Predicting Life Expectancy
Dana McGowan
Brown University Data Science Initiative
https://github.com/danamcgowan/Data1030-LifeExpProject.git

1 Introduction/Purpose

Life expectancy is a highly studied and important statistic across the world. When we think of improving our own life expectancy most of us think about personal health markers such as blood pressure, heart rate, and activity levels. However, this study focuses on the impacts of government led factors, such as immunization rates, human development levels, and economic measures.

The purpose of this project is to predict the overall life expectancy of a previously unseen country given the countries overall health levels, immunization rates, and human development statistics. This project will not only give insights into the life expectancy of a country, but it will also provide information on the factors that are most important to determining and increasing life expectancy.

The dataset was pulled together by information from the World Health Organization (WHO) data repository and corresponding economic information from the United Nations (UN) website. The dataset consists of 193 countries, 183 of those countries have data points from 2000-2015 and 10 of them only have 1 entry of data for the year 2013. There are 22 features that make up this dataset; two of the features are simply the name of the county and year of the information, six of the features are related to the economic/developmental status of the country, and the other 13 features are related to the overall health of the country [1].

2 Explanatory Data Analysis (EDA)

The purpose of EDA was to understand the spread of the target variable, the imbalances presented within the dataset, and the correlations between the features. I used .describe() to provide information about the mean and standard deviation of the data. This revealed that the mean life expectancy is 69.2 years and the standard deviation is 9.5 years. Next, I created a histogram to show the spread in the target variable. In figure 1, we can see that life expectancy ranges from about 36 to 89 years old, with the most common life expectancy age being around 74 years.

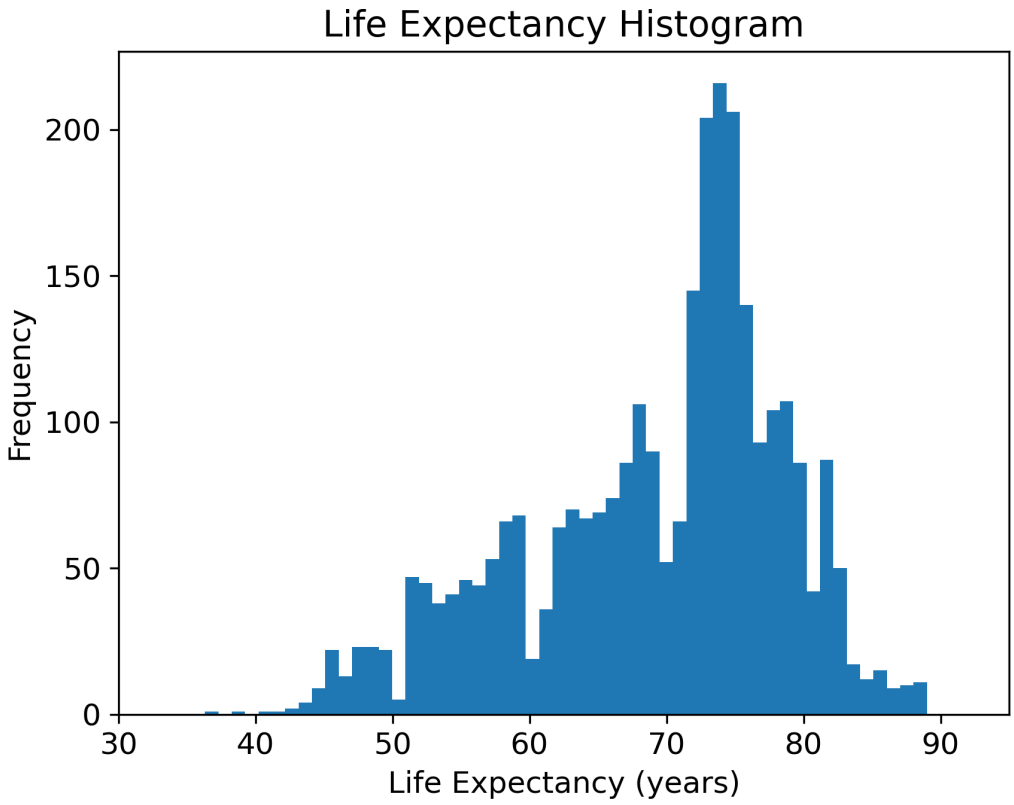

Figure 1: Histogram on the target variable, life expectancy.

In the dataset there are only two categorical features: country and status. Status refers to the development of each country, where each country can either be developed or developing. I used .value_counts() to understand the imbalances in the status feature, and plotted the results in the below pie chart. Only 20% of all countries in the dataset are considered developed, while the other 80% are developing.

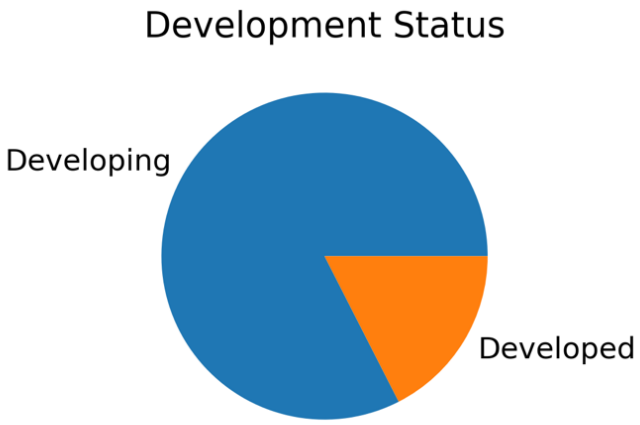

Figure 2: Pie chart showing the disparity in the data between developed and developing countries.

The boxplot below shows that the status feature has a significant effect on the life expectancy in a given country. In figure 3 we can see that the mean life expectancy of a developed country is about ten years longer then the mean of a developing country, and the minimum life expectancy of a developed country is higher then the mean of a developing country.

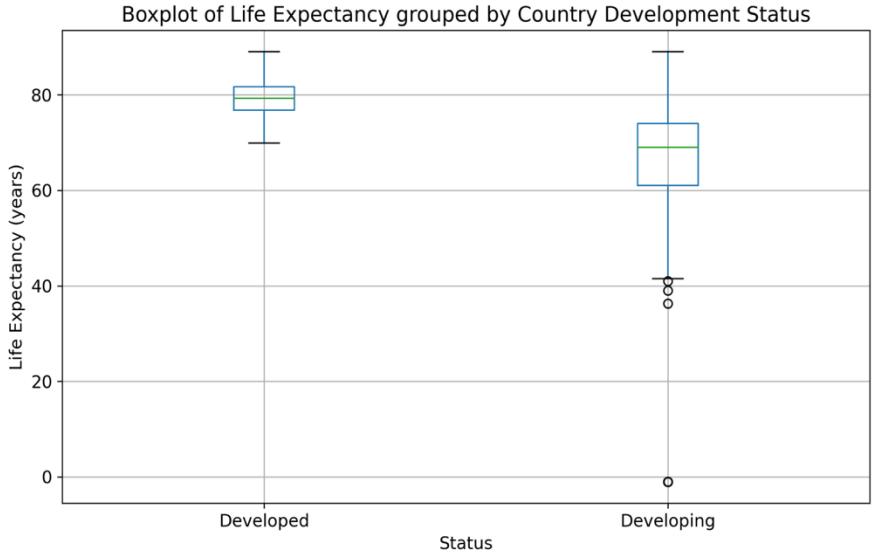

Figure 3: Boxplot for the status feature.

Heatmaps were used to show the relationship between life expectancy and all other continuous variables in the dataset. Figure 4 shows the relationship between life expectancy and polio immunizations. Figure 5 shows the relationship between life expectancy and diphtheria immunizations. Although these two diseases and vaccinations are very different, their effects on life expectancy appear to be almost identical. As the percentage of immunized infants falls, so to does the life expectancy of the country.

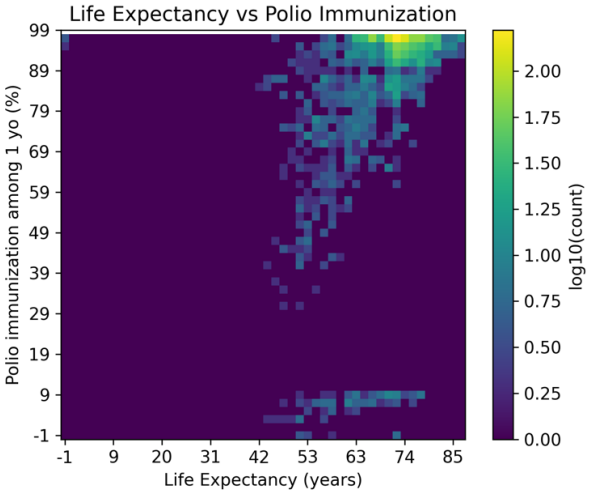

Figure 4: Heatmap of Life Expectancy vs % of 1 years old's immunized for Polio.

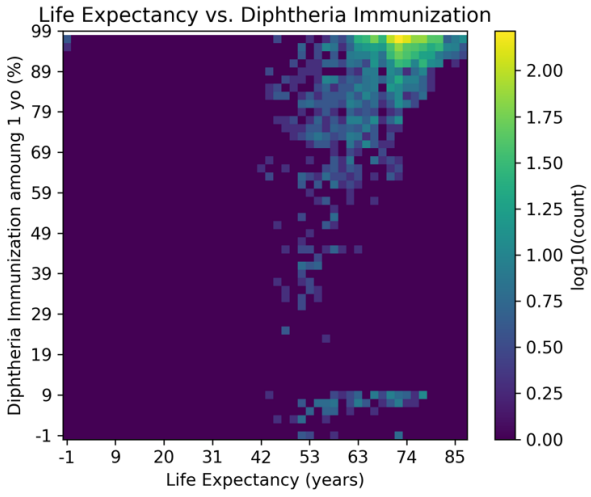

Figure 5: Heatmap of Life Expectancy vs % of 1 year old's immunized for Diphtheria.

3 Methods

3.1 Data Splitting

The dataset has a group structure, for this reason *GroupShuffleSplit* method was used to form the test and other sets with a ratio of 2:8. The other set was then split using *GroupKFold* splitting with 4 folds to create the validation and test sets. The final training set contained 114 countries, the test set contained 39 countries, and the validation set contained 40 countries.

3.2 Preprocessing

There are two categorical features in the dataset, country and status. Both of these features were preprocessed using *OneHotEncoder*, which forms a new column for each unique category in each feature. The *MinMax* transformer was used on nine continuous features that could be reasonably bound. Finally, the *StandardScaler* transformer was used on the last nine continuous features the could not be reasonably bound. These transformers were organized into separate pipelines and passed to one single column transformer that was used to preprocess the train, test, and validation sets.

3.3 Hyperparameter Tuning

Hyperparameter tuning is the process by which a machine learning model is trained on a combination of manually identified parameters in search of a parameter that optimizes the chosen evaluation metric. Once the best parameters are identified through this iteration, they are used to fit the model on the test set to obtain the final evaluation score of the model.

The goal of hyperparameter tunning is to ensure that the model is underfit and overfit by the two extremes of the parameter combinations. This ensures that a better score would not be received by further lengthening the range of the hyperparameters. Table 1 shows the hyperparameters that were tuned for each machine learning model.

ML Algorithm	Parameters Tuned	Best Parameters
Lasso	alpha: [1e-5, 1e-4, 1e-3, 1e-2, 1e-2, 1, 1e2, 1e3]	alpha = 0.01
SVR	gamma: [0.1, 1, 10, 50, 75, 100] C: [0.1, 0.5, 1, 5, 10]	gamma = 10 C = 0.1
KNeighborsRegressor	n_neighbors = [1-21]	n_neighbors = 19
XGBoost	n_estimators: [10000] max_depth: [1,3,10,30,100,300,1000,10000,100000]	n_estimators: 10000 max_depth: 3

Table 1: Machine learning algorithms trained, parameters tunned, and resulting best parameters.

3.4 Reduced Features Model

Nearly half of the dataset contained missing values, making it unable to be trained by most machine learning models without first addressing the missing values. Four entries were missing values in the target variable, so those entries were removed from the data prior to splitting. In order to train machine learning models on the dataset, the reduced features model was used. The reduced features model finds each unique combination of missing values in the dataset, returns the non-missing values of all data points that follow that same unique pattern of missing values, and trains a machine learning model on the returned non-missing values. The model repeats those steps for each unique combination of missing values [2].

To run the reduced features model, I had to first complete all splitting and preprocessing (from 3.1 and 3.2) of the data. Once the data was split and preprocessed it could be passed to the reduced features model which first finds the unique combinations of missing values, forms subgroups according to the unique combinations, and calls the machine learning model. The function that is called to run each machine learning model trains the model on the training set, evaluates the performance against the validation set, and finally fits the model to the test set for final test scores and best parameters. This process is repeated five times to account for uncertainties in the splitting method and in nondeterministic machine learning algorithms. Three machine learning algorithms were trained on the dataset using the reduced features model: Lasso, SVR, and KNeighborsRegressor.

3.5 GridSearchCV

In addition to the three machine learning algorithms (from section 3.4), XGBRegressor was also run. XGBRegressor is able to deal with missing values on its own. In order to run this algorithm, the data was first split into the other and test set using GroupShuffleSplit (from section 3.1). A pipeline was created consisting of the preprocessors (from section 3.2) and the XGBRegressor algorithm. Both the pipeline and the split sets were passed to GridSearchCV. GridSearchCV then performs the GroupKFold splitting to form the training and validation sets, trains and validates the model, and finally refits the model on the test set finding the test scores and best parameters. This process is repeated five times for each parameter combination to account for the uncertainties in data splitting.

4 Results

The scores of each machine learning model are measured by the mean and standard deviation of the models root mean squared error (RMSE). I choose RMSE as the evaluation metric because it allowed for better understanding of what the score means in relation to the target variable. If the mean RMSE score for an algorithm is 2.9 years, then we know that the model should predict the life expectancy of a country around 2.9 years of the true life expectancy of that country.

In order to measure effectiveness of the machine learning algorithms, I calculated a baseline RMSE score. This was done by finding the mean of the test set's target variable, and then

calculating the RMSE of y_test against the mean. This resulted in a baseline RMSE score of 9.4, which was then compared to the RMSE score of each machine learning algorithm. Each algorithm performed much better then the baseline model, which can be seen in figure 6. Based on the mean RMSE of the models, the XGBRegressor was the most predictive model with an RMSE score of 3.10 and therefore was chose

ML Algorithm	Mean Test Score	Standard Deviation
Lasso	5.29	0.62
SVR	4.27	0.55
KNeighborsRegressor	4.45	0.69
XGBoost	3.10	0.18

Table 2: Mean and standard deviation of the RMSE scores for each ML algorithm.

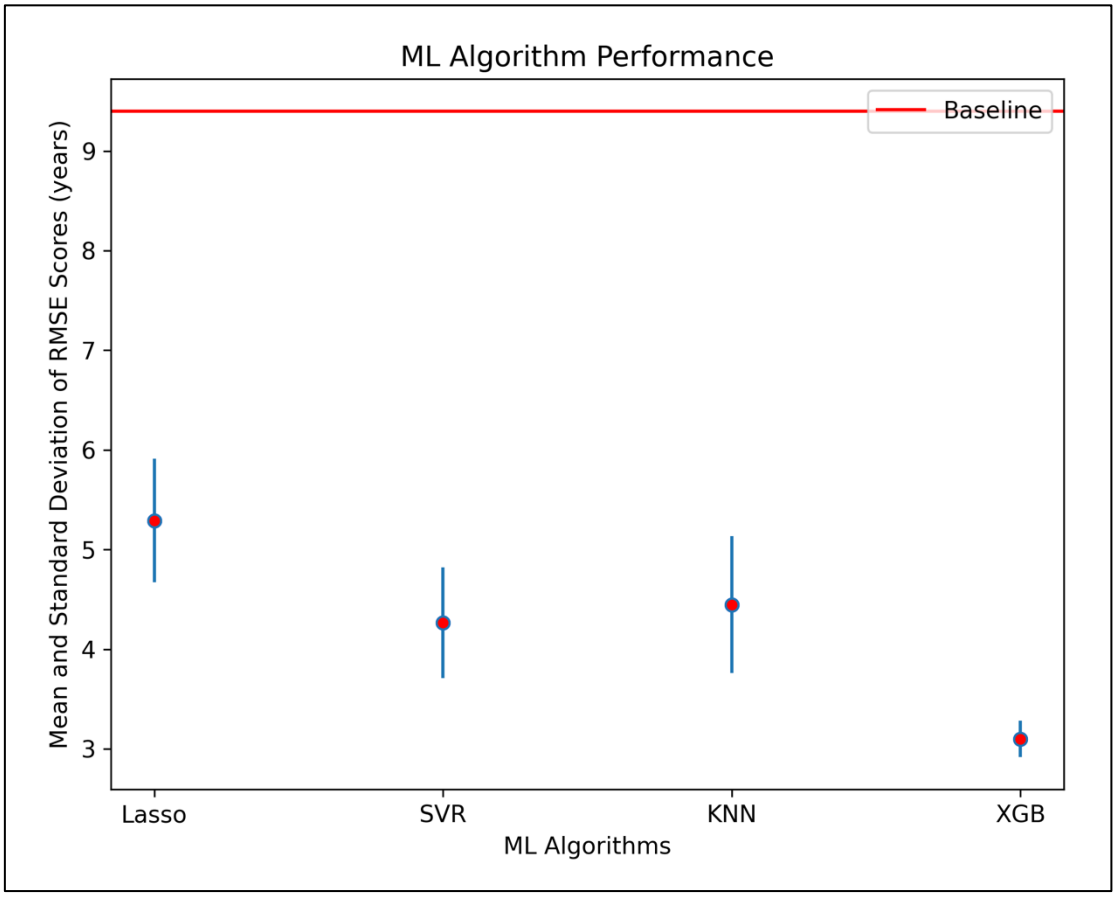

Figure 6: Resulting mean and standard deviation of each models RMSE score.

4.1 Local and Global Feature Importance

Feature importance was conducted to assist in interpretability of the XGBRegressor model. Calculating local feature importance allows us to see the features that go into each specific prediction and judge how accurate that prediction might be. Global feature importance is calculated to understand if the model is making its predictions based off of reasonable features, by returning the overall most important features when predicting the target variable. There are many methods in which to measure global feature importance, three different methods were utilized:

1. SHAP Values
2. XGBoost Weight
3. Permutation

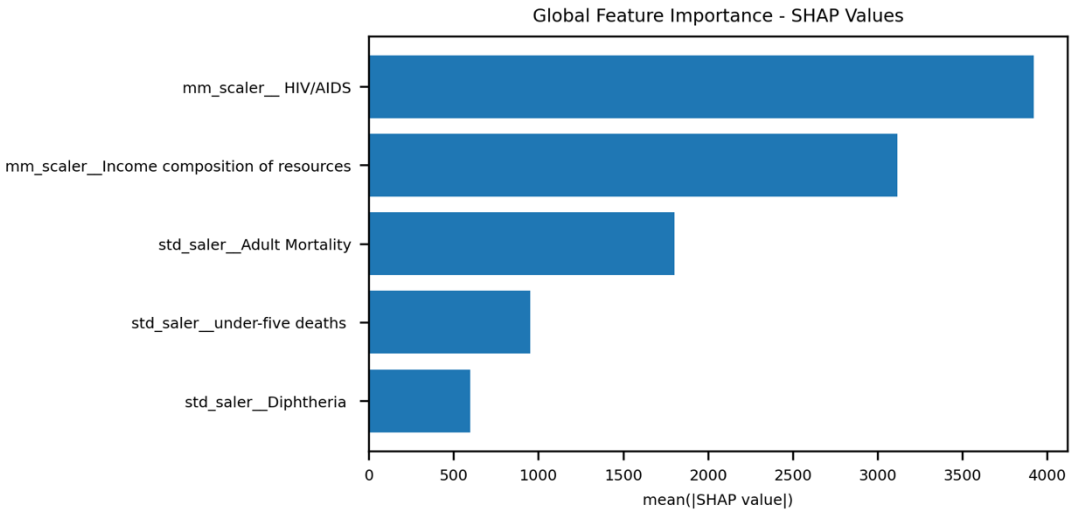

Figure 7: HIV/AIDS is the most important feature according to SHAP values.

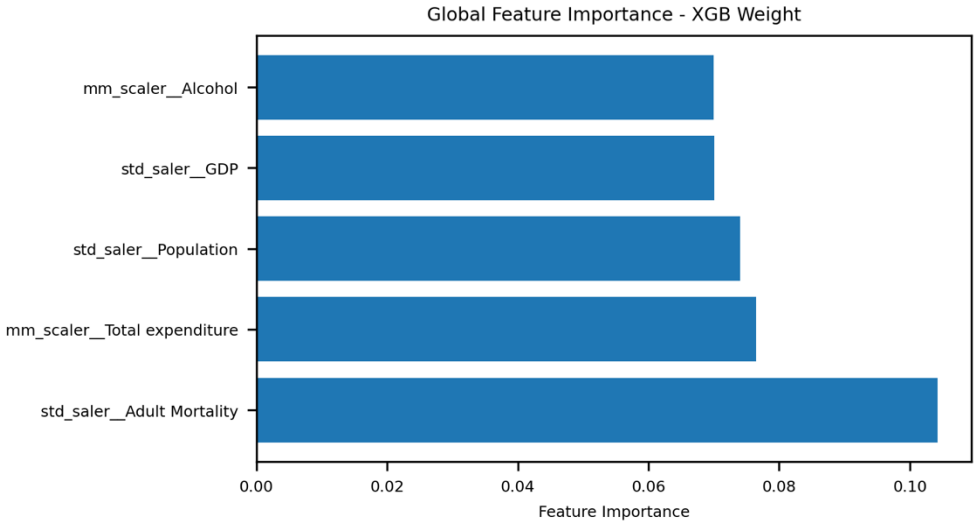

Figure 8: Adult Mortality is the most important feature according to XGB Weight

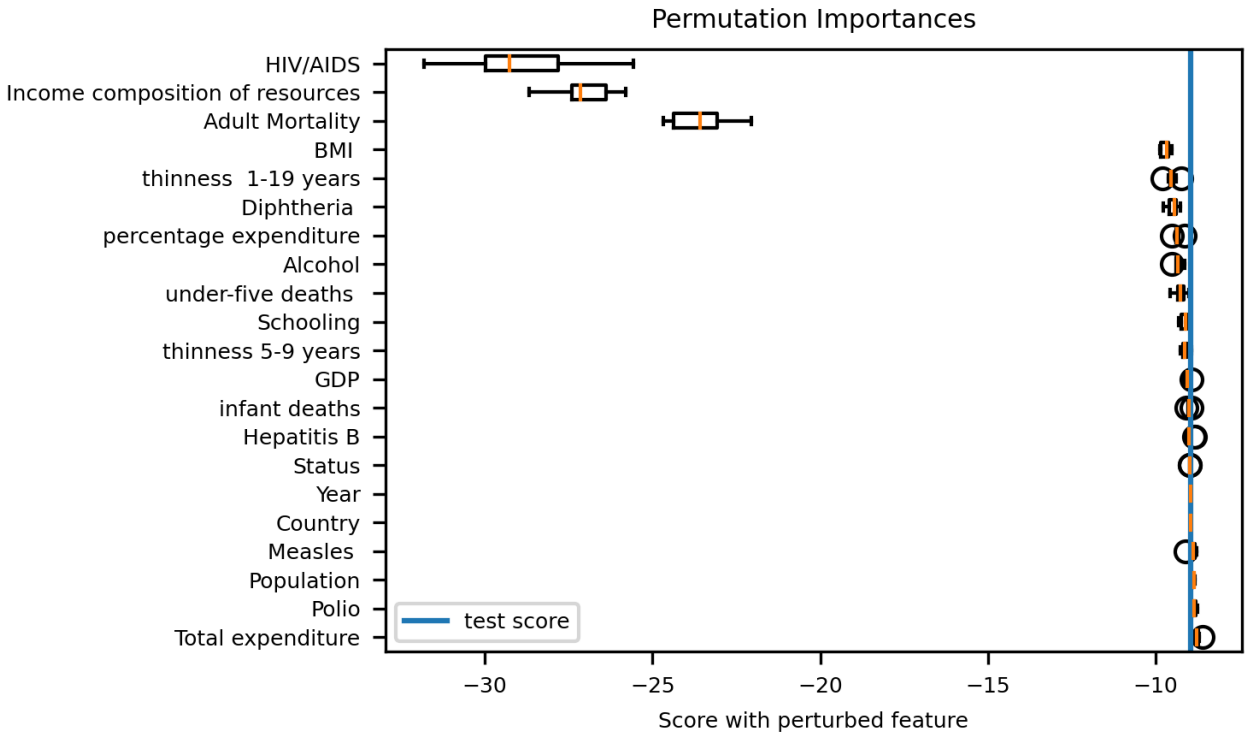

Figure 9: HIV/AIDS is the most important feature according to permutation feature importance.

When looking at the results from each of the three global feature importance methods there is one feature that appears in the top three of each method, Adult Mortality. From this we can deduce that Adult mortality is one of the most important feature calculating life expectancy. Permutation feature importance and SHAP values also share the same top three features: HIV/AIDS, Income composition of resources, and adult mortality.

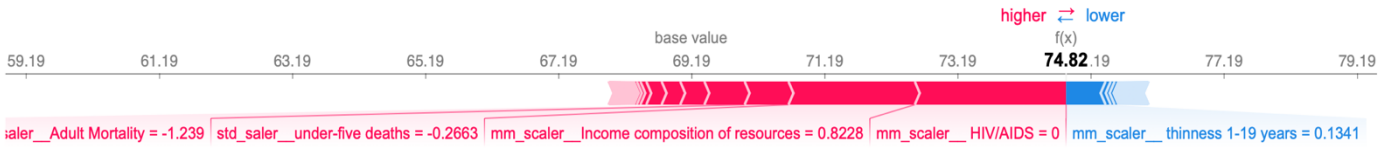

Figure 10: SHAP values for local feature importance of index 0.

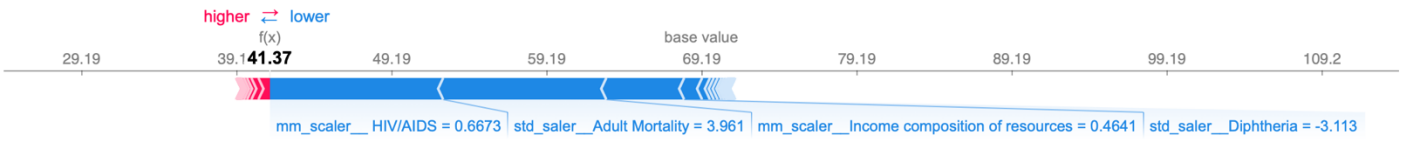

Figure 11: SHAP values for local feature importance of indes 300.

Figures 10 and 11 represent local feature importance for the prediction with index 0 and index 300. The three most important features from the global feature importance methods (HIV/AIDS, Income composition of resources, and Adult Mortality), also appear in both graphs above as having a significant roll in each prediction. This further exemplifies the importance of these three features in predicting life expectancy and proves consistent feature importance across different predictions.

5 Outlook

To further improve the predictive power and interpretability of this project, I would first train additional models to ensure that the XGBRegressor model is in fact the best model. I would look to train RandomForestRegressor and Linear Regression algorithms with L2 and elastic net regularization. The number of parameters tunned could also be increased to assist in increased predictability of each model. Specifically, I would look to increase the number of parameters tuned for the XGBRegressor model. Finally, more life expectancy experts involved in the data collection and model interpretation process would aid in a more full and effective machine learning model.

6 References

[1] https://www.kaggle.com/datasets/kumarajarshi/life-expectancy-who

[2] Saar-Tsechansky, Maytal, and Foster Provost. “Handling Missing Values when Applying Classification Models.” *Journal of Machine Learning Research* , vol. 8, 2007. *1625-1657*, https://doi.org/https://www.jmlr.org/papers/volume8/saar-tsechansky07a/saar-tsechansky07a.pdf.