Language Detection with Machine Learning

Language detection is a crucial task in natural language processing (NLP) that involves identifying the language of a given piece of text. This capability is particularly important in today's globalized environment where digital platforms often handle multilingual content. Machine learning, especially with the availability of extensive language data, has become a powerful tool to automatically and accurately detect languages. This process not only supports translation services like Google Translate but also enhances content moderation, targeted advertising, and customer support across different regions.

Workflow for Building a Language Detection Model

The workflow for developing a machine learning model for language detection typically involves the following steps:

1. **Data Collection**: Gather a dataset containing text samples labeled with their corresponding languages. This dataset should ideally be balanced across the languages of interest.
2. **Data Preprocessing**: Clean and prepare the data for modeling. This includes handling missing values, removing noise, and converting text data into a suitable format for machine learning models using techniques like tokenization and vectorization.
3. **Feature Extraction**: Use techniques like Count Vectorization to convert text into numerical features that machine learning models can process.
4. **Model Selection**: Choose a suitable machine learning algorithm for language detection. Multinomial Naive Bayes is a popular choice due to its effectiveness in dealing with discrete features and its efficiency with large datasets.
5. **Training**: Train the chosen model on the prepared dataset. This involves feeding the model with training data so it can learn to associate the features with the corresponding language labels.
6. **Evaluation**: Test the model's performance using a separate set of data (test set). This helps in assessing the accuracy and generalizability of the model.
7. **Deployment and Prediction**: Once trained and validated, the model can be deployed to make predictions in real-time. This could involve integrating the model into applications that require language detection.

By following these steps, you can build a robust machine learning-based system for detecting languages in texts, enabling applications to better handle and analyze multilingual data.

In [3]:

```python
import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB

# Load dataset from the provided URL
data = pd.read_csv("C:/Users/anike/OneDrive/Desktop/Projects/Machine Learning/Language Detection/dataset.csv")
print(data.head())
```

```
                                                Text  language
0  klement gottwaldi surnukeha palsameeriti ning ...  Estonian
1  sebes joseph pereira thomas  på eng the jesuit...   Swedish
2  ถนนเจริญกรุง อักษรโรมัน thanon charoen krung เ...      Thai
3  விசாகப்பட்டினம் தமிழ்ச்சங்கத்தை இந்துப் பத்திர...     Tamil
4  de spons behoort tot het geslacht haliclona en...     Dutch
```

In [4]:

```python
# Checking for any null values in the dataset
print(data.isnull().sum())
```

```
Text        0
language    0
dtype: int64
```

In [5]:

```python
# Display the count of each language in the dataset
print(data["language"].value_counts())
```

```
Estonian      1000
Swedish       1000
English       1000
Russian       1000
Romanian      1000
Persian       1000
Pushto        1000
Spanish       1000
Hindi         1000
Korean        1000
Chinese       1000
French        1000
Portugese     1000
Indonesian    1000
Urdu          1000
Latin         1000
Turkish       1000
Japanese      1000
Dutch         1000
Tamil         1000
Thai          1000
Arabic        1000
Name: language, dtype: int64
```

In [7]:

```
# Extract features and labels from the dataset
x = data["Text"]
y = data["language"]

# Create a CountVectorizer instance to convert text data into vectors
cv = CountVectorizer()

# Transform the text data into feature vectors
X = cv.fit_transform(x)
```

In [8]:

```
# Splitting the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)
```

In [9]:

```
# Initialize and train the Multinomial Naive Bayes classifier
model = MultinomialNB()
model.fit(X_train, y_train)

# Evaluate the model on the test data
print("Model accuracy on test set:", model.score(X_test, y_test))
```

```
Model accuracy on test set: 0.953168044077135
```

In [20]:

```
# Function to predict the language of a given text
def predict_language(text):
    transformed_text = cv.transform([text]).toarray()
    prediction = model.predict(transformed_text)
    return prediction[0]

# Taking user input and predicting the language
user_input = input("Enter a Text: ")
predicted_language = predict_language(user_input)
print(f"The language of the text is: {predicted_language}")
```

```
Enter a Text: 某物
The language of the text is: Chinese
```