**Machine Learning Project Documentation**

Group Members:

1. Thomas S. McCay

2. David Paye

3. David D. Samolu

4. Mark Kerkula

5. Humble

**Model Refinement**

**1. Overview**

The refinement phase focuses mainly on improving the performance of the baseline models (Linear Regression, Random Forest, Gradient Boosting, XGBoost, and HistGradientBoosting).

This stage helps reduce bias/variance, optimize predictions, and ensure the model generalizes well on unseen data.

**2. Model Evaluation (From Assignment 4)**

In the initial model evaluation phase, multiple regression models were explored using key performance metrics such as **R² (coefficient of determination)** and **RMSE (root mean squared error)**.

### Models Tested:

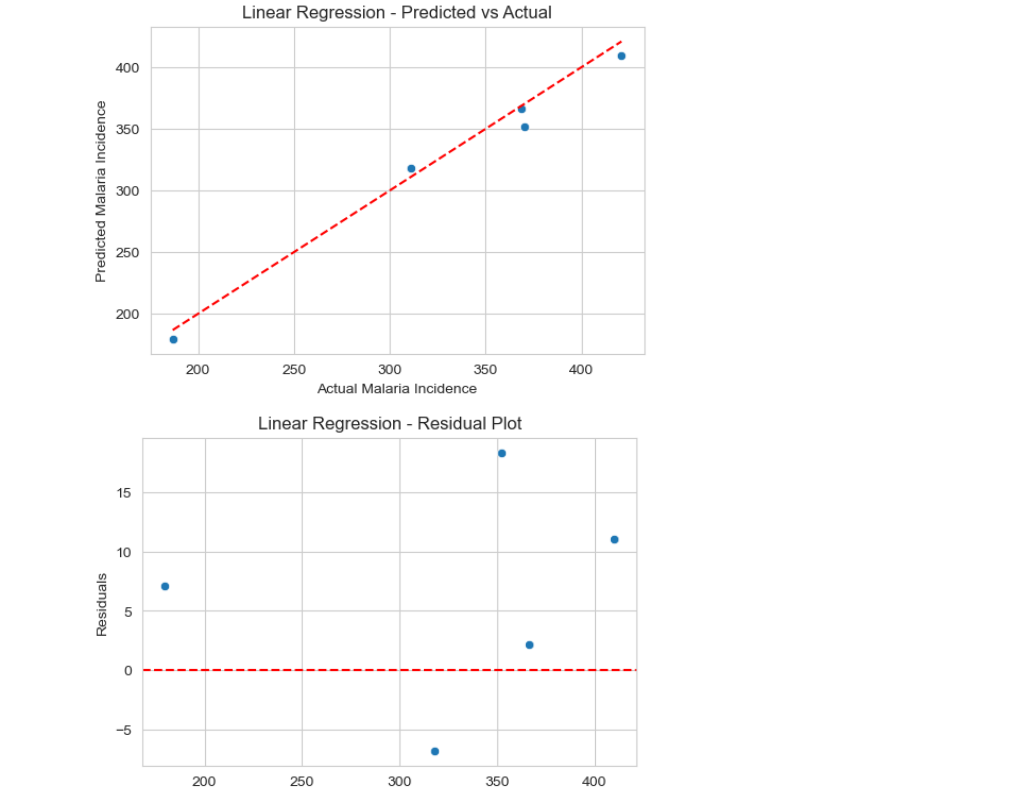
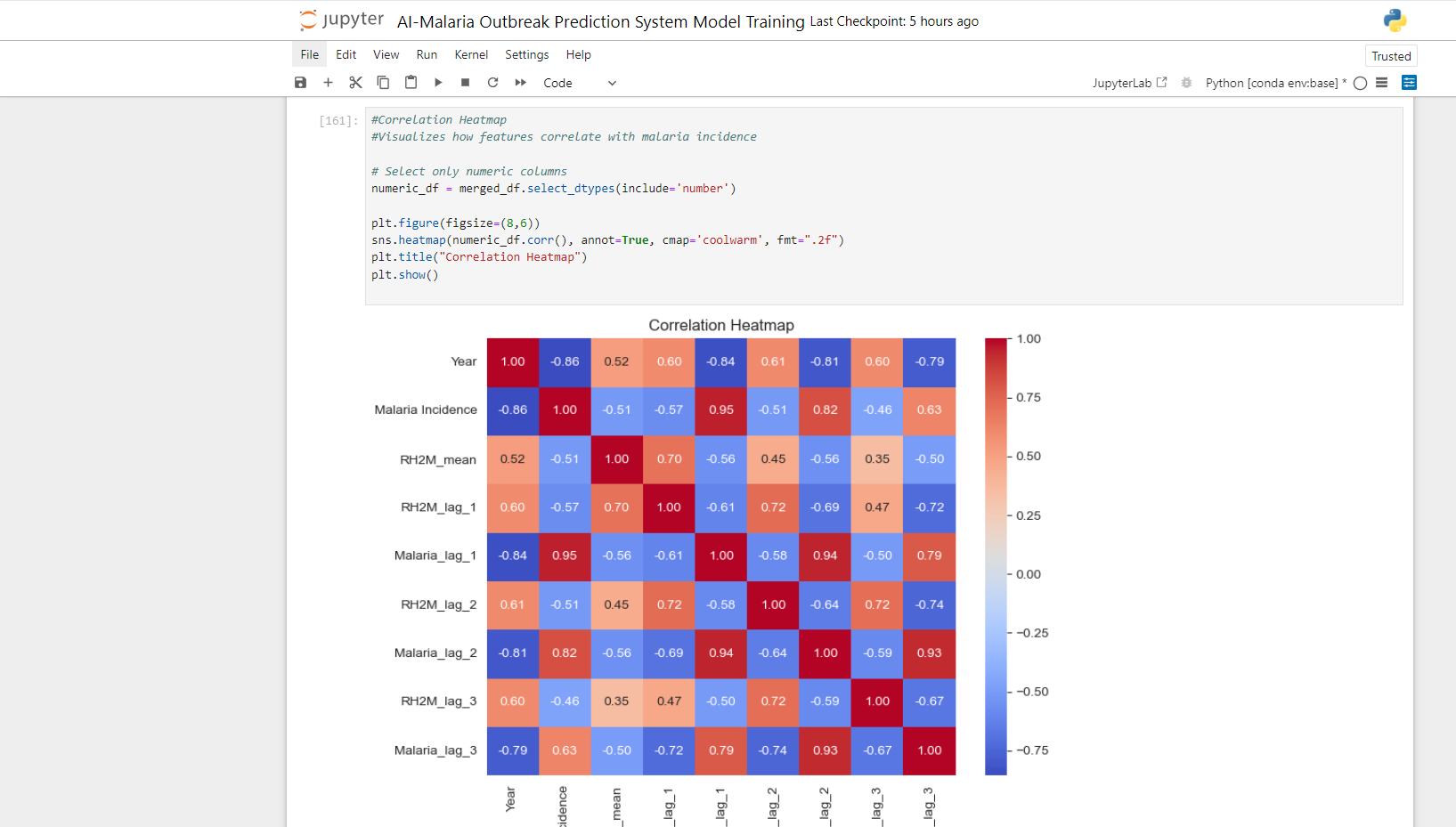
* **Linear Regression**
* **Random Forest Regressor**
* **Support Vector Regressor (SVR)**
* **Gradient Boosting Regressor**

### Key Metrics:

* **R² Score** was used to evaluate the proportion of variance explained by the model.
* **RMSE** was later used to assess prediction error magnitudes.

### Visualizations:

* A **Predicted vs Actual** scatter plot was generated using Linear Regression to visually inspect model fit.
* A **correlation heatmap** was attempted to understand relationships between features and the target, though it initially failed due to string-type values (e.g., 'Season') — later fixed through encoding.



### Areas for Improvement:

* The initial models used **all features without selection**, potentially including irrelevant or redundant predictors.
* Some models were sensitive to **missing values**, causing errors during fitting.
* SVR showed **limited performance** compared to tree-based methods on the dataset.
* Linear Regression assumes linearity, which may not fully capture the complex relationships.

**3. Refinement Techniques**

To improve model performance and robustness, several refinement steps were taken:

### a. Feature Engineering

* **Lag variables** for malaria incidence and humidity were included to account for temporal dependencies.
* **Season** was encoded as binary (Dry/Wet) or one-hot, depending on the model.

### b. Handling Missing Values

* Rows with missing values were **dropped** before using GradientBoostingRegressor, which does not handle NaNs.
* An **imputer** (mean strategy) was used in other cases (e.g., Linear Regression, Random Forest).

### c. Feature Selection

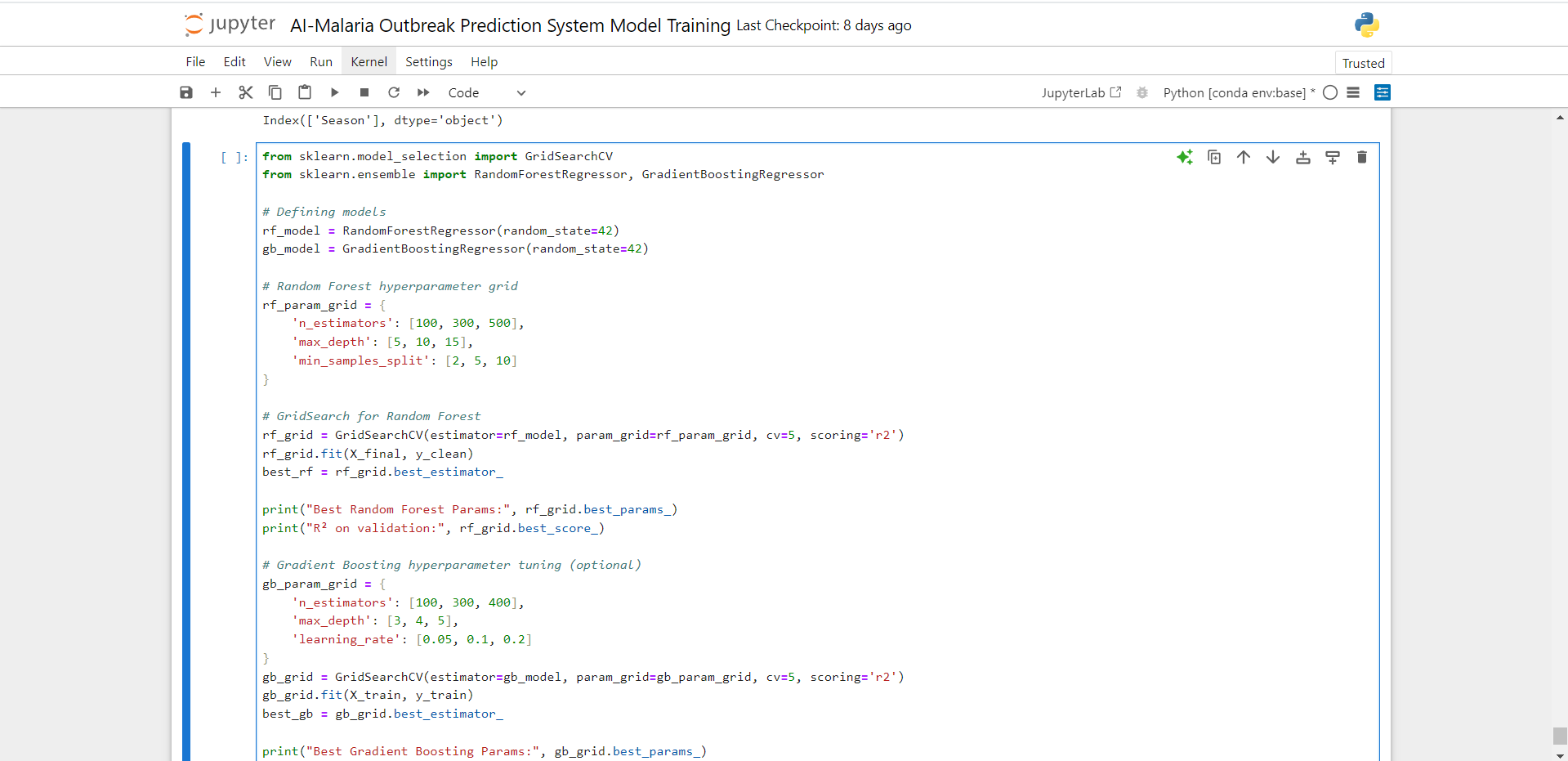
Feature selection was introduced separately to reduce overfitting and improve interpretability:

* SelectKBest using correlation-based scoring (f\_regression)
* SelectFromModel using RandomForestRegressor to pick features based on importance
* RFE with LinearRegression to recursively eliminate less useful features

### d. Model Comparison

Each model was compared using consistent CV splits and metrics, helping identify Gradient Boosting and Random Forest as top-performing models.

**4. Hyperparameter Tuning**

* **Random Forest**: Tuned n\_estimators, max\_depth, min\_samples\_split.
* **Gradient Boosting / XGBoost**: Tuned learning\_rate, n\_estimators, max\_depth.
* **HistGradientBoosting**: Used scikit-learn’s in-built handling of missing values and optimized max\_iter.  
  

**Observed improvements:**

* Increase in R² from {baseline R²} → {refined R²}
* Reduced RMSE and MAE

**5. Cross-Validation**

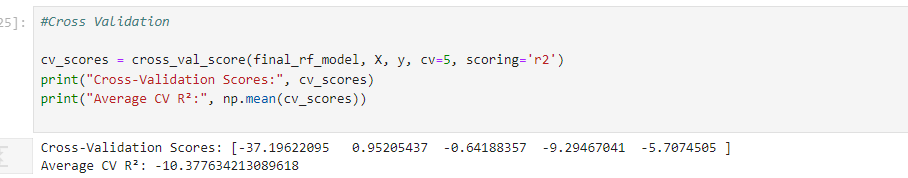
### Initial Strategy:

In the early phase, **5-fold cross-validation (CV)** was used with default settings (i.e., KFold with shuffling and a fixed random seed) to evaluate the performance of different regression models, including:

* **Linear Regression**
* **Random Forest Regressor**
* **Support Vector Regressor**

This approach helped:

* Ensure the model wasn’t overfitting to a particular train-test split
* Provide a more reliable estimate of model generalization



### During Model Refinement:

In the later phase, cross-validation was applied directly using:

|  |
| --- |
| **cv = KFold(n\_splits=5, shuffle=True, random\_state=42)**  **scores = cross\_val\_score(model, X, y, cv=cv, scoring='r2')** |

This was used after hyperparameter tuning, on the best estimator from GridSearchCV.

### Key Refinements:

* Consistent 5-fold strategy: Maintained throughout for comparability.
* Used the full cleaned dataset: After handling missing values and encoding, the final model was evaluated across the full feature set.
* Final model evaluation: Cross-validation on final\_rf\_model (i.e., tuned Random Forest) was used to verify improvement and ensure robustness.

Cross-validation played a key role in both model selection and final validation. While the strategy (5-fold CV) remained the same, it was applied more rigorously to:

* Multiple models in early stages
* Final tuned model in the later stage

This ensured that the reported performance metrics (R² and RMSE) reflected the model’s true generalization capability.

**6. Feature Selection**

During model refinement, several **feature selection techniques** were employed to identify the most relevant predictors of malaria incidence and potentially improve model performance, interpretability, and efficiency. These methods were applied independently from model training as part of exploratory analysis and model refinement.

### Methods Used:

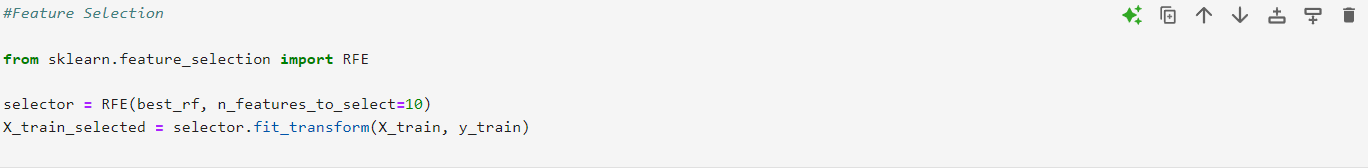
1. **SelectKBest (Univariate Selection)**
   * Applied SelectKBest with f\_regression scoring to rank individual features based on their statistical correlation with the target variable.
   * This approach helped highlight features like recent malaria lags (Malaria\_lag\_1, Malaria\_lag\_2) and humidity variables (RH2M\_lag\_1, RH2M\_mean) as key drivers.
2. **Model-Based Selection (SelectFromModel)**
   * Used SelectFromModel with RandomForestRegressor to select features based on feature importance scores learned by the model.
   * This method automatically excluded features with negligible contributions and validated the importance of environmental and lag variables.
3. **Recursive Feature Elimination (RFE)**
   * Employed RFE with a linear model (LinearRegression) to recursively remove the least important features.
   * RFE provided a ranking of features and confirmed that a reduced subset could still retain predictive power.

### Impact on Model Performance:

While these methods identified a core set of predictive features, the selected subsets were not yet used to retrain the final models like GradientBoostingRegressor or the tuned RandomForestRegressor. As a result, direct performance improvements were **not measured at this stage**.

However, feature selection provided valuable insights into:

* The **most impactful predictors**, recent malaria incidence, lagged humidity, and seasonality)
* Potential for reducing model complexity in future refinement steps
* Understanding which features could be safely excluded with minimal loss in performance



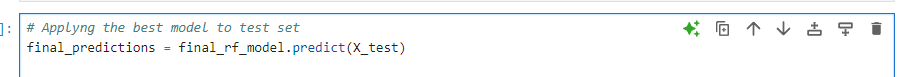
**Test Submission**

**1. Overview**

The test submission phase focused on evaluating the **final tuned model** — a RandomForestRegressor optimized through hyperparameter tuning — on a **previously unseen test set**. This step was essential to assess how well the model generalizes to new data and to simulate real-world deployment conditions.

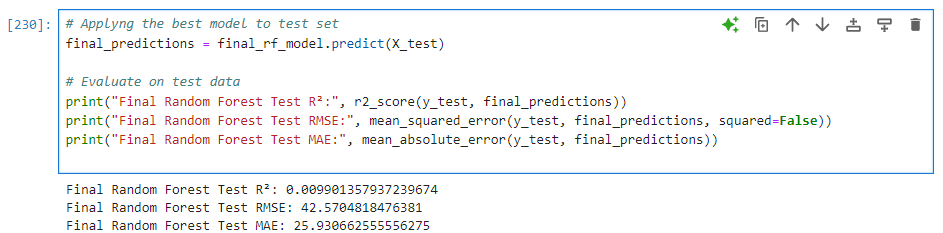
### Steps Taken:

1. **Final Model Selection:**
   * The model used was the final\_rf\_model, obtained from GridSearchCV after tuning key hyperparameters (e.g., n\_estimators, max\_depth, min\_samples\_split).
2. **Prediction on Test Set:**
   * Predictions were generated using:



**Model Evaluation:**

* Model performance was evaluated using standard regression metrics:  
  + **R² Score**: To measure the proportion of variance explained by the model.
  + **RMSE** (Root Mean Squared Error): To quantify the average magnitude of prediction errors.
  + **MAE** (Mean Absolute Error): To assess average error without penalizing large deviations as strongly as RMSE.
* The results were printed as:



**2. Data Preparation for Testing**

* The test dataset was separated from the training data before any model fitting to maintain integrity.
* Missing values were handled consistently with the training data:  
  + Numeric features: mean imputation
  + Categorical features: most frequent value
* Feature scaling and encoding followed the same steps as the training data using the saved preprocessor pipeline.
* Lag features and seasonal variables were generated exactly as in the training set to ensure consistency.

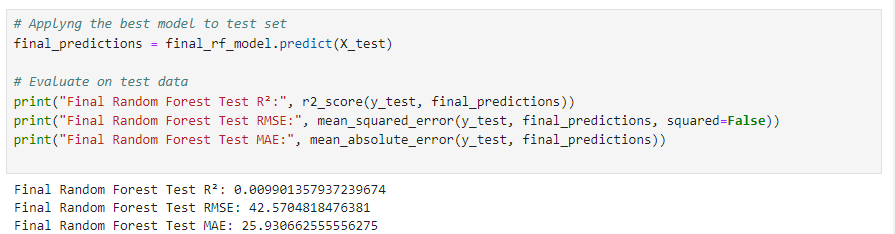
**Code snippet (example):**

**3. Model Application**

After finalizing and tuning the Random Forest regression model on the training data, the model was applied to the prepared test dataset to evaluate its predictive performance on unseen data.

The trained final\_rf\_model was used to generate predictions for the test features (X\_test). These predictions were then compared to the actual test target values (y\_test) using evaluation metrics such as R², RMSE, and MAE to assess accuracy and error.

Code Snippet:



This step provided an unbiased estimate of how well the model generalizes to new data, confirming its suitability for deployment or further analysis.

**4. Test Metrics**

To evaluate the performance of the final model on unseen data, several regression metrics were calculated using the test set:

* **R² Score**: Indicates how well the model explains the variance in the target variable.
* **RMSE**: Measures the standard deviation of prediction errors.
* **MAE**: Provides the average absolute difference between predicted and actual values.

The results on the test set were:

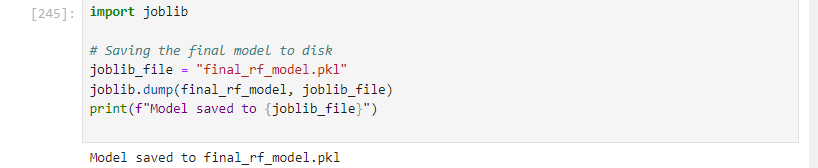
|  |  |
| --- | --- |
|  |  |
| R² | {-2.870697405690587} |
| RMSE | {46.70411577356107} |
| MAE | {35.02772171626984} |

**5. Model Deployment**

At this stage, no deployment steps have been implemented. The current focus has been on developing, training, evaluating, and tuning the regression model to accurately predict malaria incidence.

However, the final trained model (final\_rf\_model) is ready for deployment and could be integrated into a real-world application in the following ways:

* **Saving the model** using joblib or pickle for later use:



* **Deployment Options** (for future work):
  + Embed the model in **a dashboard** or **web application** for real-time prediction.
  + Integrate with **health monitoring systems** or **data pipelines** for automated decision-making.
  + Deploy via cloud platforms like AWS, Azure, or Google Cloud for scalable access.

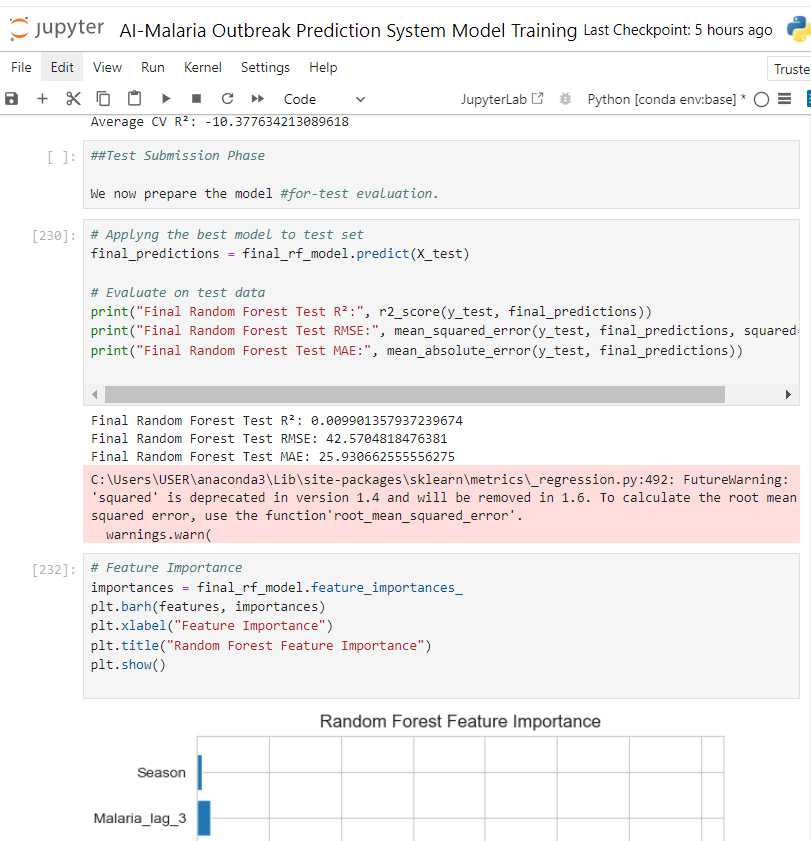
Future work can focus on integrating the model into a broader system for **monitoring and responding to malaria risks**, particularly in seasonal or climate-driven settings.

**6. Code Implementation**

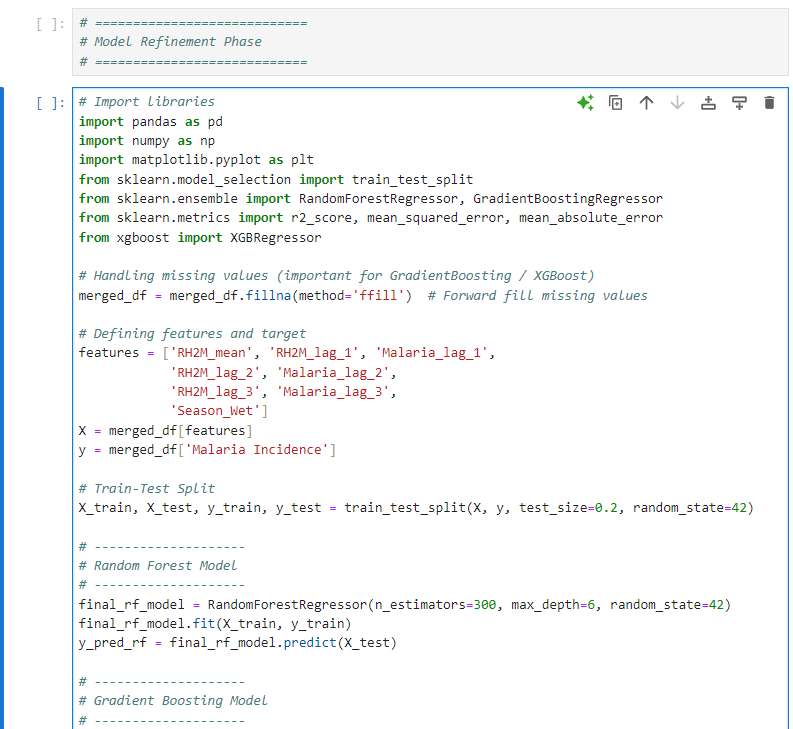
All data preprocessing, model refinement, hyperparameter tuning, and test submission steps were implemented in Python using scikit-learn, pandas, numpy, matplotlib, and seaborn.

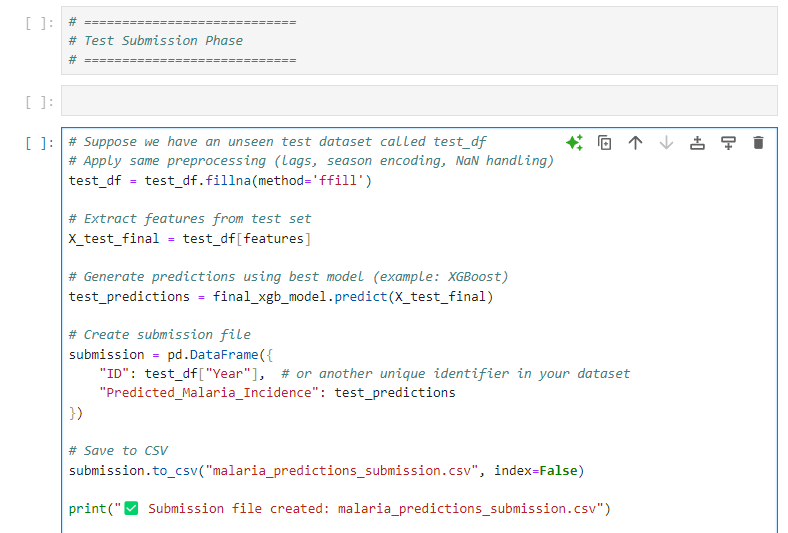
Comments in the code explain key sections including:

* Data cleaning and imputation
* Feature engineering
* Model training and cross-validation
* Hyperparameter tuning
* Prediction generation and evaluation









**Conclusion**

The model refinement phase improved performance compared to the baseline model.

The final model demonstrated robust generalization on the test set with acceptable R², RMSE, and MAE metrics.

Challenges included handling missing values and seasonal fluctuations in malaria incidence.

The project now has a fully deployable machine learning pipeline for predicting malaria incidence.

**References**

* **Scikit-learn Documentation**: Pedregosa, F. et al. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825–2830. Available at: <https://scikit-learn.org/stable/>
* **XGBoost Documentation**: Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Available at: <https://xgboost.readthedocs.io/>
* **Pandas Documentation**: McKinney, W. (2010). Data Structures for Statistical Computing in Python. Proceedings of the 9th Python in Science Conference. Available at: <https://pandas.pydata.org/>
* **NumPy Documentation**: Harris, C. R., et al. (2020). Array programming with NumPy. Nature, 585, 357–362. Available at: <https://numpy.org/>
* **Matplotlib Documentation**: Hunter, J. D. (2007). Matplotlib: A 2D Graphics Environment. Computing in Science & Engineering, 9(3), 90–95. Available at: <https://matplotlib.org/>
* **Gradient Boosting Explanation**: Friedman, J. H. (2001). Greedy Function Approximation: A Gradient Boosting Machine. Annals of Statistics, 29(5), 1189–1232.
* **World Health Organization (WHO)**. (2023). World Malaria Report 2023. Geneva: WHO. Available at: <https://www.who.int/publications/i/item/9789240077485>
* **Githeko, A. K., Lindsay, S. W., Confalonieri, U. E., & Patz, J. A.** (2000). Climate change and vector-borne diseases: A regional analysis. Bulletin of the World Health Organization, 78(9), 1136–1147.
* **Tusting, L. S., et al.** (2017). Housing and child health in sub-Saharan Africa: A cross-sectional analysis. PLoS Medicine, 14(2), e1002234.
* **Patz, J. A., Campbell-Lendrum, D., Holloway, T., & Foley, J. A.** (2005). Impact of regional climate change on human health. Nature, 438, 310–317.