Project Final Report

**Language Detection System**

# Participants:

* Varshith Reddy Kallem (11603452) [Varshithreddykallem@my.unt.edu](mailto:Varshithreddykallem@my.unt.edu)
* Sai Varshith Gandu (11706319) [saivarshithgandu@my.unt.edu](mailto:saivarshithgandu@my.unt.edu)
* Akhila Pam (11711224) [akhilapam@my.unt.edu](mailto:akhilapam@my.unt.edu)
* Sai Durga Rohith Reddy (11610068) [saidurgarohithreddy@my.unt.edu](mailto:saidurgarohithreddy@my.unt.edu)

# Workflow

**Varshith Reddy Kallem:** Role is to setup project and model selection Training.

**Sai Varshith Gandu:** Role is to Data collection, preprocessing and model evaluation.

**Akhila Pam:** Role is to Feature Engineering and Deployment.

**Sai Durga Rohith Reddy:** Role is to do the Optional, Monitoring and maintenance and Reporting.

**Project Abstract**

Proficiency in language comprehension is essential in a society growing more interconnected by the day. The goal of this project, “Language Detection System," is to develop a reliable method for accurately identifying languages in a variety of text datasets. Utilizing Natural Language Processing (NLP) and Python, the system is based on an extensive multilingual dataset that includes a wide range of languages and linguistic subtleties. The project proceeds in several stages. Data collection, preprocessing, and feature extraction are the first steps, which convert unprocessed text into numerical representations. Then, using techniques such as Naive Bayes and Support Vector Machines, a machine learning model is developed and thoroughly assessed. A fine-tuning procedure is implemented in order to improve accuracy. Finally, a smoothly integrated Python script inside Jupyter notebook provides users with an easy way to determine the language of any given text.

# Language Detection is a crucial process in today's globalized world, enabling the identification of languages based on textual input. The objective of this project is to develop a Language Detection system that accurately identifies the language of given text input.

# The SPACY algorithm utilizes advanced natural language processing techniques for more precise language detection. Users can input text via a text area and select their preferred algorithm. Upon entering the text, the system processes it using the chosen algorithm and presents the detected language to the user.

# The project is implemented in Python and relies on popular libraries, such as SPACY and Langdetect, to create a user-friendly interface. The system's performance is evaluated using various metrics, including precision, recall, F1-score, and accuracy, with both algorithms delivering impressive language detection results. In summary, this project offers an efficient and effective solution for language detection, serving the needs of those working with multilingual content.

# Project Design

Programming Language: Python IDE: Jupyter Notebook

Important Packages: Langdetect

Langdetect open source python library used for make it easy for creating the custom web apps.

The design of the Language Detection system involves key decisions. These include the selection of language detection algorithms, defining input and output formats, and managing text preprocessing tasks, such as character encoding and tokenization. The choice of algorithm is tailored to the type of text to be analyzed, ensuring precision in language detection.

Feature extraction: the process of turning text input into numerical features using methods like "TfidfVectorizer"

Machine Learning Model: We employ a variety of models, including neural networks, support vector machines, and naive Bayes.

Model Hyperparameter Tuning: Use grid search or random search to optimize hyperparameters.Model Training: - Train the selected model using the training dataset.

Use of cross-validation is necessary to guarantee robustness. Evaluate the model by computing the confusion matrices, F1-score, accuracy, precision, recall, and precision for the test and validation sets.

Utilize software such as "seaborn" or "matplotlib" to visualize model performance.

The Language Detection system's design encompasses significant decisions, algorithms, and libraries. It aims to offer precise language identification across diverse text types, catering to linguists, translators, and those dealing with multilingual content. Built upon the "Langdetect" library, the project incorporates intricate logic and language detection techniques to achieve its goal of accurate language identification.



# Project Milestone

# Project Milestones and Incremental Features:

# Data Collection and Preprocessing:

# Milestone: Successful acquisition and preprocessing of language detection dataset.

# Description: Gather and clean the dataset, ensuring it is well-structured and ready for training and evaluation.

# Basic Language Detection Model:

# Milestone: Develop a basic language detection model using a simple algorithm.

# Description: Create a language detection model that can identify languages in the dataset based on basic features.

# Integration of NLP Libraries:

# Milestone: Enhance language detection using NLP libraries like spaCy.

# Description: Integrate spaCy to improve the accuracy and performance of the language detection system.

# User Interface Development:

# Milestone: Create a user-friendly web interface for language detection.

# Description: Design and implement a web-based interface that allows users to input text and receive language detection results.

# Evaluation and Performance Metrics:

# Milestone: Implement evaluation metrics for assessing language detection accuracy.

# Description: Integrate precision, recall, F1-score, and other relevant metrics to measure the performance of the system.

# User Feedback and Refinement:

# Milestone: Incorporate user feedback and improve the language detection algorithm.

# Description: Allow users to provide feedback on language detection results, and use this feedback to refine the algorithm.

# Deployment as a Web Service:

# Milestone: Ensure the system can handle a significant volume of requests without performance issues.

# Description: Conduct scalability and load testing to confirm that the system can efficiently process a large number of language detection requests.

# Documentation and Reporting:

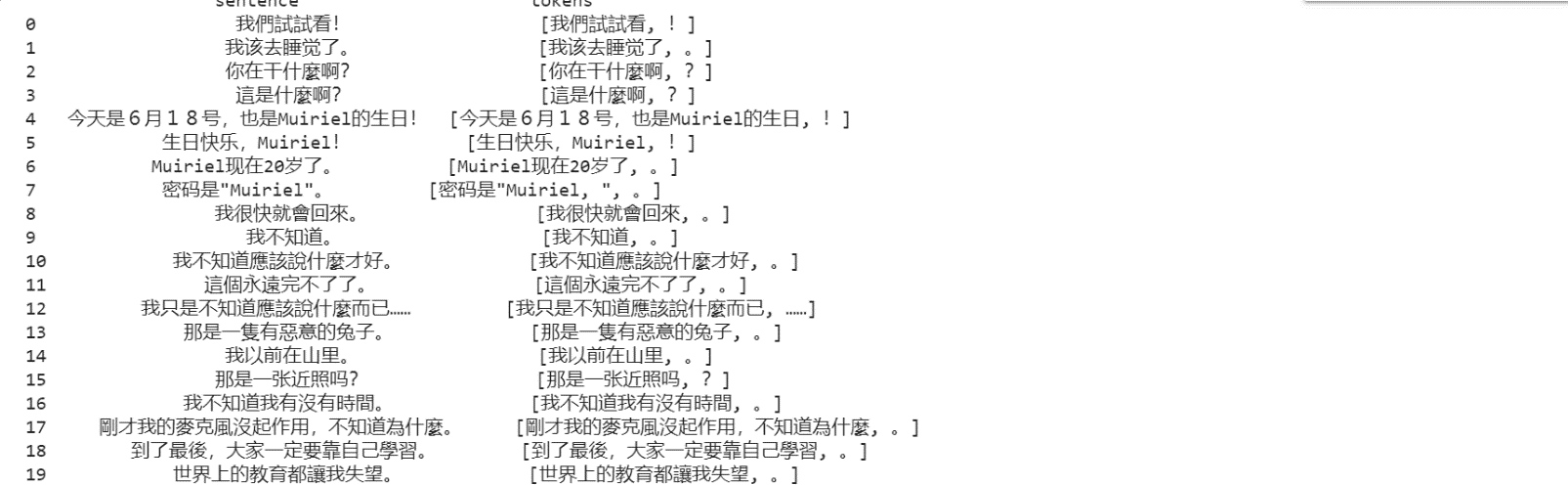
# Milestone: Create comprehensive documentation for the language detection system.

# Description: Develop detailed documentation that explains the methods, algorithms, and usage of the system, making it accessible to users and developers.

# Project Result

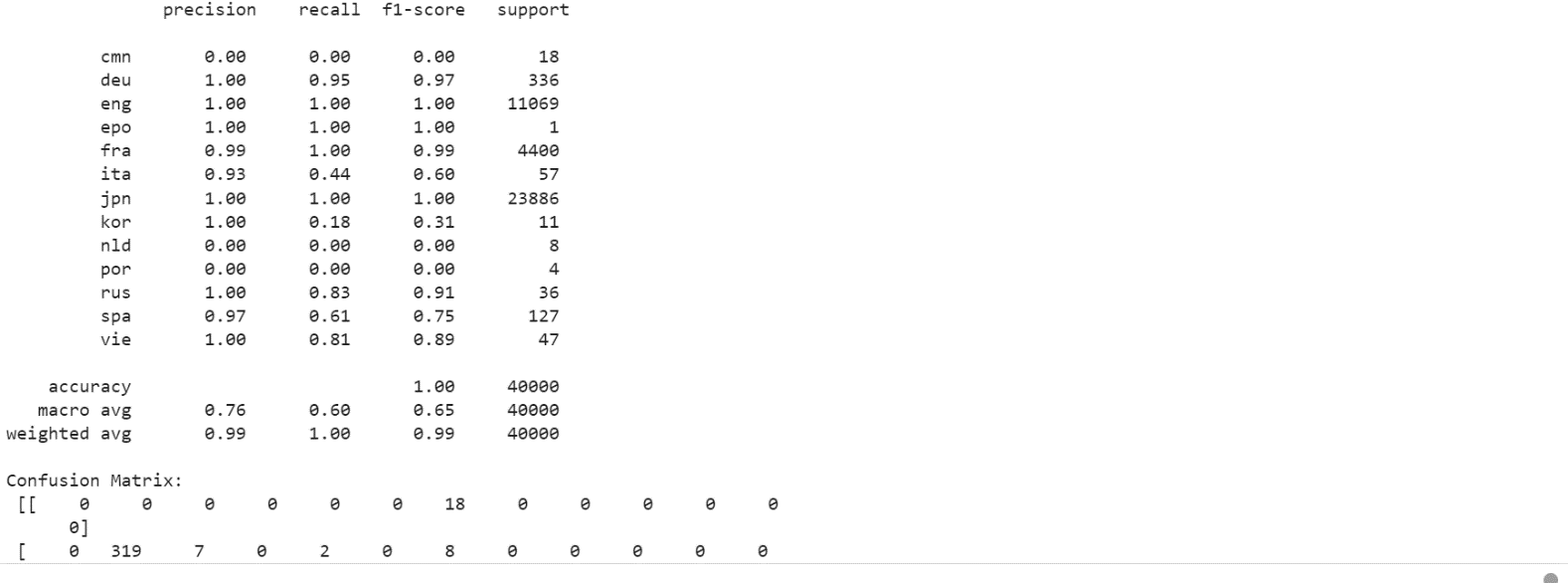
We displayed the output screen and the metrics calculation based on the output in the screenshots below.





**Result Evaluation:**

We have used the evaluated metrics f1 score, precision and accuracy to amplify our result.

****

**Project Code**

pip install langdetect

pip install pandas

import pandas as pd

import spacy

dataset = pd.read\_csv('sentences.csv')

dataset = dataset.head(20)

nlp = spacy.load('en\_core\_web\_sm')

def tokenize\_text(text):

doc = nlp(text)

tokens = [token.text for token in doc]

return tokens

dataset['tokens'] = dataset['sentence'].apply(tokenize\_text)

print(dataset[['sentence', 'tokens']])

import pandas as pd

from langdetect import detect

import ipywidgets as widgets

from IPython.display import display

dataset = pd.read\_csv("sentences.csv")

def detect\_language(text):

try:

detected\_language = detect(text)

return f"Detected language: {detected\_language}"

except:

return "Language detection failed"

text\_input = widgets.Text(placeholder="Enter a sentence")

button = widgets.Button(description="Detect Language")

output = widgets.Output()

def on\_button\_click(b):

with output:

output.clear\_output()

result = detect\_language(text\_input.value)

print(result)

button.on\_click(on\_button\_click)

display(text\_input, button)

display(output)

for index, row in dataset.iterrows():

lan\_code = row['lan\_code']

sentence = row['sentence']

print(f"Sentence {index + 1} ({lan\_code}): {sentence}")

import pandas as pd

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import classification\_report, confusion\_matrix

dataset = pd.read\_csv('sentences.csv')

dataset = dataset.head(200000)

X = dataset['sentence']

y = dataset['lan\_code']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

tfidf\_vectorizer = TfidfVectorizer()

X\_train\_tfidf = tfidf\_vectorizer.fit\_transform(X\_train)

X\_test\_tfidf = tfidf\_vectorizer.transform(X\_test)

clf = RandomForestClassifier(n\_estimators=100, random\_state=42)

clf.fit(X\_train\_tfidf, y\_train)

y\_pred = clf.predict(X\_test\_tfidf)

print("Classification Report:\n", classification\_report(y\_test, y\_pred))

print("Confusion Matrix:\n", confusion\_matrix(y\_test, y\_pred))

**REFERENCE MATERIALS**

* Bird, Steven, Edward Loper, and Ewan Klein. "Natural Language Processing with Python." O'Reilly Media, Inc., 2009.
* Pennington, Jeffrey, Richard Socher, and Christopher D. Manning. "Glove: Global Vectors for Word Representation." Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2014. [Link](https://nlp.stanford.edu/pubs/glove.pdf)
* oulin, Armand, Edouard Grave, Piotr Bojanowski, Mikolov, Tomas, Herve Jegou, and Tomas Mikolov. "FastText.zip: Compressing text classification models." arXiv preprint arXiv:1612.03651, 2016. [Link](https://arxiv.org/abs/1612.03651)
* Kaggle. "Big Language Detection Dataset." Kaggle Datasets. [Link](https://www.kaggle.com/datasets/chazzer/big-language-detection-dataset)
* Rasmussen, Catherine E. "Language Identification: A solved problem?" Proceedings of the ACL 2009 Eighth Workshop on Unsupervised and Semi-Supervised Learning in NLP, 2009. [Link](https://www.aclweb.org/anthology/W09-1404/)