**Agenda:**

* Thesis: spending time at the moment doing and writing up literature review.
* Ensemble based models
* Sensitivity analysis
* Questions about, and results from, tasks assigned during our previous meeting

**Ensemble Based Models**

Process: (repeated for the two train/test splits) -> data split by year or by country

*Description of the base estimators:*

* There are 20 base estimators
  + 10 base estimators each of LightGBM and Random Forest Regressors
    - 5 base estimators for each model type were trained on the full train dataset (1 for each of the train/test folds)
    - 5 base estimators for each model type were trained on the train dataset with a 95% missing data threshold (1 for each of the train/test folds)
* XGBoost base estimators and all estimators trained on the dataset with the 85% missing data threshold were not included in the ensemble model due to their consistent poor performance.

*Training base estimators:*

1. All base estimators were initialised with their optimal hyperparameters (from previous fine-tuning experiments) and data from their associated cross-validation fold.
2. The original dataset was initially split into train/test data in a 90:10 proportion.
   1. Each trained base estimator was used to predict labels for the entire train dataset (with all columns it was not trained on removed).
   2. Rationale: This produces a prediction for each training data sample from each base estimator that can be used to train the ensemble model.

*Voting ensemble:*

Voting ensemble models give a (possibly) weighted average of all predictions from the base estimators.

I used 300 Optuna trials to experiment with relative weightings of different base estimators’ predictions in calculating the weighted average prediction score.

Since there was no ‘training’ element of this process, there was no need to split into train/validation sets.

*Stacked ensemble:*

This ensemble method uses another machine learning technique to combine the base estimators’ predictions. I used 300 Optuna trials to experiment with the hyperparameters used in the ensemble machine learning model.

1. Take an intermediate dataset consisting of the base estimators’ predictions on the full training dataset and the associated true MMR values.
2. Split this intermediate dataset an 80:20 proportion corresponding to a train/validation set for hyperparameter tuning.
   1. Train the ensemble models (initialised with their optimal hyperparameter selection) on the train dataset, test the specific hyperparameter combination on the validation dataset, and choose the best hyperparameter combination based on the lowest mean-squared error.

*Stacking ensemble models used:*

* **Linear regression model** (Scikit Learn’s Elastic Net implementation, which combined the L1 and L2 norms to perform regularisation)
  + Alpha: specifies the overall regularisation importance
  + L1\_ratio: specifies the weighting of the L1 norm versus L2 norm (higher = closer to pure L1)
* **Random Forest**
  + Number of trees
  + Max depth
  + Minimum number of samples needed to split an internal node
  + Whether bootstrapping was used, and if so, what proportion of the dataset is used to train the individual trees
* **Support Vector Machine** 
  + Type of kernel (polynomial or RBF)
    - If polynomial, which degree (2 to 5)
  + Regularisation strength
  + Permitted error from true value
* **Multi-Layer Perceptron neural network**
  + Number of layers
  + Number of hidden units per layer (does not need to be uniform)
  + Type of activation function (logistic or ReLU)
  + Learning rate
  + Strength of L2 regularisation

*Justification for model choice:*

* Past study finding random forest and XGBoost as meta-estimator in stacking outperformed base estimators
  + However, due to XGBoost’s poor base estimator performance, I used the random forest implementation as the meta-estimator in this study
* Linear regression was also used by studies in the literature as a meta-estimator
* SVM and MLPs were used as base estimators in the literature for similar problems, but I could not use these due to missing data in my dataset. Thus, I experimented with whether they added value to the study by including them as meta-estimators.

*Model Testing*:

1. Use the base estimators to predict the MMR value from the testing dataset (with the only difference between test sets used between base estimators is that base estimators were only fed data relating to the columns they had been trained with.
2. Stack these predictions into a data frame.
3. Each (voting or stacked) ensemble model predicts the MMR value based on the base estimators’ results.

*Split by Country*

A graph with blue rectangular bars

AI-generated content may be incorrect.

A graph with blue rectangular bars

AI-generated content may be incorrect.

A graph with blue rectangular bars

AI-generated content may be incorrect.

A graph of a bar

AI-generated content may be incorrect. A graph of a bar

AI-generated content may be incorrect.

Conclusions:

* In each of the metrics, using Scikit Learn’s Random Forest model as a stacking ensemble meta-estimator produced the lowest error.
* Voting and linear regression performed very similarly
  + Which makes sense, given that their algorithm is very similar
* SVM performed the worst and MLP performed the second-worst.

Interestingly, the MLP meta-estimator model had the lowest validation score. Likely, its poor testing score was due to overfitting due to the complexity of a neural network.

A screenshot of a computer

AI-generated content may be incorrect.

As a point of comparison, I plotted the relative test error for the ensemble models next to the mean relative error for the base estimators (averaged across their cross-validation folds).

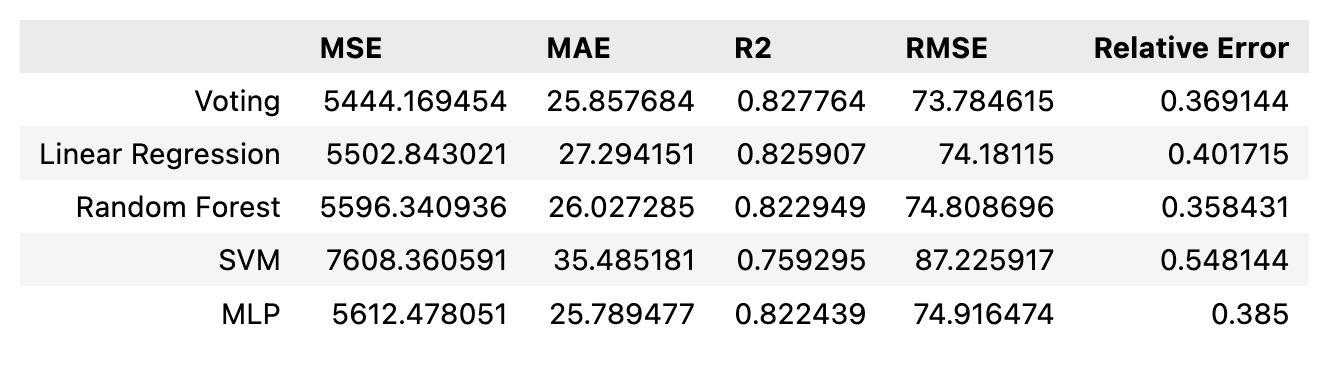
I only show this for one metric for brevity. While the ensemble-based model did not always do better than the averaged base estimator (e.g. SVM and linear regression), the Random Forest ensemble model outperformed all base estimators’ averaged scores.

A graph with different colored bars

AI-generated content may be incorrect.

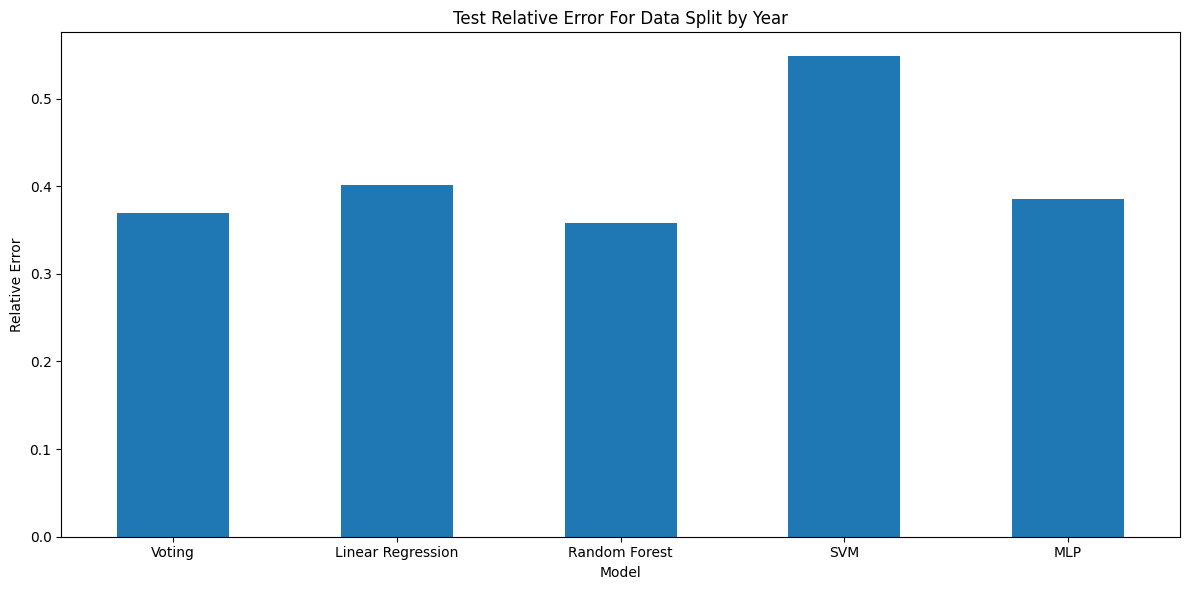
*Split by Year*

I presented a table with the ensemble models’ performance metrics because it is a little difficult to see the difference between them on the graphs included below.

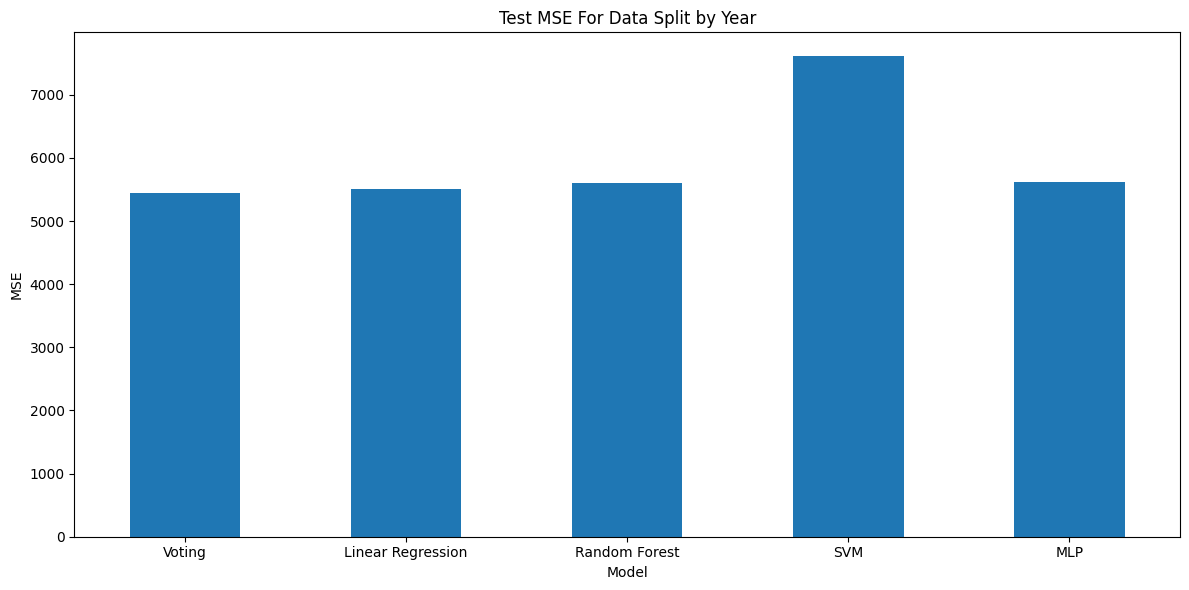
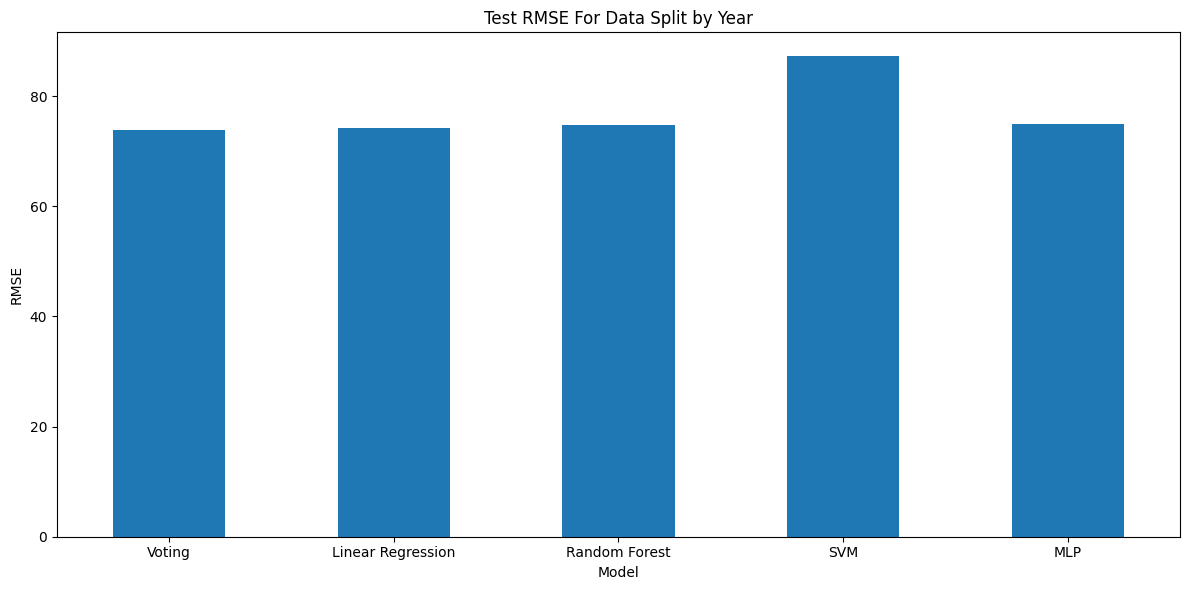




While the voting ensemble has the lowest MSE and RMSE scores & the highest R2 score, the MLP network has the higher MAE and the Random Forest model has the lowest relative error.



A graph of a graph

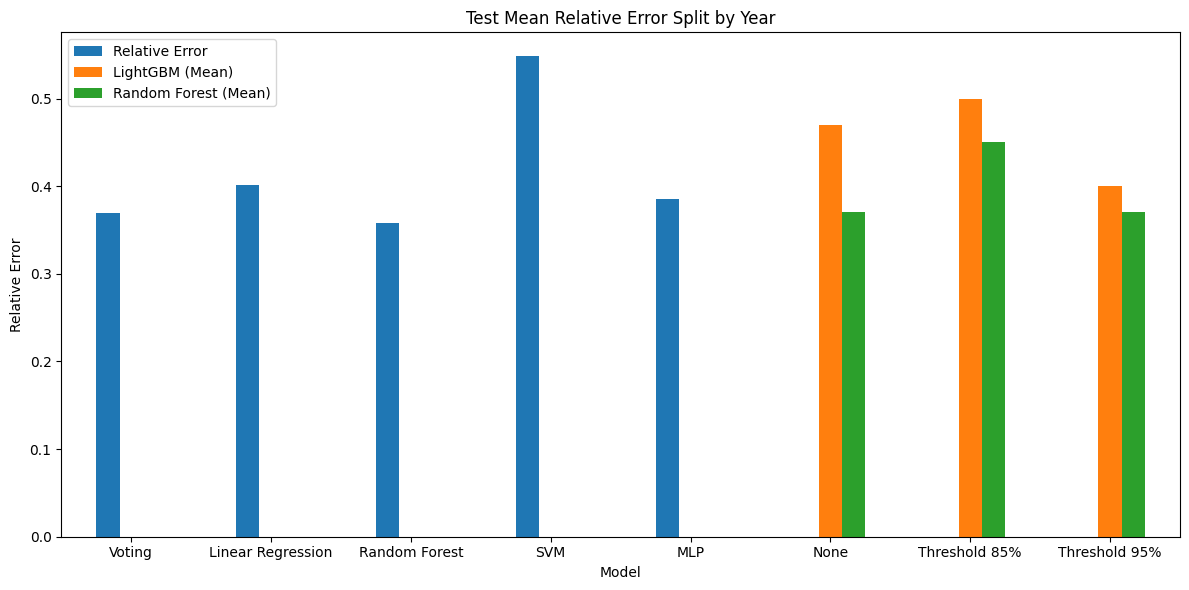
AI-generated content may be incorrect.   A graph of blue rectangular bars

AI-generated content may be incorrect.

Conclusions:

* Unlike when the data was split by country, there is not a clear best performing ensemble based model. Instead, performance varied depending on metric nad there was less difference between the models.
  + Potentially because forecasting into the future is more difficult and relies less on what is already known.

As shown below, the Random Forest meta-estimator had the lowest relative error of an ensemble or base estimator model, but the base-estimator models had higher or similar perforance as the rest of the ensemble based methods.



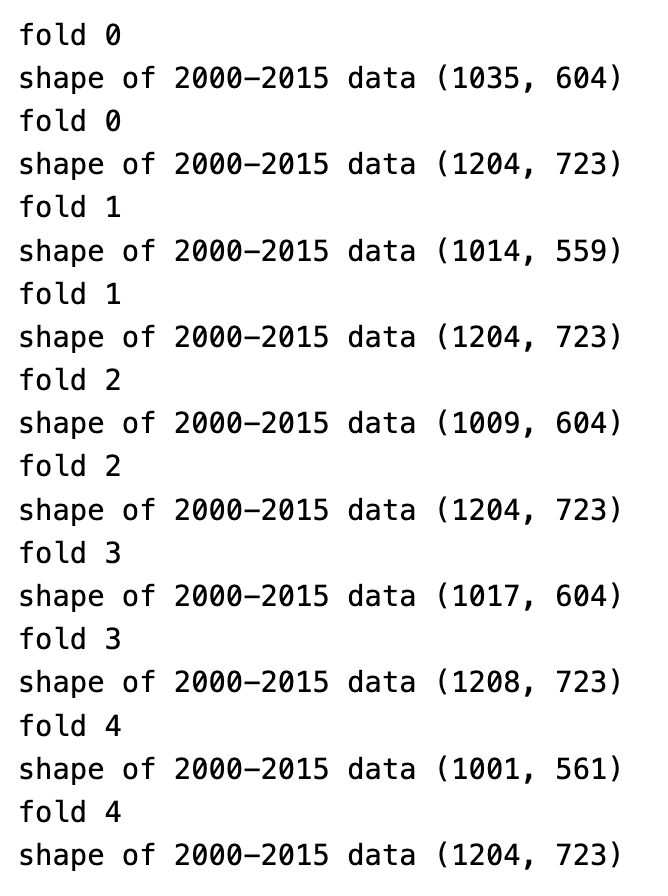
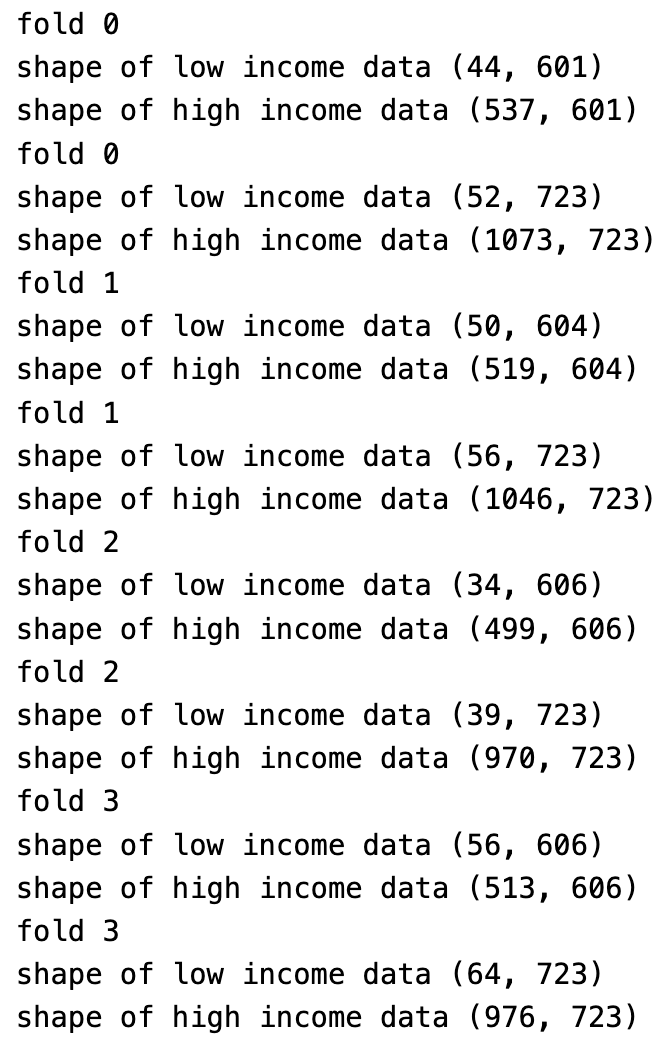
Note on MLP stacking ensemble model trained on data split by year:

* Hyperparameter tuning encouraged me to increase the number of hidden layers that the model could use. While the best hyperparameter combination was 7 hidden layers (in a range of 1 to 7), I did not continue to increase the number of possible hidden layers to prevent overfitting to training data.
  + In comparison, the optimal number of hidden layers for the MLP model used when data was split by year was 3 from a range of 1 to 5
    - Potentially, the model used to predict the future needed to detect more complex patterns and thus needed to use the additional hidden layers.

**Sensitivity analysis:**

In a previous lab meeting, we discussed the importance of a sensitivity analysis. I made an initial attempt at performing this type of analysis and would be interested to hear your feedback on the procedure.

1. Using the optimal hyperparameters for the base estimators, I retrained each estimator on their corresponding cross-validation fold.
   1. Split by country:
      1. Conducted two sensitivity analyses:
         1. Only using data from countries classified as ‘low income’
         2. Only using data from countries classified as ‘high income’
   2. Split by year:
      1. Only used years 2000-2015 for training
   3. More explicitly, I took the data for each cross-validation fold and applied the filters for the relevant sensitivity analysis
      1. This could mean that the cross-validation folds had varying amounts of data, e.g.



1. I then used these base estimators to predict the MMR value for all training data (all income levels and years in the training data regardless of sensitivity analysis).
2. These predictions were used to train the ensemble based methods.
3. All ensemble based models were then used to predict the MMR values from all testing data (all income levels and years)

Is this the correct procedure?

*Split by Country*

* Generally, the models trained solely on the low-income dataset had larger error (potentially due to the drastically smaller dataset size
* Sensitivity analysis had the largest effect on the voting-based ensemble model.
* Depending on the model, there was higher performance on the full or high income only dataset.
  + Potentially, this was due to there being more data from the high-income dataset, allowing the models to capture its patterns more effectively.
  + Additionally, with only the high-income dataset, there may have been less ‘noise’ from countries with different patterns that the model then needed to try and learn.
* The sensitivity analyses appear to have the smallest effect on the Random Forest and MLP models.

A table with numbers and text

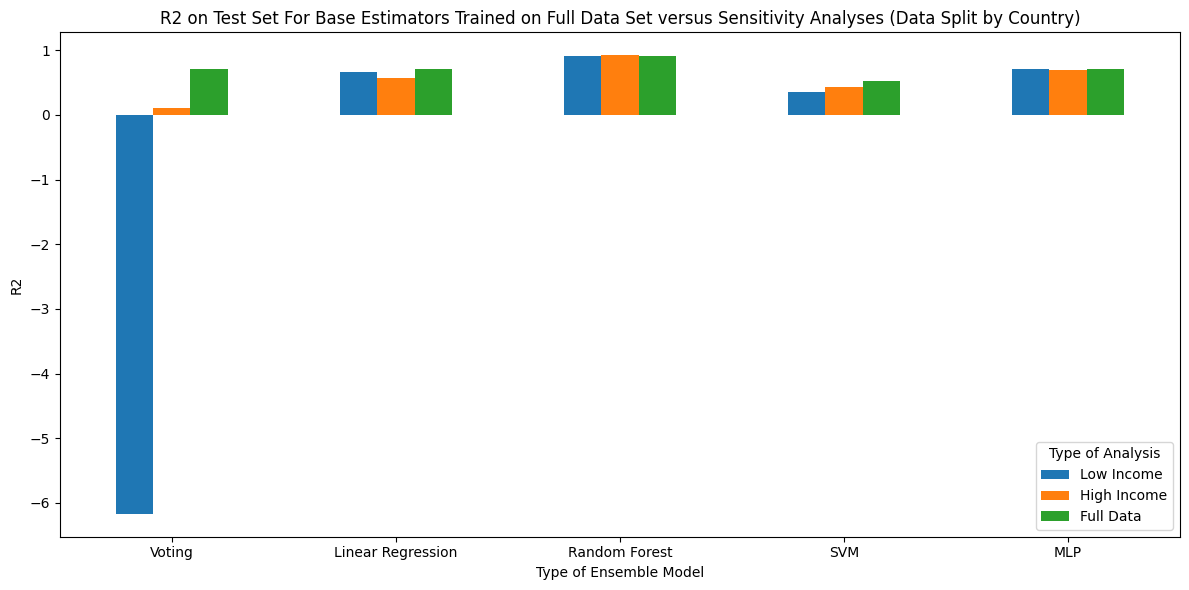
AI-generated content may be incorrect.

A graph of different colored bars

AI-generated content may be incorrect. A graph with different colored squares

AI-generated content may be incorrect. A white screen with a white background

AI-generated content may be incorrect. A graph of different colored bars

AI-generated content may be incorrect. 

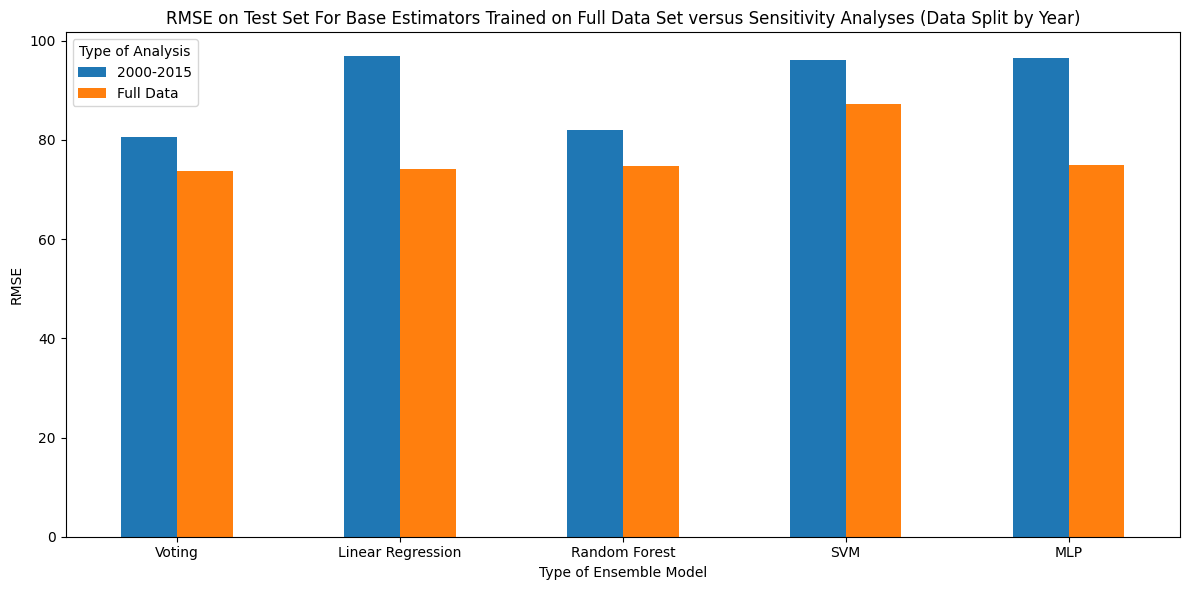
*Split by Year*

* Other than for the linear regression based ensemble model, the sensitivity analysis (restricting training data to 2000-2015) made little difference to the relative error and MAE metrics.
  + For these metrics, some models trained on this restricted training dataset produced better results.
  + Potentially because the testing set challenged the models to predict MMR values from 2015-2018, closer to the training set which now includes less ‘noise’ and irrelevant patterns from earlier years which may no longer be relevant due to institutional changes in maternal health management and healthcare systems.
* For MSE, the restricted training set always performed notably worse.
* The restriction appeared to have the largest effects on the MLP and linear regression ensemble models.

A graph of different colored bars

AI-generated content may be incorrect. A graph of different colored bars

AI-generated content may be incorrect. A graph of different colored bars

AI-generated content may be incorrect.  A graph of different colored bars

AI-generated content may be incorrect.

**Questions about, and results from, tasks assigned during our previous meeting**

Pairwise correlation analysis between features

Last week we discussed how I should explore the pairwise correlation between features, potentially dropping unnecessary features (i.e. take a highly correlated pair and drop the feature with more missing data).

*Results of exploratory analysis*:

* There are 584 pairs of features whose absolute pairwise correlation is greater than 0.95 and 923 pairs of features whose absolute correlations are greater than 0.90. I have included a histogram showing the distribution of correlations below.

A graph of a graph

AI-generated content may be incorrect.

* I took the example of feature pairings with absolute correlation coefficient greater than 0.9 and filtered by pairings whose absolute difference in their proportion of missing data was greater than 0.05 (to make it worth replacing the member of the pair with less data with the one with more data), there were only 129 of the original 923 pairs remaining.
* When I repeated this with all pairs whose absolute correlation coefficient was greater than 0.8 (n=2944) and filtered for pairs whose missing data proportions differed by more than 20%, the code returned 785 pairs.
  1. The following table shows the number of times different indicators appeared in a pairing.

|  | **count** |
| --- | --- |
| Mortality rate, infant, male (per 1,000 live births) | 57 |
| Mortality rate, infant (per 1,000 live births) | 57 |
| Mortality rate, under-5, male (per 1,000 live births) | 52 |
| Mortality rate, infant, female (per 1,000 live births) | 52 |
| Mortality rate, under-5 (per 1,000 live births) | 49 |
| ... | ... |
| Women and girls who participate in activities during menstrual period (% of women and girls ages 15-49 who had a menstrual period within the last year) | 1 |
| Teenage mothers (% of women ages 15-19 who have had children or are currently pregnant) | 1 |
| Prevalence of underweight, weight for age, female (% of children under 5) | 1 |
| 1.D.14 Intestinal nematode infections prevalence (age standardized) (per 100 000 population) female | 1 |
| Unemployment with intermediate education (% of total labor force with intermediate education) male | 1 |

How to decide which features to get rid of without losing important information.

Intersection of test features

Last week you mentioned wanting to see the intersection of features that were used across all folds. The order goes: missing data threshold 85%, missing data threshold 95%, no missing data threshold.

A close-up of a number

AI-generated content may be incorrect.

Excluding setting while testing models

Since I need to test models on the same subset of features they were trained on, do I just set the entire ‘setting’ feature column to NaN?

Logistic and linear regression:

Last week we discussed performing logistic and linear regression to use for comparison with other, more complex machine learning methods.

Unfortunately, none of the methods I found have native support for missing values; they all required imputation and/or deletion before use. The XGBoost linear regression implementation replaced all missing values with zero, which seemed that it would worse than most imputation methods.

**Miscellaneous clarifying questions**

1. How to show non-linearity in MMR?
   1. Is it enough to show this graphically?
2. I’ve done some reading of the literature to find sources that help me justify choice of ML models
   1. Does this justification go into the ‘background research’, ‘literature review’, or ‘methods’ sections?

**Possible To-Dos for This Week**:

* Improve sensitivity analysis method
* Implement feature selection based on correlation and retrain/optimise all models to compare effect
* How to calculate SHAP values?
  + For which models should this be done?