End Of Spring Quarter Report

Lana Huynh, Kyle Lew, Dylan Li, Isabella McCarty

World Bank MENA Social Change

DATA 451

12 June 2024

Table of Contents

Background

"Poverty is relatively cheap to address and incredibly expensive to ignore," notes Clint Borgen, a leading poverty reduction campaigner in the United States. In the MENA region, where coverage of household surveys has significantly dropped from 82% in 2012 to 45% in 2019, the need for accurate poverty estimation is more critical than ever. Conducting these surveys is often hampered by governmental conflicts and resource shortages. The World Bank's current poverty estimates for these countries lack necessary spatial and temporal detail, which hampers the effectiveness of policy-making and strategic intervention. This is problematic as accurate poverty measurement is not only costly but also infrequent, which limits the ability to track progress and provide aid, promptly, to worsening conditions. Our project aims to determine if public geospatial information, when integrated with machine learning techniques, can provide a reliable alternative to traditional surveys by predicting poverty levels across MENA. The main challenge lies in balancing the tradeoff between enhancing spatial and temporal granularity and ensuring the model's accuracy and broad coverage.

Methodology

We aim to integrate various datasets and employ machine learning techniques to estimate poverty levels across different regions. Our approach involves multiple steps, including data merging, fuzzy matching, utilizing advanced explanatory variables, and building the machine learning model.

Poverty metrics

For poverty estimation, it is important to note that there are two main metrics that are widely used to quantify “wealth”. Firstly, the amount of consumption for a household is used to estimate poverty. If the amount a household spends per day/month/year is below some threshold (eg. $2.15 per capita per day), then that certain household is classified as poor. The second main metric to estimate poverty is based on assets. A household is classified as poor if it owns fewer assets and has access to fewer services than some threshold.

Datasets

We are given a World Bank’s Household Survey with information containing poverty levels at $2.15, $3.65 and $6.85 per capita per day for various years in the 2000’s. The dataset contains administrative divisions one and two (ADM1 and ADM2), to specify the location of a region broadly and specifically, respectively. Additionally, we are provided a “Grid” which serves as a geographic area of MENA divided into smaller, manageable units for us to use for analysis. It also contains administrative divisions one and two in which it can be used as “ground truth” for the MENA region.

Fuzzy merging

To build our “ground truth” file, the household survey dataset with the corresponding poverty levels must first be merged onto the MENA grid which has a corresponding polygon shape to a specific location. However, due to the translational differences in values across the household survey and grid, there is not an exact match between the two ADM values. Thus, we need to employ a technique called “fuzzy matching” to merge the two datasets.

Fuzzy matching is essential for harmonizing data from diverse sources, such as the Grid and Household Survey datasets, which often contain inconsistencies in spelling and formatting. By identifying approximate matches between strings, fuzzy matching facilitates effective data integration and analysis. Our approach uses custom algorithms and Python libraries to match various administrative levels across the datasets, addressing discrepancies that traditional merge methods cannot handle.

We developed a 'fuzzy_match' function with three key parameters: 'value', 'choices', and 'threshold'. The 'value' parameter is the string to be matched, usually from the Household Survey data, while 'choices' are the strings to match against, typically from the Grid's administrative data. The 'threshold' sets the minimum score for a valid match, usually defaulting to 90 but adjustable for accommodating data quality issues. The function employs the 'extractOne' method from the 'fuzzywuzzy' library, leveraging the Levenshtein distance partial ratio to assess similarity.

During our iterative process, we adjusted the scoring mechanisms and experimented with various matching ratios to balance sensitivity and specificity. We combined different administrative levels (e.g., ADM1 and ADM2) for matching, and tested various threshold values to optimize the process. Each iteration took 10 to 15 minutes due to the processing time required for fuzzy merges and joins. This methodical approach ensured robust data reconciliation and integration, overcoming the limitations of conventional methods.

We completed the whole fuzzy merging process on only the country of Syria first due to the ease of the merge which resulted in a perfect one-to-one match. This allows us to begin the machine learning component of our project while continuing to refine the fuzzy merge process simultaneously.

As for the full MENA dataset merge, we had to perform a manual check on the ADM values that were matched to ensure that the locations were properly aligned. We noticed that our algorithm made some mistakes due to the similarities between regions, so we had to do research on the cities and countries. Afterwards, we sent this file to the WorldBank team, and they returned a file with the correct matches, ready for analysis.

As a result of the fuzzy merging algorithm and manual check, the final ground truth dataset contained information about the poverty levels, ADM1 and ADM2 locations, and the geometry value.

Explanatory variables

When thinking about what data to train our model on, there were many public geospatial datasets we highlighted that could be useful in predicting poverty levels. ACLED(conflicts), Nightlights, Weather, MOSAIKS(satellite), Population, MODIS(satellite), and Urban Areas were all the datasets we were considering. In the end, we implemented ACLED and MOSAIKS to train our Machine Learning models.

MOSAIKS

MOSAIKS("Multi Task Observation Using Satellite Imagery and Kitchen Sinks") is a dataset created by UC Berkeley and UC Santa Barabara, with the intention of making Satellite imagery data more accessible and effective. The satellite data was encompassed through 4000 numerical variables, all ranging from 0 to 1. Each of these variables(named X_0, X_1, X_3, etc), contained information describing some frequency of RGB wavelengths. This all encompassing satellite imagery dataset allowed us and the model to pick up on a variety of different geospatial information, such as forest cover, population density, nightlights, elevation, and much more.

When merging MOSAIKS onto our ground truth file, we ran into challenges given the spatial nature of this joining process. We performed the entirety of this merge in python, using the geopandas library to perform a spatial join. This provided the base for the merge, performing an inner join of the two datasets based on the spatial location(measured in latitude and longitude coordinates) of each row.

The first issue we faced when performing this spatial join was that the two datasets recorded spatial information differently. The rows in the ground truth dataset recorded spatial information through polygons, describing an area for each observation. Rows in the MOSAIKS dataset, on the other hand, recorded spatial information in specific latitude and longitude points. Given this discrepancy, we identified that we wanted our final merged dataset to record spatial information in polygons not points.

So to resolve this difference, we joined the MOSAIKS points onto the ground truth polygons, then performed an aggregation on the MOSAIKS rows to reduce them down to single values. More specifically, after joining the two datasets, when there existed multiple MOSAIKS points in the same ground truth polygon, we would average all the frequency values and record those averages as the final MOSAIKS values for that polygon. The following figure shows the first 6 of the MOSAIKS frequency values in Syria for the final merged dataset.

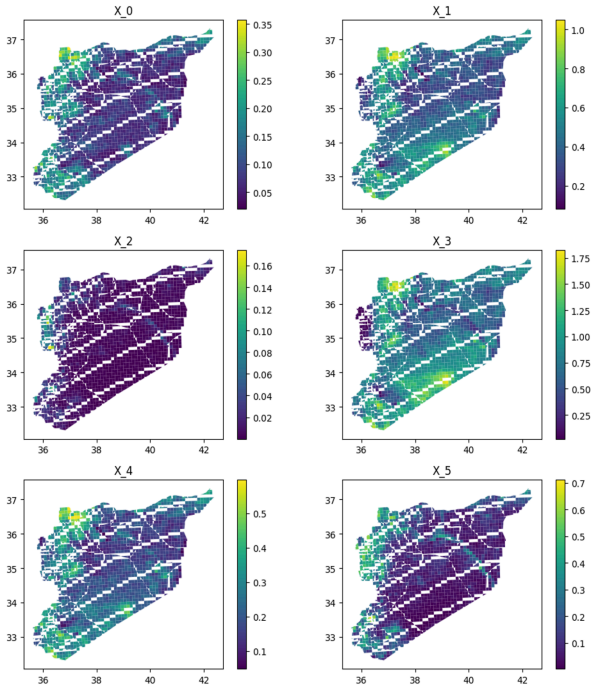

Figure 1: MOSAIKS values for Syria

ACLED

The second explanatory dataset we used to train our model was ACLED("Armed Conflict Location and Event Data"). Using this dataset, we were able to track information on conflicts in the MENA region.

When joining ACLED with our ground truth dataset, unlike the MOSAIKS merge, this merging process was made very simple because of a prematched "grid_id" variable that was present in both datasets. That being said, we still had to aggregate and feature engineer additional variables in order to prep the ACLED data. To do this, we reduced the expansive information in ACLED down to two variables, count of conflict events and total number of fatalities. Additionally, we tracked these two variables for 4 different time frames(1 year, 2 years, 4 years, 6 years). So for each grid, we tracked the count of conflicts in the past year, total number of fatalities in the past year, count of conflicts in the past 2 years, total number of fatalities in the past 2 years, and so on for the past 4 and 6 years.

Model creation

With the ground truth data and explanatory variables discussed above, we proceeded to apply machine learning techniques to predict poverty levels at the $3.65 threshold. To achieve this, we employed two distinct machine learning algorithms: Ridge Regression and Multilayer Perceptron Neural Networks. Ridge Regression, a linear regression method with an added parameter for regularization, is designed to handle high-dimensional data by preventing the model from being too closely tailored to the training data. This will ensure more reliable, interpretable results and is

particularly suited for our column-heavy MOSAIKS dataset. In contrast, the MLP Neural Network is an advanced model capable of learning complex patterns. We applied this modeling technique to both MOSAIKS and ACLED data sets as its flexibility makes it highly effective for large datasets, offering both high prediction accuracy and interpretabilitty of time-variant relationships.

To implement Ridge Regression, we designed an approach to prepare the data and build the model. First, we selected the relevant frequency columns (X_0…X_3999) from the dataset and handled missing values using imputation with average values. Categorical data was converted into numerical formal using one-hot encoding. The data was then split into training and testing sets, and a range of regularization strengths was tested to find the best fitting parameters. We trained the model using a cross-validation to prevent overfitting, made predictions, and calculated the accuracy using the Mean Squared Error (MSE). MSE represents the average squared difference between the predicted and actual values, with lower values indicating better model performance.

Following the Ridge Regression approach, we applied a Multilayer Perceptron (MLP) Neural Network using TensorFlow and Keras Tuner to optimize hyperparameters. We prepared the data similarly by selecting the relevant frequency columns, handling missing values with imputation, and converting categorical data into numerical format using one-hot encoding. The data was then split into training and testing sets. Using K-fold cross-validation with 5 folds, we evaluated the model and implemented early stopping to prevent overfitting. We performed hyperparameter tuning to optimize the number of neurons in each layer and the learning rate. Finally, we trained the best model, made predictions, and evaluated its performance using Mean Squared Error (MSE) and Mean Absolute Error (MAE), with lower values indicating better model performance.

Results

Here is part of the resulting equation for the ridge regression using the Syria with MOSAIKS dataset:

$$\hat{Poverty}_{h3651} = -0.37X_{1897} - 0.352X_{3104} - 0.333X_{2104} + 0.318X_{1037} - 0.295X_{3969}$$

Metric	*MOSAIKS Dataset*	*ACLED Dataset*
Mean Squared Error (MSE)	**0.0424**	**0.0503**
Mean Absolute Error (MAE)	**0.1767**	**0.1951**
R-squared (R²)	**0.0652**	**0.0109**

Table 1. Resulting Metrics from Syria Neural Network

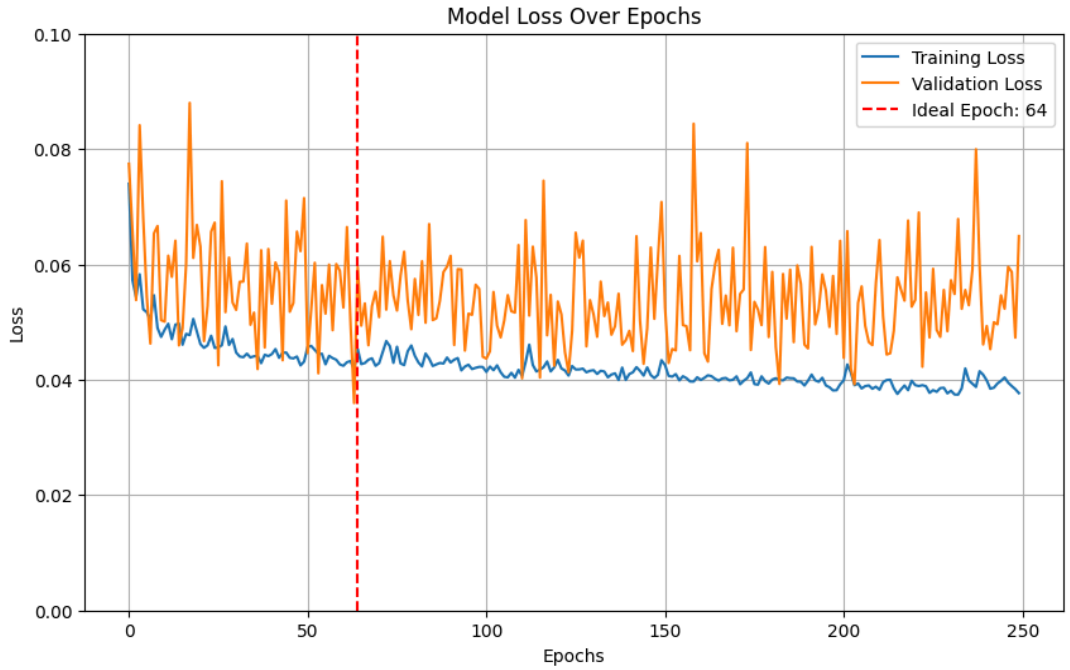

Figure 2. Model Loss Over 250 Epochs on MOSAIKS

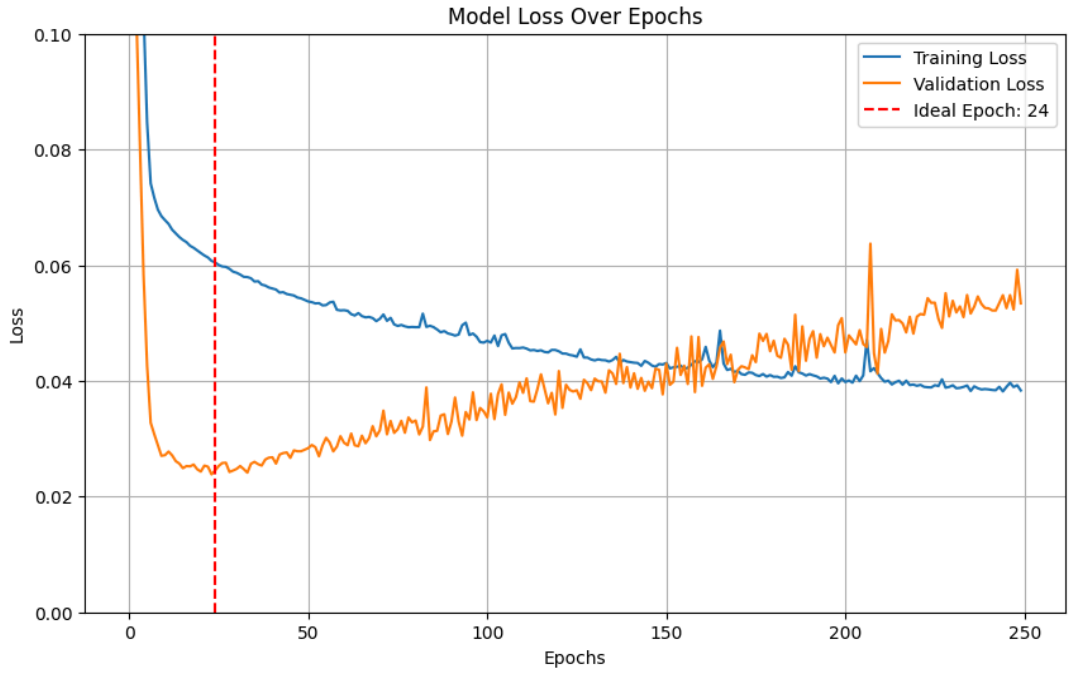

Figure 3. Model Loss Over 250 Epochs on ACLED

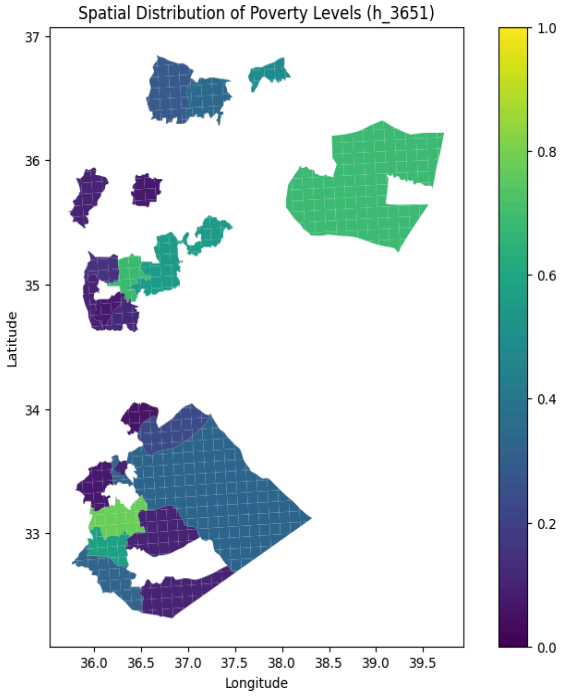

Figure 4. Actual Poverty Levels in Syria

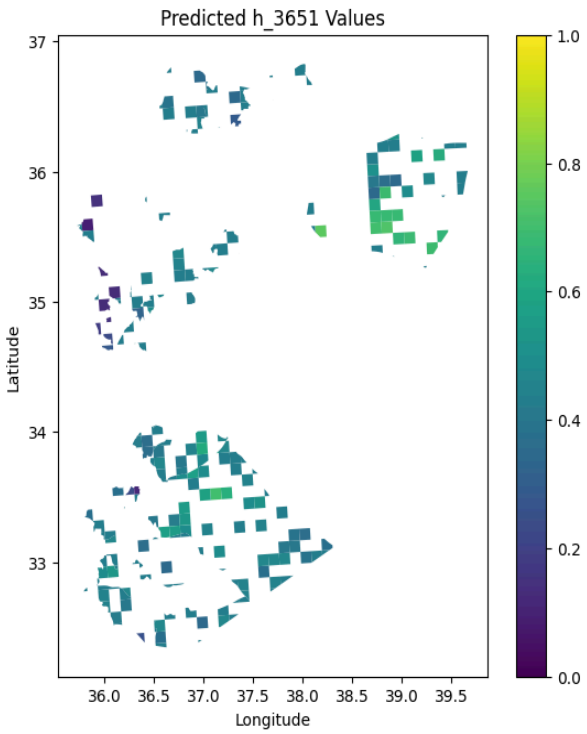

Figure 5. Predicted Poverty using MOSAIKS

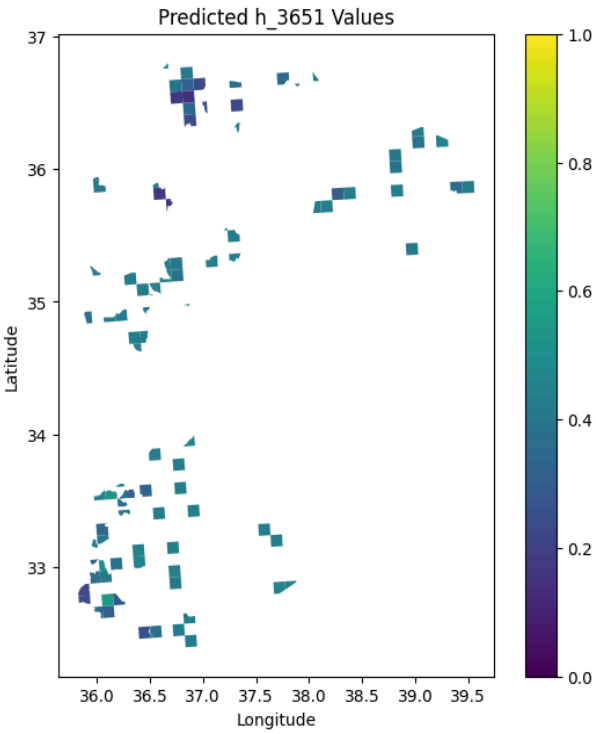

Figure 6. Predicted Poverty using ACLED

Discussion

Ridge Regression Model

For our Ridge Regression model trained on only MOSAIKS, the metric we used to evaluate the Mean Squared Error (MSE), the value for this metric on the test set was 0.035, indicating a small discrepancies between our model's predictions and the actual poverty levels. To determine the best fit of the model, we used cross-validation to find the best alpha value, which controls the model's complexity and helps prevent overfitting. Through cross-validation, we determined that the best alpha value for our model is 0.1.

Neural Network Model

Looking at Table 1, we can evaluate the effectiveness of our neural network model for each dataset. For the MOSAIKS dataset, the MSE is 0.0424, indicating a better fit compared to the ACLED dataset, which has an MSE of 0.0503. Additionally, the MOSAIKS dataset has an MAE of 0.1767, while the ACLED dataset has an MAE of 0.1951. Again, lower values indicate more accurate predictions. Finally, for the MOSAIKS dataset, the R^2 value is 0.0652, meaning the model explains about 6.52% of the variance in the data. For the ACLED dataset, the R^2 value is 0.0109, indicating the model explains about 1.09% of the variance.

The Model Loss Over Epoch graphs illustrate the MOSAIKS NN model's performance over time

as it learns from the data. In Figure 2, the blue line (training loss) steadily decreases, indicating that the model is getting better at predicting the training data. The orange line (validation loss) fluctuates but generally decreases, showing how well the model predicts new, unseen data. The red dashed line marks the ideal point to stop training (at epoch 64) to prevent the model from becoming too specialized to the training data and losing its ability to generalize to new data. Overall, the model learns effectively, it is imperative to ensure it performs well on new data.

Figure 3 illustrates the ACLED NN model's performance over 250 epochs, showing both training loss (blue line) and validation loss (orange line). Initially, both losses decrease rapidly, indicating that the model is learning effectively from the data. However, after the ideal stopping point at epoch 24 (marked by the red dashed line), the validation loss starts to increase and fluctuate, while the training loss continues to decrease. This suggests that beyond this point, the model begins to overfit to the training data, losing its ability to generalize well to new, unseen data.

Looking at Figure 4, Figure 5, and Figure 6, we can visualize how our Neural Network model performed at predicting poverty in Syria, when trained on MOSAIKS vs ACLED. The gaps in data in Figure 5 and 6 are there because of the train/test split, showing only the 30% of data where the models predicted poverty. When comparing Figures 5 and 6 to Figure 4, we see that the predicted values using MOSAIKS performed slightly better, reinforcing the results we saw above with the various test metrics. Specifically in the top right of Figure 5, we see the model using MOSIAIKS predicted very closely to the actual predicted values.

Overall from metrics predicted poverty graphs, it’s clear that our neural network model performed better when trained on MOSAIKS. We think this is because the MOSAIKS dataset, with its higher dimensionality and detailed information, allows the model to perform slightly better than with the ACLED dataset.

Conclusion and Future Work

Overall, we effectively used public geospatial data and machine learning algorithms to estimate poverty levels. By integrating the World Bank’s household survey data with the grid, MOSAIKS, and ACLED, and applying models like Ridge Regression and Multilayer Perceptron Neural Networks, we created a scalable and expandable framework.

The results indicated that the neural network model performed better with the MOSAIKS dataset compared to the ACLED dataset, as shown by lower Mean Squared Error (MSE) and Mean Absolute Error (MAE) values. This suggests that the high-dimensional and detailed nature of satellite imagery data provides more predictive power for poverty estimation.

To improve predictions of poverty levels in the MENA region, we can take two main approaches. The first approach is to increase the number of explanatory variables used in the

machine learning models. Currently, our team has included variables from two datasets: MOSAIKS and ACLED. Adding more geospatial variables, such as climate data or nightlight data, could enhance the accuracy of our models. In addition to acquiring these new datasets, they must be cleaned and integrated onto the grid using either polygon or geometry variables.

To improve predictions of poverty levels in the MENA region, we should experiment with different machine learning models. Our team has tested ridge regression and multilayer perceptron (MLP) neural networks, which show promise with our limited data. However, exploring alternative models could be a valuable investment of time. Before further neural network experiments, we recommend examining principal component analysis (PCA) due to its effectiveness with high-dimensional datasets like MOSAIKS. Additionally, testing the effectiveness of random forest models for poverty level prediction is suggested.

Another area for improvement is the merging process of household survey poverty level data onto the grid. Currently, we use a fuzzy merge process to match ADM1 and ADM2 values from the household survey to the grid, followed by manual verification. This method is time-consuming and resource-intensive. Ideally, future household survey data should record ADM1 and ADM2 labels in a format consistent with the grid. If that is not feasible, developing a word bank for spelling matches could significantly reduce the time spent on this process.

To enhance the accuracy and applicability of our poverty prediction models, we recommend expanding the analysis to include the entire MENA region, not just Syria. This broader scope will enable a more comprehensive evaluation of our machine learning models.First, the project team should merge the complete MOSAIKS and ACLED data sets with the full grid. Although we have obtained these datasets for the entire MENA region, we have yet to perform the merging. Next, they must integrate the comprehensive household survey dataset for the entire MENA region onto the completed grid. This dataset, which has been fully fuzzy merged, is available on AWS cloud servers.

Once the grid includes poverty levels, MOSAIKS, and ACLED data for the entire MENA region, the team can begin the analysis by integrating this completed grid into the existing machine learning framework. Expanding the analysis in this manner will provide a more accurate and regionally comprehensive evaluation of poverty levels.Upon completing the full analysis of the MENA region, we recommend focusing on two areas for future work. First, continue testing new machine learning models and incorporating additional explanatory variables to enhance the model's accuracy and scope. Second, develop a more efficient method for merging future household survey datasets onto the grid to streamline the primary bottleneck of this project.

Overall, the next steps involve performing a comprehensive analysis on the grid for the entire MENA region. Following this, it is recommended to enhance our model by incorporating

additional explanatory variables and experimenting with various methods to refine the machine learning predictions. Additionally, to streamline future processes, we suggest improving the fuzzy merging process of household survey data onto the grid.

Appendix

Ridge Regression Details:

I. **Data Preparation:**
   - A. **Column Selection:**
     1. Selected columns starting with 'X_' from the dataset to focus on the relevant features for the model. This was done to simplify the data and ensure that only the necessary columns are used for training.
   - B. **Predictors (X) and Target Variable (Y):**
     1. Split data into predictors (X) and target variable (Y) to separate the input features from the outcome we aim to predict. This helps in building a clear and structured model.
   - C. **Categorical Columns Identification:**
     1. Identified categorical columns for separate processing to handle them appropriately during preprocessing. This is essential because categorical data needs to be converted into a numerical format for the model to process it effectively.

II. **Preprocessing:**
   - A. **Missing Value Imputation:**
     1. Used `SimpleImputer` to fill missing values in numerical columns with their mean. This choice was made because using the mean value is a simple and effective method to handle missing data without introducing significant bias.
   - B. **One-Hot Encoding:**
     1. Applied `OneHotEncoder` to convert categorical data into a numerical format. One-hot encoding is a standard method to convert categorical variables into a format that can be provided to ML algorithms to do a better job in prediction.
   - C. **Column Transformer:**
     1. Combined both preprocessing steps using `ColumnTransformer` to streamline the preprocessing pipeline. This approach ensures that the preprocessing steps are applied consistently and efficiently to the correct columns.

III. **Model and Cross-Validation:**
   - A. **Alpha Values Definition:**
     1. Defined a range of alpha values (regularization strengths) for the Ridge Regression model using `np.logspace`. This wide range allows the model to find the optimal level of regularization, balancing between overfitting and underfitting. For example, our alpha level of 0.1 means the model applies a moderate amount of regularization to the regression coefficients, reducing the impact of less important features.
   - B. **Pipeline Creation:**
     1. Created a pipeline with the preprocessing steps and the `RidgeCV` model, which includes cross-validation to find the best alpha. Pipelines help in automating the workflow and ensuring that all steps are correctly applied in sequence.

IV. **Training and Evaluation:**
   - A. **Data Splitting:**
     1. Split the data into 70% training and 30% testing sets using `train_test_split`. This ensures that the model is trained on a substantial portion of the data while still having a separate set for unbiased evaluation.
   - B. **Model Training with Cross-Validation:**

1. Trained the Ridge Regression model on the training data with cross-validation to prevent overfitting. Cross-validation using scikit-learn functions ensures that the model's performance is robust and generalizes well to unseen data.

C. Prediction and Performance Evaluation:

1. Predicted poverty levels on the test data to evaluate the model's performance.
2. Calculated the Mean Squared Error (MSE) to measure the model's accuracy. MSE indicates how close the predicted values are to the actual values, with a lower MSE suggesting a better fit. Specifically, it is the average of the squares of the errors, where the error is the difference between the predicted and actual values. This metric was chosen because it penalizes larger errors more heavily, providing a clear indication of the model's predictive accuracy.

Neural Network Details (ACLED AND MOSAIKS):

I. **Data Preparation:**

A. **Column Selection:**

1. Selected columns starting with 'X_' from the dataset to focus on the relevant features for the model. This ensures that only the necessary columns are used for training, simplifying the data.

B. **Predictors (X) and Target Variable (Y):**

1. Split data into predictors (X) and target variable (Y) to separate the input features from the outcome we aim to predict.

C. **Categorical Columns Identification:**

1. Identified categorical columns for separate processing to handle them appropriately during preprocessing. This is essential because categorical data needs to be converted into a numerical format for the model to process it effectively.

II. **Preprocessing:**

A. **Missing Value Imputation:**

1. Used `SimpleImputer` to fill missing values in numerical columns with their mean. This choice was made because using the mean value is a simple and effective method to handle missing data without introducing significant bias.

B. **One-Hot Encoding:**

1. Applied `OneHotEncoder` to convert categorical data into a numerical format. One-hot encoding is a standard method to convert categorical variables into a format that can be provided to ML algorithms to do a better job in prediction.

C. **Column Transformer:**

1. Combined both preprocessing steps using `ColumnTransformer` to streamline the preprocessing pipeline. This approach ensures that the preprocessing steps are applied consistently and efficiently to the correct columns.

III. **Model and Hyperparameter Tuning:**

A. **Model Building Function:**

1. Defined a function `build_model` to create the neural network architecture. This function includes layers with varying numbers of neurons (units) and uses ReLU activation functions to introduce non-linearity and prevent the vanishing gradient problem.

B. **Hyperparameter Tuning:**

1. Used Keras Tuner's `Hyperband` to search for the best hyperparameters, such as the number of neurons in each layer and the learning rate. The goal was to minimize the

validation mean squared error (MSE). For example, the best hyperparameters found for both ACLED and MOSAIKS were 512 units in the first layer, 64 units in the second layer, 320 units in the third layer, and a learning rate of 0.0001.

IV. **K-Fold Cross-Validation:**
   A. **K-Fold Splitting:**
      1. Applied K-fold cross-validation with 5 folds to evaluate the model's performance on different subsets of the data. This helps in ensuring that the model generalizes well to unseen data.
   B. **Early Stopping:**
      1. Used early stopping to halt training when the model's performance on the validation set stopped improving. This prevents overfitting by ensuring the model does not train for too many epochs.

V. **Training and Evaluation:**
   A. **Data Splitting:**
      1. Split the data into training and testing sets using `train_test_split`. This ensures that the model is trained on a substantial portion of the data while still having a separate set for unbiased evaluation.
   B. **Model Training:**
      1. Trained the best model on the training data, and used a validation split to monitor its performance. This step ensures that the model's hyperparameters are fine-tuned for optimal performance.
   C. **Prediction and Performance Evaluation:**
      1. Evaluated the model's performance on the test set using Mean Squared Error (MSE) and Mean Absolute Error (MAE). MSE indicates the average squared difference between the predicted and actual values, with lower values suggesting a better fit. MAE represents the average absolute difference between the predicted and actual values, providing another measure of accuracy.
      2. Made predictions on the test set to assess how well the model generalizes to new data.
      3. The average validation loss across the 5 folds was 0.05145295262336731, indicating the model's performance during cross-validation. Lower loss values suggest better performance.