used to determine splits with look-ahead algorithms to avoid suboptimality [12].

#### 3.325 Ensemble Based Methods

Studies have found that combining predictions from multiple models can have better predictive performance than solely using predictions from a single model [13]. This is called ‘ensemble learning’ [13]. Ensemble methods can reduce generalisation error when the models being combined, called base estimators or weak learners, are independent and diverse [14]. This allows them to cover a wider range of possible outcomes [14]. Additionally, a single model may become stuck in a local optimum, but if each base estimator in an ensemble model starts in a different place and/or has a different formulation or training trajectory, it is unlikely that all base estimators will become stuck in the same local optimum [13]. Ensemble methods also perform well when complex relationships in the data can be approximated better by a combination of base estimators than by a single base estimator [13].

Ensemble models can generally be categorised as bagging, boosting, voting, or stacking algorithms [14].

##### 3.3251 Bagging

During bootstrap aggregation, or ‘bagging’, predictions from multiple versions of the same type of base estimator are combined [15]. Different versions of the same base estimator are produced by training each estimator on a bootstrap replicate of the training set [15]. In other words, datapoints are drawn at random and with replacement from the training set to form independent, bootstrapped datasets of the same size [15]. Then, each base estimator is trained on one of the bootstrap replicates of the dataset, producing an ensemble of base learners whose predictions are combined [15]. For regression tasks, the predictions are generally averaged [15]. Bagging works particularly well when models trained on different versions of the training set are substantially different, allowing the ensemble model to cover a wider variety of outcomes [15]. Additionally, bagging can reduce variability and overfitting by cancelling out noise in the dataset [14]. An example of bagging is combining the predictions of multiple decision trees made on separate, bootstrapped versions of the training dataset [14].

###### 3.32511 Random Forest

The Random Forest model is a widely used variation of the basic decision tree-based bagging ensemble [13]. In the Random Forest algorithm, each split in the base decision trees is created using a random subset of features [13]. This modification further reduces overfitting by forcing the model to learn patterns in the data based on different combinations of features [13]. However, there is no guarantee that an important feature will be used for splitting, potentially causing important information to be lost [14].

##### 3.3252 Boosting

While bagging trains base estimators independently, boosting ensemble methods train base estimators sequentially [14]. During boosting, each base estimator in the sequence tries to correct the errors of the previous estimator, giving the ensemble model higher prediction accuracy and lower bias [14, 16]. The base estimators in boosting ensembles are often decision trees due to their empirically demonstrated prediction accuracy [16].

There are many implementations of boosting algorithms, with gradient boosting being one of the most popular [16]. In gradient boosting, the first base estimator predicts the output variable. Then, each new base estimator is trained to minimise the current model’s loss [18]. To do so, the new base estimator predicts the negative gradient of the previous estimator’s loss function [18]. This negative gradient indicates the direction of greatest decrease in loss. By learning this direction, the new base estimator can move the ensemble’s prediction in a direction that most reduces its prediction loss [18]. This is expressed more formally in Equation 10, below, where is the *m*th base estimator in the sequence, is the base estimator trained on the negative gradient of , and is the weight attached to , quantifying its importance [16]. The final prediction from a boosting ensemble is the sum of predictions from its base learners [17].

When the base learners are decision trees, gradient boosting is referred to as the gradient boosting decision tree algorithm (GBDT) [16]. Studies have shown that GBDT is accurate, efficient, and interpretable, precipitating its use in a wide variety of disciplines [18]. Two of the most common GBDT methods are Extreme Gradient Boosting (XGBoost) and Light Gradient Boosting (LightGBM).

###### 3.32521 Extreme Gradient Boosting (XGBoost)

The XGBoost algorithm is a high-performance, scalable GBDT method [16, 17].

In the base gradient boosting method described above, the *m*th base estimator, , predicts the negative first-order gradient of the (*m-1*)th base estimator to move the ensemble’s predictions in the direction that most reduces loss [16]. The XGBoost model takes this a step further. It constructs a 2nd-order Taylor approximation of the current model’s loss function using the loss function’s first and second-order derivatives [17]. This Taylor approximation is minimised to find the optimal leaf node weights, which are the tree’s predictions [17]. This method provides a more controlled error correction mechanism, as the second-order derivative indicates how quickly the gradient is changing, guiding how much change should be made in response to the gradient [17].

When building a new base estimator, the algorithm must decide which feature value to use as the ‘value’ part of the *feature <= value* logical tests on the internal nodes [17]. When evaluating a candidate logical test, the algorithm separates the input data into two groups – the data that would push the model to the left child and the data that would push it to the right child [17]. The model then calculates the approximated loss function for each group using the current predictions from the previous base estimators [17]. By minimising this loss, it determines the optimal node weight for the children nodes and can determine the potential reduction in loss produced by this specific split [17].

A major advantage of XGBoost is its ability to work with missing data [17]. As described above, the logical tests at a base estimator’s internal nodes are determined during training [17]. This process is completed with non-missing data only [17]. Then, the model determines the ‘default direction’ for each internal node. This is the direction taken when the feature dimension used in the node’s logical test has a missing value [17]. The default direction is set to left or right, depending on whether moving to the left or right child node produced lower predictive error during training [17].

Unlike the base GBDT algorithm, the loss function used in the XGBoost model has an additional regularisation term [17]. This regulariser is a function of the number of leaves in the base estimator’s decision tree and the squared absolute values of the leaf node scores [17]. Adding this regulariser to the loss function increases loss when the number of leaf nodes increase [17]. Consequently, the regularisation term penalises model complexity, as it encourages the model to have fewer internal nodes and input space partitions to reduce the number of terminal nodes [17]. XGBoost also supports feature subsampling, like the Random Forest model, to further reduce overfitting [17].

One of the key hurdles to constructing GBDT ensembles is the need to trial all possible feature values in the ‘value’ part of the *feature <= value* logical test when determining the optimal structure for the base estimator [17]. In the exact greedy algorithm approach, all possible splits for all features must be tested, with the split’s performance quantified by how much it reduces loss [17]. While this has strong performance, it is computationally demanding, especially when evaluating all possible splits for continuous feature variables and when the input data does not fit into memory [17]. The approximate algorithm was introduced to address this problem [17]. This algorithm splits the continuous features’ distributions into percentiles, with the differences between percentiles using as candidate split points, thus reducing the number of possible splits needed to be evaluated [17].

###### 3.32522 Light Gradient Boosting Machine (LightGBM)

LightGBM is another commonly used GBDT algorithm that also places a strong emphasis on maximising computational efficiency [18]. One of the main modifications proposed by the LightGBM algorithm is gradient-based one-side sampling (GOSS) [18]. GOSS reduces the number of samples used to determine internal node splits. Instead of using all data points to determine each split, GOSS uses the most informative data points and samples a subset of less informative points to maintain the same general data distribution [18]. Its choice of samples is derived from the observation that datapoints associated with small gradients offer smaller potential reduction in error and are thus less useful for increasing model performance [18]. Using this observation, GOSS takes all datapoints with gradient greater than a certain threshold and samples randomly among the remaining datapoints with smaller gradients [18]. It uses this subsampled dataset to determine the internal node split, increasing computational efficiency [18].

Another innovation used in the LightGBM model is exclusive feature bundling [18]. This approach can be applied to sparse feature spaces, which generally have mutually exclusive feature variables, or groups of features where no more than one feature takes a non-zero value at the same time [18]. Groups of mutually exclusive features can be ‘bundled’ together into a single feature, further increasing computational efficiency [18].

##### 3.3253 Voting

The voting ensemble model is another method of aggregating predictions from multiple base estimators [14]. In contrast to bagging, base estimators in the voting ensemble model can have different model architectures, and all models are trained on the same dataset [14]. In regression, the final prediction from a voting ensemble model is the unweighted or weighted average of the base estimators’ predictions [14]. Using a weighted average allows more importance to be placed on specific base estimators [14]. The voting strategy benefits from combining the strengths of each model class in the ensemble but can show lower performance if the base estimators are too similar [14].

##### 3.3254 Stacking

In a stacking ensemble model, predictions from base estimators serve as inputs to a meta-learning model, which combines the inputs to produce a single, final output [13]. In other words, the predictions from each base estimator serve as the input dataset for the meta-estimator, which learns patterns within these predictions to output a final, low-error prediction [13]. The meta-estimator can learn which base estimators are the most important and how to most effectively combine predictions from base estimators [14]. The meta-learner can have a different structure from the base estimators, with examples of meta-learners being Random Forests, support vector machines, linear regressors, and neural networks [14]. While stacking ensembles can improve performance in similar ways to those discussed above, they can be computationally expensive to train, as all the base estimators and meta-learning model must be fit to the data [14].

Citations:

1. Independent Group of Scientists appointed by the Secretary-General, Global Sustainable Development Report 2023: Times of crisis, times of change: Science for accelerating transformations to sustainable development, (United Nations, New York, 2023).
   1. <https://sdgs.un.org/sites/default/files/2023-09/FINAL%20GSDR%202023-Digital%20-110923_1.pdf>
2. <https://icdcdn.who.int/icd11referenceguide/en/html/index.html#standards-and-reporting-requirements-related-for-maternal-mortality>
3. Trends in maternal mortality estimates 2000 to 2023: estimates by WHO, UNICEF, UNFPA, World Bank Group and UNDESA/Population Division. Geneva: World Health Organization; 2025. Licence: CC BY-NC-SA 3.0 IGO.
4. <https://www.thelancet.com/journals/langlo/article/PIIS2214-109X(24)00560-6/fulltext>
   1. Global and regional causes of mmr
5. <https://www.nature.com/articles/s41580-021-00407-0?fromPaywallRec=false>
   1. Information about ML basics
6. <https://www.science.org/doi/pdf/10.1126/science.aaa8415>
   1. Intro to ML
7. <https://link.springer.com/article/10.1007/s10462-025-11198-7>
   1. Loss functions and performance metrics
8. <https://academic.oup.com/jrsssb/article/67/2/301/7109482>
   1. Elastic net
9. <file:///Users/rosalitarosenberg/Downloads/B_STCO.0000035301.49549.88.pdf>
   1. A tutorial on support vector regression
   2. Smola and Sch¨olkopf
10. <https://academic.oup.com/bmb/article/121/1/121/2926155>
    1. Recording MMR methods
11. <https://onlinelibrary.wiley.com/doi/epdf/10.1002/sim.9335?getft_integrator=tfo&src=getftr&utm_source=tfo>
    1. BMIS methods
12. <https://link.springer.com/article/10.1007/s10462-022-10275-5>
    * Decision tree review
13. <https://www.sciencedirect.com/science/article/pii/S095219762200269X?pes=vor&utm_source=scopus&getft_integrator=scopus>
    1. Ensemble review
14. <https://pmc.ncbi.nlm.nih.gov/articles/PMC10298658/#abstract1>
    1. Ensemble review medicine context
    2. Mahajan, P.; Uddin, S.; Hajati, F.; Moni, M.A. Ensemble Learning for Disease Prediction: A Review. Healthcare 2023, 11, 1808. https://doi.org/10.3390/ healthcare11121808
15. <https://link.springer.com/article/10.1007/BF00058655>
    1. Bagging predictors
16. <https://journalofbigdata.springeropen.com/articles/10.1186/s40537-025-01071-3>
    1. Gradient boosting trees
17. <https://arxiv.org/pdf/1603.02754>
    1. Xgboost paper
18. <https://proceedings.neurips.cc/paper_files/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf>
    1. Lightgbm
19. <https://onlinelibrary.wiley.com/doi/epdf/10.1111/insr.12016>
    1. Decision tree (missing, gain metrics)
20. <https://dl.acm.org/doi/pdf/10.1145/3292500.3330701>
    1. Optuna
    2. Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, Masanori Koyama. 2019. Optuna: A Next-generation Hyperparameter Optimization Framework. In The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD ’19), August 4–8, 2019, Anchorage, AK, USA. ACM, New York, NY, USA, 9 pages. <https://doi.org/10.1145/3292500.3330701>
21. <https://bmcpregnancychildbirth.biomedcentral.com/articles/10.1186/s12884-023-06077-4>
    1. Maternal mortality recording errors
22. <https://www.tandfonline.com/doi/epdf/10.1080/2330443X.2023.2286313?needAccess=true>
    1. About the latest WHO model
23. <https://obgyn.onlinelibrary.wiley.com/doi/epdf/10.1111/1471-0528.12735>
    1. Obstetrics – maternal and perinatal morbidity
24. <https://pmc.ncbi.nlm.nih.gov/articles/PMC11010755/#section14-21501319241245847>
    1. Review of AI in healthcare
25. <https://journalofbigdata.springeropen.com/articles/10.1186/s40537-023-00839-9#:~:text=The%20gradient%20descent%20(GD)%20is,computationally%20expensive%20for%20large%20datasets>.
    1. Gradient descent overview
26. <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=363438>
    1. Gradient descent formula