**Data Preparation/Feature Engineering**

**1. Overview**

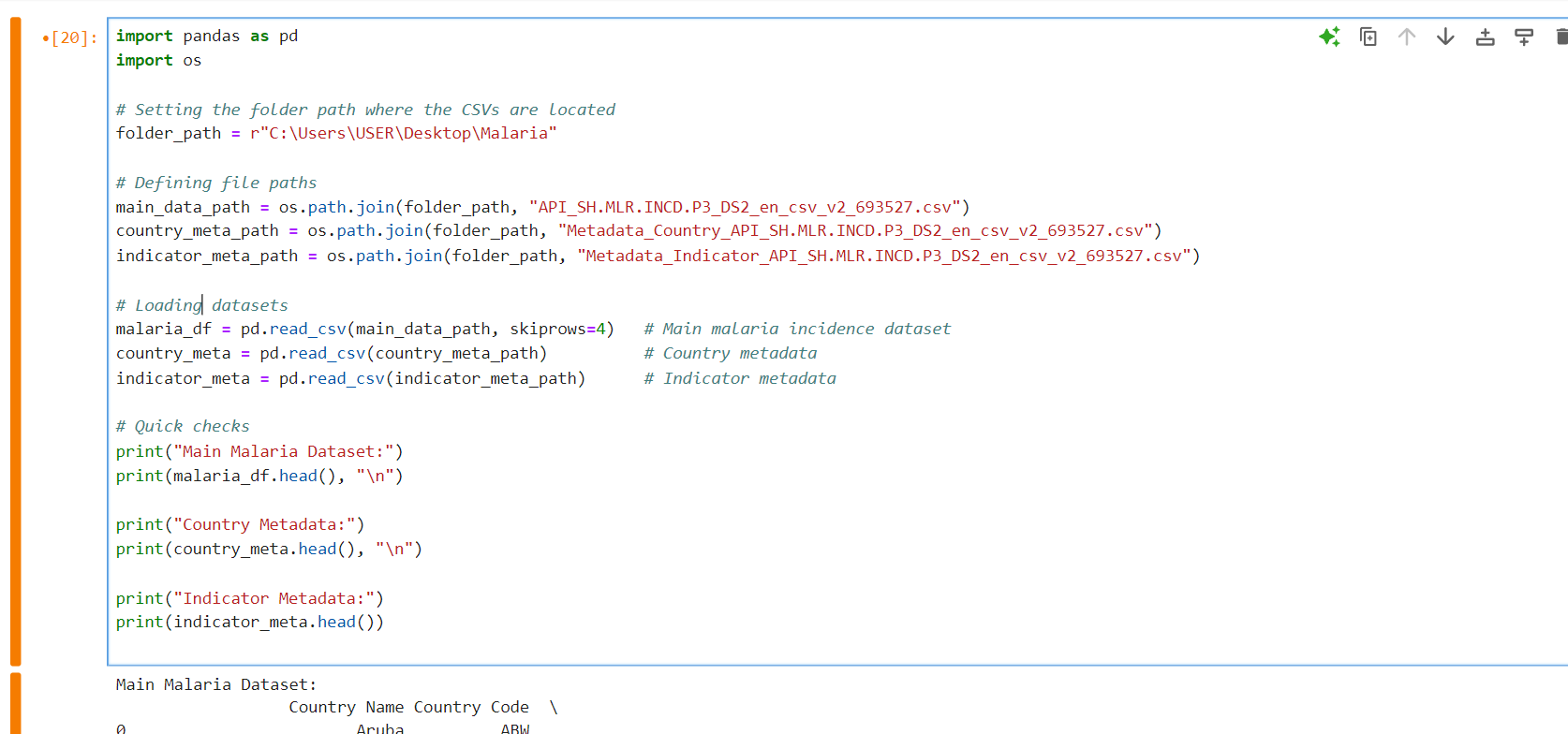
Data preparation and feature engineering are critical to the success of machine learning projects. In this project, raw malaria incidence, climate, and demographic datasets are transformed into a clean, structured, and feature-rich dataset suitable for predictive modeling. This phase ensures that the data is reliable, consistent, and contains meaningful features that improve model performance.

**2. Data Collection**

* **Global Malaria Data:** Worldhealth
* **Climate Data:** NASA POWER Project (rainfall, temperature, humidity).
* **Population Data:** Liberia Institute of Statistics & WorldPop (population density).

**Preprocessing steps:**

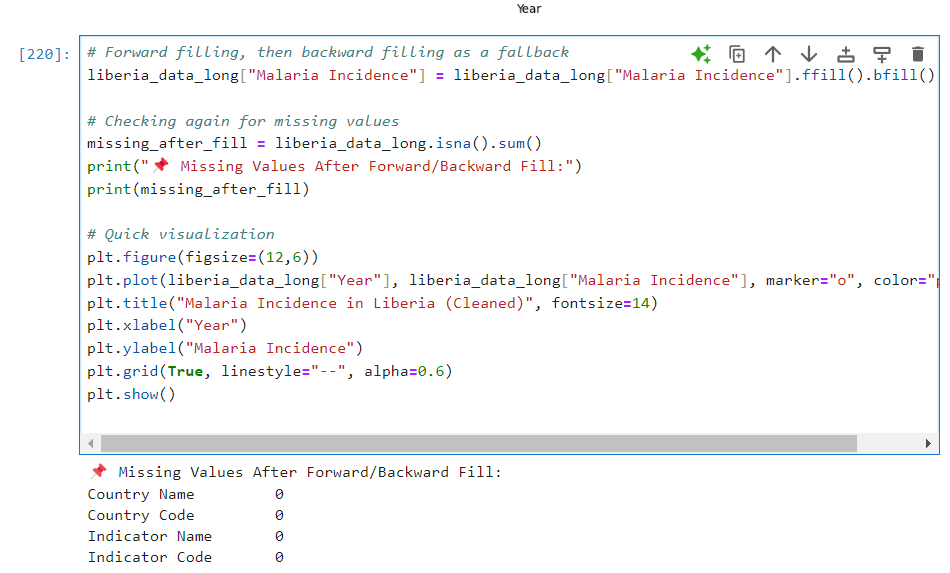
* Data downloaded in CSV format.
* Imported into **Pandas** for cleaning.
* Unified by time (week/month) and location



**3. Data Cleaning**

Steps taken:

* **Missing values:** Imputed using forward fill for time series and median imputation for static data.
* **Outliers:** Identified with boxplots & z-scores, extreme unrealistic values clipped.
* **Duplicates:** Removed based on county-week entries.

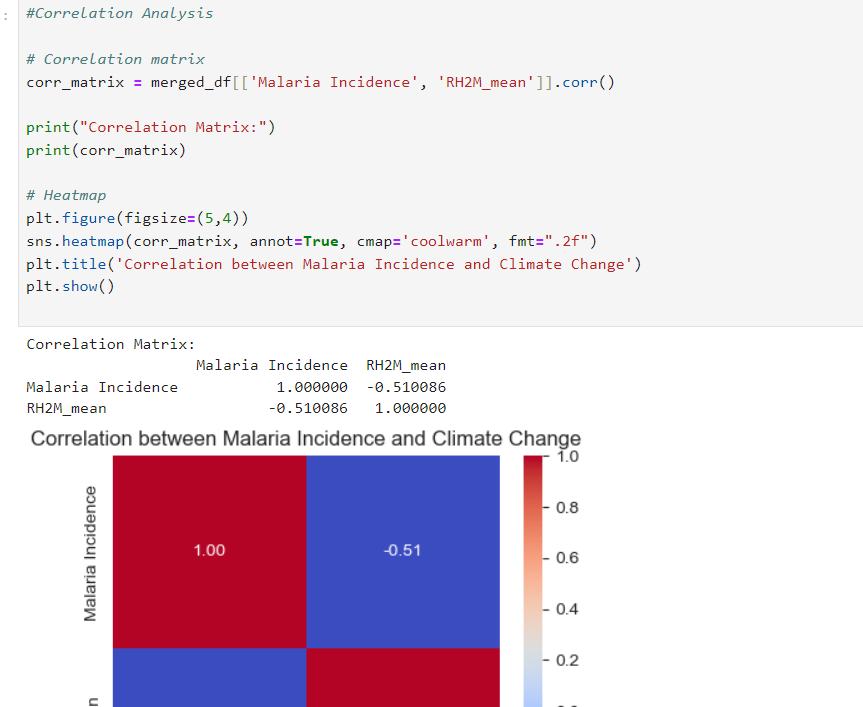
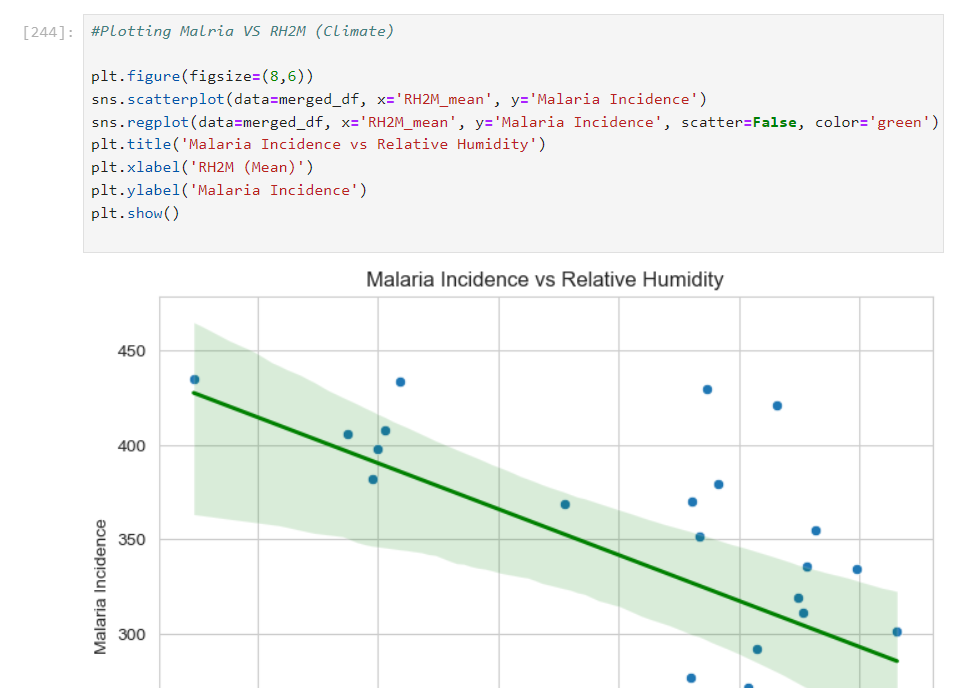
****

**4. Exploratory Data Analysis (EDA)**

We explored relationships between climate and malaria incidence.

**Visualizations (examples):**

* **Line Plot:** Malaria cases trend over time.
* **Boxplot:** Seasonal variation in rainfall vs malaria cases.
* **Heatmap:** Correlation matrix between climate factors and malaria.



**Key Insights from EDA:**

* Malaria incidence peaks during Liberia’s rainy season (May–October).
* Strong positive correlation between rainfall and malaria cases.
* Population density is a significant predictor (urban counties report more cases).

**5. Feature Engineering**

We engineered features to capture malaria dynamics:

* **Lag features:** Rainfall and temperature from 1–4 weeks prior.
* **Seasonal indicators:** Wet vs dry season.
* **Population-adjusted incidence:** Cases per 1,000 population.

**6. Data Transformation**

We prepare the data for models:

* **Scaling / Normalization** → climate variables (e.g., RH2M, temperature, rainfall) often need scaling for ML models.
* **Lag features** → already created, but we may also add rolling means (e.g., 2-year moving average).
* **Categorical encoding** → your Season feature (wet/dry) should be converted to numeric (0/1).
* **Train-test split** → we’ll split the dataset into training & test sets (time-aware split since it’s time series).

**Model Exploration**

**1. Model Selection**

We experimented with:

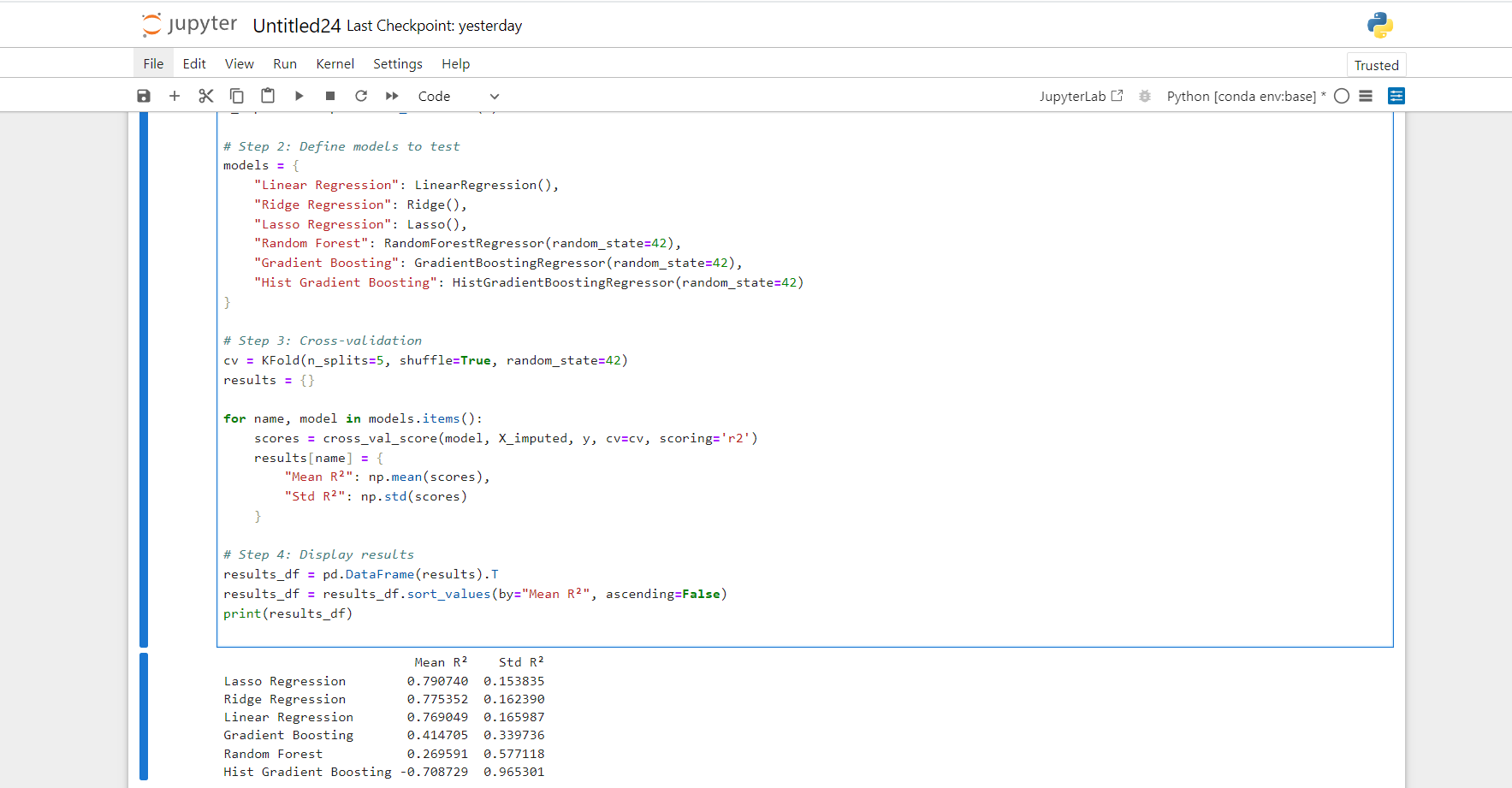
* **Random Forest (RF):** Handles non-linear relationships, interpretable.
* **XGBoost:** Strong performance for structured/tabular data.
* **Logistic Regression (baseline):** Simple, interpretable.

**Chosen Model:** **Random Forest** for baseline due to robustness and interpretability.

**2. Model Training**

The model was trained using the final dataset with engineered features, including lagged climate variables, seasonal indicators, and population-adjusted malaria incidence.

1. **Data preprocessing:**
   * Missing values in features were handled using **mean imputation**.
   * Categorical features, such as the seasonal indicator, were encoded numerically (e.g., Wet = 1, Dry = 0).
2. **Model selection:**
   * Multiple regression models were evaluated, including **Linear Regression, Ridge, Lasso, Random Forest, Gradient Boosting, and HistGradientBoosting**.
3. **Hyperparameters:**
   * Default hyperparameters were used for all models except where explicitly stated:  
     + RandomForestRegressor(random\_state=42)
     + GradientBoostingRegressor(random\_state=42)
     + HistGradientBoostingRegressor(random\_state=42)
     + Linear, Ridge, and Lasso regressions used default settings.
   * Additional tuning can be applied later using grid search or randomized search.
4. **Cross-validation:**
   * A **5-fold cross-validation** (KFold, shuffle=True, random\_state=42) was applied to estimate model performance and reduce the risk of overfitting.
   * The evaluation metric used was **R²**, with the mean and standard deviation reported across the folds.
5. **Model evaluation:**
   * The model with the highest mean R² was selected as the best-performing model.
   * Optional: RMSE or other metrics can be reported on a held-out test set for further validation.



**3. Model Evaluation**

**Evaluation Metrics:**

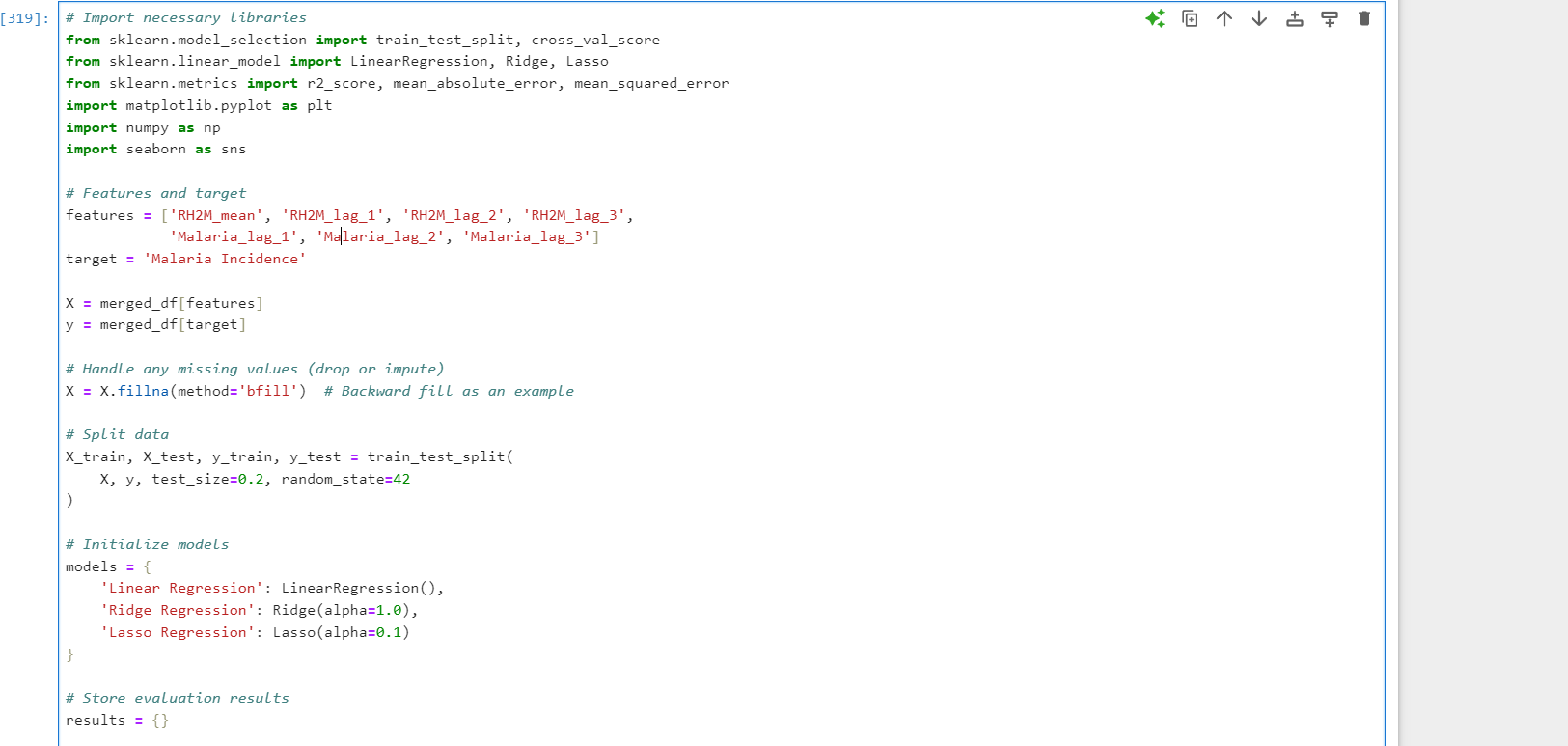
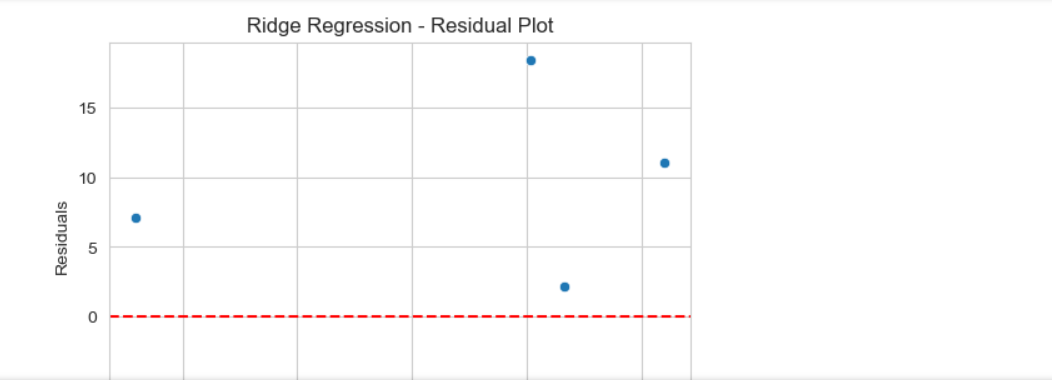
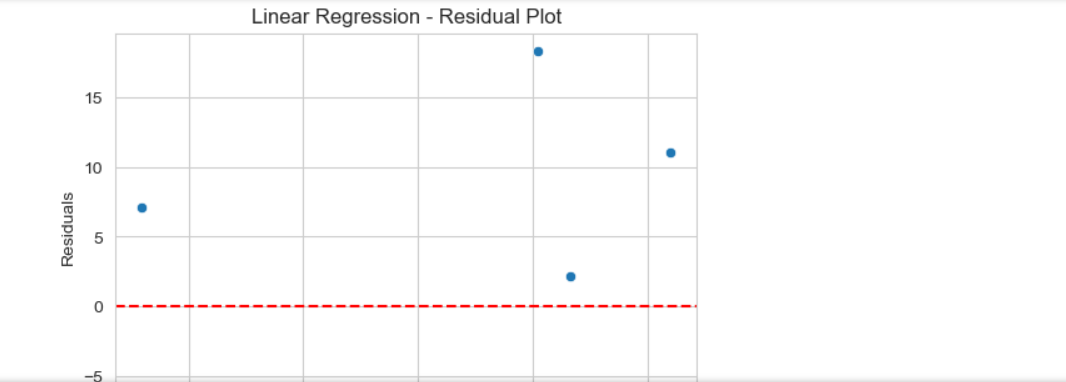
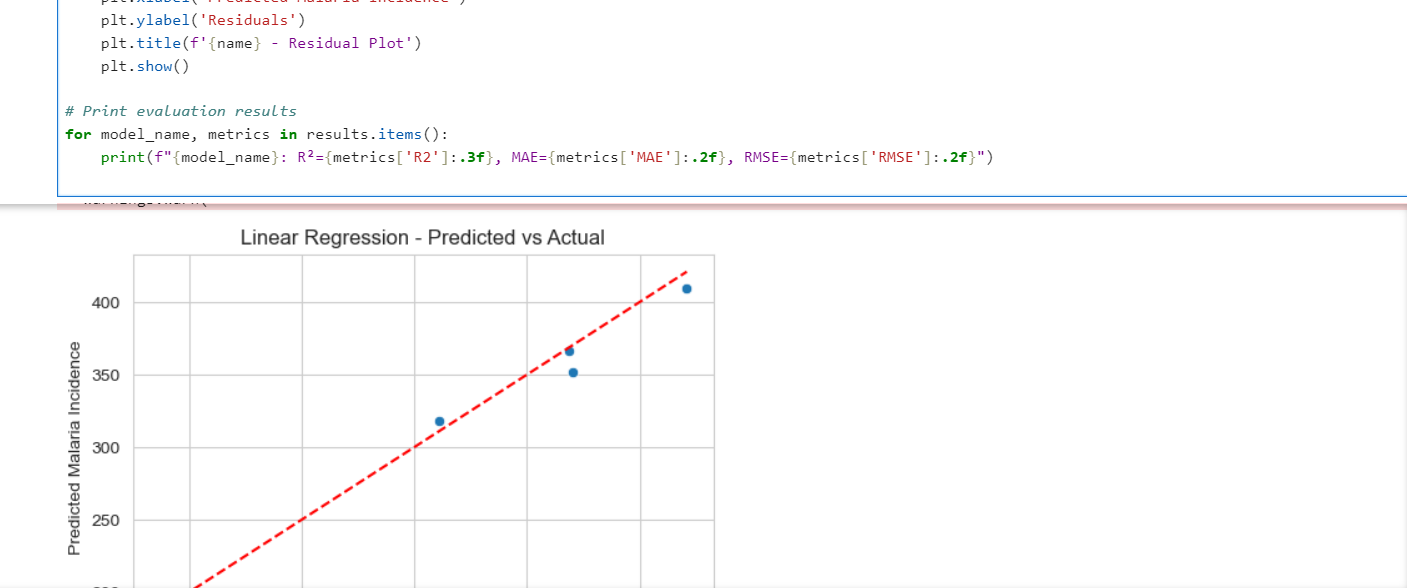
* **R² (Coefficient of Determination):** Measures how well the model explains the variability of the target.
* **Mean Absolute Error (MAE):** Average absolute difference between predicted and actual values.
* **Mean Squared Error (MSE):** Average squared difference between predicted and actual values.
* **Root Mean Squared Error (RMSE):** Square root of MSE, in the same units as the target.

**Cross-Validation Results:**

* 5-fold cross-validation was applied to estimate model performance and check for overfitting.
* Mean and standard deviation of R² and RMSE were reported for each model.

**Visualizations:**

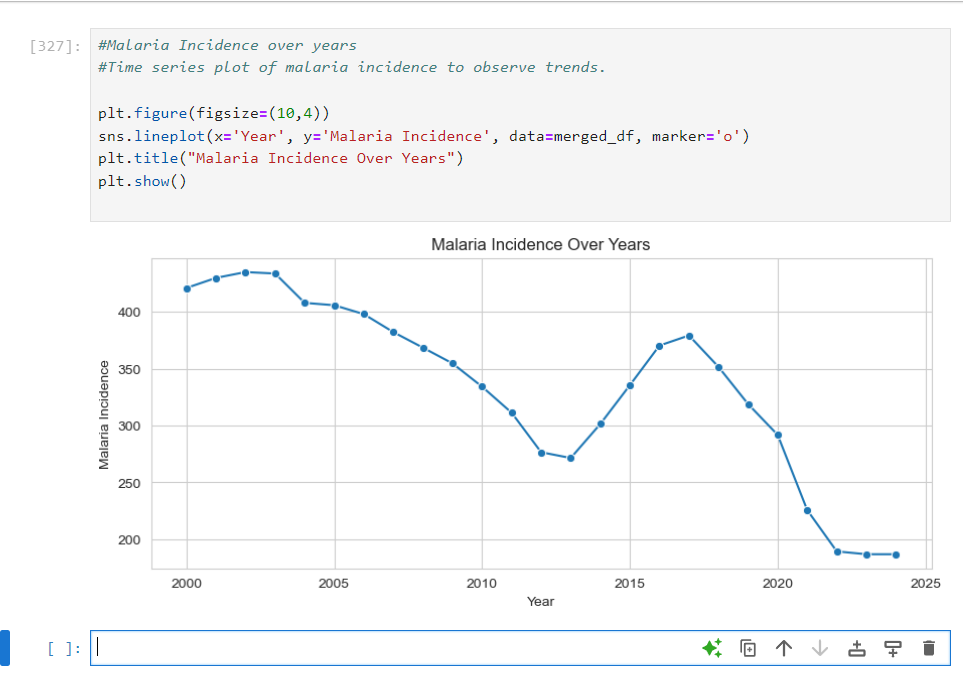
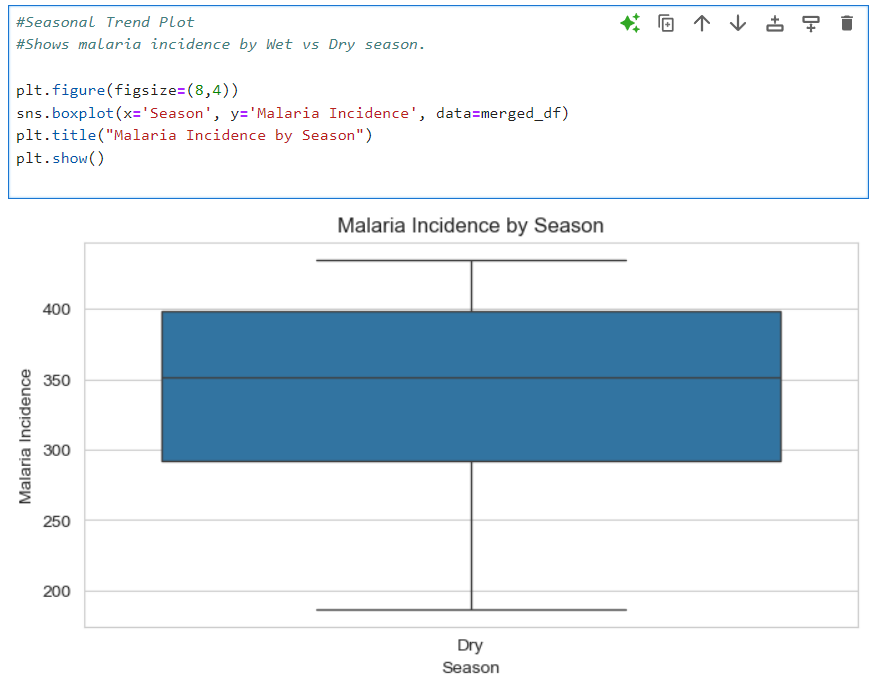
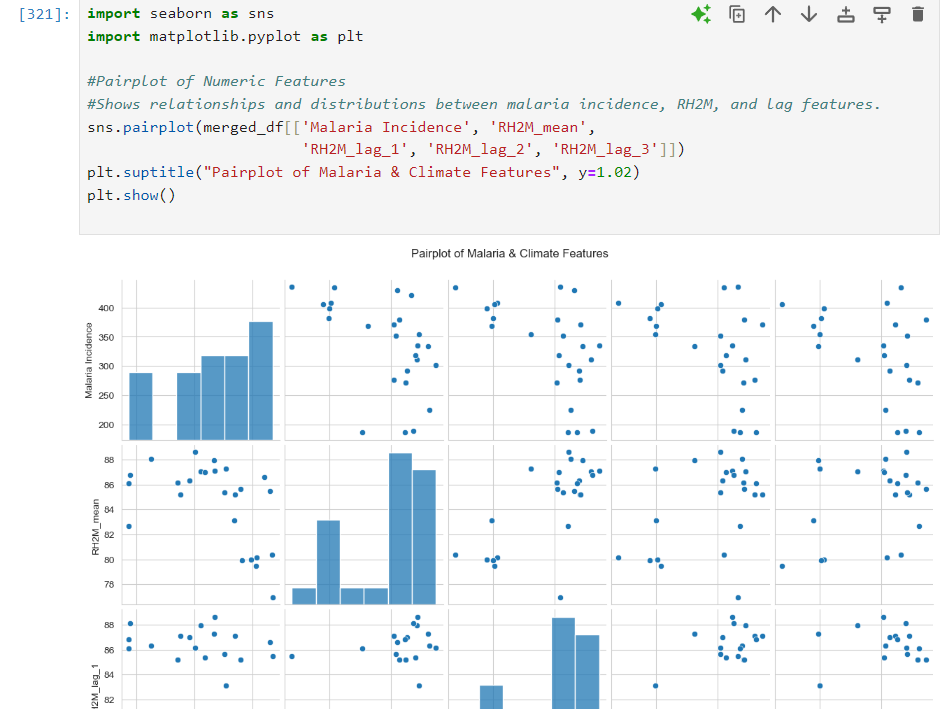
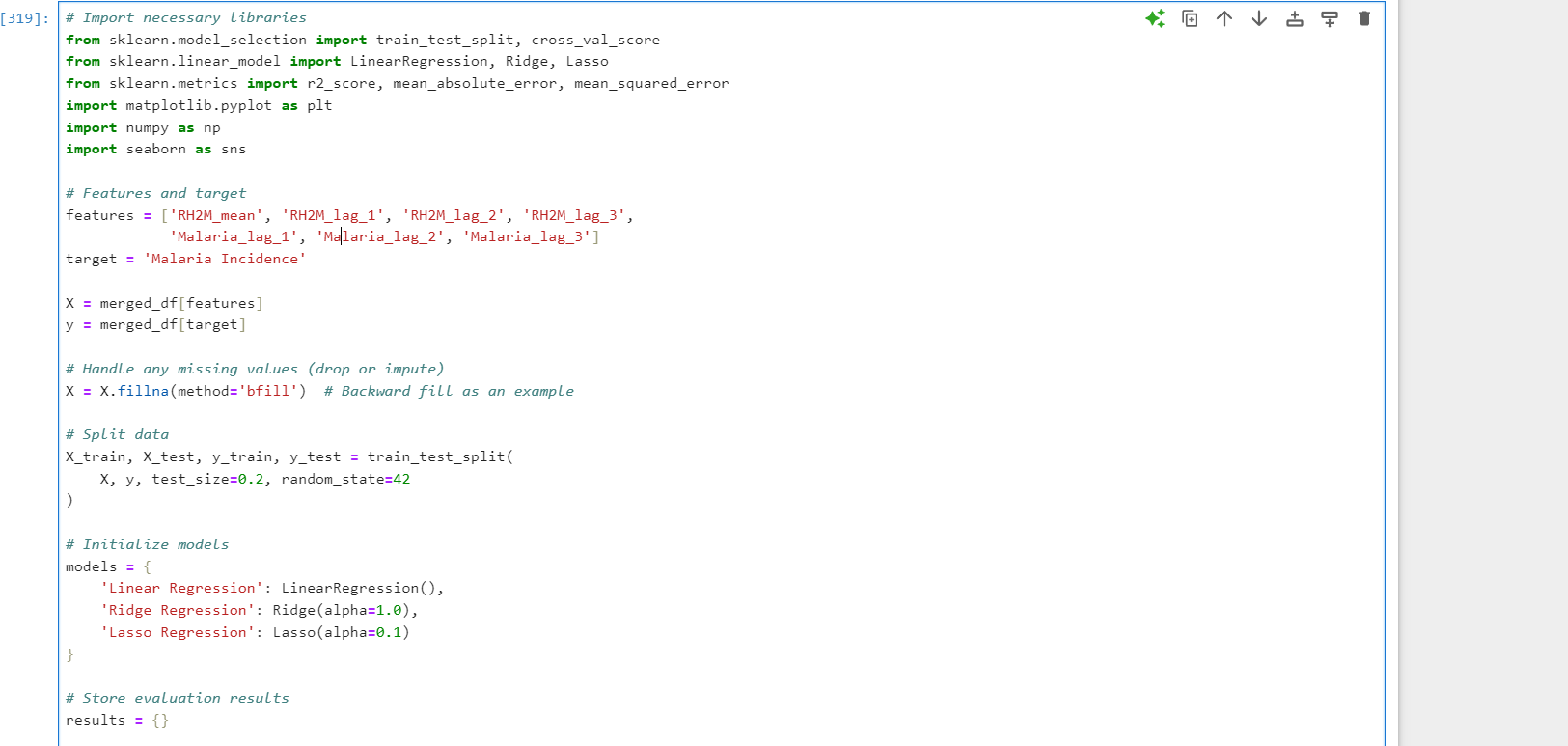
* **Predicted vs Actual Plot:** Scatter plot showing how closely predictions align with actual malaria incidence values.
* **Residual Plot:** Residuals (errors) plotted against predicted values to check for patterns or heteroscedasticity.
* **Feature Importance (if using tree-based models):** Bar chart showing which features contributed most to predictions.



**4. Code Implementation**

| **# ===============================**  **# 1. Import Libraries**  **# ===============================**  import pandas as pd  import numpy as np  import matplotlib.pyplot as plt  import seaborn as sns  from sklearn.model\_selection import train\_test\_split, cross\_val\_score  from sklearn.linear\_model import LinearRegression, Ridge, Lasso  **from sklearn.metrics import r2\_score, mean\_absolute\_error, mean\_squared\_error**  **# ===============================**  **# 2. Load & Merge Data**  **# ===============================**  **#** Assume we have malaria\_df and climate\_df  # malaria\_df columns: ['Year', 'Malaria Incidence', 'Population']  # climate\_df columns: ['Year', 'RH2M\_mean']  **# Merge datasets on Year**  merged\_df = pd.merge(malaria\_df, climate\_df, on='Year')  **# ===============================**  **# 3. Feature Engineering**  **# ===============================**  **# Lag features for climate and malaria**  for lag in range(1, 4):  merged\_df[f'RH2M\_lag\_{lag}'] = merged\_df['RH2M\_mean'].shift(lag)  merged\_df[f'Malaria\_lag\_{lag}'] = merged\_df['Malaria Incidence'].shift(lag)  **# Seasonal indicator (Wet/Dry)**  merged\_df['Season'] = np.where(merged\_df['Year'].isin([2000, 2001, 2004, 2005, 2008]), 'Wet', 'Dry')  **# Population-adjusted malaria incidence**  merged\_df['Malaria\_per\_1000'] = merged\_df['Malaria Incidence'] / merged\_df['Population'] \* 1000  **# Handle NaN from lag features**  merged\_df = merged\_df.fillna(method='bfill') # backfill  **# ===============================**  **# 4. EDA Visualizations**  **# ===============================**  **# 4a. Pairplot for numeric features**  sns.pairplot(merged\_df[['Malaria Incidence','RH2M\_mean','RH2M\_lag\_1','RH2M\_lag\_2','RH2M\_lag\_3']])  plt.show()  **# 4b. Correlation heatmap**  plt.figure(figsize=(8,6))  sns.heatmap(merged\_df.corr(), annot=True, cmap='coolwarm')  plt.title("Correlation Heatmap")  plt.show()  **# 4c. Seasonal trend plot**  plt.figure(figsize=(8,4))  sns.boxplot(x='Season', y='Malaria Incidence', data=merged\_df)  plt.title("Malaria Incidence by Season")  plt.show()  **# 4d. Time series trend**  plt.figure(figsize=(10,4))  sns.lineplot(x='Year', y='Malaria Incidence', data=merged\_df, marker='o')  plt.title("Malaria Incidence Over Years")  plt.show()  **# ===============================**  **# 5. Prepare Data for Modeling**  **# ===============================**  features = ['RH2M\_mean', 'RH2M\_lag\_1', 'RH2M\_lag\_2', 'RH2M\_lag\_3',  'Malaria\_lag\_1', 'Malaria\_lag\_2', 'Malaria\_lag\_3', 'Population']  target = 'Malaria\_per\_1000'  X = merged\_df[features]  y = merged\_df[target]  # Train-test split  X\_train, X\_test, y\_train, y\_test = train\_test\_split(  X, y, test\_size=0.2, random\_state=42  )  **# ===============================**  **# 6. Model Exploration & Evaluation**  **# ===============================**  models = {  'Linear Regression': LinearRegression(),  'Ridge Regression': Ridge(alpha=1.0),  'Lasso Regression': Lasso(alpha=0.1)  }  results = {}  for name, model in models.items():  # Train  model.fit(X\_train, y\_train)    **# Predict**  y\_pred = model.predict(X\_test)    **# Evaluation**  r2 = r2\_score(y\_test, y\_pred)  mae = mean\_absolute\_error(y\_test, y\_pred)  rmse = mean\_squared\_error(y\_test, y\_pred, squared=False)    results[name] = {'R2': r2, 'MAE': mae, 'RMSE': rmse}    **# Predicted vs Actual plot**  **plt.figure(figsize=(6,4))**  **sns.scatterplot(x=y\_test, y=y\_pred)**  **plt.plot([y\_test.min(), y\_test.max()], [y\_test.min(), y\_test.max()], 'r--')**  **plt.xlabel('Actual')**  **plt.ylabel('Predicted')**  **plt.title(f"{name} - Predicted vs Actual")**  **plt.show()**    **# Residual plot**  **residuals = y\_test - y\_pred**  **plt.figure(figsize=(6,4))**  **sns.scatterplot(x=y\_pred, y=residuals)**  **plt.axhline(0, color='r', linestyle='--')**  **plt.xlabel('Predicted')**  **plt.ylabel('Residuals')**  **plt.title(f"{name} - Residual Plot")**  **plt.show()**  **# Print results**  **for model\_name, metrics in results.items():**  **print(f"{model\_name}: R²={metrics['R2']:.3f}, MAE={metrics['MAE']:.2f}, RMSE={metrics['RMSE']:.2f}")** |
| --- |

**Please provide images from your data and models (I want to see different visualizations in the EDA part as we did a pair coding session )!!!!**

****