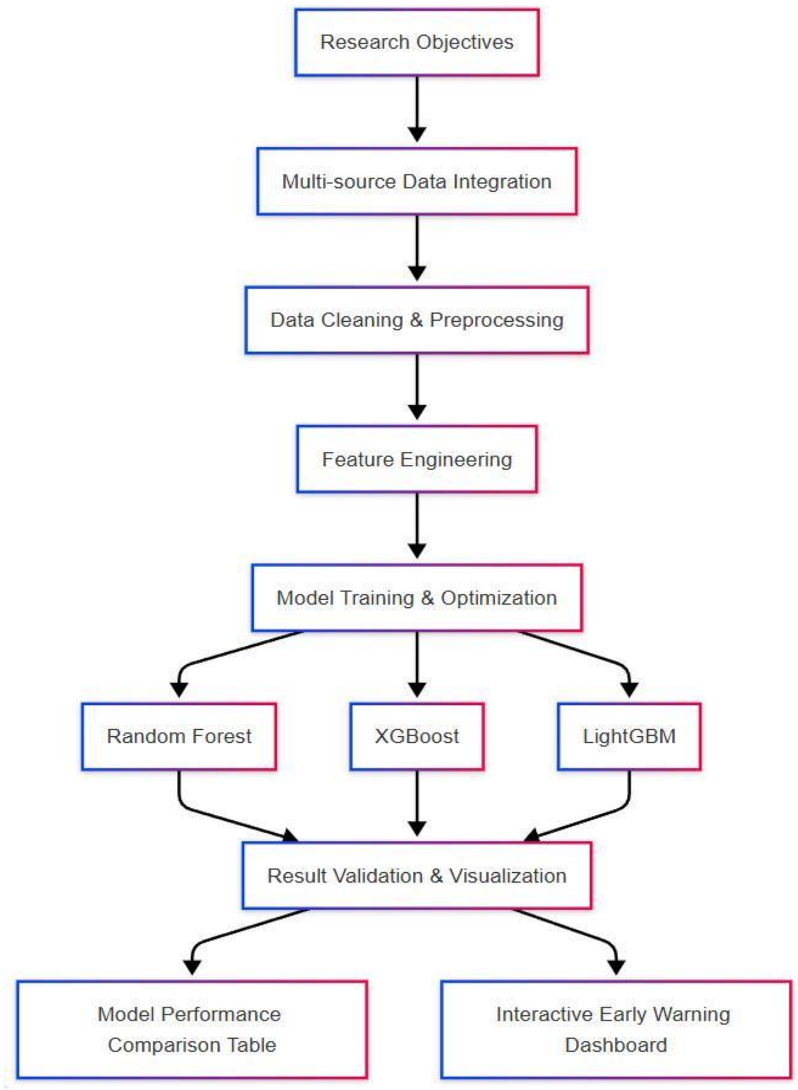
RESEARCH ON SOIL ENVIRONMENTAL AND HEALTH RISK ANALYSIS BASED ON MACHINE LEARNING.

ZHAO ZHIHAN

UNIVERSITI TEKNOLOGI MALAYSIA

3.1 The Framework

The details of the research framework for this study are shown in the Figure below.


Research Objectives
Multi-source Data Integration
Data Cleaning & Preprocessing
Feature Engineering
Model Training & Optimization
Random Forest
XGBoost
LightGBM
Result Validation & Visualization
Model Performance Comparison Table
Interactive Early Warning Dashboard

3.2 Problem Statement

With the rapid advancement of industrialization and agricultural modernization, soil pollution has become a globally - concerned environmental and health issue. However, the spatial distribution characteristics of soil pollutants (such as heavy metals and pesticides) and the specific associated mechanisms with human health effects remain unclear. Based on a dataset of soil pollution and health cases covering multiple regions around the world, this study attempts to analyze the distribution patterns of different pollutant types in soil and the differences in their health impacts on people of various age groups (children, adults, and the elderly).

The data shows that there are significant outliers in the concentrations of soil pollutants in some regions, suggesting the possible existence of high - risk scenarios with intensive industrial activities or excessive use of agricultural chemicals. This study intends to focus on the following questions: first, whether there is a spatial aggregation relationship between the concentrations of soil pollutants and the nearby industrial types and agricultural practices; second, whether the exposure to different pollutants is significantly associated with the risk of specific disease types (such as neurological diseases and gastrointestinal diseases); third, how the physical and chemical properties of soil, such as pH value and organic matter content, affect the bioavailability of pollutants. By revealing the driving factors of health risks from soil pollution, this study aims to provide preliminary data support for environmental governance and population health protection in high - risk areas, and lay the foundation for subsequent in - depth risk modeling and policy - making.

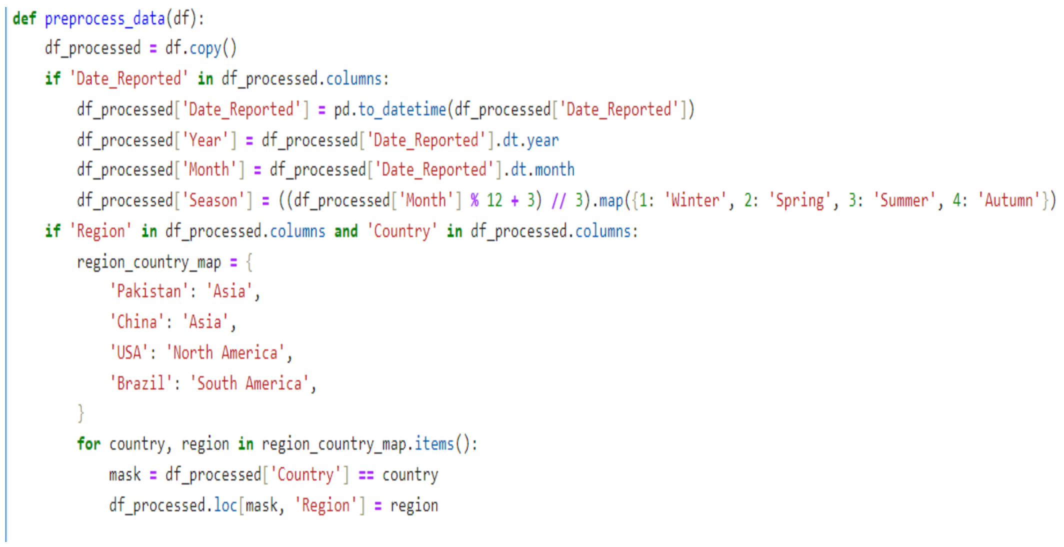
3.3 Data Collection

The data for this study is sourced from the publicly available Kaggle dataset titled "Soil Pollution and Health Impact Cases". This dataset integrates soil pollution monitoring data and health case records from multiple regions, spanning the time period from 2023 to 2024. It contains 3,000 samples and 24 variables, covering multi - dimensional information such as soil pollutant concentrations, meteorological conditions, agricultural practices, and disease types.

3.4 Data Pre-processing

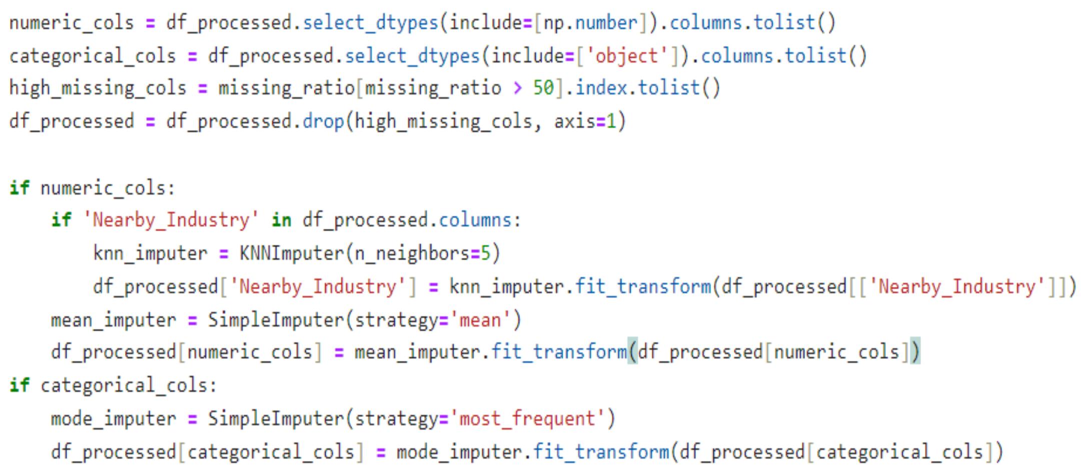
It is necessary to complete preliminary analysis prior to moving on to further pre-processing. A data merging procedure is necessary to bring all of the raw data into one data frame once we have a firm grasp of the features provided in the dataset. Several data wrangling and data transformation procedures will be used on the dataset in an effort to further unify the disorganised raw data.

Data type conversion: Although some columns store specific types of data such as dates and classifications, their current data type is "object". It may be necessary to convert them into appropriate data types for subsequent analysis. For example, "Date Reported" may need to be converted into a date type.



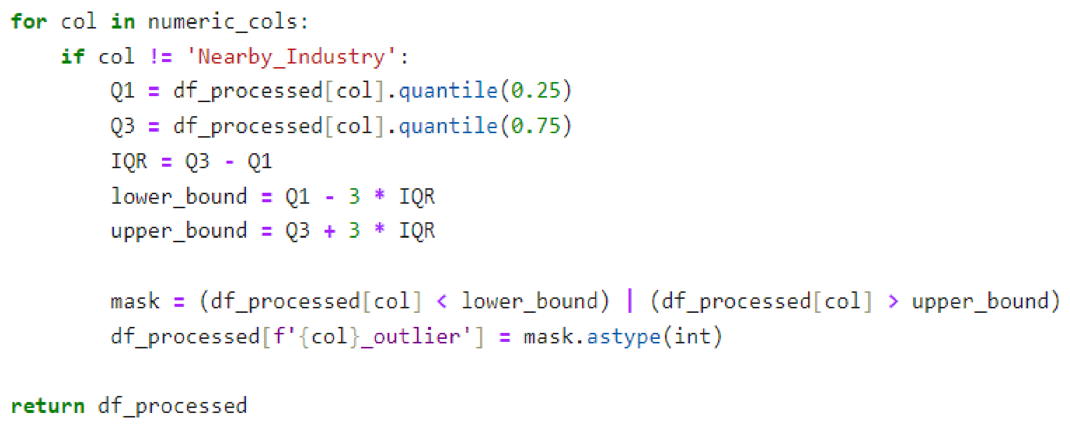
```python
def preprocess_data(df):
    df_processed = df.copy()
    if 'Date_Reported' in df_processed.columns:
        df_processed['Date_Reported'] = pd.to_datetime(df_processed['Date_Reported'])
        df_processed['Year'] = df_processed['Date_Reported'].dt.year
        df_processed['Month'] = df_processed['Date_Reported'].dt.month
        df_processed['Season'] = ((df_processed['Month'] % 12 + 3) // 3).map({1: 'Winter', 2: 'Spring', 3: 'Summer', 4: 'Autumn'})
    if 'Region' in df_processed.columns and 'Country' in df_processed.columns:
        region_country_map = {
            'Pakistan': 'Asia',
            'China': 'Asia',
            'USA': 'North America',
            'Brazil': 'South America',
        }
        for country, region in region_country_map.items():
            mask = df_processed['Country'] == country
            df_processed.loc[mask, 'Region'] = region
```

2.Missing value handling: There are missing values in the "Nearby dustry" column. It is necessary to select appropriate methods for handling according to the characteristics of the data and analysis requirements, such as deleting rows with missing values or filling in the missing values.



```
numeric_cols = df_processed.select_dtypes(include=[np.number]).columns.tolist()
categorical_cols = df_processed.select_dtypes(include=['object']).columns.tolist()
high_missing_cols = missing_ratio[missing_ratio > 50].index.tolist()
df_processed = df_processed.drop(high_missing_cols, axis=1)

if numeric_cols:
    if 'Nearby_Industry' in df_processed.columns:
        knn_imputer = KNNImputer(n_neighbors=5)
        df_processed['Nearby_Industry'] = knn_imputer.fit_transform(df_processed[['Nearby_Industry']])
    mean_imputer = SimpleImputer(strategy='mean')
    df_processed[numeric_cols] = mean_imputer.fit_transform(df_processed[numeric_cols])
if categorical_cols:
    mode_imputer = SimpleImputer(strategy='most_frequent')
    df_processed[categorical_cols] = mode_imputer.fit_transform(df_processed[categorical_cols])
```

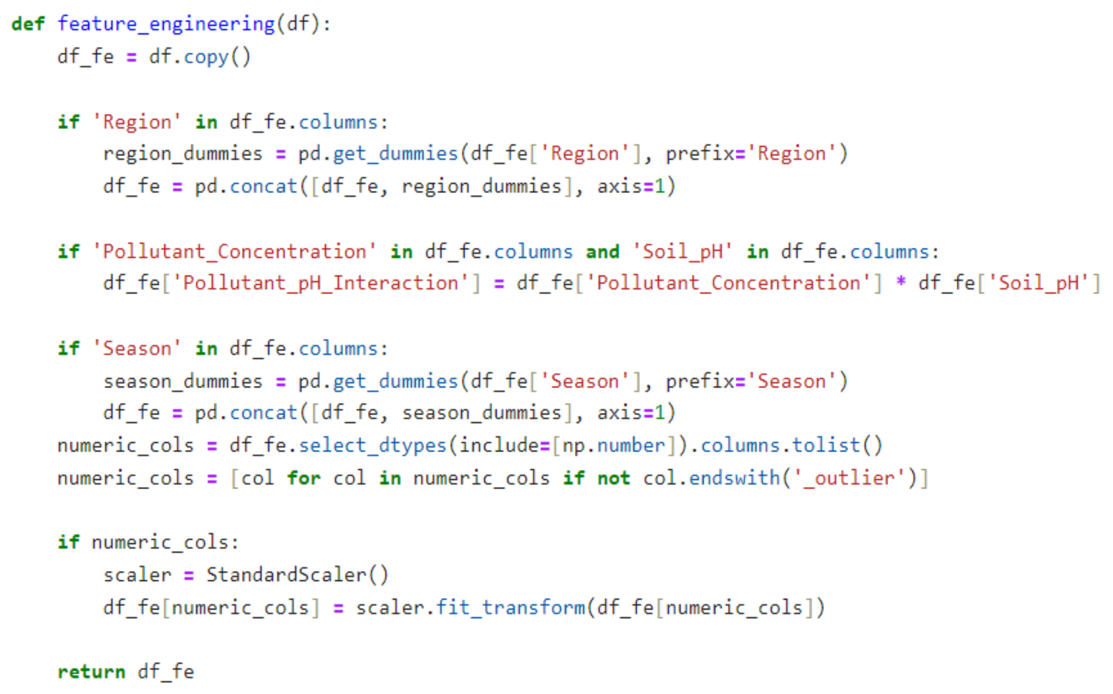
3. Handle outliers: Calculate the interquartile range (IQR) of the data and mark the outliers.



```
for col in numeric_cols:
    if col != 'Nearby_Industry':
        Q1 = df_processed[col].quantile(0.25)
        Q3 = df_processed[col].quantile(0.75)
        IQR = Q3 - Q1
        lower_bound = Q1 - 3 * IQR
        upper_bound = Q3 + 3 * IQR

        mask = (df_processed[col] < lower_bound) | (df_processed[col] > upper_bound)
        df_processed[f'{col}_outlier'] = mask.astype(int)

return df_processed
```

3.5 Feature Engineering

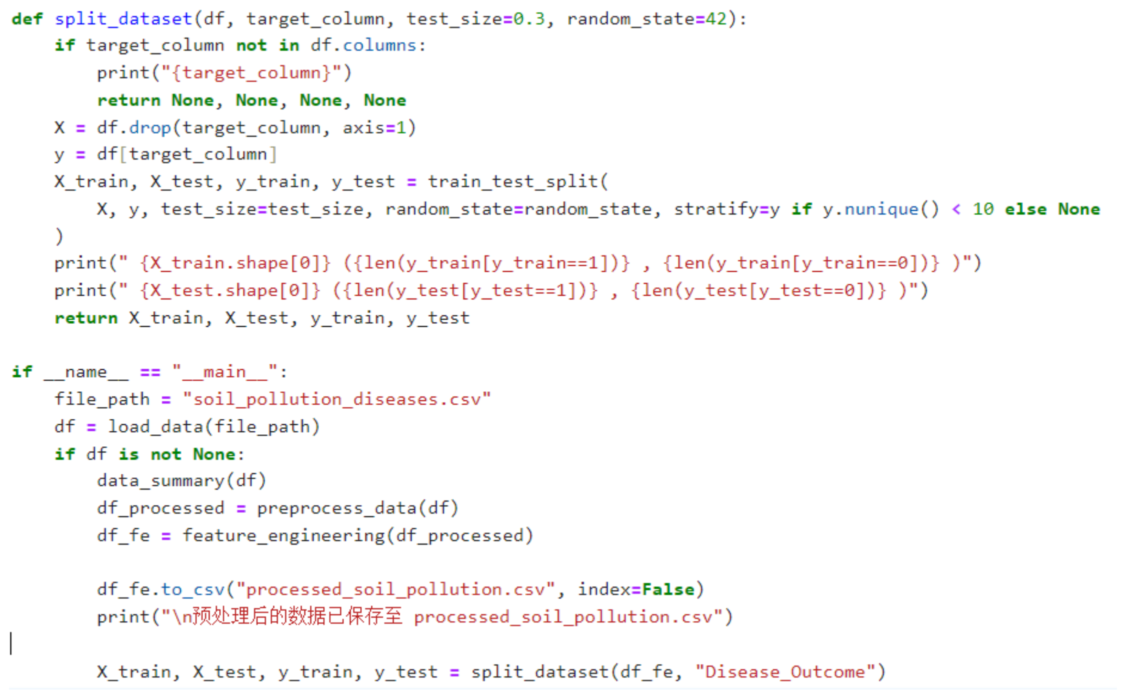
First of all, feature engineering can improve the performance of the model and uncover implicit relationships from the original data. In addition, feature selection can reduce noise, making the model more focused and easier to screen out more useful information.



```python
def feature_engineering(df):
    df_fe = df.copy()

    if 'Region' in df_fe.columns:
        region_dummies = pd.get_dummies(df_fe['Region'], prefix='Region')
        df_fe = pd.concat([df_fe, region_dummies], axis=1)

    if 'Pollutant_Concentration' in df_fe.columns and 'Soil_pH' in df_fe.columns:
        df_fe['Pollutant_pH_Interaction'] = df_fe['Pollutant_Concentration'] * df_fe['Soil_pH']

    if 'Season' in df_fe.columns:
        season_dummies = pd.get_dummies(df_fe['Season'], prefix='Season')
        df_fe = pd.concat([df_fe, season_dummies], axis=1)
    numeric_cols = df_fe.select_dtypes(include=[np.number]).columns.tolist()
    numeric_cols = [col for col in numeric_cols if not col.endswith('_outlier')]

    if numeric_cols:
        scaler = StandardScaler()
        df_fe[numeric_cols] = scaler.fit_transform(df_fe[numeric_cols])

    return df_fe
```

3.6 Data Modelling

Split the data into a 7:3 ratio for training and testing respectively. This enables the quantification of the non - linear relationship between pollution and safety, identification of key risk factors and more vulnerable populations. As a result, it becomes possible to conduct a quantitative analysis of the health effects of soil pollution, identify key factors, and predict risks.



```python
def split_dataset(df, target_column, test_size=0.3, random_state=42):
    if target_column not in df.columns:
        print("{target_column}")
        return None, None, None, None
    X = df.drop(target_column, axis=1)
    y = df[target_column]
    X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=test_size, random_state=random_state, stratify=y if y.nunique() < 10 else None
    )
    print(" {X_train.shape[0]} ({len(y_train[y_train==1])} , {len(y_train[y_train==0])} )")
    print(" {X_test.shape[0]} ({len(y_test[y_test==1])} , {len(y_test[y_test==0])} )")
    return X_train, X_test, y_train, y_test

if __name__ == "__main__":
    file_path = "soil_pollution_diseases.csv"
    df = load_data(file_path)
    if df is not None:
        data_summary(df)
        df_processed = preprocess_data(df)
        df_fe = feature_engineering(df_processed)

        df_fe.to_csv("processed_soil_pollution.csv", index=False)
        print("\n预处理后的数据已保存至 processed_soil_pollution.csv")

        X_train, X_test, y_train, y_test = split_dataset(df_fe, "Disease_Outcome")
```