```
!pip install unidecode
```

```
Collecting unidecode
  Downloading Unidecode-1.3.8-py3-none-any.whl.metadata (13 kB)
Downloading Unidecode-1.3.8-py3-none-any.whl (235 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 235.5/235.5 kB 5.1 MB/s eta 0:00:00
Installing collected packages: unidecode
Successfully installed unidecode-1.3.8
```

```
!pip install datasets
```

```
Collecting datasets
  Downloading datasets-3.0.1-py3-none-any.whl.metadata (20 kB)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-packages (f
Collecting pyarrow>=15.0.0 (from datasets)
  Downloading pyarrow-17.0.0-cp310-cp310-manylinux_2_28_x86_64.whl.metadata (3.3 kB)
Collecting dill<0.3.9,>=0.3.0 (from datasets)
  Downloading dill-0.3.8-py3-none-any.whl.metadata (10 kB)
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (from d
Requirement already satisfied: requests>=2.32.2 in /usr/local/lib/python3.10/dist-packag
Requirement already satisfied: tqdm>=4.66.3 in /usr/local/lib/python3.10/dist-packages (
Collecting xxhash (from datasets)
  Downloading xxhash-3.5.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.me
Collecting multiprocess (from datasets)
  Downloading multiprocess-0.70.16-py310-none-any.whl.metadata (7.2 kB)
Requirement already satisfied: fsspec<=2024.6.1,>=2023.1.0 in /usr/local/lib/python3.10/
Requirement already satisfied: aiohttp in /usr/local/lib/python3.10/dist-packages (from
Requirement already satisfied: huggingface-hub>=0.22.0 in /usr/local/lib/python3.10/dist
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (fro
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-packages (f
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in /usr/local/lib/python3.10/dist
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.10/dist-packag
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.10/dist-packa
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.10/dist-pac
Requirement already satisfied: yarl<2.0,>=1.0 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: async-timeout<5.0,>=4.0 in /usr/local/lib/python3.10/dist
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.10/d
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dis
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-pack
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-pack
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from
Downloading datasets-3.0.1-py3-none-any.whl (471 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 471.6/471.6 kB 12.7 MB/s eta 0:00:00
Downloading dill-0.3.8-py3-none-any.whl (116 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 116.3/116.3 kB 7.2 MB/s eta 0:00:00
Downloading pyarrow-17.0.0-cp310-cp310-manylinux_2_28_x86_64.whl (39.9 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 39.9/39.9 MB 16.2 MB/s eta 0:00:00
```

```
Downloading multiprocess-0.70.16-py310-none-any.whl (134 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 134.8/134.8 kB 8.9 MB/s eta 0:00:00
Downloading xxhash-3.5.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (194
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 194.1/194.1 kB 8.9 MB/s eta 0:00:00
Installing collected packages: xxhash, pyarrow, dill, multiprocess, datasets
  Attempting uninstall: pyarrow
    Found existing installation: pyarrow 14.0.2
    Uninstalling pyarrow-14.0.2:
      Successfully uninstalled pyarrow-14.0.2
ERROR: pip's dependency resolver does not currently take into account all the packages t
cudf-cu12 24.4.1 requires pyarrow<15.0.0a0,>=14.0.1, but you have pyarrow 17.0.0 which i
Successfully installed datasets-3.0.1 dill-0.3.8 multiprocess-0.70.16 pyarrow-17.0.0 xxh
```

```python
import json
import os
from unidecode import unidecode
```

TASK 1

Sentences were extracted for 27 languages that use Roman-script characters.Each language is identified by a locale (e.g., 'en-US' for English spoken in the US). Sentences were filtered out for the required locales and each locale's sentences were saved in a separate text file. Deaccenting: Some languages contain accented characters, which were normalized using Python's unidecode library, depending on the desired preprocessing.

```python
from datasets import load_dataset
dataset = load_dataset("qanastek/MASSIVE")
```

```
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_token.py:89: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https:/
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public model
  warnings.warn(
MASSIVE.py: 100%                                            32.3k/32.3k [00:00<00:00, 660kB/s]

README.md: 100%                                             34.1k/34.1k [00:00<00:00, 1.26MB/s]

The repository for qanastek/MASSIVE contains custom code which must be executed to corre
You can avoid this prompt in future by passing the argument `trust_remote_code=True`.

Do you wish to run the custom code? [y/N] y
Downloading data: 100%                                      39.5M/39.5M [00:01<00:00, 42.6MB/s]

Generating train split:      587214/0 [03:57<00:00, 1615.58 examples/s]

Generating validation split:      103683/0 [00:49<00:00, 2657.42 examples/s]

Generating test split:      151674/0 [01:14<00:00, 3026.94 examples/s]
```

```python
roman_script_locales = [
    'af-ZA', 'da-DK', 'de-DE', 'en-US', 'es-ES', 'fr-FR', 'fi-FI', 'hu-HU', 'is-IS', 'it-
    'jv-ID', 'lv-LV', 'ms-MY', 'nb-NO', 'nl-NL', 'pl-PL', 'pt-PT', 'ro-RO', 'ru-RU', 'sl-
    'sv-SE', 'sq-AL', 'sw-KE', 'tl-PH', 'tr-TR', 'vi-VN', 'cy-GB'
]

output_dir = 'language_files'
os.makedirs(output_dir, exist_ok=True)

def extract_and_save(dataset, locales, partition='train'):
    for locale in locales:
        subset = dataset[partition].filter(lambda x: x['locale'] == locale)

        with open(f'{output_dir}/{locale}.txt', 'w', encoding='utf-8') as f:
            for item in subset:
                utt = unidecode(item['utt'])
                f.write(utt + '\n')

extract_and_save(dataset, roman_script_locales, partition='train')
```

```
Filter: 100%    587214/587214 [01:23<00:00, 10172.56 examples/s]
Filter: 100%    587214/587214 [01:06<00:00, 9435.64 examples/s]
Filter: 100%    587214/587214 [01:16<00:00, 9649.33 examples/s]
Filter: 100%    587214/587214 [01:10<00:00, 8160.74 examples/s]
Filter: 100%    587214/587214 [01:05<00:00, 10435.23 examples/s]
Filter: 100%    587214/587214 [01:14<00:00, 4796.60 examples/s]
Filter: 100%    587214/587214 [01:23<00:00, 9930.76 examples/s]
Filter: 100%    587214/587214 [01:08<00:00, 9413.24 examples/s]
Filter: 100%    587214/587214 [01:07<00:00, 6799.97 examples/s]
Filter: 100%    587214/587214 [01:06<00:00, 8552.44 examples/s]
Filter: 100%    587214/587214 [01:05<00:00, 11078.16 examples/s]
Filter: 100%    587214/587214 [01:05<00:00, 10589.23 examples/s]
Filter: 100%    587214/587214 [01:06<00:00, 9791.36 examples/s]
Filter: 100%    587214/587214 [01:05<00:00, 8991.34 examples/s]
Filter: 100%    587214/587214 [01:07<00:00, 10493.33 examples/s]
Filter: 100%    587214/587214 [01:05<00:00, 9234.42 examples/s]
Filter: 100%    587214/587214 [01:07<00:00, 10821.15 examples/s]
Filter: 100%    587214/587214 [01:06<00:00, 8788.26 examples/s]
Filter: 100%    587214/587214 [01:05<00:00, 6617.10 examples/s]
Filter: 100%    587214/587214 [01:06<00:00, 7861.75 examples/s]
Filter: 100%    587214/587214 [01:07<00:00, 10607.44 examples/s]
Filter: 100%    587214/587214 [01:05<00:00, 10006.79 examples/s]
Filter: 100%    587214/587214 [01:05<00:00, 9581.58 examples/s]
Filter: 100%    587214/587214 [01:04<00:00, 11499.37 examples/s]
Filter: 100%    587214/587214 [01:04<00:00, 11852.81 examples/s]
Filter:  39%    229000/587214 [00:25<01:08, 5232.49 examples/s]
Filter:   0%|          | 0/587214 [00:00<?, ? examples/s]
```

TASK 2

The text data is vectorized using CountVectorizer, which transforms the sentences into numerical feature vectors based on word frequency (bag-of-words model). **Model Training:** The Multinomial Naive Bayes classifier is then trained on the vectorized text data. This method is well-suited for text classification tasks, as it works with categorical data and can handle the high-dimensional feature space generated from text.

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import train_test_split
from sklearn import metrics
import glob
```

```
def load_sentences_from_files(folder):
    sentences = []
    labels = []

    for filepath in glob.glob(f"{folder}/*.txt"):
        locale = os.path.basename(filepath).replace('.txt', '')
        with open(filepath, 'r', encoding='utf-8') as file:
            for line in file:
                sentences.append(line.strip())
                labels.append(locale)

    return sentences, labels
```

```
X, y = load_sentences_from_files(output_dir)

X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state=42)

vectorizer = CountVectorizer()
X_train_vec = vectorizer.fit_transform(X_train)
X_val_vec = vectorizer.transform(X_val)

model = MultinomialNB()
model.fit(X_train_vec, y_train)

y_pred_train = model.predict(X_train_vec)
y_pred_val = model.predict(X_val_vec)

print("Training Accuracy:", metrics.accuracy_score(y_train, y_pred_train))
print("Validation Accuracy:", metrics.accuracy_score(y_val, y_pred_val))
print(metrics.classification_report(y_val, y_pred_val))
```

```
Training Accuracy: 0.989782953092456
Validation Accuracy: 0.980217447246526
              precision    recall  f1-score   support
```

```
       af-ZA       0.98      0.97      0.98      2313
       cy-GB       1.00      0.99      0.99      2295
       da-DK       0.92      0.94      0.93      2281
       de-DE       0.99      0.98      0.99      2265
       en-US       0.86      0.99      0.92      2219
       es-ES       0.98      0.98      0.98      2349
       fi-FI       1.00      0.98      0.99      2254
       fr-FR       0.99      0.99      0.99      2345
       hu-HU       0.99      0.98      0.99      2304
       is-IS       1.00      0.98      0.99      2337
       it-IT       0.99      0.99      0.99      2284
       jv-ID       0.99      0.98      0.99      2254
       lv-LV       0.99      0.99      0.99      2360
       ms-MY       0.98      0.99      0.99      2269
       nb-NO       0.95      0.92      0.93      2287
       nl-NL       0.98      0.96      0.97      2320
       pl-PL       0.99      0.99      0.99      2312
       pt-PT       0.98      0.98      0.98      2296
       ro-RO       0.99      0.98      0.99      2287
       ru-RU       0.99      0.99      0.99      2238
       sl-SL       1.00      0.98      0.99      2380
       sq-AL       0.99      0.99      0.99      2344
       sv-SE       0.98      0.97      0.98      2219
       sw-KE       1.00      0.99      0.99      2306
       tl-PH       0.99      1.00      0.99      2335
       tr-TR       0.99      0.98      0.99      2333
       vi-VN       0.99      1.00      0.99      2390

    accuracy                           0.98     62176
   macro avg       0.98      0.98      0.98     62176
weighted avg       0.98      0.98      0.98     62176
```

TASK-3

To simplify the classification task further, the 27 languages are grouped into 4 continent-based categories: Asia, Africa, Europe, and North America. The sentences for each language group are collapsed into four files corresponding to these continents.

This step reduces the complexity of the classification task by focusing on continent-level classification rather than language-level classification. The dataset is now treated as a four-class classification problem.

```
continent_groups = {
    'Africa': ['af-ZA', 'sw-KE'],
    'Asia': ['ms-MY', 'tl-PH', 'jv-ID', 'vi-VN', 'tr-TR'],
    'Europe': ['da-DK', 'de-DE', 'en-US', 'es-ES', 'fr-FR', 'fi-FI', 'hu-HU', 'is-IS',
               'nl-NL', 'pl-PL', 'pt-PT', 'ro-RO', 'ru-RU', 'sl-SL', 'sv-SE', 'sq-AL',
    'North America': ['en-US']
}
```

The goal here is to build a Regularized Discriminant Analysis (RDA) model. RDA is a hybrid of Linear Discriminant Analysis (LDA) and Quadratic Discriminant Analysis (QDA), with a hyperparameter lambda that controls the trade-off between the two models.

Feature Representation: Sentences are converted into numerical features using TfidfVectorizer (Term Frequency-Inverse Document Frequency), which provides a more refined feature representation than simple word counts by considering how important a word is across all documents. LDA vs. QDA: LDA assumes the same covariance matrix across all classes (simpler, less variance). QDA allows different covariance matrices for each class (more flexible, but potentially overfits). Blending the Models: RDA is simulated by blending the predictions of LDA and QDA based on lambda. For lambda = 0, LDA is used; for lambda = 1, QDA is used ; for values in between, the predictions are blended.

```
def group_by_continent(language_folder, continent_groups, output_dir):
    continent_sentences = {continent: [] for continent in continent_groups.keys()}
    for filepath in glob.glob(f"{language_folder}/*.txt"):
        locale = os.path.basename(filepath).replace('.txt', '')
        for continent, locales in continent_groups.items():
            if locale in locales:
                with open(filepath, 'r', encoding='utf-8') as file:
                    sentences = file.readlines()
                    continent_sentences[continent].extend(sentences)
    for continent, sentences in continent_sentences.items():
        with open(f"{output_dir}/{continent}.txt", 'w', encoding='utf-8') as f:
            for sentence in sentences:
                f.write(sentence)
continent_output_dir = 'continent_files'
os.makedirs(continent_output_dir, exist_ok=True)
group_by_continent('language_files', continent_groups, continent_output_dir)
```

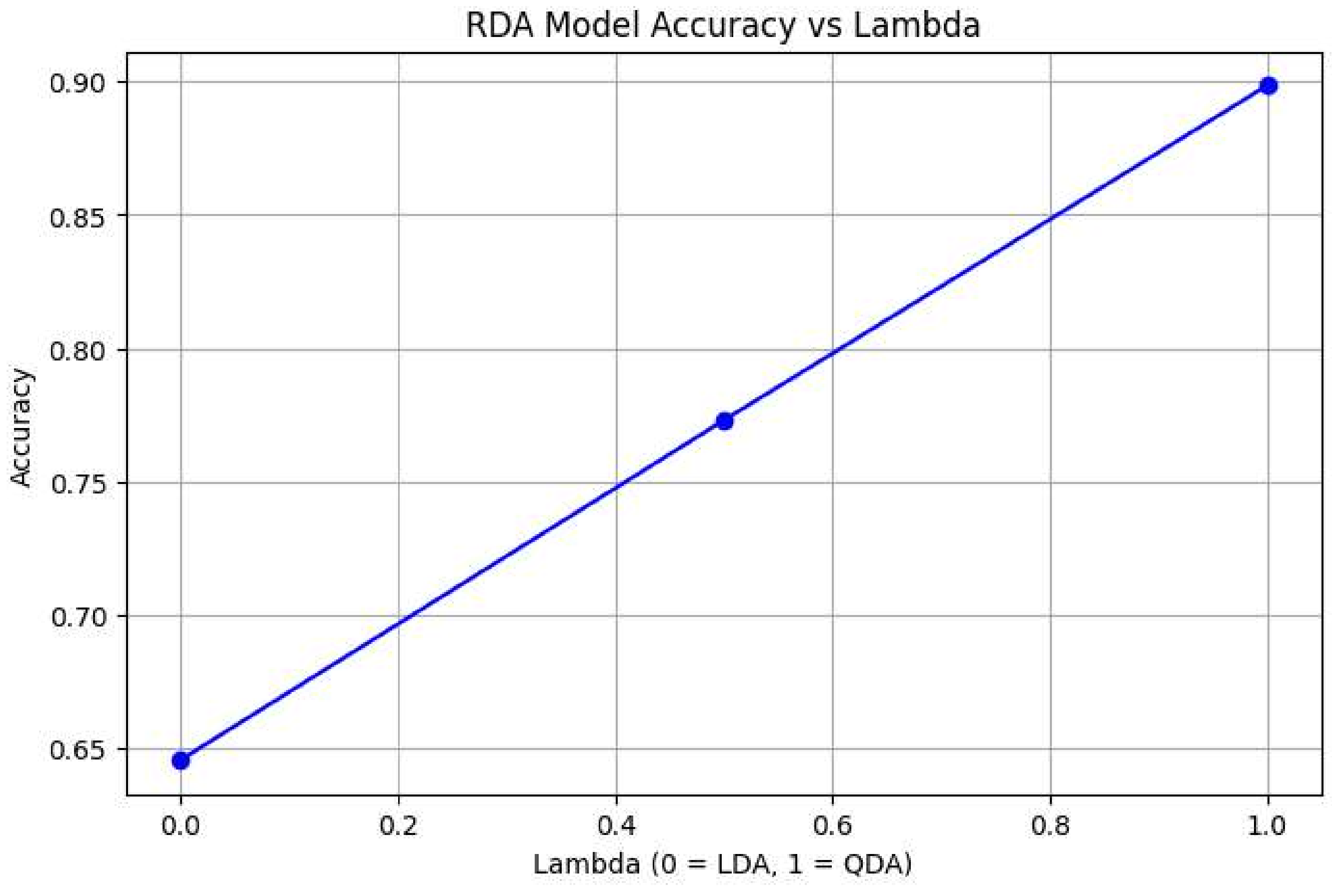
```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis as QDA
import numpy as np
```

```
def load_continent_data(continent_folder):
    X = []
    y = []

    for filepath in glob.glob(f"{continent_folder}/*.txt"):
        continent = os.path.basename(filepath).replace('.txt', '')
        with open(filepath, 'r', encoding='utf-8') as file:
            sentences = file.readlines()
            X.extend(sentences)
            y.extend([continent] * len(sentences))

    return X, y
X, y = load_continent_data(continent_output_dir)
vectorizer = TfidfVectorizer(max_features=500, stop_words='english'
X vec = vectorizer.fit transform(X)
```

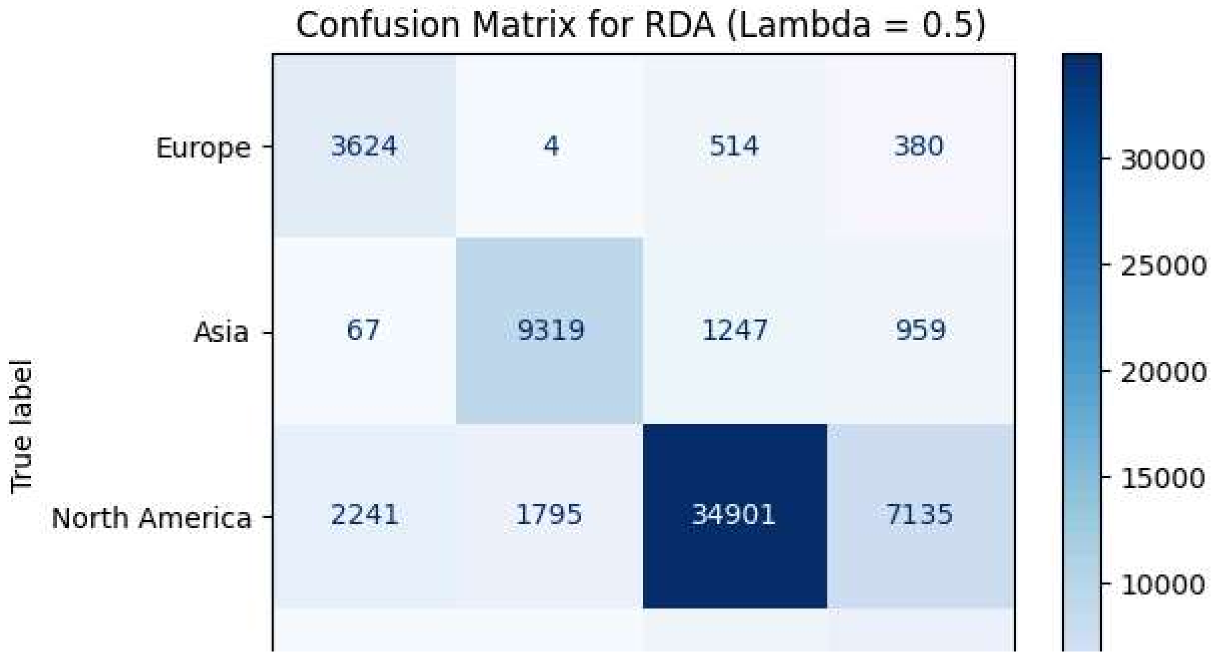
```
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis as QDA
import numpy as np
from sklearn.metrics import ConfusionMatrixDisplay, confusion_matrix

# Train-test split
X_train, X_val, y_train, y_val = train_test_split(X_vec, y, test_size=0.2, random_state=

# Function to implement RDA with lambda and return accuracy
def RDA_optimized(X_train, y_train, X_val, y_val, lambda_param):
    lda_model = LDA()
    qda_model = QDA()

    # Train LDA and QDA models
    lda_model.fit(X_train.toarray(), y_train)
    qda_model.fit(X_train.toarray(), y_train)

    # Get predictions from both LDA and QDA
    lda_pred = lda_model.predict(X_val.toarray())
    qda_pred = qda_model.predict(X_val.toarray())

    # Blend predictions using lambda
    blended_pred = np.where(np.random.rand(len(lda_pred)) < lambda_param, lda_pred, qda_

    # Calculate accuracy
    accuracy = metrics.accuracy_score(y_val, blended_pred)

    return accuracy, blended_pred

# Lambda values to test
lambda_values = [0.0, 0.5, 1.0]
accuracies = []

# Run the RDA model for each lambda and store accuracies
for lambda_param in lambda_values:
    accuracy, preds = RDA_optimized(X_train, y_train, X_val, y_val, lambda_param)
    accuracies.append(accuracy)
    print(f"Lambda: {lambda_param}, Accuracy: {accuracy}")

# Visualizing Accuracy vs Lambda
```

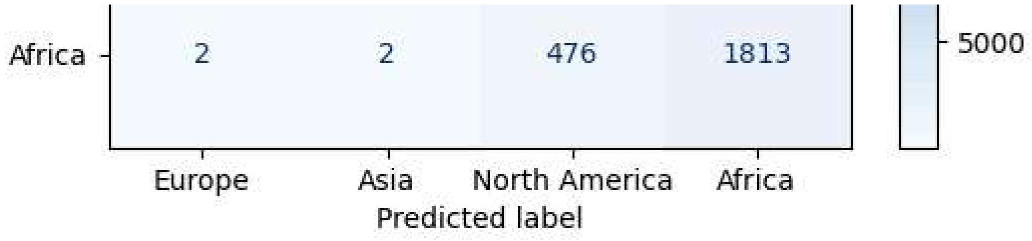
```
plt.figure(figsize=(8, 5))
plt.plot(lambda_values, accuracies, marker='o', linestyle='-', color='b')
plt.title('RDA Model Accuracy vs Lambda')
plt.xlabel('Lambda (0 = LDA, 1 = QDA)')
plt.ylabel('Accuracy')
plt.grid(True)
plt.show()

# Pick a lambda value for confusion matrix visualization
lambda_param = 0.5
accuracy, preds = RDA_optimized(X_train, y_train, X_val, y_val, lambda_param)

# Generate the confusion matrix
cm = confusion_matrix(y_val, preds)

# Plot confusion matrix
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=set(y_val))
disp.plot(cmap=plt.cm.Blues)
plt.title(f'Confusion Matrix for RDA (Lambda = {lambda_param})')
plt.show()

```

```
/usr/local/lib/python3.10/dist-packages/sklearn/discriminant_analysis.py:947: UserWarnir
  warnings.warn("Variables are collinear")
Lambda: 0.0, Accuracy: 0.6453108764093736
/usr/local/lib/python3.10/dist-packages/sklearn/discriminant_analysis.py:947: UserWarnir
  warnings.warn("Variables are collinear")
Lambda: 0.5, Accuracy: 0.7729028055646024
/usr/local/lib/python3.10/dist-packages/sklearn/discriminant_analysis.py:947: UserWarnir
  warnings.warn("Variables are collinear")
Lambda: 1.0, Accuracy: 0.8986646815242172
```



```
/usr/local/lib/python3.10/dist-packages/sklearn/discriminant_analysis.py:947: UserWarnir
  warnings.warn("Variables are collinear")
```





Double-click (or enter) to edit