AIML ASSIGNMENT

NAME-ADITYA PRATAP SINGH SISODIA

BATCH-11

SAP ID- 500121985

R2142230333

Libraries and Modules

Pandas (pd): For data manipulation and to load the dataset from Google Drive. It makes many tasks such as reading in the CSV, handling missing values, and manipulating columns much easier.

Scikit-learn Modules:

train_test_split: This function is used to split your data into training and test sets if you need them (although it isn't used here).

KFold: It provides K-fold cross-validation to help you get an idea of how a model performs on different subsets of your dataset.

cross_val_predict: This cross-validation estimator produces cross-validated estimates of model predictions on each fold, which facilitates calculating metrics over the whole dataset in a consistent manner.

GridSearchCV: This class is used for hyperparameter optimization. It tests several parameter combinations and finds the best one based on cross-validation scores.

RF and LR: These are the two classification algorithms selected for this model. RF is a tree-based ensemble model, while LR is a simple, linear model. It is truly interesting to see how a complex model may do compared to a simpler model.

Metrics: accuracy_score, precision_score, recall_score, and many others. They show how the model has done; overall accuracy, precision, recall, F1, AUC is considered as these provide balanced estimation of prediction quality.

Imbalanced-learn: SMOTE

```
# Apply SMOTE to handle class imbalance
smote = SMOTE()
X_resampled, y_resampled = smote.fit_resample(X, y)
```

SMOTE stands for Synthetic Minority Over-sampling Technique: It uses over sampling of synthetic examples in the minority class to balance out the classes. Since Imbalanced datasets-eg-fewer high foodwaste compared to the many low foodwaste countries-can make the models biased toward majority class, SMOTE helps the patterns learn both in classes.

LabelEncoder: It converts the categorical labels into numeric numbers, which is the requirement for the machine learning algorithms that need numeric input. It has been used here to assign country names with integers.

StandardScaler: This module standardizes features by subtracting the mean and scaling to unit variance. The reason for doing this is so that all features are on approximately the same scale. In many machine learning algorithms (for example, Logistic Regression), this prevents a dominant feature from dominating the entire model because of its scale.

Loading and Preprocessing the Data:

```
# Load the dataset
file_path = '/content/drive/My Drive/google collab/Food Waste data and research - by country.csv'
df = pd.read_csv(file_path)

# Preprocessing
# Fill missing values
numeric_df = df.select_dtypes(include=['number'])
df[numeric_df.columns] = numeric_df.fillna(numeric_df.mean())
```

Missing Value Handling: Fill missing values in numeric columns by mean, not changing the dataset and would bring no errors during model training.

Categorical Variable Encoding: Label Encoding is done because machine learning algorithms work only on numerical data. Thus convert categorical text data into a number

Target and Features Definition:

```
# Define target and features
threshold = 100  # Example threshold to categorize high/low food waste
df['food_waste_category'] = (df['combined figures (kg/capita/year)'] > threshold).astype(int)  # 1 for high, 0 for low
X = df.drop(['food_waste_category', 'Country', 'Confidence in estimate', 'M49 code', 'Region', 'Source'], axis=1)
y = df['food_waste_category']
```

Threshold for Target: When the target variable is of combined figure, for making an instance as high food waste or low food waste, a threshold is assigned to the target variable which turns this problem into a binary classification problem. This makes the problem even simpler and helps in finding the performance of a model.

Removing Unnecessary Columns: All the irrelevant columns like Country and Confidence in estimate are removed so that no noise comes into the model and it is ensured that only those features are considered which are relevant.

Cross-validation and Hyperparameter Tuning:

```
# K-Fold Cross Validation and Hyperparameter Tuning
kf = KFold(n_splits=5, shuffle=True, random_state=42)
```

```
# Cross-validation and metrics calculation
results = {}
for name, model in models.items():
    y_pred = cross_val_predict(model, X_resampled, y_resampled, cv=kf)
    accuracy = accuracy_score(y_resampled, y_pred)
    precision = precision_score(y_resampled, y_pred)
    recall = recall_score(y_resampled, y_pred)
    f1 = f1_score(y_resampled, y_pred)
    roc_auc = roc_auc_score(y_resampled, y_pred)
    conf_matrix = confusion_matrix(y_resampled, y_pred)

    results[name] = {
        'Accuracy': accuracy,
        'Precision': precision,
        'Recall': recall,
        'F1 Score': f1,
        'ROC AUC': roc_auc,
        'Confusion Matrix': conf_matrix
    }
```

K-Fold Cross Validation (KFold): This is used in getting a more generalized measure of performance by averaging across multiple splits of the data such that there is no overfitting to a single data split. Hyperparameter Tuning by GridSearchCV: Optimization of the parameters for the best models, such as n_estimators in Random Forest, C in Logistic Regression, and so on, leading to an improvement in performance of the model. Evaluation Metrics:

Accuracy, Precision, Recall, F1, ROC AUC: Each evaluation metric tells you something specific about how well your models are doing. For example:

Accuracy: Proportion of correct labels predicted

Precision and Recall: Balancing false positives and false negatives

F1 Score: A combination of Precision and Recall. One single performance metric

ROC AUC: Aggregate performance measure across all classification thresholds; useful if you are dealing with imbalanced datasets

Confusion Matrix: True positives, False positives, False negatives, True negatives; it diagnoses the error in predictions.

ABOUT THE DATASET

Food waste statistics per country. For the columns below: Country Country names. Combined figures (kg/capita/year) Total yearly per capita food loss in kg. Household estimate (kg/capita/year) Household yearly per capita food loss in kg. Household estimate (tonnes/year) Household yearly food waste in tonnes. Retail estimate (kg/capita/year) Food waste retail yearly per capita in kg.

Retail estimate (tonnes/year): Total retail food waste in tonnes per year.

Food service estimate (kg/capita/year): Food service sector food waste per capita per year, in kilogrammes/capita/year.

Food service estimate (tonnes/year): Food waste in the food service sector in tonnes per year

Confidence in estimate: Level of confidence in the food waste estimates: e.g. "Very Low Confidence," "Low Confidence."

M49 code: The UN's M49 code for each country.

Region: The region of the country (e.g., "Southern Asia," "Northern Africa").

This data is organized by country and by stage (household, retail, food service) of food waste: it includes both per capita and total annual waste, as well as information about the reliability of the estimates.