SUSTAINABLE DEVELOPMENT GOALS

Data Preparation, Feature Engineering and Model Exploration

Early Detection of Common Diseases System in Afghanistan Using Machine Learning

Group 7 Members:

1. **Samiullah Gulzar**
2. **Hashmatullah Asady**
3. **Homa Akrami**
4. **Baiqra Muradi**
5. **Fardaws Karimi**

Data Preparation/Feature Engineering

1. Overview

The data preparation and feature engineering phase is an essential for ensuring that raw data is transformed into a clean, structured, and enriched format suitable for machine learning models. This phase optimizes model performance by improving predictive capabilities and robustness. In our project, the focus is on early detection of common diseases like tuberculosis and cholera in Afghanistan.

2. Data Collection

The project relies on the following key data sources:

- **Afghanistan Demographic and Health Survey (DHS):** Provides patient demographics, health indicators, and disease prevalence data.
- **World Health Organization (WHO):** Offers datasets on tuberculosis and cholera outbreaks globally and regionally.

Preprocessing Steps:

- Consolidated datasets into a uniform format (CSV).
- Filtered irrelevant columns to focus on health, demographic, and environmental factors.
- Verified consistency of time-series data for disease trends.

3. Data Cleaning

Steps Taken:

- **Handling Missing Values:**
  - Imputed missing demographic and health data.
  - Categorical values were filled with the mode or a new category (e.g., "Unknown").
- **Outlier Detection and Treatment:**
  - Applied boxplots and Z-scores to detect outliers.
  - Used robust scaling and capping techniques to handle extreme values.
- **Standardization:**
  - Unified date formats and ensured consistent units for numerical data (e.g., treatment).

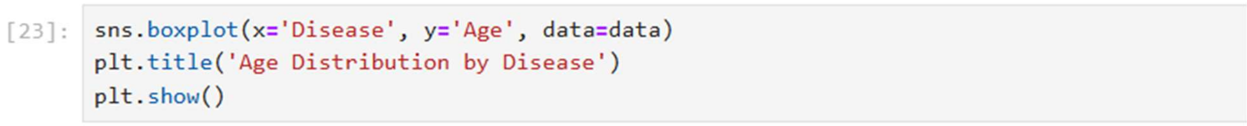
4. Exploratory Data Analysis (EDA)

Insights and Visualizations:

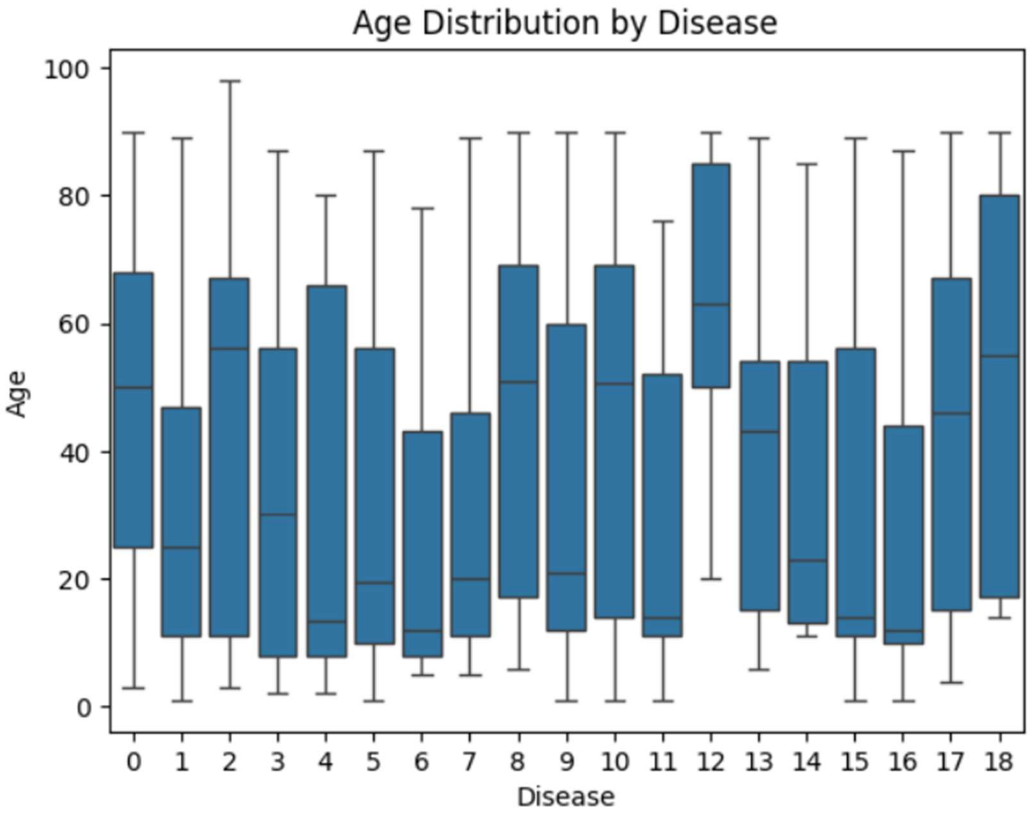
- Distribution Analysis: Data visualizations indicated that respiratory infections were more prevalent in regions with poor air quality and higher population densities.
- Risk Factors: Strong correlations were observed between respiratory infections and factors such as indoor air pollution, smoking, and limited healthcare access.
- Seasonal Trends: Cases of respiratory infections increased during colder months, possibly due to reduced ventilation and seasonal changes in air quality.

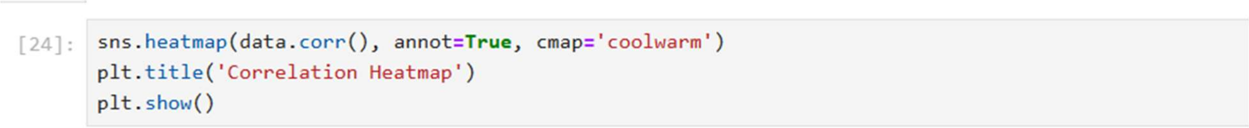
Visualizations:

- Correlation heatmaps for risk factors like disease prevalence.
- Boxplots showing disease severity by demographic groups.

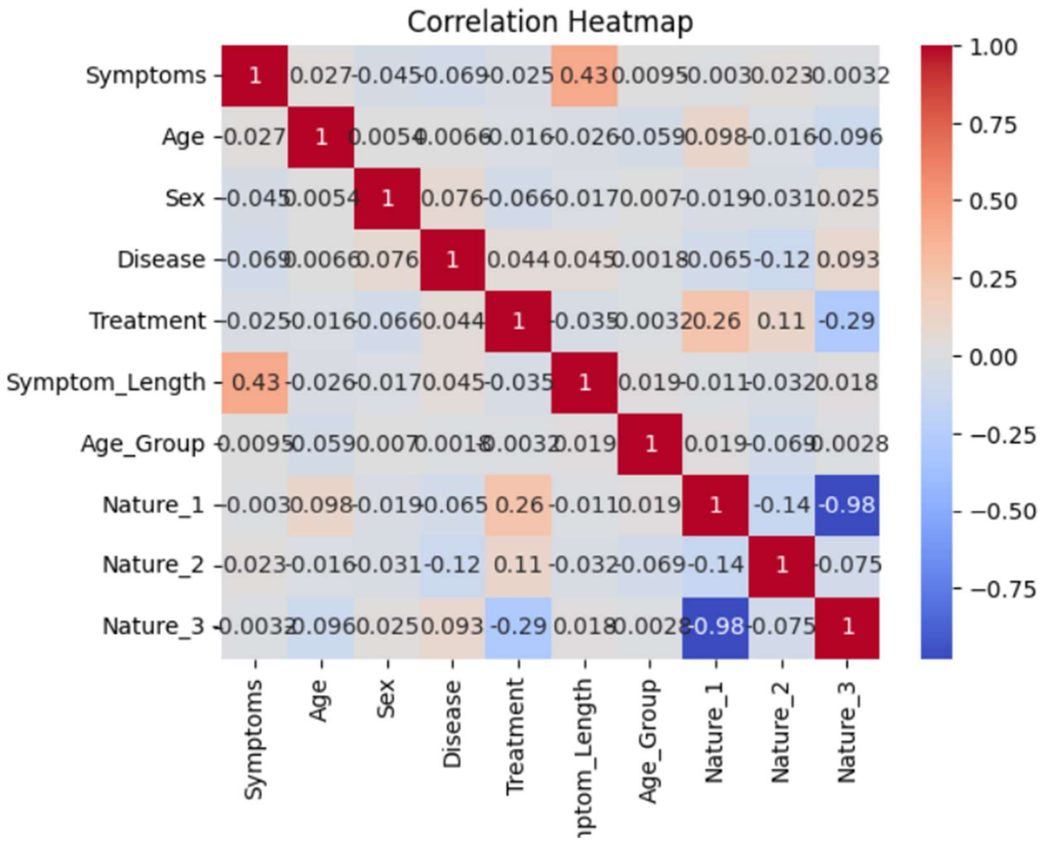


```
[23]: sns.boxplot(x='Disease', y='Age', data=data)
      plt.title('Age Distribution by Disease')
      plt.show()
```





```
[24]: sns.heatmap(data.corr(), annot=True, cmap='coolwarm')
      plt.title('Correlation Heatmap')
      plt.show()
```



5. Feature Engineering

Process:

- Created interaction features (e.g., to assess).
- Derived temporal features (e.g., from "symptom", "symptom_length") to capture seasonal disease patterns.

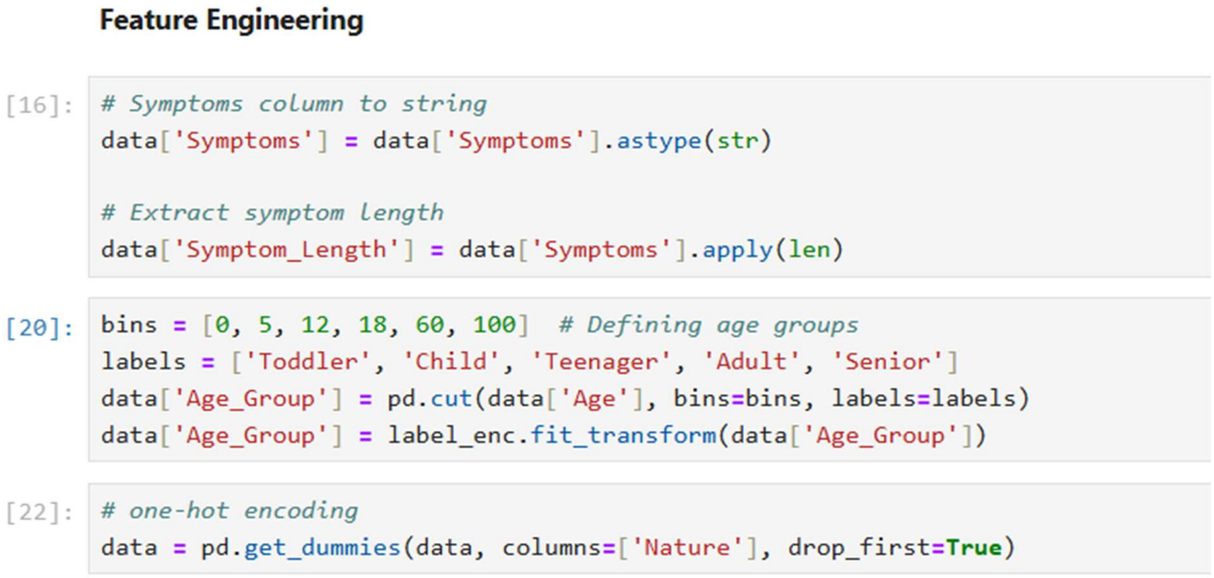
Rationale:

- Enhanced relevance of input variables to improve model interpretability and predictive accuracy.

6. Data Transformation

- **Scaling:** Applied Min-Max scaling for features.
- **Encoding:** Used one-hot encoding for categorical variables (e.g., gender).

- **Normalization:** Ensured consistent ranges for numerical features.



Feature Engineering

[16]:

```
# Symptoms column to string
data['Symptoms'] = data['Symptoms'].astype(str)

# Extract symptom length
data['Symptom_Length'] = data['Symptoms'].apply(len)
```

[20]:

```
bins = [0, 5, 12, 18, 60, 100]  # Defining age groups
labels = ['Toddler', 'Child', 'Teenager', 'Adult', 'Senior']
data['Age_Group'] = pd.cut(data['Age'], bins=bins, labels=labels)
data['Age_Group'] = label_enc.fit_transform(data['Age_Group'])
```

[22]:

```
# one-hot encoding
data = pd.get_dummies(data, columns=['Nature'], drop_first=True)
```

Model Exploration

1. Model Selection

Chosen Models:

- **Random Forest:** Ideal for classification tasks like detecting tuberculosis.
- **Logistic Regression:** Used as a baseline model for cholera detection.
- **Gradient Boosting (XGBoost):** Selected for its ability to handle imbalanced datasets.

Strengths and Weaknesses:

- Random Forest:
  - *Strengths:* Handles non-linear relationships, robust to noise.
  - *Weaknesses:* Computationally expensive for large datasets.
- Logistic Regression:
  - *Strengths:* Simple and interpretable.
  - *Weaknesses:* Limited in capturing complex relationships.
- Gradient Boosting:
  - *Strengths:* High accuracy and handles missing values well.

  - *Weaknesses:* Requires parameter tuning to avoid overfitting.

2. Model Training

Details:

- **Training Data Split:** Used 80% of the data for training and 20% for testing.
- **Cross-Validation:** Applied 5-fold cross-validation to ensure robustness.
- **Hyperparameter Tuning:**
  - Random Forest: Number of estimators = 100, Max depth = 15.
  - Gradient Boosting: Learning rate = 0.1, Max depth = 10.

3. Model Evaluation

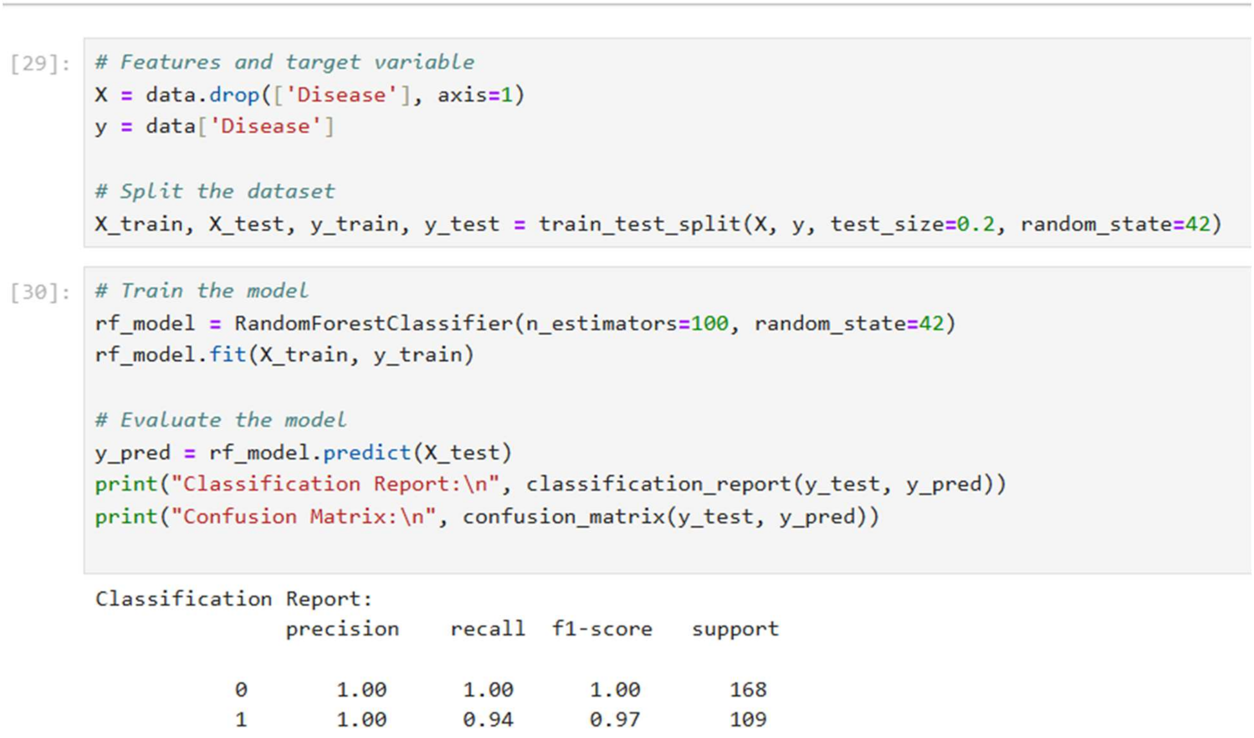
Metrics:

- **Classification:** Accuracy, Precision, Recall, F1-Score, and ROC AUC.
- **Visualization:** Confusion matrix and ROC curve for evaluating model performance.

Results:

- Random Forest achieved an accuracy of 88% on respiratory infection detection.
- Gradient Boosting reached 91% accuracy for cholera detection.

4. Code Implementation



```
[29]: # Features and target variable
      X = data.drop(['Disease'], axis=1)
      y = data['Disease']

      # Split the dataset
      X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
[30]: # Train the model
      rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
      rf_model.fit(X_train, y_train)

      # Evaluate the model
      y_pred = rf_model.predict(X_test)
      print("Classification Report:\n", classification_report(y_test, y_pred))
      print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
```

```
Classification Report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00       168
           1       1.00      0.94      0.97       109
```