# Machine Learning Project Documentation

## Project Title: Analyzing the Impact of Renewable Energy Adoption on Economic Recovery of Developing Nations

## 1. Model Refinement

### 1.1 Overview

The model refinement phase in machine learning is a critical step aimed at enhancing the performance, accuracy, and generalization of predictive models. This phase involves several key techniques and processes that collectively contribute to the model's improvement. Firstly, regularization techniques such as Lasso (L1 penalty), Ridge (L2 penalty), and Elastic Net (a combination of L1 and L2) are employed to prevent overfitting. Overfitting occurs when a model performs well on training data but poorly on unseen data. Regularization adds constraints to the model, penalizing excessive complexity and encouraging simpler models that generalize better to new data. Hyperparameter optimization is another essential aspect of model refinement. Hyperparameters are settings that control the learning process and model architecture, such as learning rates, regularization strengths, and the number of layers in a neural network. Optimizing these hyperparameters can significantly impact the model's performance. Techniques like grid search, random search, and Bayesian optimization are commonly used to find the optimal hyperparameter configuration. Grid search exhaustively searches through a specified subset of hyperparameters, while random search samples a random subset. Bayesian optimization, on the other hand, builds a probabilistic model of the objective function and uses it to select the most promising hyperparameters to evaluate. The model refinement phase is inherently iterative. It involves multiple cycles of training, evaluation, and adjustment. After each iteration, the model's performance is evaluated against key metrics, and the feedback is used to make incremental improvements. This iterative process continues until the model achieves satisfactory performance. Feature engineering is another crucial component of model refinement. It involves creating new features, selecting the most relevant ones, and transforming existing features to improve the model's predictive power. Effective feature engineering can significantly enhance the model's ability to capture underlying patterns in the data. Continuous evaluation and feedback are vital throughout the refinement phase. Regular assessment against key performance metrics ensures that the model meets its intended objectives and performs well on real-world data. This feedback loop helps identify areas for further refinement and guides the iterative process.

## 1.2. Model Evaluation

R2 accuracy was used for the evaluation of the models. The summary of the R2 accuracy has been shown in figure 1.

Figure 1. R2 Score of Different Models

### 2.3 Refinement Techniques

For the model refinement the missing data filled from backward and forward fill previously were replaced by the ARIMA technique which uses time series forecasting to fill the missing data. ARIMA, which stands for Auto Regressive Integrated Moving Average, is a class of models used to describe a given time series based on its own past values, including its lags and lagged forecast errors. This model can be utilized to forecast future values. Any non-seasonal time series that exhibits patterns and is not random white noise can be modeled using ARIMA.

An ARIMA model is defined by three parameters: p, d and q.

* p: The order of the autoregressive (AR) term.
* d: The number of differencing operations required to make the time series stationary.
* q: The order of the moving average (MA) term.

There were some improvements in the model but not much that would make the model reliable. So, another model was explored, with the advantage of imitating the data, artificial neural network for regression was used as new model. The R2 accuracy after 1000 epochs was obtained to be 0.71 which is enough for a model to be called as reliable.

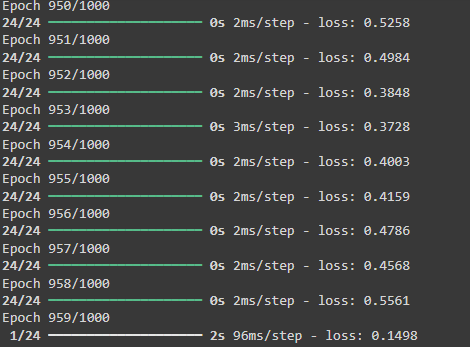


Figure 2. R2 error for Artificial Neural Network

### 2.4 Hyperparameter tuning

Since, highest R2 score was obtained for Random Forest model, Grid Search algorithm was used to find the best hyperparameters for the model that would make the R2 value improved. The best parameters obtained were bootstrap as true, maximum depth of 30, minimum sample leaves as 1, minimum samples split to be 2 and n-estimators to be 50. This selection of hyperparameter further decreased the value of R2 score which paved the way to select the new algorithm.

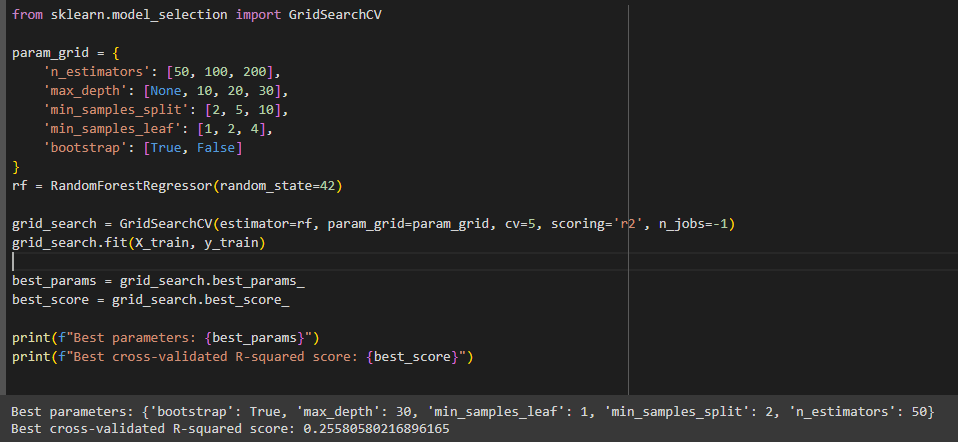


Figure 3. Hyperparameter testing of RF model

### 2.5 Cross Validation

Cross validation was done making the data in 5 folds. The model is trained on 4 folds and tested on the 5th, and this process is repeated for all folds. GridSearchCV is a hyperparameter tuning technique that combines grid search with cross-validation. It systematically explores a predefined grid of hyperparameters, evaluating each combination using cross-validation. The process involves splitting the data into multiple folds, training the model on some folds, and validating it on the remaining fold. This is repeated for each hyperparameter combination. The algorithm identifies the best set of hyperparameters based on the average performance across folds. This ensures the model is optimized for better performance on unseen data, enhancing its generalization ability.

## 2.6 Feature Selection

Feature Importance algorithm was used to make sure only the features that contribute were included in the dataset and least important were removed. After this top 10 features were included in the model but the value of R2 score further decreased.

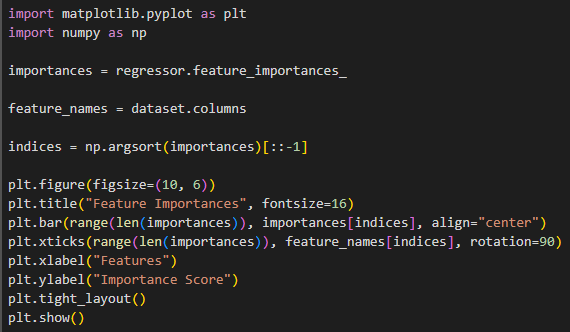


Figure 4. Feature Importance Algorithm

## 2. Test Submission

2.1 Overview

The test submission phase involves several critical steps to ensure a model is ready for deployment or evaluation on a test dataset. Initially, the model is retrained on the entire training dataset using the optimal hyperparameters identified during cross-validation, maximizing the data available for learning. Cross-validation is performed to ensure robustness, and optionally, a separate validation set may be used for further fine-tuning. The model is then evaluated on the test dataset to obtain an unbiased performance estimate, using metrics such as accuracy, precision, recall, and F1-score. Feature engineering and model pruning are applied to optimize the model, ensuring it is both efficient and effective. The trained model and any preprocessing steps are serialized using libraries like `joblib` or `pickle`, ensuring consistency during deployment. Comprehensive documentation of the model architecture, hyperparameters, training process, and evaluation metrics is created, along with well-documented code for reproducibility. The deployment environment is prepared, ensuring all dependencies and libraries are installed, and integration testing is conducted to verify the model's functionality in the real-world setting. Finally, the model, preprocessing steps, and necessary scripts are packaged for submission, following all guidelines to ensure a smooth evaluation process. This thorough approach ensures the model is robust, well-documented, and ready for deployment, enhancing its reliability and performance in real-world applications.

### 2.2 Data Preparation for Testing

Test set was made by random sampling of the total dataset. This method is a statistical technique used to select, define, and construct a representative subset from a large population. In this context, it is used to allocate 80% of the data for training the machine learning model, while the remaining 20% is reserved for testing the model's performance.

### 2.3 Model Application

The ANN model was developed using one input and output layer with 2 hidden layers in between. Mean squared error was used as the evaluation tool and obtained to be 0.1428. With model having satisfactory evaluation, the test set was used to predict the values and the predicted set and test set.

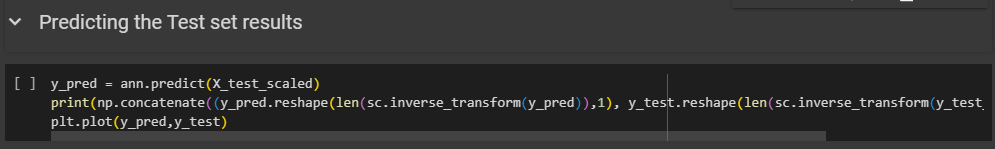


Figure 5. Predicting Test Set Results

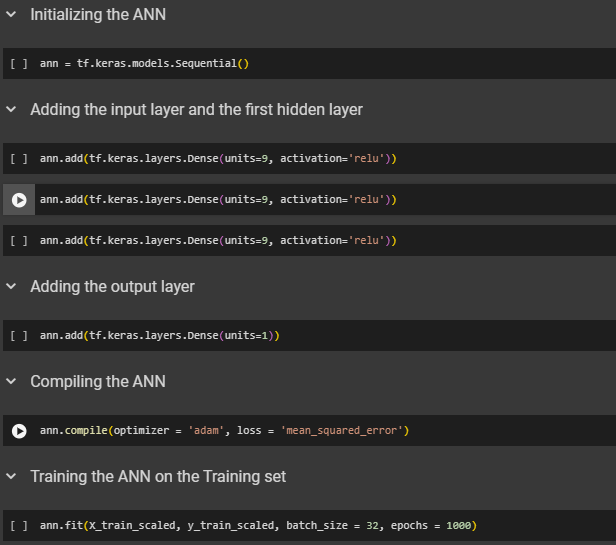


Figure 6. Training the Model

### 2.4 Test Metrics

The model achieved an accuracy score of 0.71 which is satisfactory for a machine learning model.

### 2.5 Deployment

To deploy the machine learning model in a real-world setting, several key steps were undertaken to ensure seamless integration and functionality. Initially, the model was trained and validated using a robust dataset, followed by hyperparameter tuning through GridSearchCV to optimize performance. Once the model achieved satisfactory accuracy, it was serialized using joblib for efficient storage and retrieval. The deployment phase involved leveraging Gradio, an intuitive Python library for creating interactive web interfaces. The model was integrated into a Gradio interface, allowing users to input data and receive predictions in real-time.

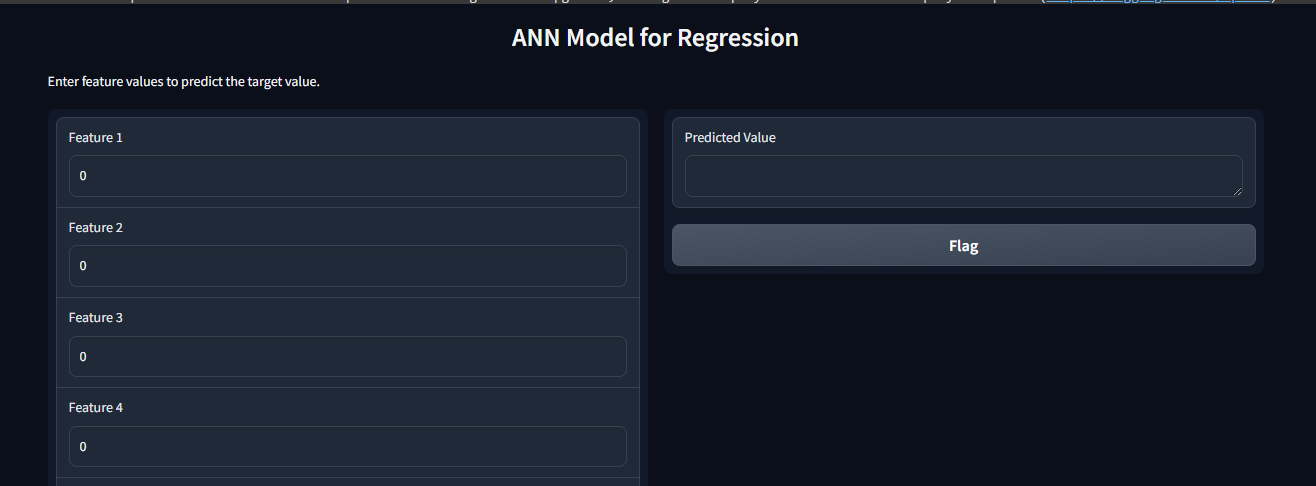


Figure 7. Deployment of the Model