# 4. Materials and Methods

This thesis used a variety of socio-economic and health-related features to predict the MMR for each (country, year) sample. Separate models were trained to perform country-level prediction and forecasting. Development of the final, highest performing model involved 4 major steps.

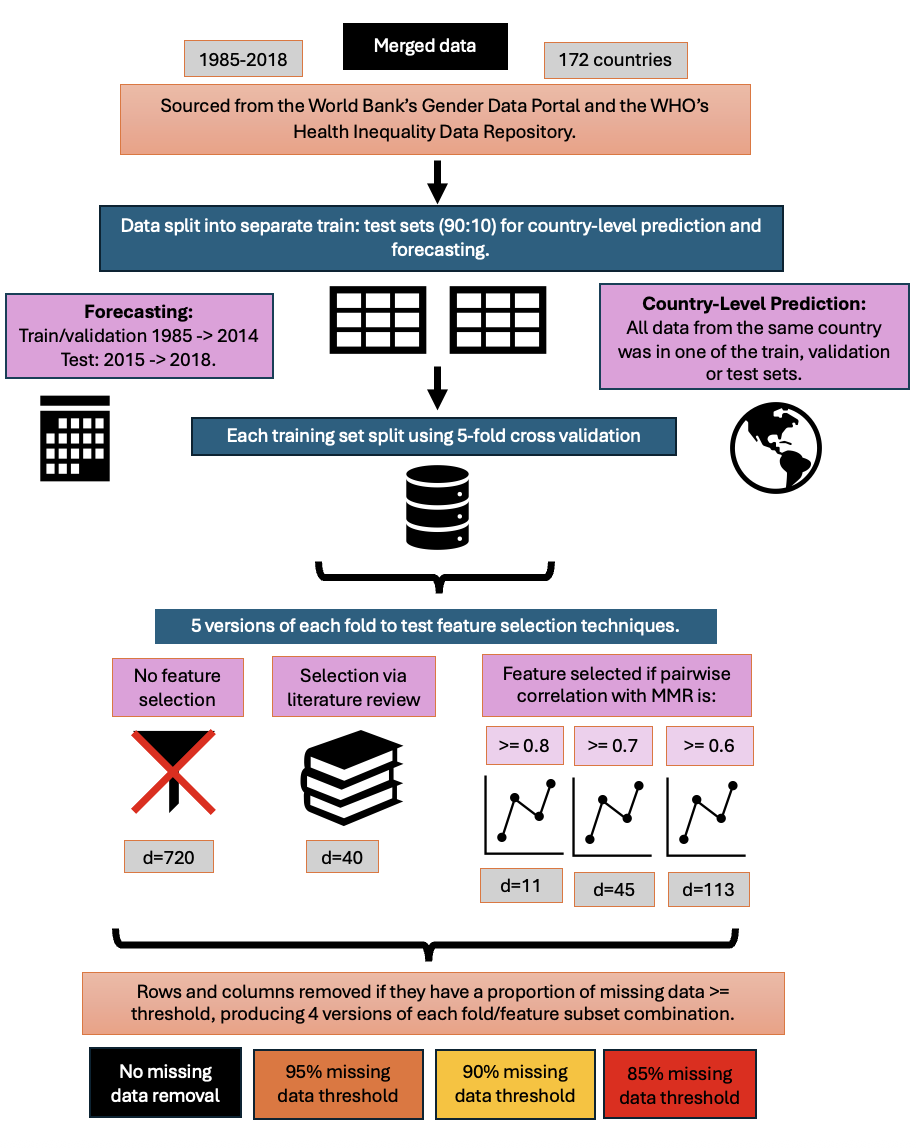
1. Applying cross-fold validation, feature selection, and missing data removal to generate different train datasets.
2. Training Random Forest, XGBoost, and LightGBM regressors on different versions of the training data to explore various pre-processing techniques.
3. Training stacking and voting ensemble models on different combinations of base estimators to reduce predictive error.
4. Evaluate best-performing model by analysing feature importance and sensitivity to input data as well as comparing its predictions to MMR estimates in the literature.

All code was written using Python3 and run in Visual Code Studio or on the Gadi supercomputer at the National Computational Infrastructure. Where appropriate, the random seed was set to 42 for reproducibility. All data and code were uploaded to a public GitHub repository (<https://github.com/R0sle/health_economics_honours>). Unfortunately, some of the datasets were too large to be uploaded. All data is available upon request.

In the following chapter, I present information about the datasets used to develop my models, as well as the data processing and exploratory analysis pipeline (section 4.0-4.3) and computational workflow (section 4.4-4.8).

## 4.0 Data Pre-Processing and Exploratory Analysis

Figure 2 gives a high-level overview of the data cleaning and exploratory analysis applied to the raw data, as well as the pre-processing steps used to generate different versions of the training data. I first merged various World Health Organisation and World Bank datasets into a single, raw dataset (Figure 2a). This raw data was cleaned and split into train/test sets (Figure 2b). Separate train/test sets were used to train models for country-level prediction and forecasting. Each train set was further split into 5 cross-validation folds (Figure 2c). I applied five different feature selection techniques to each training fold, creating five versions of each fold (Figure 2d). Finally, I removed rows and columns from each training fold/feature subset combination that had more than a threshold proportion of missing data (Figure 2e). I tested four missing data thresholds, creating four versions of each training fold/feature subset. This process generated 100 versions of the training data for both country-level prediction and forecasting. These steps will now be explained in more details in the following subsections.



**a)** data merging and cleaning

**e)** missing data removal

**d)** feature selection

**c)** cross-validation

**b)** train/test split

**Figure 2**: High-level overview of the process of generating train, validation and test datasets. Data from the WHO and World Bank was merged into a single dataset (a) before being split into train/test sets (b). Separate splits were conducted for country-level prediction and forecasting. Each training set was further divided into 5 cross-validation folds (c). Five versions of each fold were produced with different feature selection methods (d). Finally, four versions of each feature subset dataset were produced by removing missing data with different levels of flexibility (e). This process generated 100 versions of the training data.

## 4.1 Data Sources and Merging

In this subsection, I describe the data merging and cleaning steps (Figure 2a).

### 4.11 Data Sources

National MMR data for 242 regions, countries, territories, and areas between 1985 and 2018 was sourced from the World Bank Group’s Gender Data Portal [30]. The data was derived from information provided by countries’ national data collection systems, such as from national surveys, hospital records, and civil registration and vital statistics systems [30]. **These MMR values served as the ‘ground truth’** for training my model.They were different from the outputs of the BMat, CODEm, and GMatH models discussed in the literature review**.**

Feature data was retrieved from a variety of World Health Organisation (WHO) and World Bank Group repositories. More specifically, 5 datasets were downloaded from these repositories, with each component dataset itself a compilation of variables, sometimes provided by a range of sources. Information about each component dataset was summarised in Table 1. See the GitHub repository for the specific variables gathered from each data source.

Briefly, the feature dataset sourced from the World Bank Group’s Gender Data Portal describes various health and socio-economic outcomes, which were each estimated by UN/WHO divisions or other partner organisations [31]. For example, one of the indicators included in this dataset was the ‘Probability of Survival to Age 5’, which was calculated by the UNESCO Institute for Statistics. Similarly, data describing health determinants related to the environment, employment, education and social protection was compiled in The World Bank Catalogue by various agencies, with many of the monitored indicators used to track progress toward the UN’s Sustainable Development Goals [33].

The dataset describing illness incidence and prevalence was compiled by the Institute of Health Metrics and Evaluation, which publishes the Global Burden of Disease Study [34]. Thus, illness incidence and prevalence would be determined using disease-specific versions of the CODEm framework.

The WHO Collaborating Centre for Health Equity monitoring re-analysed data from Demographic and Health Surveys to provide information about women’s empowerment [32]. More specifically, this re-analysis measured women’s social independence, such as women’s ability to complete schooling and achieve their goals, women’s ability to make household decisions, and women’s attitudes to violence [32].

This use of data from a variety of data sources was motivated by Onambele et al. (2023), who recommended combining multiple sources to take advantage of the different datasets offered by the WHO.

Each country was categorised as low, lower-middle, upper-middle, or high-income by the World Bank. These categories were converted into numbers using ordinal encoding, where low-income was denoted as ‘1’ and high-income as ‘4’, to preserve their implicit order. No other features were modified from their raw format using methods like one-hot encoding, ordinal encoding, or log transformation.

**Table 1:** Summary information about the datasets used in this study.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Type of Dataset** | **Number of Features** | **Date Range** | **Number of Areas Covered** | **Demographic Subsets Used** | **Source** |
| National MMR estimates | 1 | 1985-2018 | 242 | NA | World Bank Group’s Gender Data Portal [30] |
| Health outcomes & literacy, agency | 198 | 1960-2023 | 265 | NA | World Bank Group’s Gender Data Portal [31] |
| Illness incidence and prevalence | 193 | 2000-2019 | 194 | Sex | WHO Health Inequality Data Repository & IHME [34] |
| Empowerment | 9 | 1991-2023 | 120 | Economic status (quintiles 1, 5) | WHO Health Inequality Data Repository after re-analysis by the WHO Center for Health Equity Monitoring [32] |
| Socioeconomic, education, environmental variables | 64 | 1970-2023 | 195 | Sex, economic status (quintiles 1 and 5), residence (urban, rural) | WHO Health Inequality Data Repository, sourced from the World Bank Data Catalogue [33] |
| World Bank’s Categorisation of a Country’s Income level | 1 | 2024 |  | NA | WHO Health Inequality Data Repository [35] |

Some of the datasets contained disaggregated data. For example, features were sex or economic status specific. However, the ground truth MMR values were not disaggregated. Including the disaggregation as its own feature column would therefore produce a missing value in the associated MMR estimate column when merging the datasets. To prevent this, I created subgroup specific versions of the feature. See Table 2 for an illustrative example. If the data was disaggregated on a scale (e.g. Feature A was reported for wealth quintiles 1 through 5), I only used values from the most extreme subgroups (e.g. quintiles 1 and 5) to prevent the number of features, and thus the dimensionality of the dataset, from becoming too large.

**Table 2**: Illustrative example of subgroup specific versions of a single feature, ‘Feature 1’, with the bolded text defining the demographic being represented.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Country | Date | Feature 1 **Female** | Feature 1 **Male** | Feature 1 **Rural** | Feature 1 **Urban** | Feature 1 **Wealth Quintile 1** | Feature 1 **Wealth Quintile 5** |
|  |  |  |  |  |  |  |  |

As a note, my proposed model uses different input data than the BMat, CODEm, and GMatH models discussed in the literature review. This was due to variation across the models’ covariates/features/parameters and methodologies. While the BMat and CODEm models use some pre-compiled datasets, they mainly rely on processed mortality data collected from a mixture of sources, including surveys and official reporting mechanisms [27, 28]. GMatH relies both on mortality databases, Demographic and Health Surveys, the WHO database, and the medical literature [29].

### 4.12 Merging Data

All datasets used in this report contained columns specifying the country and its associated ISO3 country code, as described in the ISO 3166 international standard [36]. However, different datasets sometimes used a different version of the same country’s name (e.g. United States versus United States of America). Therefore, datasets were merged using the unique ISO3 code and year.

The national MMR estimates were collected between 1985 and 2018. Therefore, all data collected before 1985 and after 2018 was excluded. Features with no data between 1985 and 2018 were also excluded.

### 4.13 Data Cleaning

All (country, year) samples that were missing an associated MMR estimate were removed from the dataset. This avoided needing to impute the ground truth variable, which may have caused the models to be trained on incorrect feature/MMR estimate pairings, introducing inaccuracy.

Additionally, I removed the following feature variables:

* 'Number of maternal deaths’
* ‘Lifetime risk of maternal death (1 in: rate varies by country)’
* ‘Lifetime risk of maternal death (%)’

The ‘number of maternal deaths’ is the numerator of the MMR. Similarly, the two features measuring the ‘lifetime risk of maternal death’, as a rate or percentage, are calculated using the MMR [UN MMEIG report]. The Pearson’s correlation coefficient between the MMR estimate and the ‘lifetime risk of maternal death (%) was 0.93, showing their strong positive correlation. Therefore, these three variables were excluded from the feature dataset to prevent the model from using them to predict the MMR instead of learning the relationship between MMR and socio-economic and health-related data.

The final data cleaning step involved removing the ‘country’ and ‘year’ as feature variables, instead using them as unique sample identifiers. This step was performed because ‘country’ and ‘year’ could introduce bias in the data, where the model learns the typical MMR for a country or year instead of learning to use the relationships between MMR and the features.

## 4.2 Exploratory Data Analysis

An initial exploratory data analysis was conducted to gain a better understanding of the dataset and motivate choice of pre-processing techniques.

### 4.21 Trends in Missing Input Data

No data imputation was used despite the high level of missing data. This was done to prevent imputation from introducing bias into the dataset, especially given the high level of missing data. More specifically, the pattern of missing data in this dataset would typically be categorised as ‘missing not at random’ (MNAR). In other words, the probability of data being missing relies on both observed and missing data, or on other, unobserved variables [2, 3]. This dataset would be considered MNAR because the probability of missing data is heavily related to the robustness of the country’s data collection systems, which is an unseen variable, but which may be related to a country’s MMR. Additionally, some years have a much lower proportion of missing data than others (20 to 35% missing data versus 80 to 90%). This indicates structural differences in data collection due to periodic data reporting, which is an unobserved variable.

Given that data MNAR is dependent on unseen data, it is extremely difficult to remove or impute the missing data without ignoring the important unseen variables and introducing bias [2]. For example, removing all rows and columns with missing data or imputing the missing values based on the observed datapoints would bias the data toward countries reporting greater amounts of data. As a result, I only used ML models that could work with missing data.

### 4.22 Key Statistics

To give deeper insight into the feature dataset, I presented key summary statistics about some of the features that the literature describes as having a particularly meaningful relationship with MMR (Section 5.22). The proportion of missing data was calculated after data cleaning, explaining why the MMR estimates have a missing data proportion of 0%. Both mean and median were included to give an indication of outlier occurrence.

### 4.23 Principal Component Analysis

Principal component analysis (PCA) was employed for dimensionality reduction purposes. Rather than trying to visualise patterns by plotting all 721 feature dimensions, PCA was used to project samples into the 10 principal components that captured the maximum amount of total variance across the dataset.

PCA cannot be performed on a sparse dataset. Thus, Scikit Learn’s k-Nearest Neighbours imputation method was used to impute missing data before applying PCA [4]. This method imputed missing values using datapoints that were most similar to the sparse datapoint in their non-missing dimensions [4]. The data was standardised to a zero mean and unit standard deviation before applying the Scikit Learn PCA method [5]. This prevented variance calculations from being skewed by features with high magnitudes [5]. Note that imputation was not used at any other point in this research.

The input feature data was plotted across its two most important principal component axes, with samples being coloured according to their income level, MMR estimate, and year to better identify patterns and clusters in the data.

### 4.24 Correlation Analysis

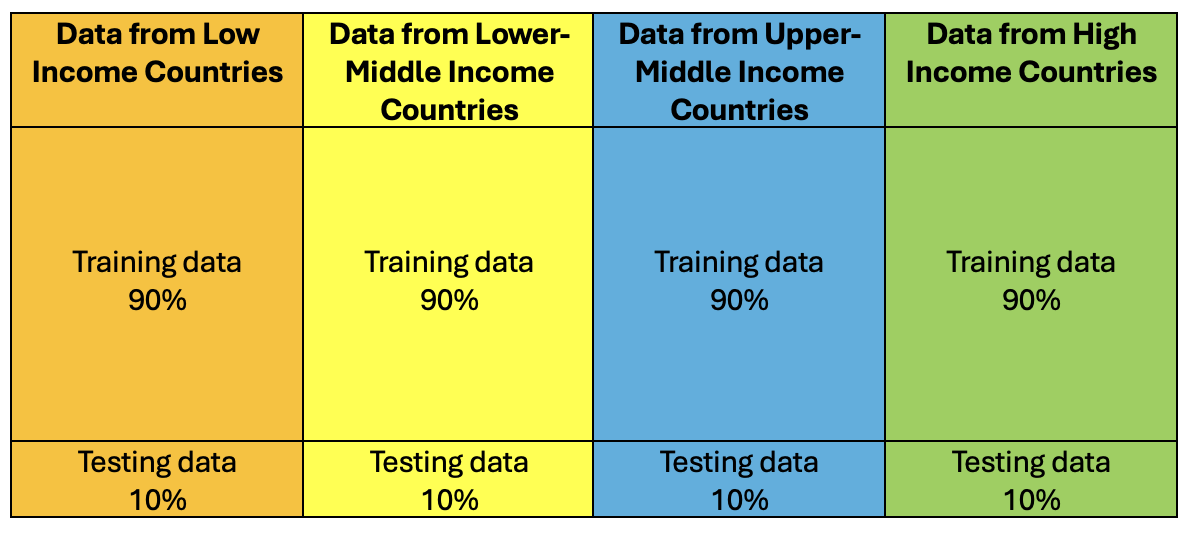
I calculated Pearson’s pairwise correlation coefficients between each feature and the maternal mortality ratio. Analysis of these pairwise correlations were used to inform feature selection strategy.

## 4.3 Processing for Machine Learning Pipeline

Figure 2 visualises the flowchart overview of the data pre-processing process. Different versions of the train dataset were produced to explore the effect of various pre-processing techniques. The distribution of data across the train and test sets was presented in Section 5.3 to give insight into model performance and generalisation.

### 4.31 Splitting Input Data into Train/Test Sets

The cleaned dataset was split into train/test subsets in two different ways (Figure 2b). Separate train/test datasets were used for country-level prediction (Figure 4a) and forecasting (Figure 4b).



a)

A graph showing a number of data

AI-generated content may be incorrect.

b)

**Figure 4:** Train/test split visualisation. a) When split for country-level prediction, 90% of the data from each income level was placed into the train set and 10% into the test set, with all data from the same country in only one set. b) When split for forecasting, all data between 1985 and 2014 was placed in the training set (88% of available data) and all data between 2015 and 2018 was put in the test set (12%).

#### 4.311 Country-Level Prediction (CLP):

I split the input dataset so all data from a specific country was in either the train or test set. Splitting the input data by country prevented data leakage by preserving independence between the train and test sets. Models trained in this way could estimate missing MMR values and thus inform policy makers about their country’s maternal health status.

Each country in the input dataset was classified as high-income, upper-middle income, lower-middle income, or low-income by the World Bank. I split the original dataset into 4 subsets, each containing all countries from a specific income level. Each of these subsets were split into train/test datasets in a 90:10 ratio. The 4 train sets were merged into one complete train set, and the 4 test sets were merged into a complete test set (Figure 4a). This ensured that samples from each income level were in both the train and test sets to allow the test set to effectively evaluate whether the model could generalise to different income levels. However, ensuring all data from the same country was in either the train or test set meant the train/test split was not exactly 90:10. Instead, the true ratio was within one or two percent of 90:10, as the number of rows per country did not allow an exact 90:10 split.

#### 4.312 Forecasting

MMR forecasts can give policy-makers information about future MMR trends as well as allow researchers to predict the effects of candidate policies. I simulated this scenario by placing all data from 1985-2014 in the train set and all data from 2015-2018 in the test set (Figure 4b). The goal of this strategy was for the model to learn patterns in the historical data to use to predict the future.

This division produced an 88:12 train/test split. I included data from 2015 in the test set to ensure the test set contained sufficient non-missing data to be useful, as 2015 contained more than 50% less missing data than average.

### 4.32 Cross-Validation

Each of the CLP and forecasting train datasets were further divided into 5 cross-validation folds, each of which being a specific permutation of the train/validation 80:20 split (Figure 2c).

If the data was being split for CLP, all data for the same country was placed in either the train or validation set to prevent data leakage and evaluate the model’s ability to predict sparse country data. If the data was split to perform forecasting, all data from the same year was placed in either the train or validation set to prevent data leakage and assess the model’s ability to predict for an unknown year. These conditions resulted in slight deviations from the 80:20 ratio, but only within a percentage point.

The train data was split into cross-validation folds using Scikit Learn’s GroupKFold method, which ensures that entries of the same group only appear in one validation set [15]. Members of the groups were countries for CLP and years for forecasting. This method ensured that the train and validation sets varied across the different folds, allowing me to assess how changes in the composition of the training dataset affected model performance.

### 4.33 Feature Selection

As discussed in the literature review, decision-tree based models can work with high-dimensional data. I tested whether this ability meant they achieved the greatest performance when using high-dimensional data, or if they achieved greater performance when working with a subset of features. This difference may be due to a higher number of features introducing noise, sparsity, and additional computational complexity, as discussed in Section 3.2.

I created 5 versions of each fold, each with a different selection of features (see Table 3 and Figure 2d for a summary and the GitHub repository for a spreadsheet giving the specific features used in each subset). In the first case, no features were removed to evaluate model performance on the full feature dataset, which contained 720 features. To create the second feature subset, I surveyed a number of papers about maternal mortality to learn which features researchers believed most strongly influence MMR [7, 8, 9, 10, 11, 12, 13, 14]. As a result, I hand-picked 40 biological and socio-economic feature variables. While there were many other relevant features I could have chosen from the available dataset, I believed these covered the major MMR determinants.

The final three feature subsets were produced using the correlation between feature variables and MMR. More, specifically, I computed the pairwise Pearson’s correlation coefficient of all feature columns with the MMR estimate. I used the Pandas correlation method, which ignores rows where either feature pair has a missing value. I created a dataset containing features whose absolute pairwise correlation coefficient with MMR was at least 0.8. I produced two additional datasets containing features whose absolute correlation coefficient with MMR was at least 0.7 and 0.6, respectively. This allowed me to test the strength of correlation needed for the features to improve model performance.

**Table 3:** The 5 feature selection methods used to create 5 versions of each train fold.

|  |  |
| --- | --- |
| **Feature Selection Method** | **Number of Features** |
| No feature selection employed. | 720 |
| Features that the literature describes as having a strong influence on MMR [7, 8, 9, 10, 11, 12, 13, 14]. | 40 |
| All features whose absolute Pearson’s pairwise correlation coefficient with MMR >= 0.8. Called ‘Correlation 0.8’ from this point forward. | 11 |
| All features whose absolute Pearson’s pairwise correlation coefficient with MMR >= 0.7. Called ‘Correlation 0.7’ from this point forward. | 45 |
| All features whose absolute Pearson’s pairwise correlation coefficient with MMR >= 0.6. Called ‘Correlation 0.6’ from this point forward. | 113 |

As a note, feature selection was performed after cross-fold validation to ensure that the training samples used in each fold were consistent across the feature subsets.

### 4.34 Iterative Removal of Rows and Columns with a Higher Proportion of Missing Data Than a Specific Threshold

As discussed in Section 4.21, the input data contained information missing not at random. In this case, imputation and/or data removal can introduce bias, as the presence of missing data may signal important information about the country’s health system dynamics. However, if the specific feature has a very high proportion of missing data, such as over 85%, the model may overfit to its small dataset. Therefore, I experimented with removing rows and columns with very high proportions of data. More specifically, I iteratively removed columns and rows that contained a higher proportion of missing values than a pre-defined threshold. Higher thresholds meant rows and columns with a greater proportion of missing values were kept in the dataset.

Given that removal of a sparse row could affect the proportion of missing data in a column, and vice versa, the removal of rows and columns was conducted iteratively until the dataset stabilised. Similarly, since each feature subset had a different number of columns, the iterative removal of missing data had to be performed per feature subset/fold pair, as the number of columns influenced the proportion of missing data per row.

Iterative data removal was performed per fold to prevent data leakage between the folds (Figure 2e). This procedure was only applied to the training data, not the validation or testing sets. This allowed evaluation results from different training datasets to be compared. However, to allow the models to function, columns dropped from a model’s train set due to feature selection or missing data removal were also dropped from the validation and test sets.

I produced 4 versions of each feature subset/fold combination using missing data thresholds of 85%, 90%, 95%, and 100% (no missing data removed). I included missing data thresholds above and below 90% because researchers have hypothesised that even principled imputation methods can introduce bias into datasets with greater than 90% missing data (where data is missing at random) [37]. I used this range of missing data thresholds to determine whether a similar cutoff point occurs with data missing not at random when using decision-tree based methods like default directions to handle missing data.

Figure 7 (Section 5.12) shows how iterative data removal affected the size of the entire input dataset, which gives a rough idea of the effects of data removal per fold.

### 4.35 Summary of Datasets Produced Via Pre-Processing

The cleaned data was split into train/test subsets in a 90:10 ratio, with different versions of the split implemented for CLP and forecasting (Figure 2b). These subsets were each further split into 5 cross-validation folds (Figure 2c). Five versions of each fold were created by applying different feature selection mechanisms (Figure 2d). Finally, 4 versions of each of each feature subset/fold combination was produced by applying iterative missing data removal with different thresholds of missing data allowed (Figure 2e).

This produced 100 datasets (5 folds x 5 feature selections x 4 missing data thresholds) for each of country-level prediction and forecasting analyses.

## 4.4 Computational Workflow

Figure 3 visualises the development, testing, and investigation of a variety of decision-tree based machine learning models, with each of the following steps again done separately for country-level prediction and forecasting. I first trained a Random Forest, XGBoost, and LightGBM model on each version of the training data, creating 300 models (Figure 3a). I fine-tuned each of the models’ parameters over 1,000 Optuna trials (Figure 3b). I compared the performance of models trained on each version of the training data to determine which model type and combination of pre-processing techniques produced the best-performing model. I then experimented with combining the models’ predictions using a voting or stacking ensemble (Figure 3c). I trialled each of the Elastic Net, Random Forest, and Support Vector Machine architectures as the stacking ensemble’s meta-learner. Each of the voting and stacking ensembles’ hyperparameters were tuned over 1,000 Optuna trials. I then investigated whether using a subset or randomly permuted ordering of base estimators’ predictions reduced the predictive error of the best-performing voting or stacking ensemble (Figure 3d). Next, I conducted a feature importance analysis for the best-performing model to find the variables with the highest predictive power for MMR (Figure 3e). I then determined whether the predictive error of the best-performing model changed when it was trained on data from only one income level, thus investigating its sensitivity to its input data (Figure 3f). Finally, I compared my best-performing model’s MMR predictions to estimates from the BMat, CODEm, and GMatH models described in the literature review. These steps will be described in detail in the following subsections.

A diagram of a stacking ensemble

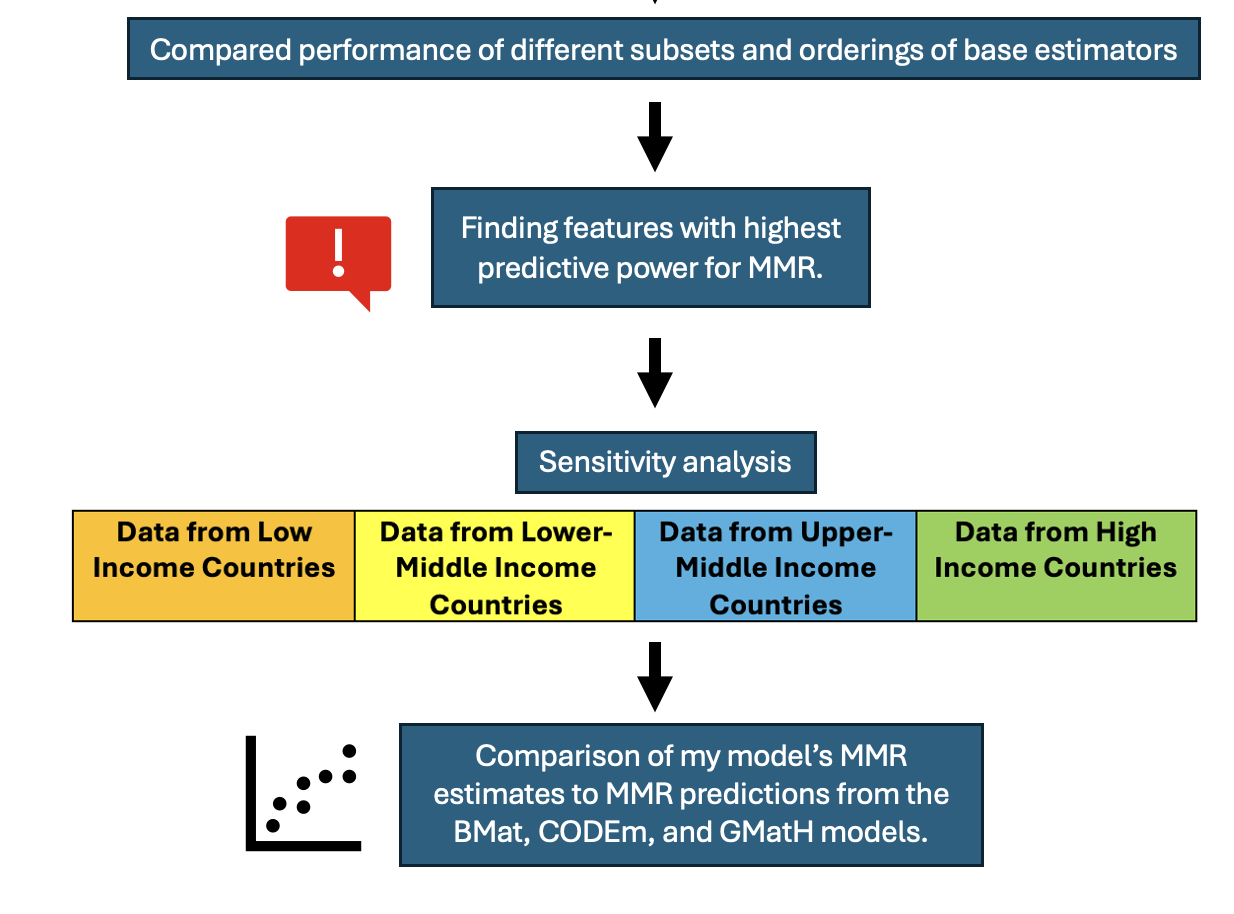
AI-generated content may be incorrect.

**d)** testing base estimator subsets

**b)** fine-tuning

**a)** training decision-tree based models on each of the 100 versions of the training data

**c)** training and fine-tuning the voting and stacking ensembles



**g)** benchmarking against the literature

**f)** sensitivity analysis

**e)** feature importance analysis

**Figure 3:** Experimental overview of the development of decision-tree based machine learning models used to estimate MMR. Development was done separately for models used to perform country-level prediction and forecasting. Each of the 100 versions the training data was used to fit a single Random Forest, XGBoost, and LightGBM model (a), with each model’s hyperparameters fine-tuned over 1000 Optuna trials (b). Different combinations of predictions from each of these base estimators were used to train voting and stacking ensembles (c, d). The features with the highest predictive power for MMR in the best-performing model were identified (e). Finally, I tested the best-performing model’s sensitivity to the input data and similarity to MMR predictions from existing models (f, g).

## 4.5 Base Model Training and Fine-Tuning

This section details the methodology used to train and fine-tune the Random Forest, XGBoost, and LightGBM base models (Figure 3a and 3b).

I did not use deep learning methods in this research because deep learning models cannot natively handle missing data [missing\_dl]. To use deep learning models with my sparse data, I would need to remove, ignore, or impute the missing values. However, this could introduce bias, especially given that information was missing not at random in my dataset. Therefore, I only used decision-tree based models, which can natively handle missing data by learning a default direction to move through the tree when they encounter missing data [cite].

### 4.51 Base Model Development

#### 4.511 Training and Fine-Tuning

Scikit Learn’s Random Forest Regressor, XGBoost’s XGBRegressor and LightGBM’s LGBMRegressor were trained to predict the MMR for a specific country, year datapoint [16, 17, 18]. 100 versions of each model type were trained for country-level prediction (CLP), with each version corresponding to one of the 100 versions of the train dataset produced during pre-processing. Another 100 versions of each model type were trained to perform forecasting.

The models’ hyperparameters were fine-tuned using the Optuna hyperparameter optimisation framework [19]. Finetuning occurred over 1,000 Optuna trials, where each trial represented a choice of values for the subset of hyperparameters being tuned. Tables 4, 5, and 6 show the specific hyperparameters fine-tuned for Scikit Learn’s Random Forest, XGBoost and LightGBM, respectively. All other hyperparameters were set to their default values. During each trial, the model being fine-tuned was fit to its associated train data using the chosen hyperparameter values. Its performance was evaluated by calculating the mean squared error (MSE) of its predictions on the associated validation fold. The set of hyperparameters with the lowest validation MSE across the 1,000 trials was saved and used to define the highest performing model. These best-performing hyperparameters can be accessed via the Optuna trial objects in the linked GitHub repository.

This method produced 300 fine-tuned models (100 each of XGBoost, LightGBM, and Random Forest models) for CLP and 300 models for forecasting. Thus, 600 models were fine-tuned in total, which took 1 to 2 days. Due to this computational demand, fine-tuning was conducted with only one metric as opposed to multiple metrics capturing different information. MSE was used because it heavily penalises outliers, which in this context would most likely be errors in the high MMR estimates for lower-income countries.

**Table 4:** Hyperparameter Tuning for Scikit-Learn’s Random Forest Regressor [16]

|  |  |  |
| --- | --- | --- |
| **Hyperparameter Name in Scikit-Learn** | **Hyperparameter Function** | **Range of Potential Values** |
| n\_estimators | The number of trees. | 10 to 300 |
| max\_depth | The maximum depth of trees. | 3 to 25 |
| min\_samples\_split | The minimum number of samples/rows for which an internal node can be split. | 2 to 10 |
| bootstrap | Whether each tree was trained on a random subset of samples. | True or False |
| max\_samples | Proportion of the full dataset used to train each base estimator. This parameter was not used when bootstrap is set to False. | 0.01 to 1.0 |

**Table 5:** Hyperparameter Tuning for XGBoost’s XGBRegressor [17]

|  |  |  |
| --- | --- | --- |
| **Hyperparameter Name in XGBoost** | **Hyperparameter Function** | **Range of Potential Values** |
| n\_estimators | Number of trees/boosting iterations. | 10 to 300 |
| max\_depth | The maximum depth of trees. | 3 to 25 |
| learning\_rate | Controls the extent to which each new tree influenced the model’s predictions. | 0 to 1 |
| reg\_alpha | Constant used for L1 regularisation. | 0 to 0.001 |
| reg\_lambda | Constant used for L2 regularisation. | 0 to 0.001 |
| booster | ‘gbtree’ was the XGBoost gradient boosting method. ‘dart’ modified ‘gbtree’ to randomly drop trees to reduce overfitting. | ‘gbtree’ or ‘dart’ |
| subsample | Proportion of dataset randomly chosen for each boosting iteration during training. | 0.1 to 1 |

**Table 6:** Hyperparameter Tuning for LightGBM’s LGBMRegressor [18]

|  |  |  |
| --- | --- | --- |
| **Hyperparameter Name in LightGBM** | **Hyperparameter Function** | **Range of Potential Values** |
| n\_estimators | Number of trees/boosting iterations. | 10 to 300 |
| max\_depth | The maximum depth of trees. | 3 to 25 |
| learning\_rate | Controls the extent to which each new tree influenced the model’s predictions. | 0 to 1 |
| reg\_alpha | Constant used for L1 regularisation. | 0 to 0.001 |
| reg\_lambda | Constant used for L2 regularisation. | 0 to 0.001 |
| boosting | ‘gbdt’ was the LightGBM gradient boosting method. ‘dart’ modified ‘gbdt’ to randomly drop trees to reduce overfitting. | ‘gbdt’ or ‘dart’ |
| bagging\_freq | Every k-th iteration, a random subset of data was used for the next k iterations of training. | 0 to 10 |
| bagging\_fraction | Proportion of input data randomly chosen for training. Used if bagging\_freq was not zero. | 0.1 to 1.0 |

#### 4.512 Testing and Comparison

The best performing hyperparameter values for each model were saved at the end of the Optuna fine-tuning process. When being evaluated, each model was re-fit on its associated training data using these hyperparameter values.

The 300 fine-tuned models for CLP were evaluated on the same test set, which had no missing data removed. The only difference between the test sets used for the various models was that each test set contained only the features included in the model’s training data.

The fine-tuned models were evaluated on the accuracy of their test set predictions. Specifically, the MSE, root mean-squared error (RMSE), mean absolute error (MAE), R2, and mean relative error of its test set predictions were calculated. Using a wide range of metrics enabled a more nuanced evaluation of the model’s performance, as the metrics placed different emphasis on outliers. The mean relative error was a symmetrical version of the mean absolute percentage error (MAPE) described in Section 2.3211. The formula for the mean relative error (Equation 11) modified the base MAPE formula (Equation 5) to adjust for its asymmetrically.

Each combination of model type, feature subset and missing data threshold had 5 associated training folds. The combination’s performance was the average test performance of the models trained on these folds. The performances of different combinations were then compared.

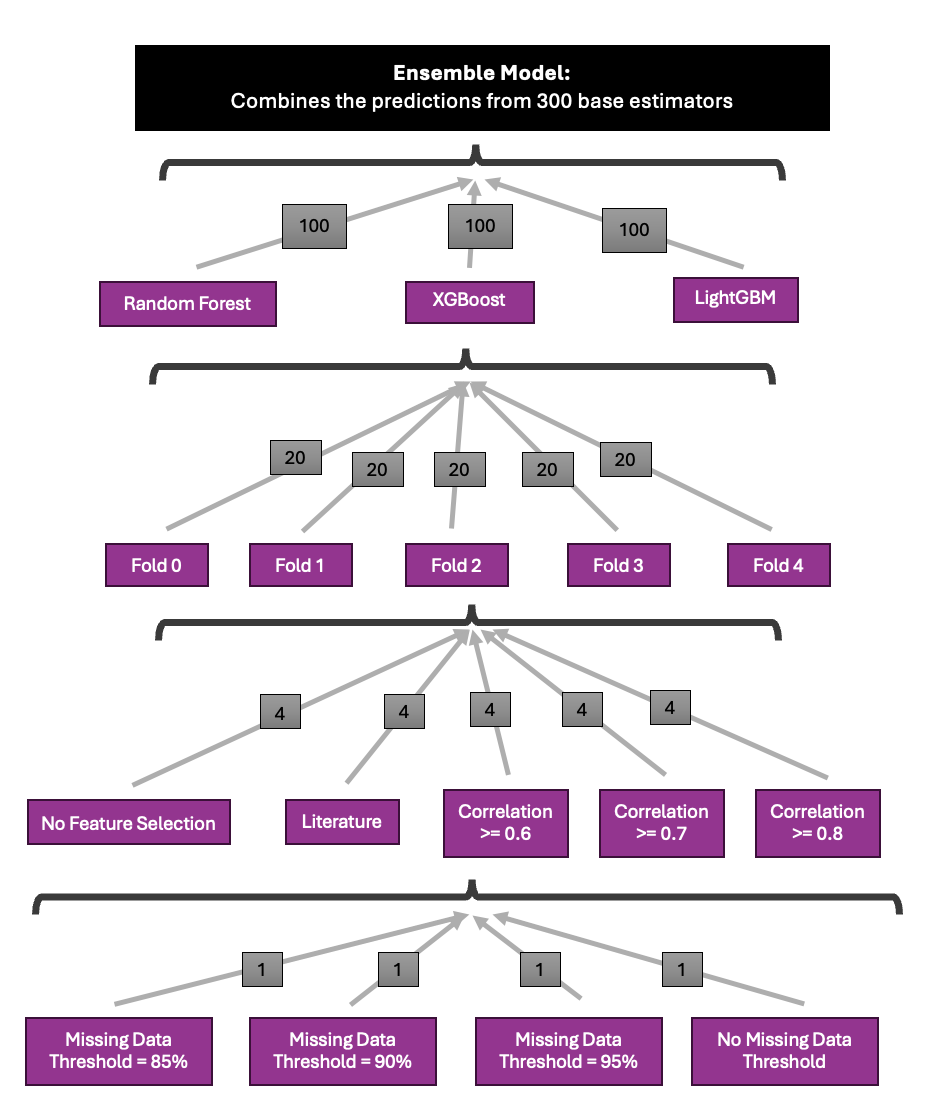
The 300 fine-tuned models that performed forecasting were similarly evaluated and compared.

#### 4.513 Feature Importance Analysis

Each feature’s importance was calculated using the models’ in-built methods. Scikit Learn’s Random Forest Regressor used the mean decrease in impurity/MSE across the feature’s splits [16]. XGBoost used the mean decrease in loss across the feature’s splits taking into account information from the loss function’s derivatives [17]. LightGBM calculated feature importance in a similar way to XGBoost but used the total decrease in loss rather than the average decrease [17, 18]. Feature importance was used to determine which variables had the highest predictive power for MMR, which was one of the primary aims of this thesis. Additionally, variation in feature importance across models was used to investigate differences in model performance.

### 4.52 Development of Voting and Stacking Ensemble Models

This section discusses how the 300 MMR predictions from the fine-tuned Random Forest, XGBoost, and LightGBM models were combined using voting and stacking ensemble methods (Figures 3c, 3d, and 3e).



a)

b)

c)

d)

**Figure 5:** Visualisation of where the 300 base estimators used as input into the ensemble models came from. The grey boxes gave the number of base estimators trained per method. The 300 base estimators consisted of 100 Random Forest, XGBoost, and LightGBM models (a). 20 versions of each model-type were trained on each cross-validation fold (b), with the 4 models from the same fold trained on different feature subsets (c). Each of these 4 models were trained on a dataset with a different missing data threshold (d).

While the fine-tuned XGBoost, LightGBM and Random Forest models discussed in Section 4.41 are bagging and boosting ensembles, they were referred to as “base estimators” for the remainder of this thesis, with the stacking and voting ensembles used to combine them referred to as “ensemble models”. Ensemble modelling was performed for both the CLP and forecasting analyses. Figure 4 visualises the ensemble models’ inputs.

#### 4.521 Using all Base Predictors in Different Types of Voting and Stacking Ensemble Models

Each (country, year) sample in the input dataset was associated with a single ground truth MMR estimate. Using the previously described method, the 300 fine-tuned models would each predict sample’s MMR, giving 300 predictions per sample. The ensemble model treated each prediction as a feature, learning how to combine the 300 features to produce a final, accurate MMR estimate (Figure 5a).

More specifically, the base estimators were fit on their associated training data using their best hyperparameter settings, which were determined through the fine-tuning process described above. Then, each of the fine-tuned models predicted the MMR for the full, concatenated training and validation sets. In other words, the fine-tuned models predicted on the training data before it was exposed to cross-fold validation, feature selection, and missing data removal. These predictions served as the training dataset for the ensemble models.

I used voting and stacking ensemble models to combine the base estimators’ predictions, as a review of ensemble methods given in Section 3.31 found that, of all models tested within a study, voting and stacking ensembles most frequently had the highest performance [20].

#### 4.5211 Voting Ensemble

The voting ensemble model produced a weighted average of the base estimators’ predictions for each country, year sample. The weighting given to each base model was determined through 1,000 Optuna fine-tuning trials, with weights ranging from 0 to 1. The weights that produced the lowest MSE on the ensemble training data were chosen as the optimal hyperparameters. No validation set was used because the voting ensemble was not ‘trained’, it was simply given different sets of weights to use to combine the various base estimators. Thus, the ‘training data’ served the same function as ‘out of sample validation data’, as it was ‘unseen’ by the ensemble.

#### 4.5212 Stacking Ensemble

The stacking ensemble method uses a meta-learner, which itself was an ML model. This meta-estimator learned how to combine the predictions from the 300 base estimators to produce the lowest predictive error. I created three versions of the stacking ensemble to compare the performance of different meta-estimators. More specifically, I used the Elastic Net linear regression model, Random Forest regressor, and Support Vector Regressor as candidate meta-learners. All models were used with the Scikit Learn’s implementation [22, 16, 23]. Elastic Net was used as a progression of the voting ensemble model, where the combination of L1 and L2 regression could both perform feature selection and reduce the possibility of overfitting, as described in the background [24]. Additionally, as described in the literature review, decision-tree based stacking ensembles outperform base estimators solely based on bagging and boosting, thus motivating use of the Random Forest regressor as a meta-learner [26]. Support vector regression was used as a meta-learner because its approach of only using datapoints outside its error tolerance margin could have interesting effects on how it uses the predictions from different base learners [25].

Each meta-estimator had internal parameters that needed to be tuned, rather than just hyperparameters like in the case of the voting ensemble. Table 7 describes the hyperparameters tuned for each meta-learner. As a result, the training dataset had to be split into train/validation sets. This allowed the stacking ensemble models to fit their internal parameters on the train set and fine-tune their hyperparameters on the validation set over 1,000 Optuna trials. The ensemble training data was split into train/validation sets in an 80:20 ratio using Scikit Learn’s ‘train\_test\_split’ method [21]. The hyperparameter values that produced the lowest MSE on the validation set were used in the final stacking ensemble models. These best-performing hyperparameters can be accessed via the Optuna trial objects in the linked GitHub repository.

**Table 7:** Hyperparameters Tuned for Each Stacking Ensemble Meta-Estimator

|  |  |  |  |
| --- | --- | --- | --- |
| **Ensemble Model** | **Hyperparameter Name** | **Hyperparameter Function** | **Range of Potential Values** |
| Elastic Net Stacking Ensemble | Alpha | Specifies the extent of regularisation. | 0.1 to 1 |
| L1\_ratio | Controls the weighting of the L1 versus L2 norm. Higher values push the regulariser closer to the L1 norm. | 0 to 1 |
| Random Forest Stacking Ensemble | Same parameters as described in Table 4. | | |
| Support Vector Machine Stacking Ensemble | kernel | Type of kernel used to transform input into a non-linear space. If ‘poly’, degrees tested were 2-5. | polynomial or radial basis function |
| C | Strength of regularisation term | 0.1 to 1 |
| epsilon | Error tolerance, used to determine support vectors. | 0.05 to 1 |

#### 4.522 Evaluating the Voting and Stacking Ensemble Models

A test set was generated to be able to evaluate the voting and stacking ensembles’ predictive performance on out-of-sample data. The ensembles’ input test data consisted of the base estimators’ predictions on their test sets. The ensemble models used these predictions to give final MMR estimates, which were compared to the test ground truth. The ensembles’ models’ test performance was used to assess their generalisability and determine the best ensemble.

#### 4.523 Analysis of Base Estimator Importance

To better understand the variation in the ensemble models’ predictive performance, I explored how each ensemble valued the contribution of the various base estimators. More specifically, I investigated whether different ensemble models placed the most importance on predictions from the same set of base estimators or on different base estimators.

As described earlier, the different base estimators can be thought of as “features” in the ensemble model. Thus, the importance placed on each base estimator by the ensemble model was quantified using the model’s built-in feature importance methods. Model importance in the Random Forest ensemble was determined using the same Scikit Learn feature importance calculation described in Section 4.413 [16]. The weighting of each base estimator in the voting ensemble’s final prediction was used as a proxy for model importance. Similarly, base estimator importance in the Elastic Net model was determined using the coefficient attached to each base estimator’s predictions.

#### 4.524 Testing the Performance of the Best Voting/Stacking Ensemble with Different Subsets of Base Estimators

After establishing the best performing voting or stacking ensemble, I tested whether its performance could be improved by using a different combination of base estimators (Figure 3d). I compared its test performance when its input dataset only consisted of predictions from the following subsets of base estimators for both CLP and forecasting.

* 300 base estimators consisting of XGBoost, LightGBM, and Random Forest regressors (original ensemble model).
* 100 base estimators consisting of just XGBoost regressors.
* 100 base estimators consisting of just LightGBM regressors.
* 100 base estimators consisting of just Random Forest regressors.

#### 4.525 Investigating Base Estimator Selection in the Best Performing Voting/Stacking Ensemble

To gain a deeper understanding of the best performing voting/stacking ensemble, I investigated possible reasons for its choice of base estimators. More specifically, I explored whether the chosen base estimators had the lowest mean-squared error, as this was the metric used to train and fine-tune the ensembles. I compared the predictive performance of all base estimators given an importance score of at least 0.03. I also tested whether choice of base estimator was arbitrarily determined by its position in the input dataset used for the stacking/voting ensemble. I did this by permuting the base estimators’ positions and re-estimating base model importance in the ensemble.

#### 4.526 Feature Importance Analysis in the Best Performing Ensemble

Determining predictive power of various socio-economic and health-related variables was a primary aim of this thesis. Thus, I calculated the importance scores of the feature variables used by the most important base estimators in the highest performing ensemble (Figure 3e).

#### 4.527 Analysis of the Best Performing Ensemble’s Prediction Error by Income Level

The distribution of prediction errors per income level made by the best performing voting/stacking ensemble was visualised and analysed. This experiment gave greater insight into how the model performed in different settings.

#### 4.528 Analysis of the Best Performing Ensemble’s Uncertainty

There were 300 base estimator predictions for each ground truth MMR estimate. I calculated the standard deviation among the 300 predictions for every datapoint in the test set to explore the base estimators’ agreement on the true MMR prediction. Lack of consensus among the base predictors would likely make the ensemble’s prediction less stable. As a result, this analysis provides an approximation for uncertainty in the ensemble’s predictions.

## 4.6 Sensitivity Analysis

I conducted a sensitivity analysis to gain a deeper understanding of how the input dataset affected the quality of the best performing ensemble’s predictions (Figures 3f and 6). The sensitivity analysis was conducted in the same way both country-level prediction and forecasting.

A diagram of a method

AI-generated content may be incorrect.

**Figure 6:** Visualisation of the sensitivity analysis procedure, where the ensemble’s base estimators were fit on data from a single income level (in this case, high-income). The sensitivity analysis created a separate ensemble model for each subset of the input data corresponding to a specific income level.

To perform the sensitivity analysis, I created 4 new versions of each fold/feature subset dataset by filtering the input dataset by income level. I then conducted missing data thresholding on each filtered dataset, as above. The filtered datasets were referred to as “sensitivity datasets” from this point forward, with the four filtered datasets characterised in the following list. This process generated 300 base estimators trained on each sensitivity dataset.

* Data from low-income countries only.
* Data from lower-middle income countries only.
* Data from upper-middle income countries only.
* Data from high-income countries only.

Each sensitivity analysis ensemble was fit on the base estimators’ predictions for a filtered version of the concatenated train, validation data, which only contained data from the relevant income level. Fine-tuning followed the same procedure described in Section 4.411.

The sensitivity analysis ensemble models were then evaluated using a version of the test dataset filtered to only contain data from the relevant income level. Each sensitivity analysis ensemble was compared to the original best performing ensemble, which was trained on data from all income levels but only used to predict on the filtered test set. For example, the sensitivity analysis ensemble trained and tested on data from low-income countries was compared to the original ensemble trained on all data but tested on data from low-income countries. Using the same test set ensured comparability between the original and sensitivity analysis models.

## 4.7 Comparison to Literature

I compared my best performing ensemble, trained on data from all income levels, to the latest versions of the GMatH simulation model, BMat model, and CODEm model (Figure 3f). More specifically, I compared my model’s MMR predictions for each country/year sample to the predictions made by models described in the literature [27, 28, 29]. The MMR estimates from the literature were given with their 95% confidence intervals.

First, I calculated the percentage difference between my test set MMR estimates and the corresponding estimates from each literature model using Equation 12, below. This gave an indication of the similarity between my MMR estimates and the literature’s MMR predictions.

Then, I determined the percentage of test set MMR estimates from my best performing ensemble that fell in the 95% confidence interval of the corresponding estimates from the literature. This analysis was performed for all test set estimates and per income level, where I calculated the proportion of test set estimates for countries from a particular income that fell within the 95% confidence intervals of the associated estimates from the literature. I also calculated the proportion of ground truth MMR estimates used to test my model that fell within the 95% confidence interval of the literature models’ estimates. This compared the MMR estimates that my model was trained to predict with the literature’s MMR estimates to provide more information about whether differences between my model and the literature were due to poor performance or training data.

Finally, I visualised the difference between my model’s MMR estimates and the literature’s estimates for an exemplar country from each income level. I visualised the model’s MMR estimates for country-level prediction and forecasting separately. I attempted to maximise geographic coverage by comparing estimates for at least one country in the Americas, Africa, Europe, Oceania, and Asia.

I performed this comparison for MMR estimates produced by both the country-level and forecasting models.

## 4.8 Note About Limited Computational Resources

Further experiments that involved training different base estimators and/or ensemble combinations of base estimators were not conducted due to a lack of computational resources. Many of the experiments described in this chapter were performed on the Gadi supercomputer, as the model training process ran too slowly to be feasible on my personal computer. However, after completing the experiments in this chapter, my team had used all the computational resources allocated to them on the supercomputer this quarter, preventing further experiments from being run. In total, 527 kilo-Service Units were used this quarter by the team, with my research using 123.22 KSUs.

Citations

1. Onambele, L.; Guillen-Aguinaga, S.; Guillen-Aguinaga, L.; Ortega-Leon, W.; Montejo, R.; Alas-Brun, R.; Aguinaga-Ontoso, E.; Aguinaga-Ontoso, I.; Guillen-Grima, F. Trends, Projections, and Regional Disparities of Maternal Mortality in Africa (1990–2030): An ARIMA Forecasting Approach. Epidemiologia 2023, 4, 322–351. https://doi.org/ 10.3390/epidemiologia4030032
2. <https://link.springer.com/article/10.1186/s40537-021-00516-9>
   1. Journal of big data
   2. Missing data mechanisms
3. <https://link.springer.com/article/10.1007/s40273-023-01297-0#Sec2>
   1. More about missing data mechanisms
4. <https://scikit-learn.org/stable/modules/generated/sklearn.impute.KNNImputer.html>
5. <https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html>
6. <https://www.nature.com/articles/s41592-018-0019-x>
   1. Curse of dimensionality
7. <https://jmai.amegroups.org/article/view/8590/html>
8. <https://www.sciencedirect.com/science/article/pii/S0169023X23000587?pes=vor&utm_source=scopus&getft_integrator=scopus>
9. <https://iris.who.int/bitstream/handle/10665/381012/9789240108462-eng.pdf?sequence=1>
10. <https://onlinelibrary.wiley.com/doi/10.5694/mja2.52452>
    1. Non-communicable disease and pregnancy
    2. Ramson et al., 2024
11. Akselrod S, Banerjee A, Collins TE, Acharya S, Artykova N, Askew I, Berdzuli N, Diorditsa S, Eggers R, Farrington J, Jakab Z, Ferreira-Borges C, Mikkelsen B, Azzopardi-Muscat N, Olsavszky V, Park K, Sobel H, Tran H, Vujnovic M, Weber M, Were W, Yaqub N, Berlina D, Dunlop CL and Allen LN (2023) Integrating maternal, newborn, child health and non-communicable disease care in the sustainable development goal era. Front. Public Health 11:1183712. doi: 10.3389/fpubh.2023.1183712
12. [**https://pubmed.ncbi.nlm.nih.gov/27642018/**](https://pubmed.ncbi.nlm.nih.gov/27642018/)
13. [**https://pmc.ncbi.nlm.nih.gov/articles/PMC11117177/pdf/jogh-14-04128.pdf**](https://pmc.ncbi.nlm.nih.gov/articles/PMC11117177/pdf/jogh-14-04128.pdf)
14. [**https://www.sciencedirect.com/science/article/pii/S2589537024002323**](https://www.sciencedirect.com/science/article/pii/S2589537024002323)
15. <https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GroupKFold.html>
    1. K-fold cross validation mechanism
16. <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html#sklearn.ensemble.RandomForestRegressor>
    1. Scikit learn random forest
17. <https://xgboost.readthedocs.io/en/stable/parameter.html>
    1. Xgboost
18. <https://lightgbm.readthedocs.io/en/latest/Parameters.html>
    1. Lightgbm
19. [**https://dl.acm.org/doi/10.1145/3292500.3330701**](https://dl.acm.org/doi/10.1145/3292500.3330701)
    1. Optuna
20. <https://pmc.ncbi.nlm.nih.gov/articles/PMC10298658/>
    1. Ensemble methods review
21. <https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html>
    1. Train test split method
22. <https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.ElasticNet.html>
    1. Elastic net
23. <https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html#sklearn.svm.SVR>
    1. Support vector regressor
24. <https://academic.oup.com/jrsssb/article/67/2/301/7109482>
    1. Elastic net
25. <file:///Users/rosalitarosenberg/Downloads/B_STCO.0000035301.49549.88.pdf>
    1. A tutorial on support vector regression
    2. Smola and Sch¨olkopf
26. <https://www.nature.com/articles/s41598-024-71934-x#Abs1>
    1. Ensemble article with high performance from DT based methods
27. <https://www.who.int/data/gho/data/indicators/indicator-details/GHO/maternal-mortality-ratio-(per-100-000-live-births)>
    1. UN MMEIG data
28. Global Burden of Disease Collaborative Network.
    1. Global Burden of Disease Study 2021 (GBD 2021) Results.
    2. Seattle, United States: Institute for Health Metrics and Evaluation (IHME), 2022.
    3. Available from <https://vizhub.healthdata.org/gbd-results/>.
29. Ward, Zachary, 2023, "Simulation Results: Global, regional, and country-level maternal mortality by cause, 1990-2050", <https://doi.org/10.7910/DVN/UBGY9P>, Harvard Dataverse, V1, UNF:6:YbDwFpLTGF39vO7onL7qww== [fileUNF]
30. <https://genderdata.worldbank.org/en/indicator/sh-sta-mmrt?estimate=National>
    1. Mmr data
31. <https://genderdata.worldbank.org/en/topics/health#idAllIndicators>
    1. World bank data
32. <https://www.who.int/data/sets/health-inequality-monitor-dataset#swper>
    1. Empowerment data
33. <https://www.who.int/data/sets/health-inequality-monitor-dataset#wb>
    1. Health determinants data (mostly economic)
34. <https://www.who.int/data/sets/health-inequality-monitor-dataset#ihme-gbd>
    1. Illness prevalence data
35. <https://www.who.int/data/sets/health-inequality-monitor-dataset#gho>
    1. Income level data
36. <https://www.iban.com/country-codes>].
    1. ISO3 codes
37. <https://link.springer.com/article/10.1007/s42979-022-01249-z>