

Machine Learning Final Project

Student Information:

1. Kontos Christos – AEM: 224

Table of Contents

1	Introduction.....	4
2	Data Analysis.....	4
2.1	Preprocessing	6
2.1.1	Drop hhid, com and survey_id features	6
2.1.2	Handle Employment.....	6
2.1.3	Handle Missing Values.....	6
2.2	Categorical Encoding and Feature Alignment	7
2.2.1	Feature Schema Alignment.....	7
3	Train-Test Split	7
4	Model Implementation and Hyperparameter Tuning	7
4.1	Random Forest Regressor	7
4.2	XGBoost Regressor	8
4.3	Lasso Regression	9
4.4	Multi-Layer Perceptron (MLP)	10
5	Prediction Pipeline and Submission Generation	12
6	Conclusion.....	13
6.1	Ways to Improve Data Quality.....	13

Table of Figures

Figure 4-1 Random Forest Feature Importance	8
Figure 4-2 XGBoost Feature Importance	9
Figure 4-3 Lasso Regression Feature Importance	10
Figure 4-4 MLP Architecture	10
Figure 4-5 MLP MAE and MAPE History.....	11
Figure 4-6 MAE Comparison.....	11
Figure 4-7 MAPE Comparison	12
Figure 5-1 Competition Result	12

1 Introduction

The objective of this project is to develop a predictive model for household consumption (cons_ppp17) based on survey data, as accurate consumption prediction is a fundamental step in economic analysis, allowing for the estimation of poverty headcount ratios across different geographic and social strata. By leveraging machine learning, we aim to identify patterns in household characteristics that correlate with economic well-being.

First, the various dataset files were downloaded from the competition website and uploaded to the Google Colab environment for further processing. Various files were contained in the .zip files but only test_hh_features.csv, train_hh_features.csv and train_hh_gt.csv were uploaded, with feature_descriptions.csv being used to further investigate what each feature represents, something which will be further analyzed later in this report.

After the various libraries needed for this implementation were imported, 3 dataframes were created from the train_hh_features.csv, train_hh_gt.csv and test_hh_features.csv files, those being, train_features, train_targets and test_features accordingly.

2 Data Analysis

Along with the feature_descriptions.csv file that contains all the feature representations which will be explained, a Profiling Report .pdf file was created using the ydata_profiling library, for easier analysis and processing of the data.

The various features in the dataframe are as follows:

1. **hhid** represents the unique household identifier.
2. **com** represents the identifier of the specific household member providing data.
3. **weight** indicates the household sampling weight used to represent the broader population.
4. **strata** represents the stratification variable defined during the survey design.
5. **utl_exp_ppp17** represents the household's daily expenditure on utilities, adjusted for Purchasing Power Parity (PPP17).
6. **male** is a binary indicator of whether the household head is male.
7. **hsize** records the total number of members residing in the main household.
8. **num_children5** represents the number of children under 5 years old in the household.
9. **num_children10** represents the number of children aged 5 to 10 years old in the household.
10. **num_children18** represents the number of children aged 10 to 18 years old in the household.
11. **age** displays the age of the household head.
12. **owner** indicates whether the household owns their current dwelling.

13. **water** indicates if the dwelling has access to a formal water supply system.
14. **toilet** indicates the presence of toilet facilities within the dwelling.
15. **sewer** indicates whether the toilet facilities are linked to a public sewer system.
16. **elect** states whether the dwelling has access to electricity.
17. **water_source** records the specific primary sources of drinking water used by the household.
18. **sanitation_source** describes the main type of sanitation facility used by the household.
19. **dweltyp** classifies the specific type of dwelling inhabited by the household.
20. **num_adult_female** counts the number of adult females (aged 18 to 69) in the household.
21. **num_adult_male** counts the number of adult males (aged 18 to 69) in the household.
22. **num_elderly** counts the number of elderly members (aged 70 and above) in the household.
23. **employed** indicates whether the household head is currently employed.
24. **sworkershh** represents the share of all adults in the household who are currently working.
25. **share_secondary** represents the share of adults in the household who have completed a secondary education.
26. **educ_max** indicates the highest level of education attained by any member of the household.
27. **sworkershh** represents the share of working adults in the household who are in formal employment.
28. **any_nonagric** indicates if any household member is employed in a non-agricultural sector.
29. **sector1d** identifies the employment sector of the household head using 1-digit ISIC codes.
30. The **region{n}** variables (region1 through region7) indicate the specific geographic region of the household.
31. **urban** serves as an indicator of whether the household is located in an urban or rural area.
32. The **consumed{n}** variables indicate whether the household consumed specific items, such as bread, meats, or vegetables.
33. **cons_ppp17** is the target variable, representing the daily per-capita expenditure adjusted for PPP17.

2.1 Preprocessing

2.1.1 Drop hhid, com and survey_id features

Firstly, the hhid, com and survey_id feature columns were dropped as they don't serve any other purpose than identification and would only hinder predictions. Following up, a check for missing values was performed in order to check the dataset and decide how to follow up on the preprocessing.

2.1.2 Handle Employment

Since the sector1d feature contains information about the sector where the household leader works and the employed characteristic contains their employment status, with some modification, sector1d can contain the information of employed by creating and adding a new "Unemployed" value to it and dropping the employed column. The sector1d values are not null only when the employed feature has a value of "Employed", so it can be assumed that when it is null they are unemployed and thus the new value can replace those entries in the data.

2.1.3 Handle Missing Values

To address the presence of null values in critical columns like dweltyp, sector1d, and educ_max, a localized imputation strategy is implemented within the impute_by_region function, where instead of utilizing global means or modes, which could obscure geographic and socioeconomic variations, the solution constructs a temporary region_str identifier by concatenating all regional indicators. By grouping the data by both this regional string and the strata, categorical missing values are filled using the most frequent value (mode) specific to that sub-group. This methodology ensures that missing household attributes are replaced with values common to their immediate neighbours and social circle.

For numerical expenditure data, specifically utl_exp_ppp17, the implementation utilizes a median-based imputation grouped by region and strata. This mitigates the influence of outliers common in consumption data and to guarantee dataset completeness, a secondary global median fallback is applied if a specific regional group lacks sufficient data to calculate a local statistic.

Furthermore, a domain-specific imputation rule is applied to the share_secondary feature via the handle_education function, where rather than relying solely on statistical averages, missing values are first inferred based on the household leader's maximum education level (educ_max).

- **Logical Zero Imputation:** For households where the leader has "Never attended," "Incomplete Primary," or "Incomplete Secondary" education, it is logically consistent to assume the share_secondary (share of household members with secondary education) is 0.
- **Median Fallback:** For remaining missing cases where this logical inference does not apply, the global median is used to preserve the variable's distribution without introducing bias. This two-step process leverages known semantic relationships between features to improve data quality before modelling.

After these functions are defined the full preprocessing happens where they are all run in sequence.

2.2 Categorical Encoding and Feature Alignment

To prepare the dataset for regression analysis, **One-Hot Encoding** was applied to all categorical variables. A specific focus was placed on the **strata** variable, which was first explicitly converted from a numerical to a categorical data type. By encoding **strata** alongside the other text-based features, we ensured that the models treated each sampling group as a distinct entity rather than a continuous number, preventing errors where "Strata 5" would be interpreted as having five times the magnitude of "Strata 1."

2.2.1 Feature Schema Alignment

Following the encoding process, a synchronization step was performed to align the training and test datasets. Since different splits might contain different subsets of categories, the test set was **reindexed** to match the training set's feature schema exactly. Any dummy columns representing categories (such as specific strata or regions) that existed in the training data but were absent in the test data were created and filled with zeros, ensuring identical input dimensionality for the models.

3 Train-Test Split

Before initializing the algorithms, the complete feature set underwent normalization using a **StandardScaler**. This transformation centers the data distribution around a mean of zero with a standard deviation of one, effectively removing magnitude disparities between variables. This step is mathematically critical for the **Lasso** and **Multi-Layer Perceptron (MLP)** models, which are highly sensitive to input scales and can fail to converge or exhibit bias toward larger values if the data is not standardized.

Following scaling, the dataset was partitioned into training and validation subsets using a **90/10 split** ratio. A fixed random state 42 was utilized to ensure the deterministic nature of this split, allowing for exact reproducibility of the results. Crucially, the survey **weights** were partitioned in parallel with the feature matrix and target vector. This synchronization ensured that the loss functions during training and the final error metrics accurately reflected the true population distribution, rather than treating every survey respondent as equally significant.

4 Model Implementation and Hyperparameter Tuning

4.1 Random Forest Regressor

The first algorithm implemented was the **Random Forest Regressor**, selected for its robustness in handling non-linear relationships and high-dimensional data. To ensure the model generalized well to unseen data, a rigorous **cross-validation process** was executed to identify the optimal hyperparameters. This search focused on balancing the number of estimators with tree depth to minimize variance without inducing overfitting. The optimal parameters that were found are as follows: Best Random Forest Params: {'n_estimators': 100, 'min_samples_leaf': 5, 'max_depth': None}

During the training phase, the model explicitly incorporated the survey **weights**. This ensured that the learning process prioritized households based on their actual representativeness in the population rather than treating every sample equally. The evaluation yielded the following results:

RF Validation MAE: 3.2101

RF Validation MAPE: 33.35%

Feature importance analysis highlighted variables such as utility expenses and household size as critical predictors.

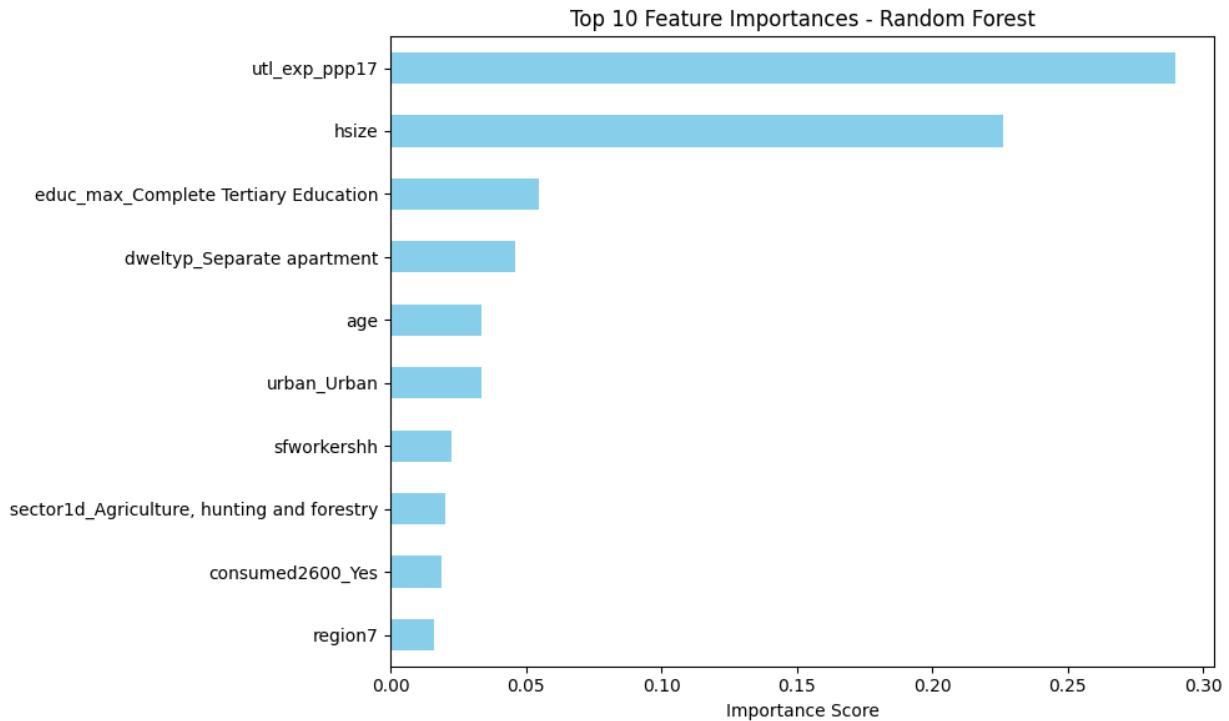


Figure 4-1 Random Forest Feature Importance

4.2 XGBoost Regressor

The second approach utilized **XGBoost (Extreme Gradient Boosting)**, a powerful ensemble method that builds decision trees sequentially and unlike Random Forest, where trees are independent, each new tree in XGBoost attempts to correct the residual errors of its predecessors. Key hyperparameters governing the learning rate, tree depth, and regularization were optimized via **cross-validation** to prevent the model from memorizing noise in the training data, with the results being as follows: Best XGBoost Params: {'tree_method': 'hist', 'n_estimators': 1000, 'max_depth': 6, 'learning_rate': 0.05, 'device': 'cuda'}

Similar to the other models, the training was weighted to align with the survey's population design. This configuration proved highly effective, outperforming the Random Forest model with the following evaluation results:

XGBoost Validation MAE: 3.0509

XGBoost Validation MAPE: 30.88%

The feature importances from this model were as follows:

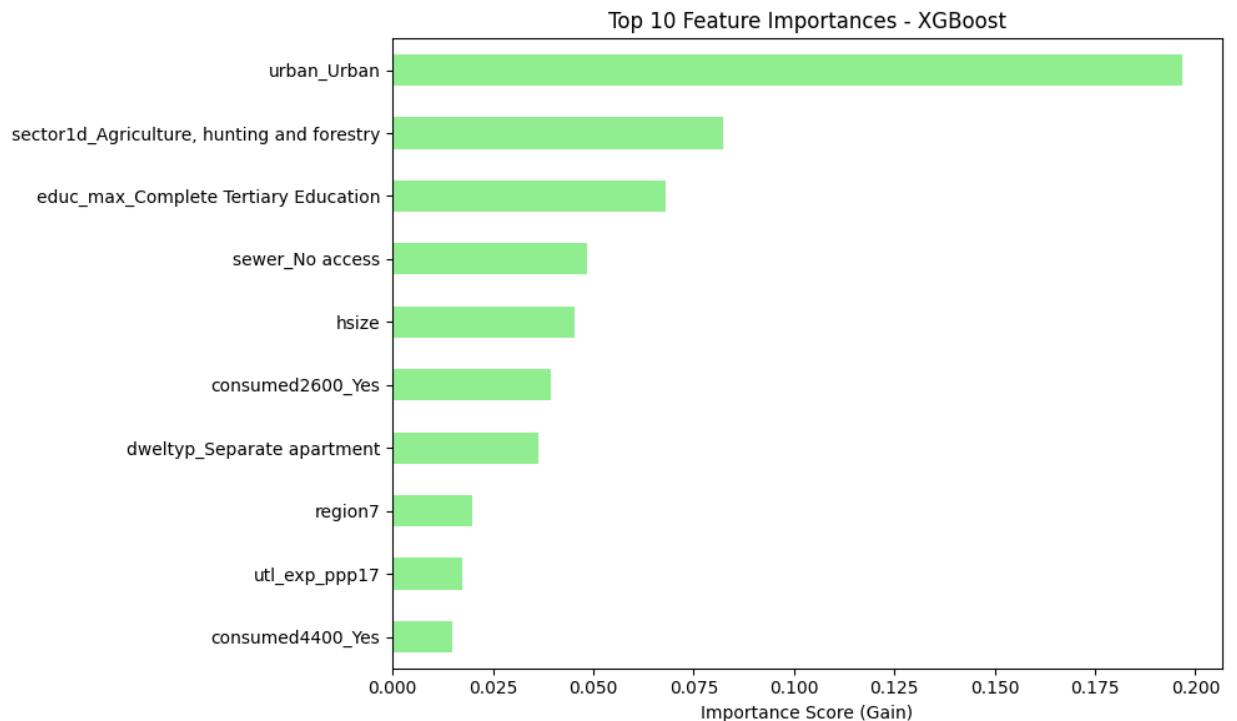


Figure 4-2 XGBoost Feature Importance

4.3 Lasso Regression

To provide a linear baseline with intrinsic feature selection capabilities, a **Lasso Regression** model was deployed. This algorithm applies an L₁ regularization penalty, which shrinks the coefficients of less relevant variables to zero, effectively filtering out noise and reducing dataset complexity. Before training, all features were standardized to ensure that the regularization penalty was applied uniformly across variables with different magnitudes. The regularization strength (alpha) was strictly determined through a **5-fold Cross-Validation (LassoCV)** procedure, which returned an alpha of 0.004968064868881013. While effective at feature reduction, the model showed limited flexibility compared to non-linear methods, producing the following evaluation results:

Lasso Validation MAE: 3.7181

Lasso Validation MAPE: 42.42%

The feature importances from this model were as follows:

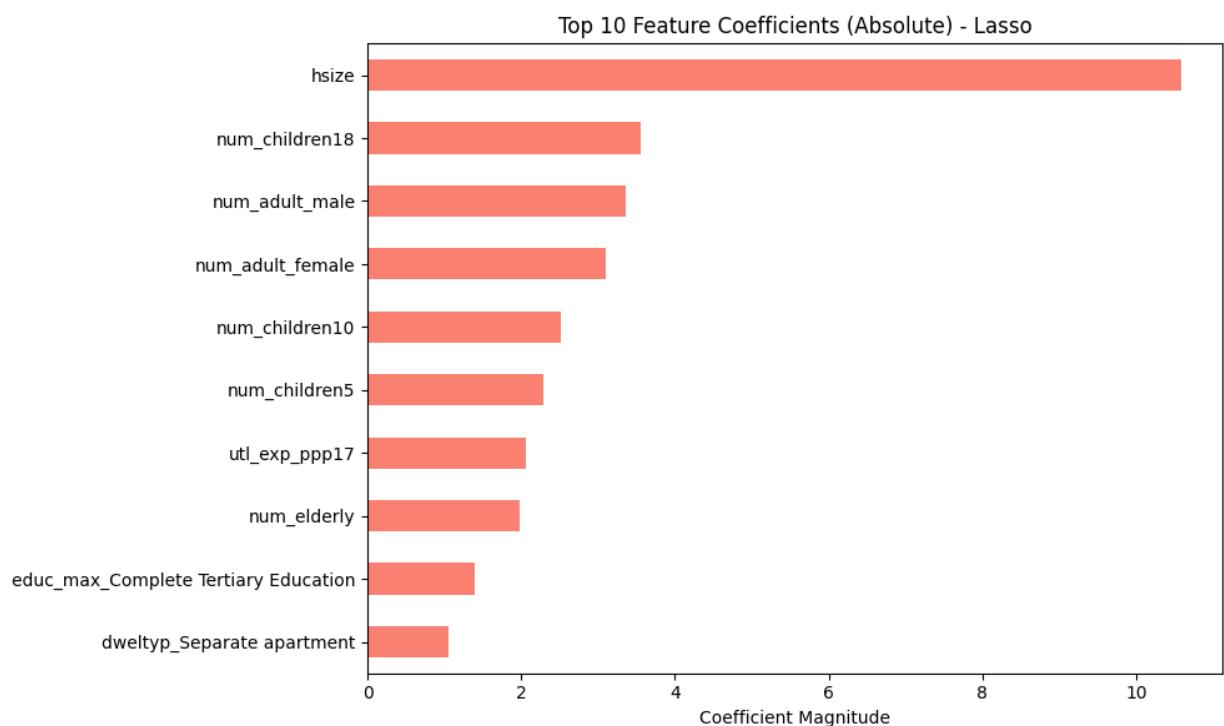


Figure 4-3 Lasso Regression Feature Importance

4.4 Multi-Layer Perceptron (MLP)

The final and most effective architecture implemented was a **Multi-Layer Perceptron (MLP)**. This deep learning model was designed to capture complex, high-order interactions between variables that traditional regression trees might miss. The network architecture consists of an input layer followed by a series of dense hidden layers.

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 512)	62,976
batch_normalization (BatchNormalization)	(None, 512)	2,048
dropout (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 256)	131,328
batch_normalization_1 (BatchNormalization)	(None, 256)	1,024
dropout_1 (Dropout)	(None, 256)	0
dense_2 (Dense)	(None, 128)	32,896
dropout_2 (Dropout)	(None, 128)	0
dense_3 (Dense)	(None, 1)	129

```

Total params: 230,401 (900.00 KB)
Trainable params: 228,865 (894.00 KB)
Non-trainable params: 1,536 (6.00 KB)

```

Figure 4-4 MLP Architecture

To ensure training stability and mitigate overfitting, **BatchNormalization** and **Dropout** layers were integrated into the design. The model was trained using the Adam optimizer, with the learning process monitored to identify the optimal number of epochs where validation loss stabilized. The results are as follows:

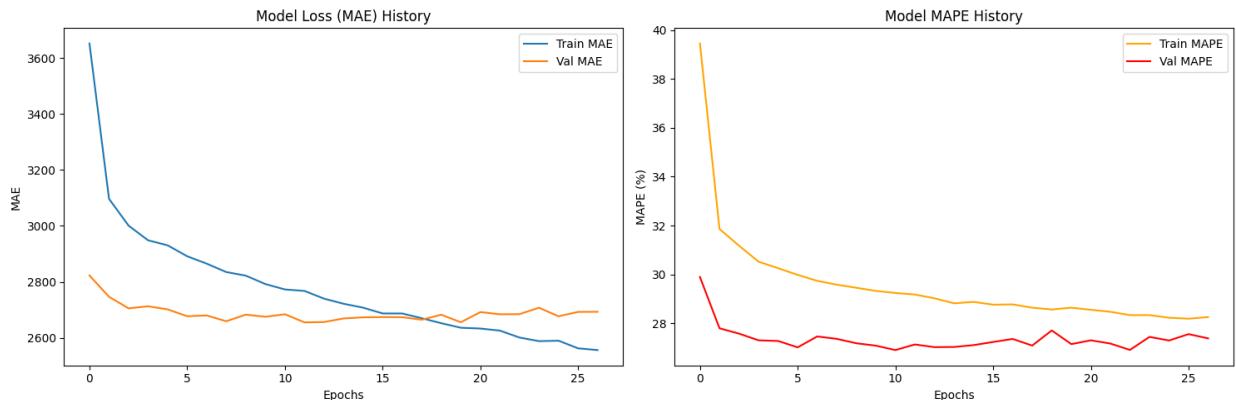


Figure 4-5 MLP MAE and MAPE History

The MLP model far surpasses the rest, as expected and, as such, was the final submission. Following are various comparison graphs between the various models to better gauge their abilities when compared to each other closely.

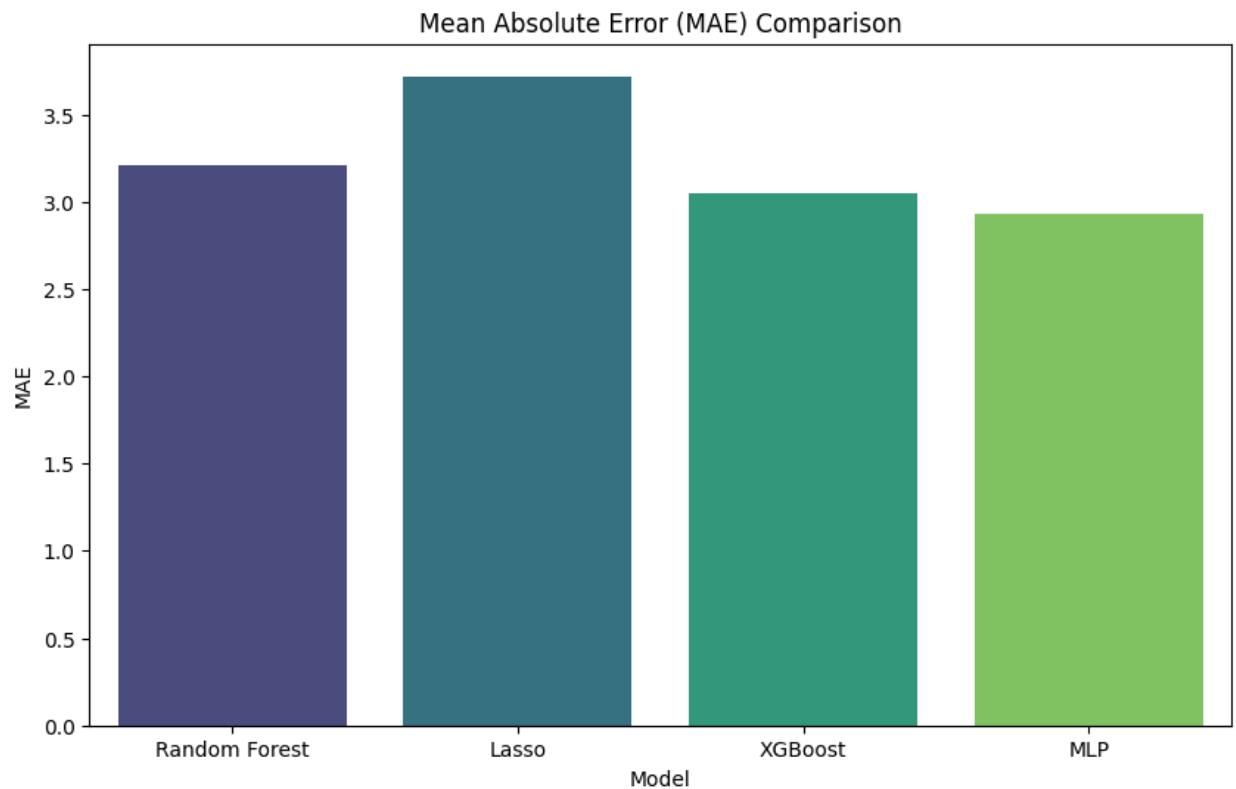
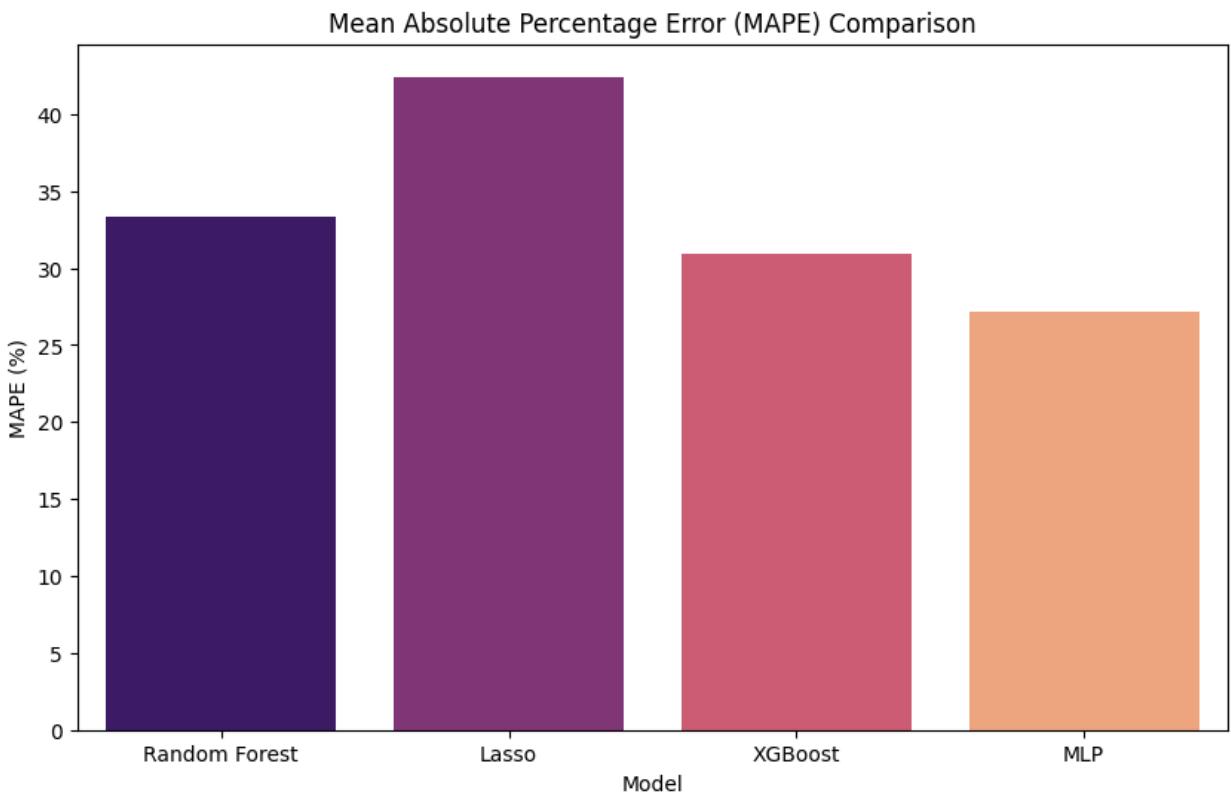


Figure 4-6 MAE Comparison



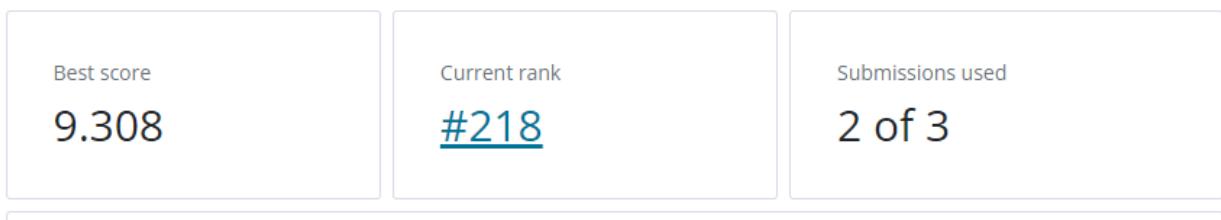
[Figure 4-7 MAPE Comparison](#)

5 Prediction Pipeline and Submission Generation

To generate the final competition submissions, the `test_hh_features.csv` dataset underwent a preprocessing pipeline identical to that of the training data. This ensured that the feature distributions remained consistent for inference. The process involved the rigorous imputation of missing values, the one-hot encoding of categorical variables, and the specific alignment of feature columns to match the training schema.

Non-predictive identifiers, including `hhid`, `com`, and `survey_id`, were temporarily excluded to isolate the mathematical feature matrix. Predictions were then generated using the four trained models, and the resulting values were mapped back to their respective household IDs. These results were exported into the specific CSV format required for the competition leaderboard.

The final result achieved from the MLP model was as follows:



[Figure 5-1 Competition Result](#)

6 Conclusion

The comparative analysis of the competition results highlights a significant performance gap between deep learning and traditional regression techniques. The **Multi-Layer Perceptron (MLP)** was the undisputed superior architecture, achieving the highest ranking on the leaderboard. This demonstrates that the dataset contains highly complex, non-linear dependencies that simple algebraic formulas cannot effectively capture.

In contrast, the **Lasso Regression** model failed to provide competitive results. While its feature selection capability successfully reduced noise, the high error rate confirms that assuming a linear relationship between household attributes and consumption is fundamentally flawed for this economic data. Similarly, the tree-based ensembles—**XGBoost** and **Random Forest**—struggled significantly, likely suffering from overfitting to the training split and failing to generalize global economic trends to the test population. The results conclusively show that only a deep neural network had the capacity to map the intricate socio-economic patterns required for accurate prediction.

6.1 Ways to Improve Data Quality

The current dataset relies exclusively on household-specific survey responses, often ignoring the broader economic environment. Integrating external **geospatial features** would provide critical context:

- **Market Access:** Calculating the distance to the nearest urban centre or major market.
- **Infrastructure:** Analysing road density within the specific region.
- **Satellite Data:** Utilizing "nightlight intensity" data as a proxy for regional economic activity and development.

Furthermore, the current dataset provides a static "snapshot" of the household, which can be misleading depending on the time of year the survey was conducted. Consumption patterns—especially in agricultural or developing regions—fluctuate wildly between harvest seasons and "lean" seasons.

- **Time-Series Data:** Collecting data across multiple months would smooth out temporary spikes in spending.
- **Seasonality Flag:** Adding a feature indicating the specific **month of interview** would allow the model to adjust for seasonal heating costs or holiday-related spending surges.

Lastly, in modern economic modeling, digital behavior is often a stronger predictor of wealth than reported income. Including data on **mobile phone usage**, airtime expenditure, or internet data consumption would provide a high-fidelity proxy for disposable income. Households with consistent digital connectivity almost invariably belong to higher consumption strata, even if their reported employment status is informal or missing.