IndiSpeak: Multilingual Language Detection using Feature-based Audio Classification

MSDA CAPSTONE PROJECT FOR SUMMER 2023

BY SONAL GANVIR

MENTOR:

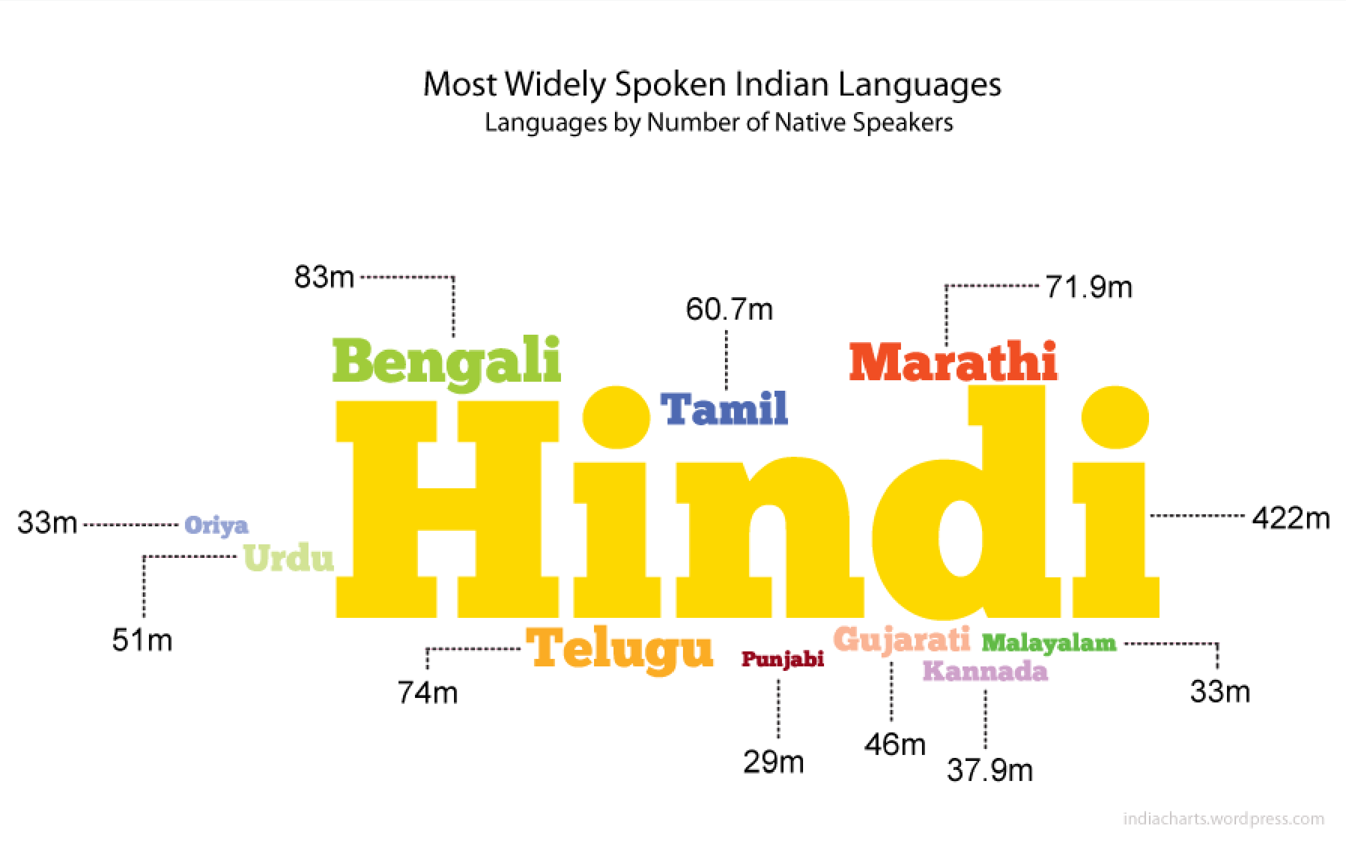
DR.DVIJESH SHASTRI

Language recognition is a critical problem domain with immense practical applications.

• India is a country known for its rich linguistic diversity, with 22 languages officially recognized by the Indian Constitution [1].

• Language identification is essential for various applications, including:

- **Customer Support**: Providing personalized assistance in the customer's native language for better user experience.
- **Voice Assistants**: Enabling voice assistants to comprehend and respond in multiple Indian languages.
- **Language Research**: Facilitating linguistic studies and preserving indigenous languages.



[1] Languages of India – Wikipedia: https://en.wikipedia.org/wiki/Languages_of_India

The dataset comprises a vast collection of audio samples representing 10 diverse Indian languages.

Massive Collection of Audio Samples

- 250k + voice samples
- mp3 format

Standardized Duration

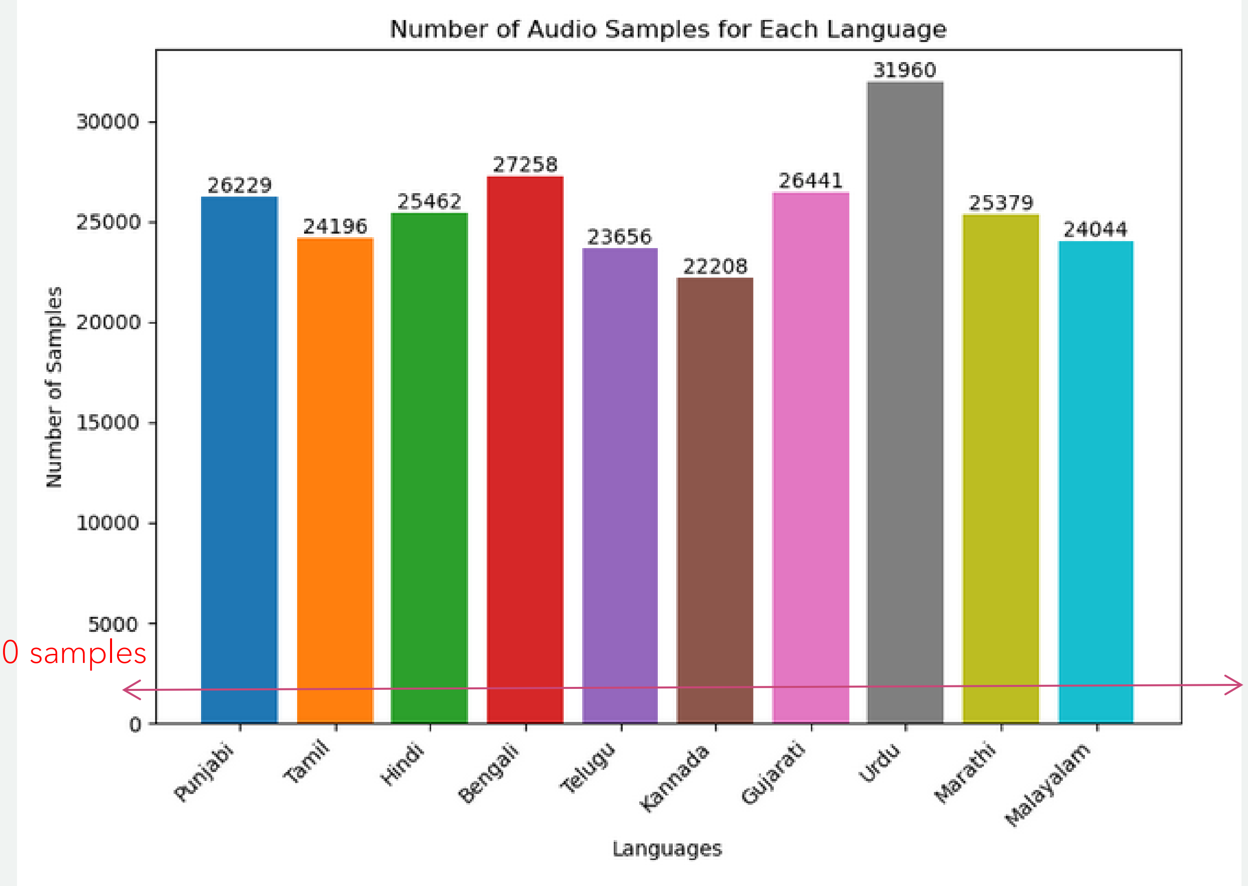
- Each voice sample ~ 5 sec long

Source of Dataset

- Extracted from YouTube videos

Supported Languages

- 10 Indian Languages : Punjabi, Tamil, Hindi, Bengali, Telugu, Kannada, Gujarati, Urdu, Marathi, Malayalam



1000 samples

Audio Dataset with 10 Indian Languages – Kaggle Dataset: https://www.kaggle.com/datasets/hbchaitanyabharadwaj/audio-dataset-with-10-indian-languages

Tools and Technologies Used

Language

python™

IDE

kaggle

Packages

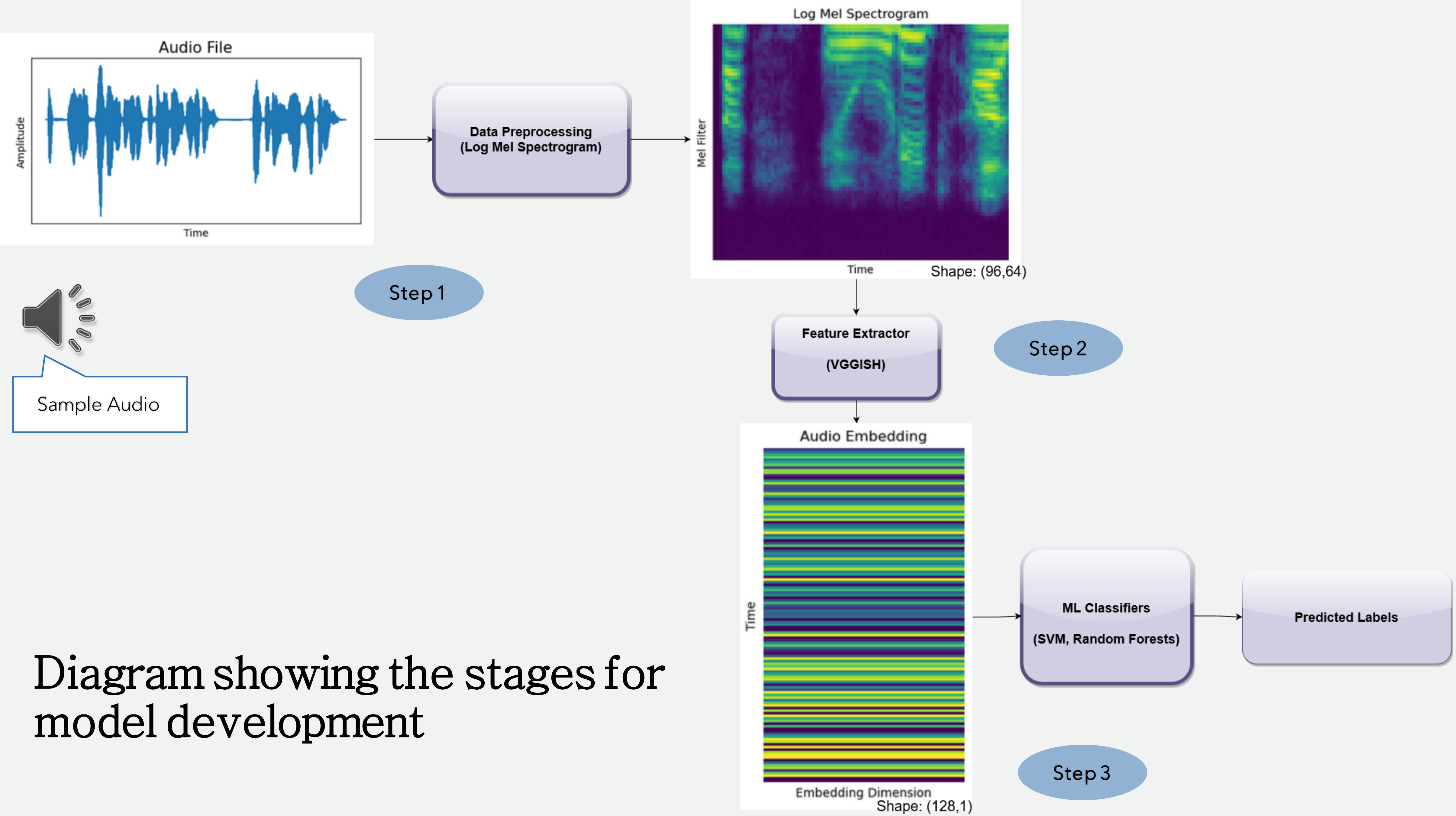
K + TensorFlow

NumPy

librosa

scikit learn

Diagram showing the stages for model development



Data Preprocessing (Log Mel Spectrogram)

Step 1

Shape: (96,64)

Sample Audio

Feature Extractor

(VGGISH)

Step 2

Shape: (128,1)

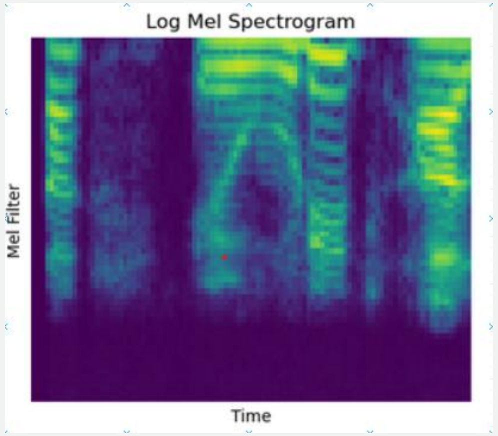
ML Classifiers

(SVM, Random Forests)

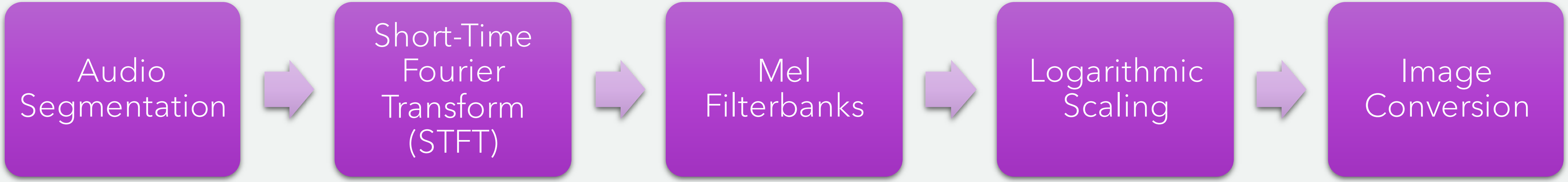
Predicted Labels

Step 3

Converting the audio classification problem to an image classification task enhances language recognition accuracy.



Mel Spectrogram: A mel spectrogram is a visual representation of the spectrum of frequencies in an audio signal, with frequencies on the y-axis and time on the x-axis.

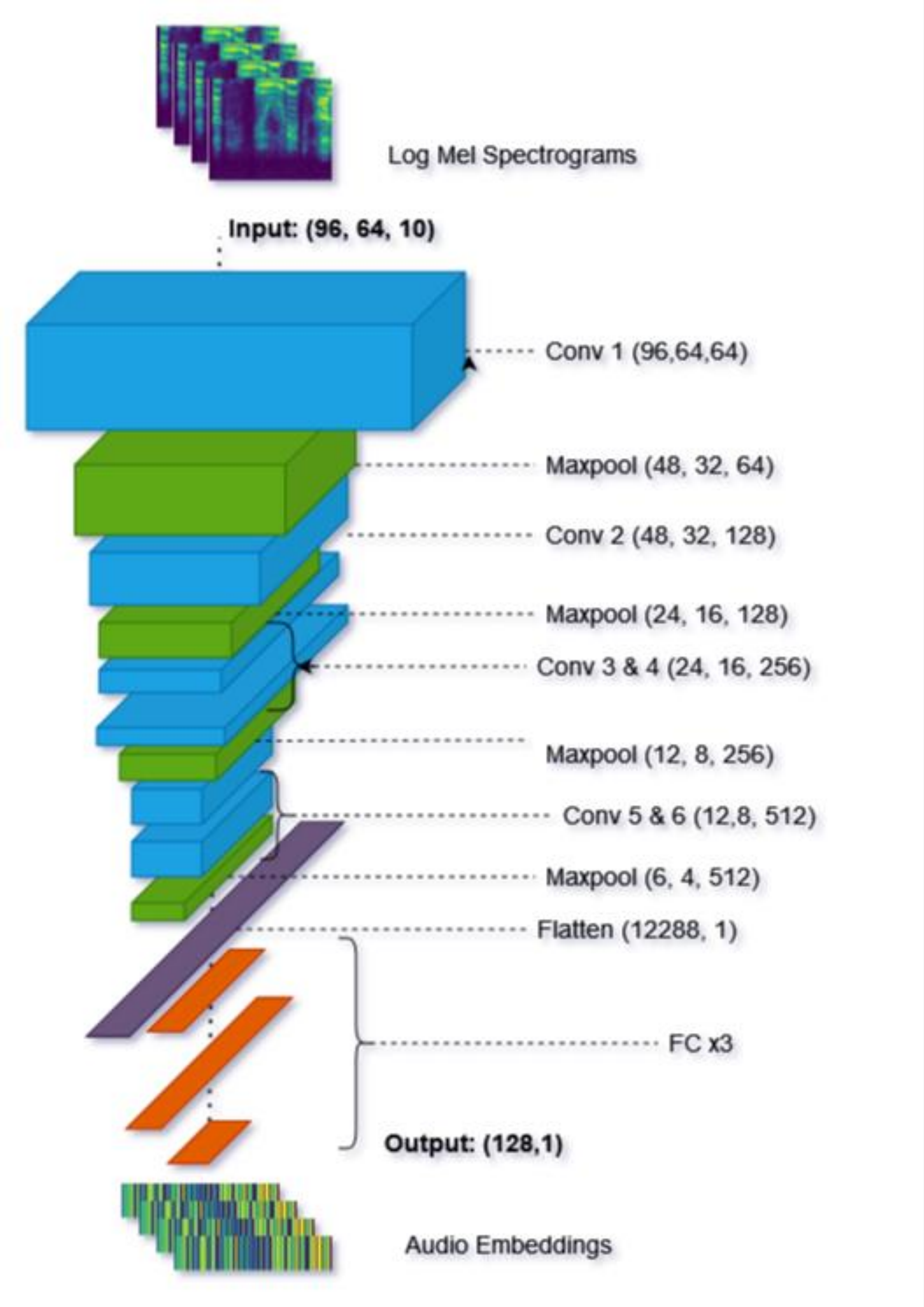


Audio Segmentation → Short-Time Fourier Transform (STFT) → Mel Filterbanks → Logarithmic Scaling → Image Conversion

VGGish Model – A Powerful Feature Extractor

- **VGGish Model:** VGGish is a pre-trained deep learning model developed by Google's Magenta team.
- It was originally designed for audio feature extraction, particularly for audio classification tasks.

- **Architecture:** VGGish is based on the VGG-16 architecture, modified for audio processing.
- The model consists of 6 convolutional layers and 4 max-pooling layers.
- The final layer is a fully connected layer with 128 units, followed by a ReLU activation.

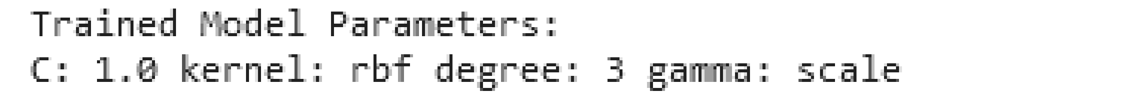


VGGISH Github Repository: https://github.com/tensorflow/models/tree/master/research/audioset/vggish

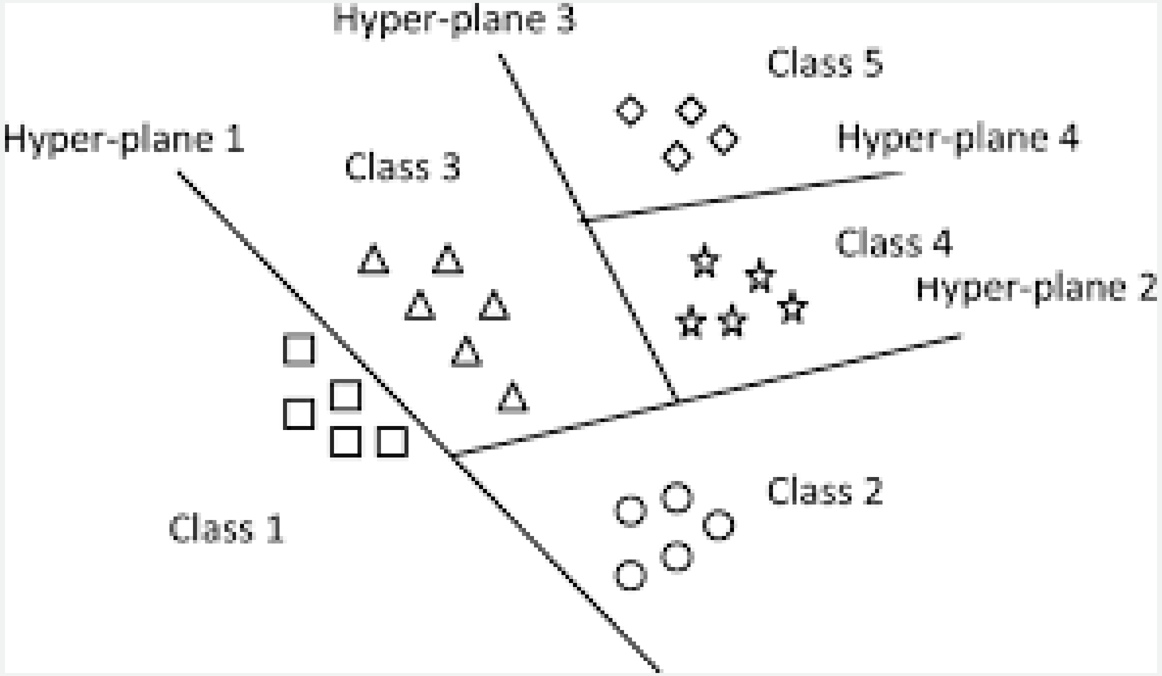
Two powerful Machine Learning (ML) classifiers are employed for language identification.

1. Support Vector Machine (SVM) Classifier:

- **Working:** SVM is a supervised learning algorithm that finds an optimal hyperplane to separate data points into different classes, maximizing the margin between classes.
- **Usage:** SVM is applied to classify audio embeddings into Indian languages, effectively handling high-dimensional data and multiclass classification tasks.
- **Advantages:** SVM is robust against overfitting, efficient in handling large feature spaces, and performs well with limited training data.
- **Default Parameters:**



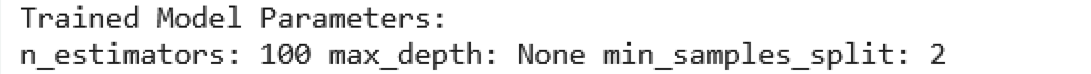
```
Trained Model Parameters:
C: 1.0 kernel: rbf degree: 3 gamma: scale
```



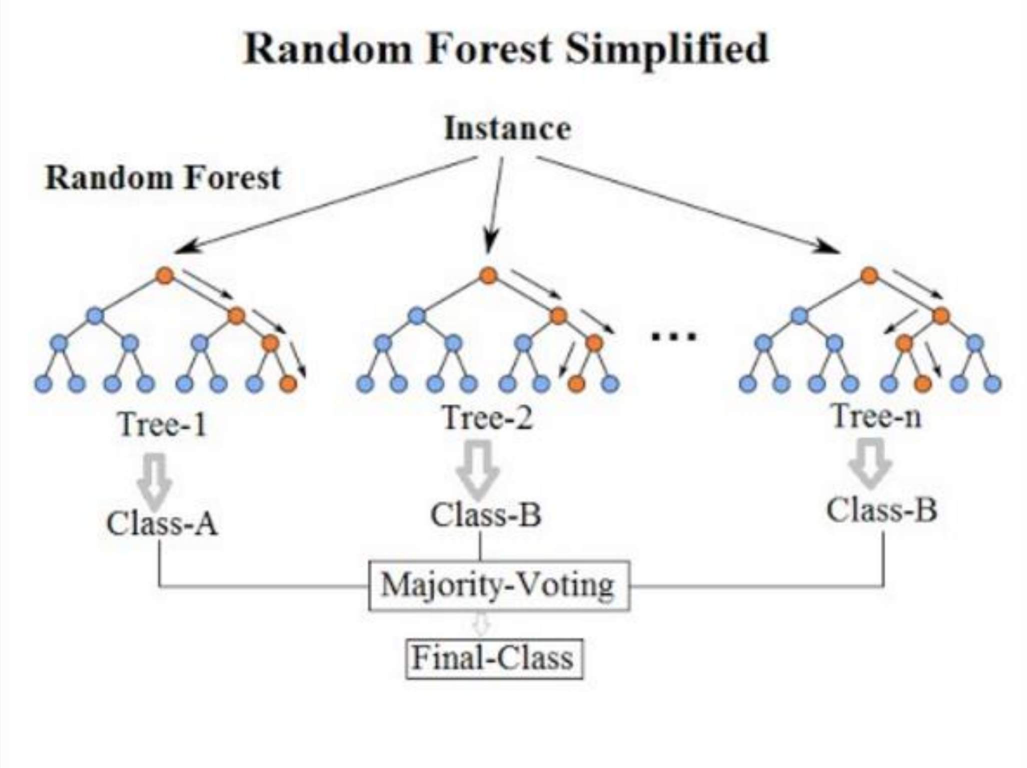
Two powerful Machine Learning (ML) classifiers are employed for language identification.

2. Random Forest (RF) Classifier:

- **Working:** Random Forest (RF) is an ensemble learning method that combines multiple decision trees, trained on random subsets of data and features, using majority voting for final predictions.
- **Usage:** RF is utilized to classify audio embeddings into Indian languages, benefiting from its ensemble nature that provides robustness and reduces overfitting risk.
- **Advantages:** RF effectively handles high-dimensional data, captures complex relationships, and performs well with imbalanced datasets.
- Default Parameters:

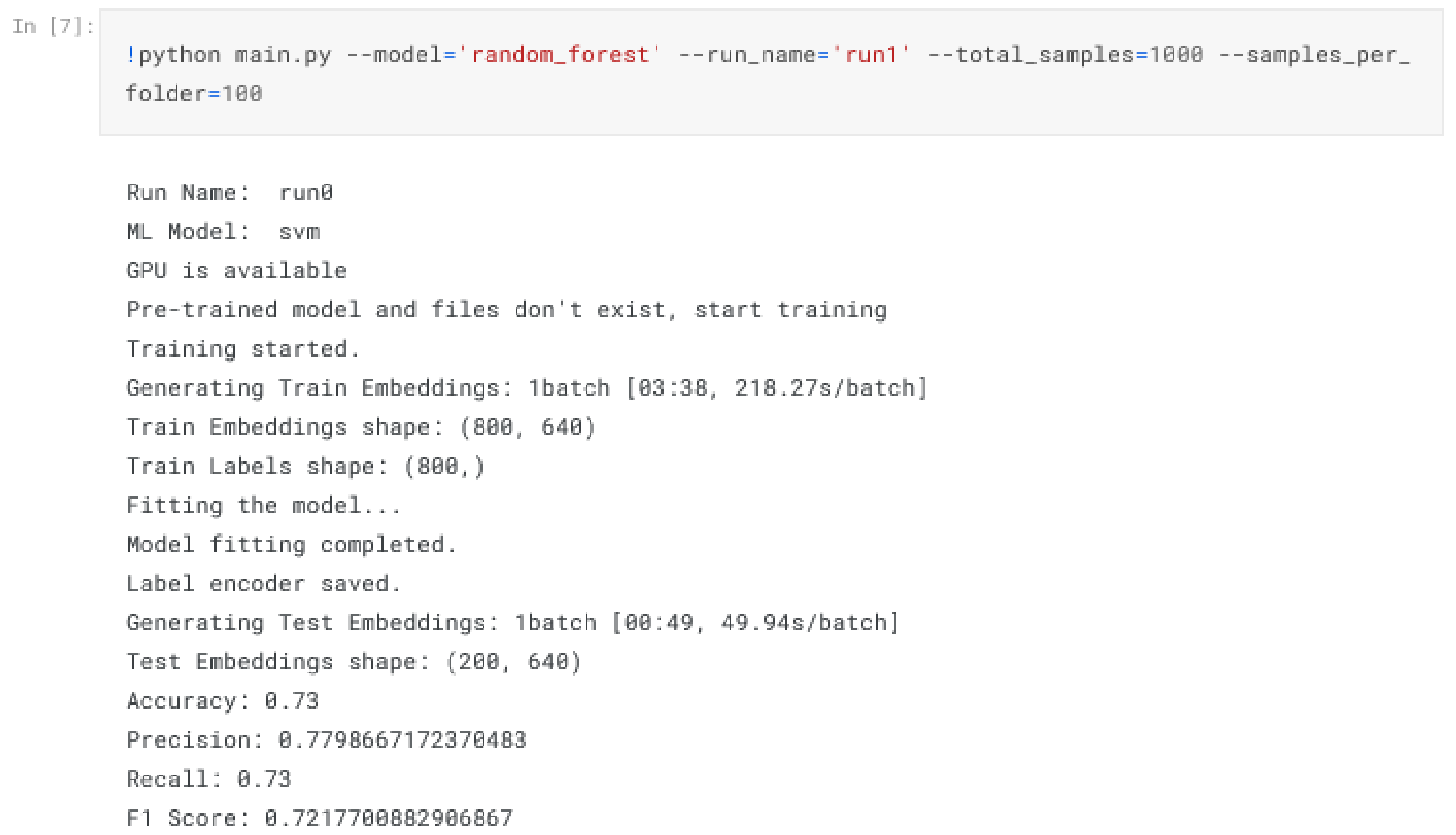


```
Trained Model Parameters:
n_estimators: 100 max_depth: None min_samples_split: 2
```



Model Training and Evaluation

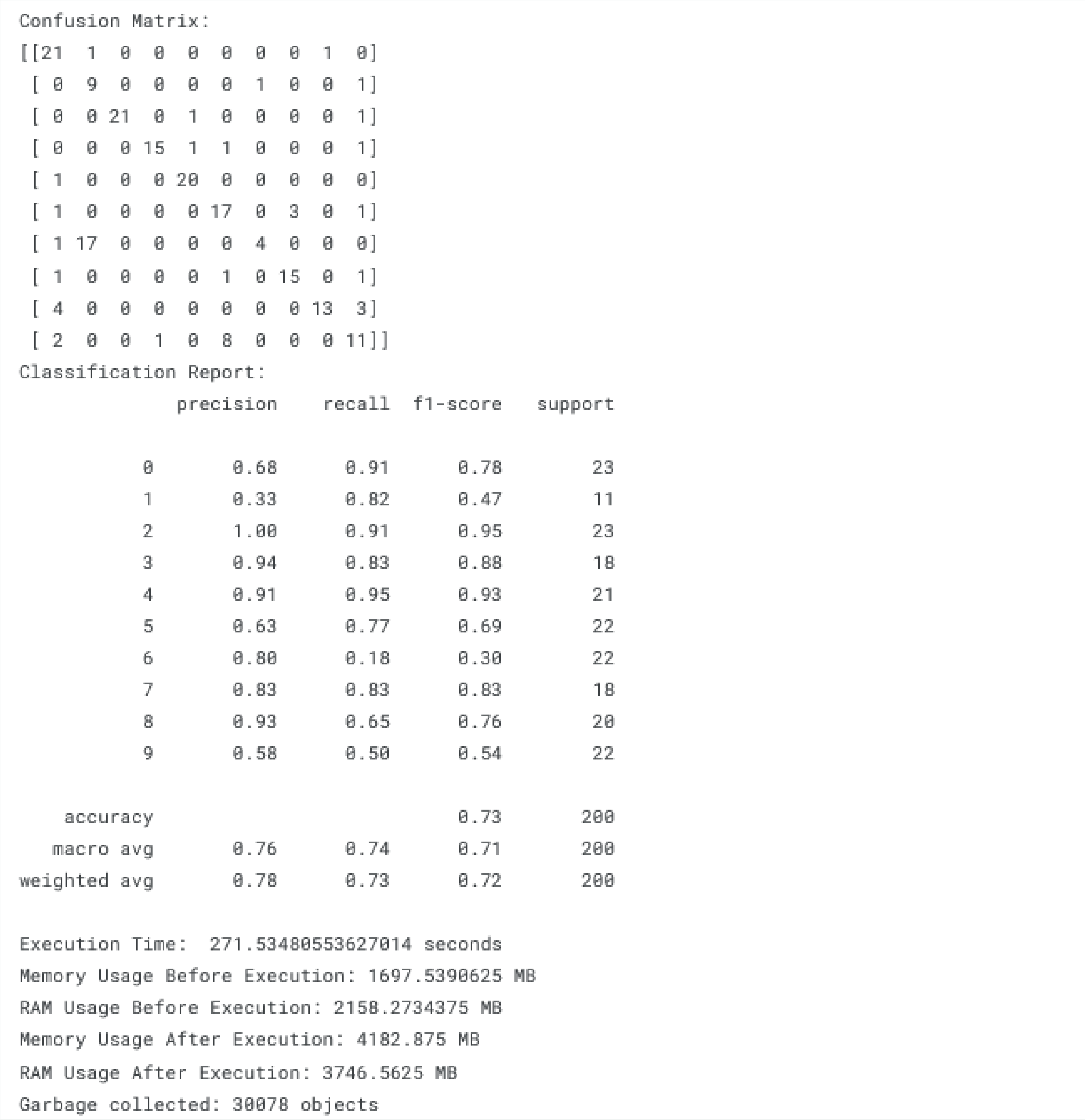
1. Training Process:
   1. Samples Range: We train both classifiers for sample sizes ranging from 1,000 to 10,000 samples.
   2. Data Split: 80% of the samples are used for training, ensuring the classifiers learn from a substantial portion of the data.
   3. Class Balancing: To prevent bias towards dominant classes, we ensure class balancing during training.
   4. Hyperparameter Tuning: We used GridSearchCV with cross-validation (cv=10) to find the best combination of hyperparameters for the model..
2. Evaluation Process:
   1. Testing Set: The remaining 20% of samples serve as the testing set, ensuring an unbiased evaluation.
   2. Performance Metrics: The classifiers are evaluated using accuracy, precision, recall, and F1-score to measure their effectiveness in language detection.
   3. Confusion Matrix: Confusion matrices are generated to visualize classifier predictions and identify misclassifications.
3. Time and Space Complexity Analysis:
4. Kaggle Compute Instance: NVIDIA T4 (X2) with 30 GB RAM
   1. Execution Time: We measure the time taken by each classifier to train and make predictions.
   2. Memory Usage: We monitor the memory consumption during the execution of the classifiers.



```
In [7]: !python main.py --model='random_forest' --run_name='run1' --total_samples=1000 --samples_per_
        folder=100
```

```
Run Name:  run0
ML Model:  svm
GPU is available
Pre-trained model and files don't exist, start training
Training started.
Generating Train Embeddings: 1batch [03:38, 218.27s/batch]
Train Embeddings shape: (800, 640)
Train Labels shape: (800,)
Fitting the model...
Model fitting completed.
Label encoder saved.
Generating Test Embeddings: 1batch [00:49, 49.94s/batch]
Test Embeddings shape: (200, 640)
Accuracy: 0.73
Precision: 0.7798667172370483
Recall: 0.73
F1 Score: 0.7217700882906867
```

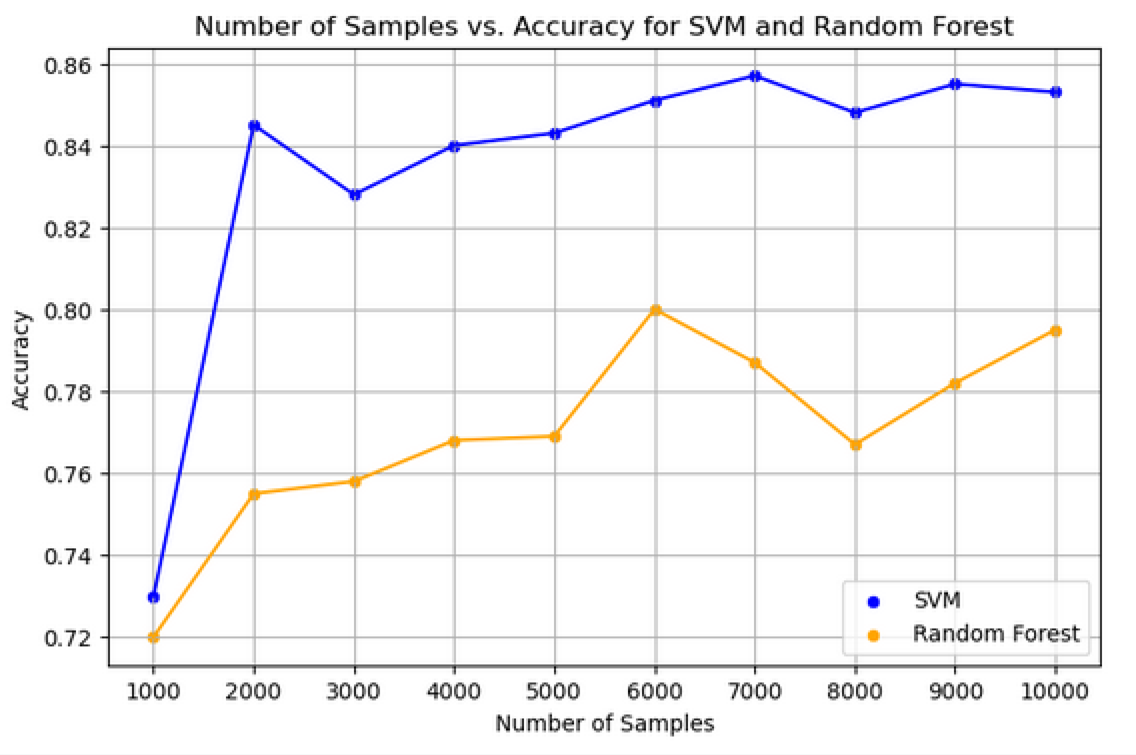
Sample Output

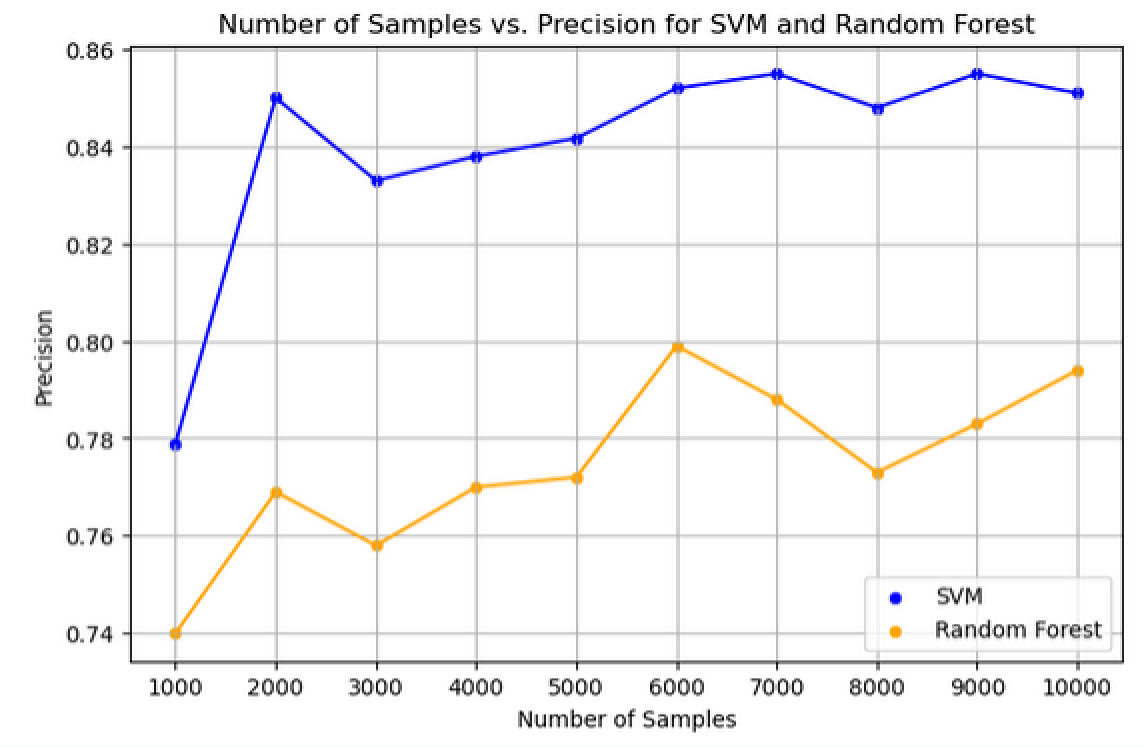


```
Confusion Matrix:
[[21  1  0  0  0  0  0  0  1  0]
 [ 0  9  0  0  0  0  1  0  0  1]
 [ 0  0 21  0  1  0  0  0  0  1]
 [ 0  0  0 15  1  1  0  0  0  1]
 [ 1  0  0  0 20  0  0  0  0  0]
 [ 1  0  0  0  0 17  0  3  0  1]
 [ 1 17  0  0  0  0  4  0  0  0]
 [ 1  0  0  0  0  1  0 15  0  1]
 [ 4  0  0  0  0  0  0  0 13  3]
 [ 2  0  0  1  0  8  0  0  0 11]]
Classification Report:
              precision    recall  f1-score   support

           0       0.68      0.91      0.78        23
           1       0.33      0.82      0.47        11
           2       1.00      0.91      0.95        23
           3       0.94      0.83      0.88        18
           4       0.91      0.95      0.93        21
           5       0.63      0.77      0.69        22
           6       0.80      0.18      0.30        22
           7       0.83      0.83      0.83        18
           8       0.93      0.65      0.76        20
           9       0.58      0.50      0.54        22

    accuracy                           0.73       200
   macro avg       0.76      0.74      0.71       200
weighted avg       0.78      0.73      0.72       200

Execution Time:  271.53480553627014 seconds
Memory Usage Before Execution: 1697.5390625 MB
RAM Usage Before Execution: 2158.2734375 MB
Memory Usage After Execution: 4182.875 MB
RAM Usage After Execution: 3746.5625 MB
Garbage collected: 30078 objects
```

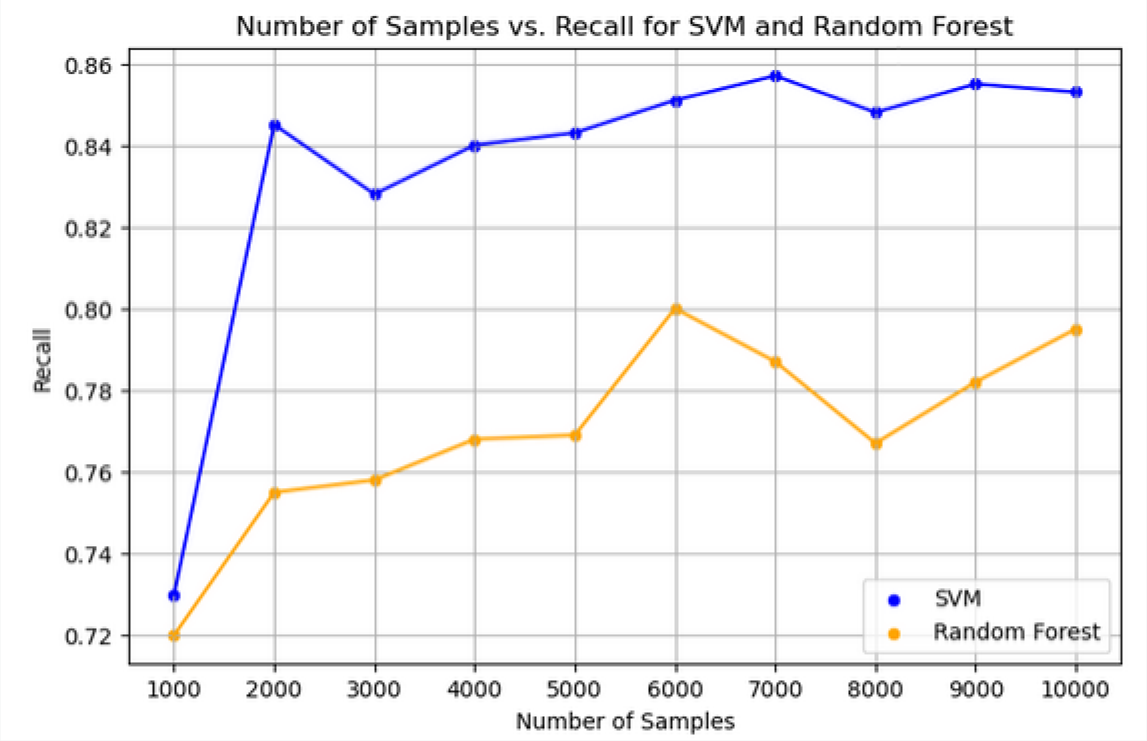
Sample Output

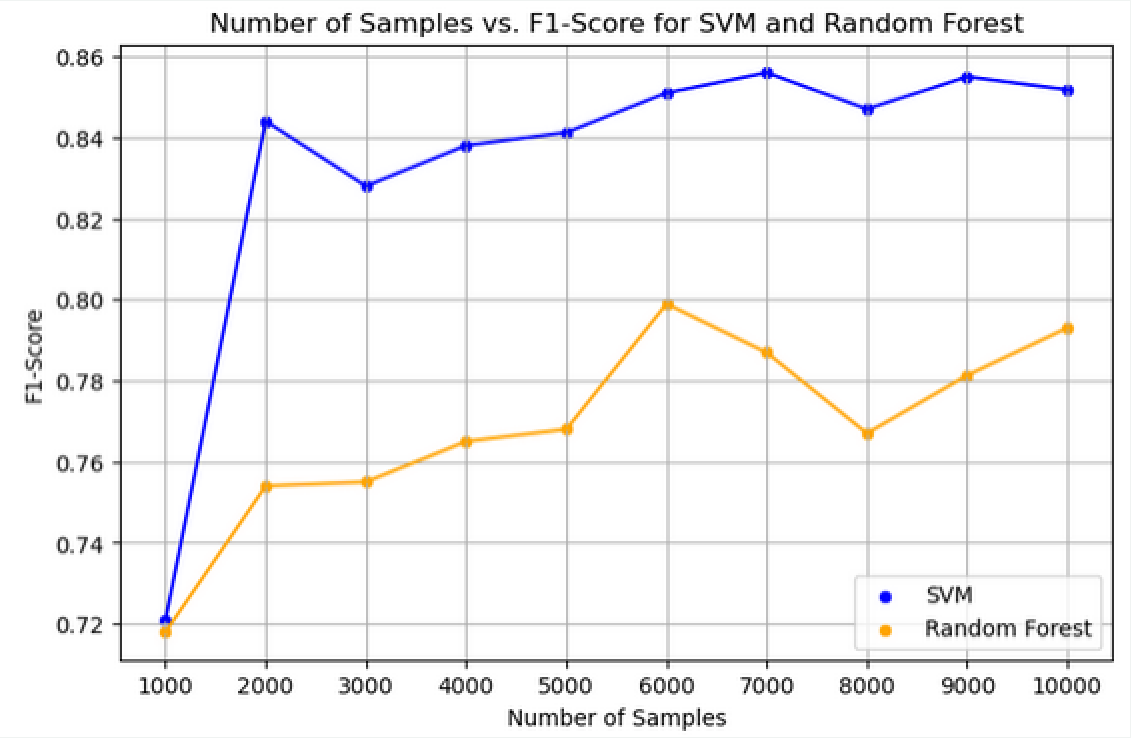
Performance Metrics (Accuracy + Precision)



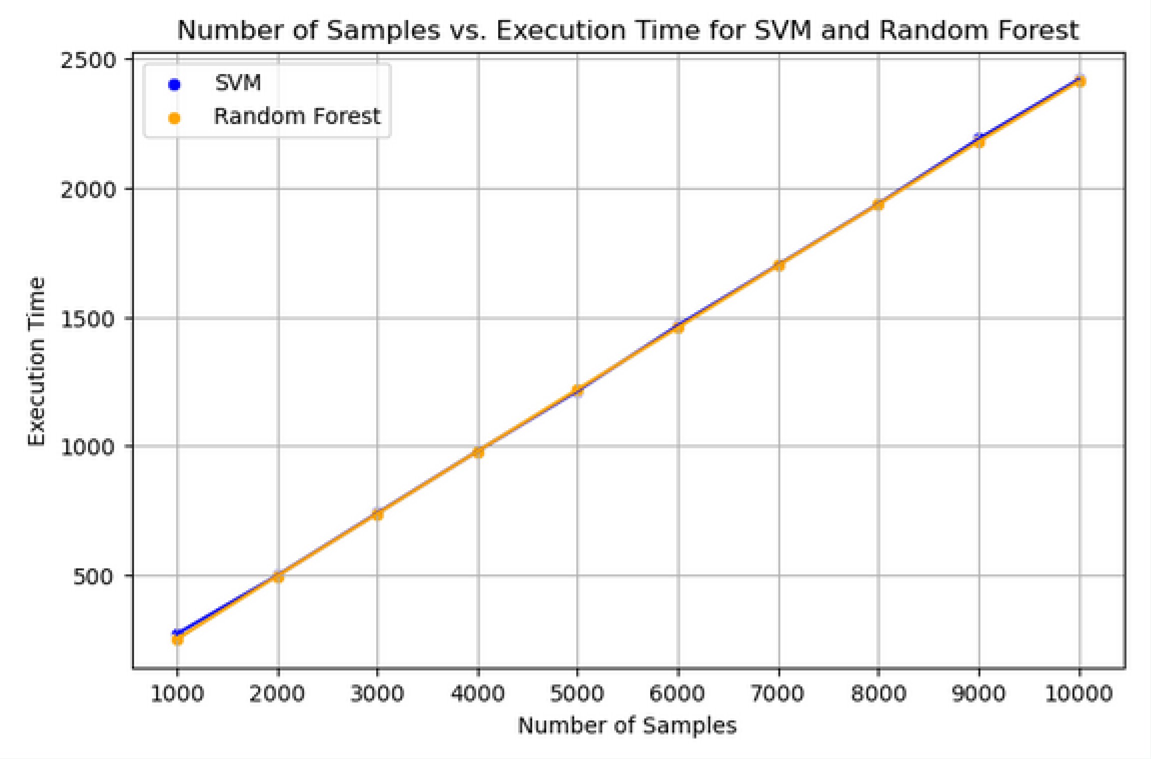


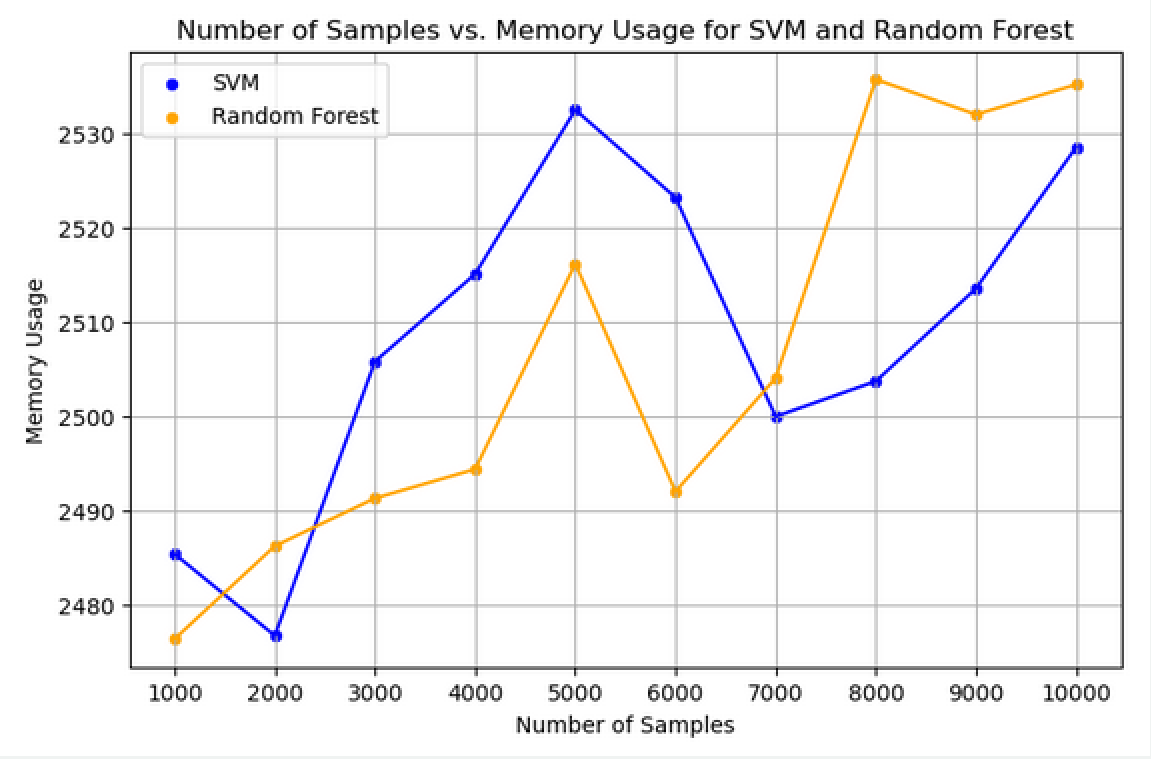
Performance Metrics (Recall + F1-Score)



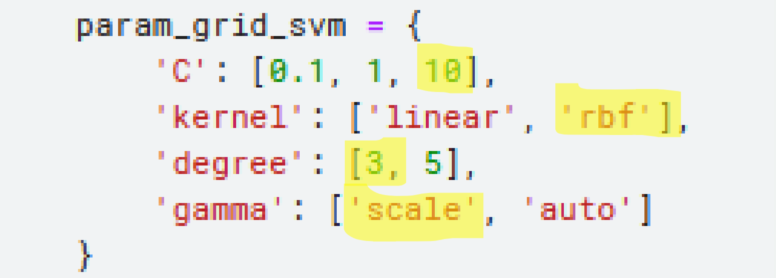


Performance Metrics (Execution Time + RAM Usage)

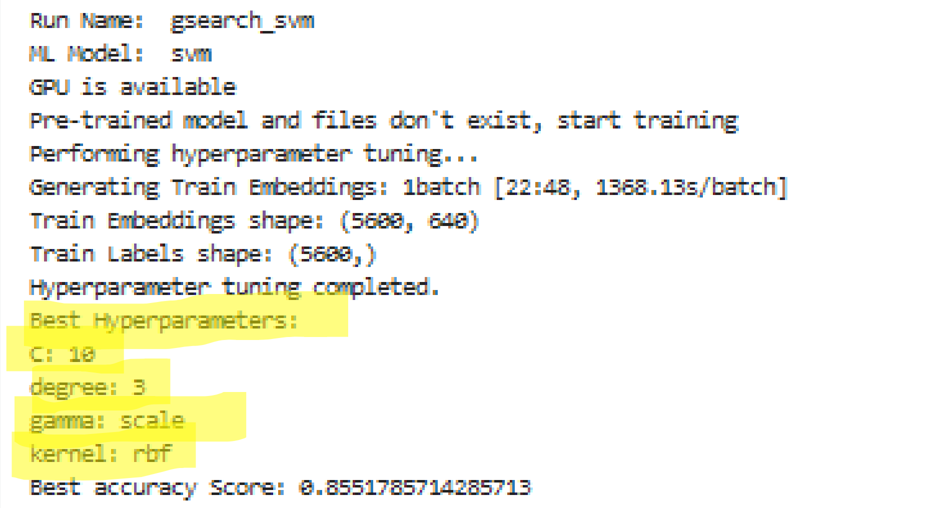




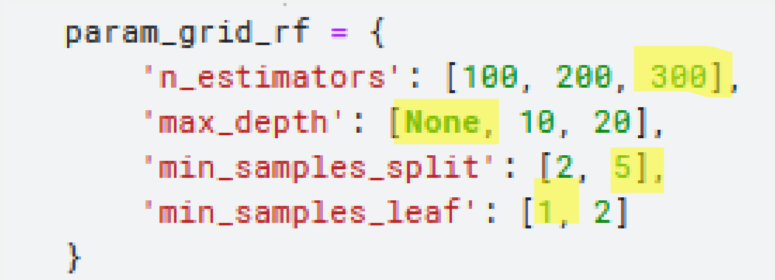
Hyperparameter Tuning



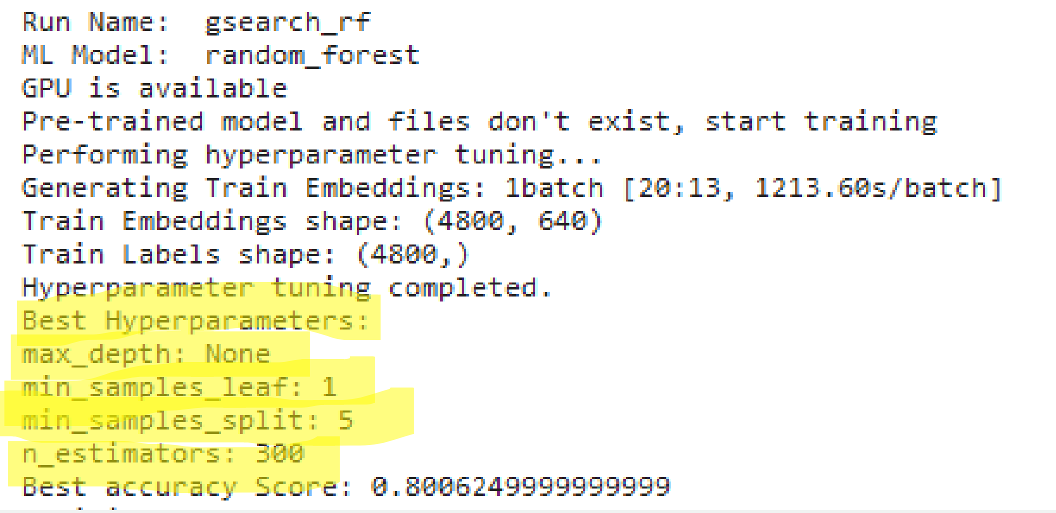
```
param_grid_svm = {
    'C': [0.1, 1, 10],
    'kernel': ['linear', 'rbf'],
    'degree': [3, 5],
    'gamma': ['scale', 'auto']
}
```



```
Run Name:  gsearch_svm
ML Model:  svm
GPU is available
Pre-trained model and files don't exist, start training
Performing hyperparameter tuning...
Generating Train Embeddings: 1batch [22:48, 1368.13s/batch]
Train Embeddings shape: (5600, 640)
Train Labels shape: (5600,)
Hyperparameter tuning completed.
Best Hyperparameters:
C: 10
degree: 3
gamma: scale
kernel: rbf
Best accuracy Score: 0.8551785714285713
```



```
param_grid_rf = {
    'n_estimators': [100, 200, 300],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5],
    'min_samples_leaf': [1, 2]
}
```



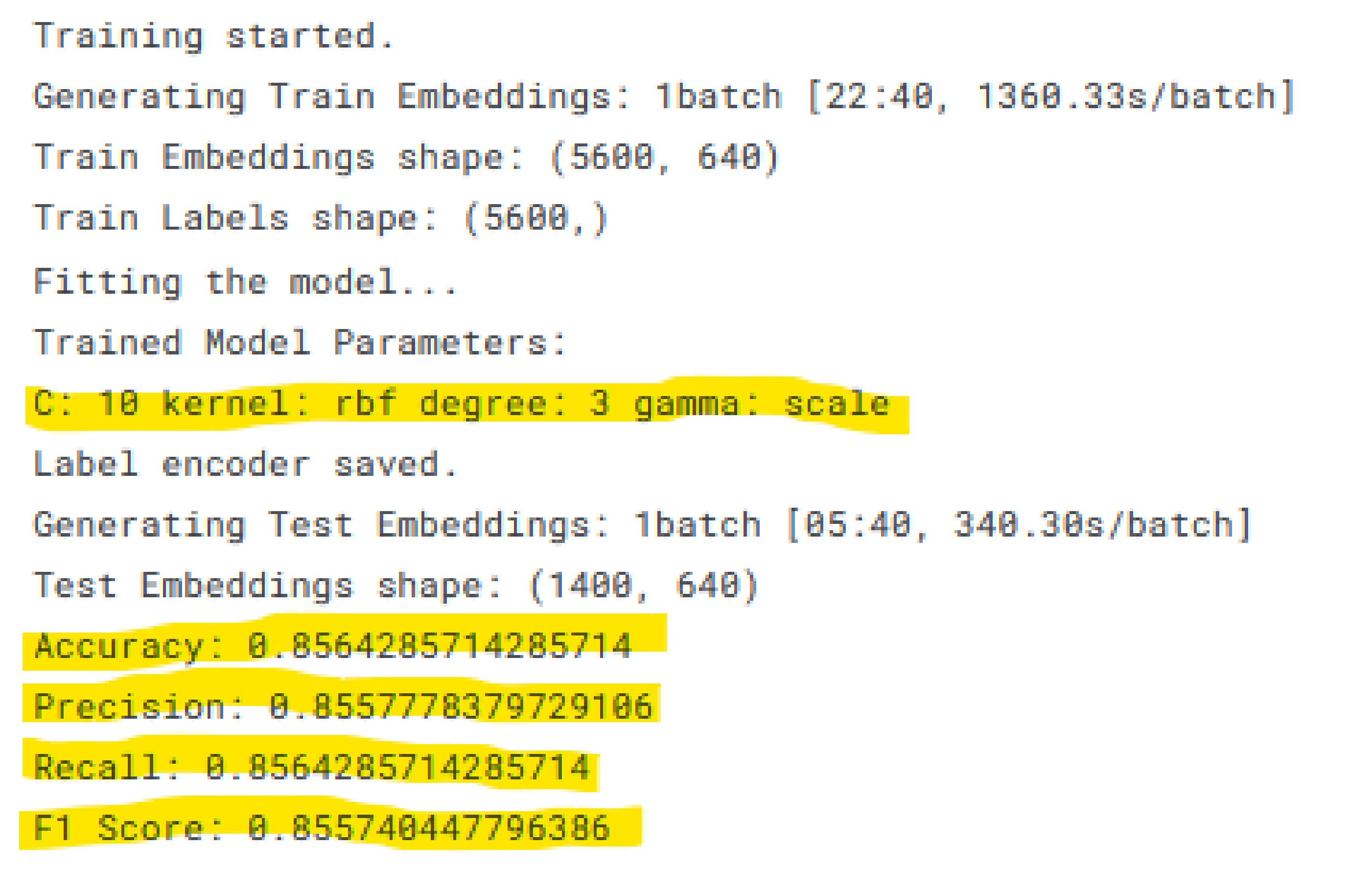
```
Run Name:  gsearch_rf
ML Model:  random_forest
GPU is available
Pre-trained model and files don't exist, start training
Performing hyperparameter tuning...
Generating Train Embeddings: 1batch [20:13, 1213.60s/batch]
Train Embeddings shape: (4800, 640)
Train Labels shape: (4800,)
Hyperparameter tuning completed.
Best Hyperparameters:
max_depth: None
min_samples_leaf: 1
min_samples_split: 5
n_estimators: 300
Best accuracy Score: 0.8006249999999999
```

ML Training Summary

- Total Models Built : 10 models for 2 ML Algorithm i.e. 20 + 2 GridSearchCV Models
- Minimum Execution Time = 4 min
- Maximum Execution Time = 40 min
- Average Execution time for a regular ML model = 22 min

- Average Execution Time for hyperparameter tuning = 1 h 48 min
- Total Combinations tried for SVM = 24
- Total Combinations tried for RF = 36

Conclusion – Best Performing Model

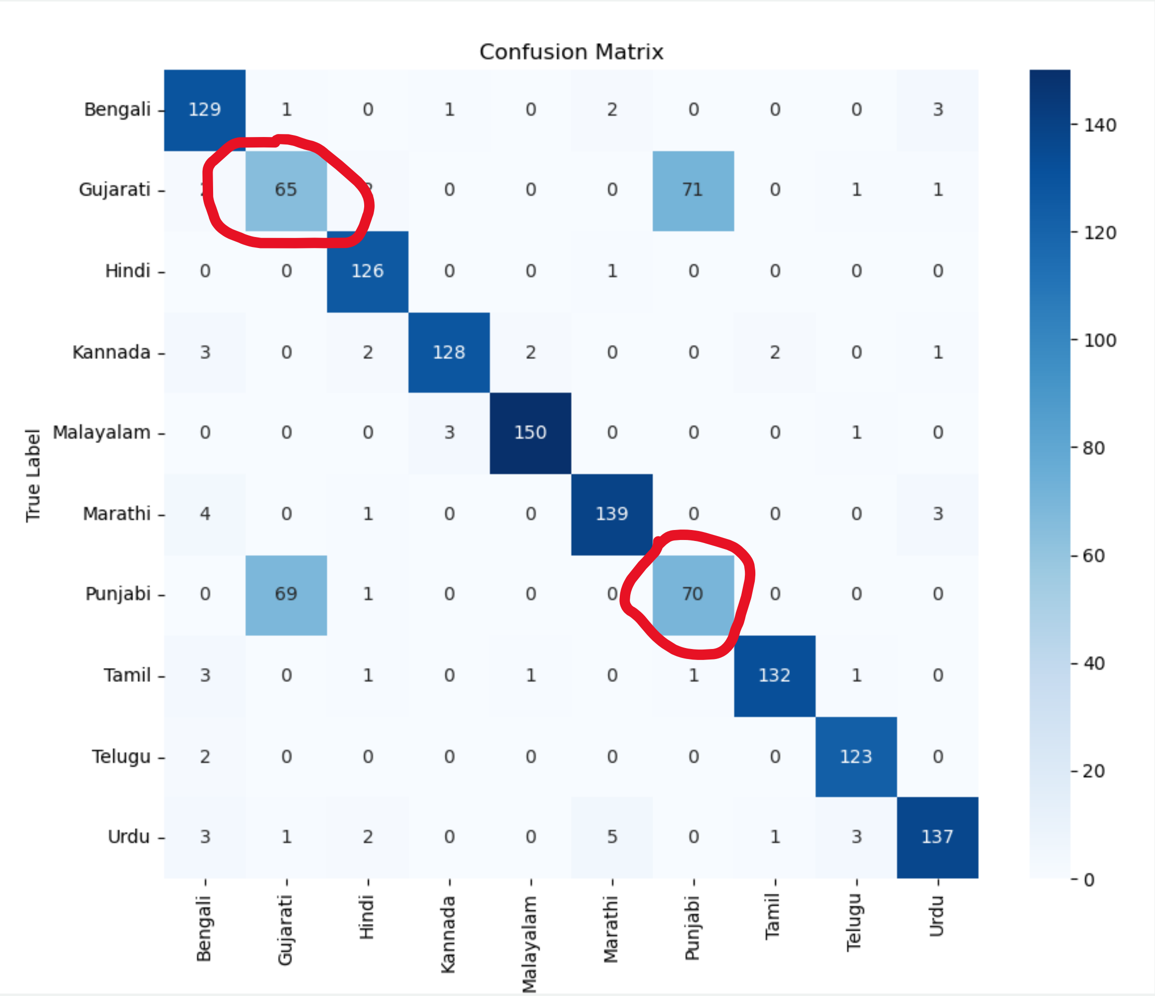


```
Training started.
Generating Train Embeddings: 1batch [22:40, 1360.33s/batch]
Train Embeddings shape: (5600, 640)
Train Labels shape: (5600,)
Fitting the model...
Trained Model Parameters:
C: 10 kernel: rbf degree: 3 gamma: scale
Label encoder saved.
Generating Test Embeddings: 1batch [05:40, 340.30s/batch]
Test Embeddings shape: (1400, 640)
Accuracy: 0.8564285714285714
Precision: 0.8557783797291 06
Recall: 0.8564285714285714
F1 Score: 0.855740447796386
```

- Best Model: *gsearch_svm* (SVM Classifier)
- Accuracy: 0.85
- Precision: 0.85
- Recall: 0.85
- F1 Score: 0.85

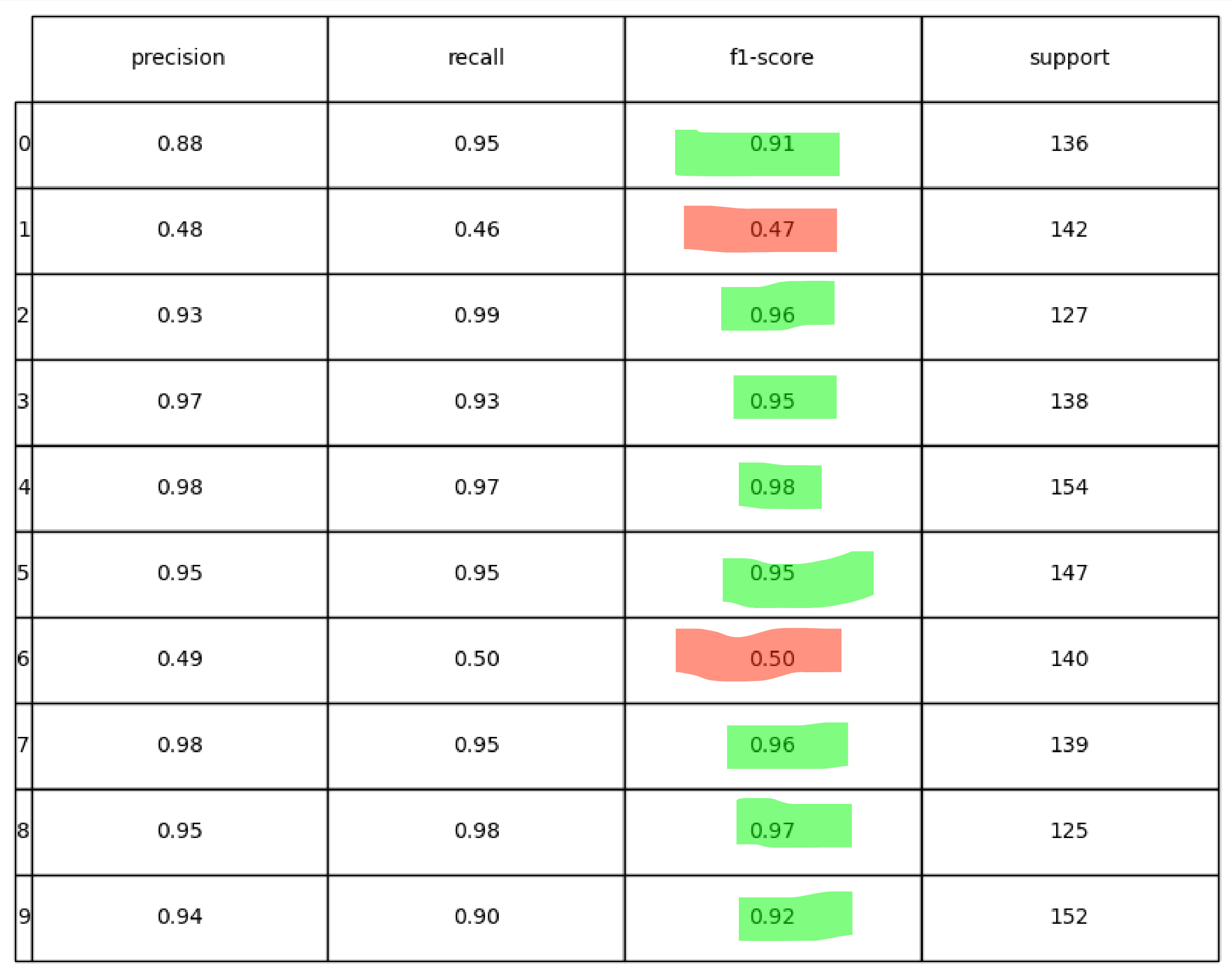
- Note: The selected model, 'gsearch_svm', demonstrates high accuracy and a good balance between precision and recall, making it a reliable and effective choice for language detection in real-world applications.

Performance Reports for *gsearch_svm*



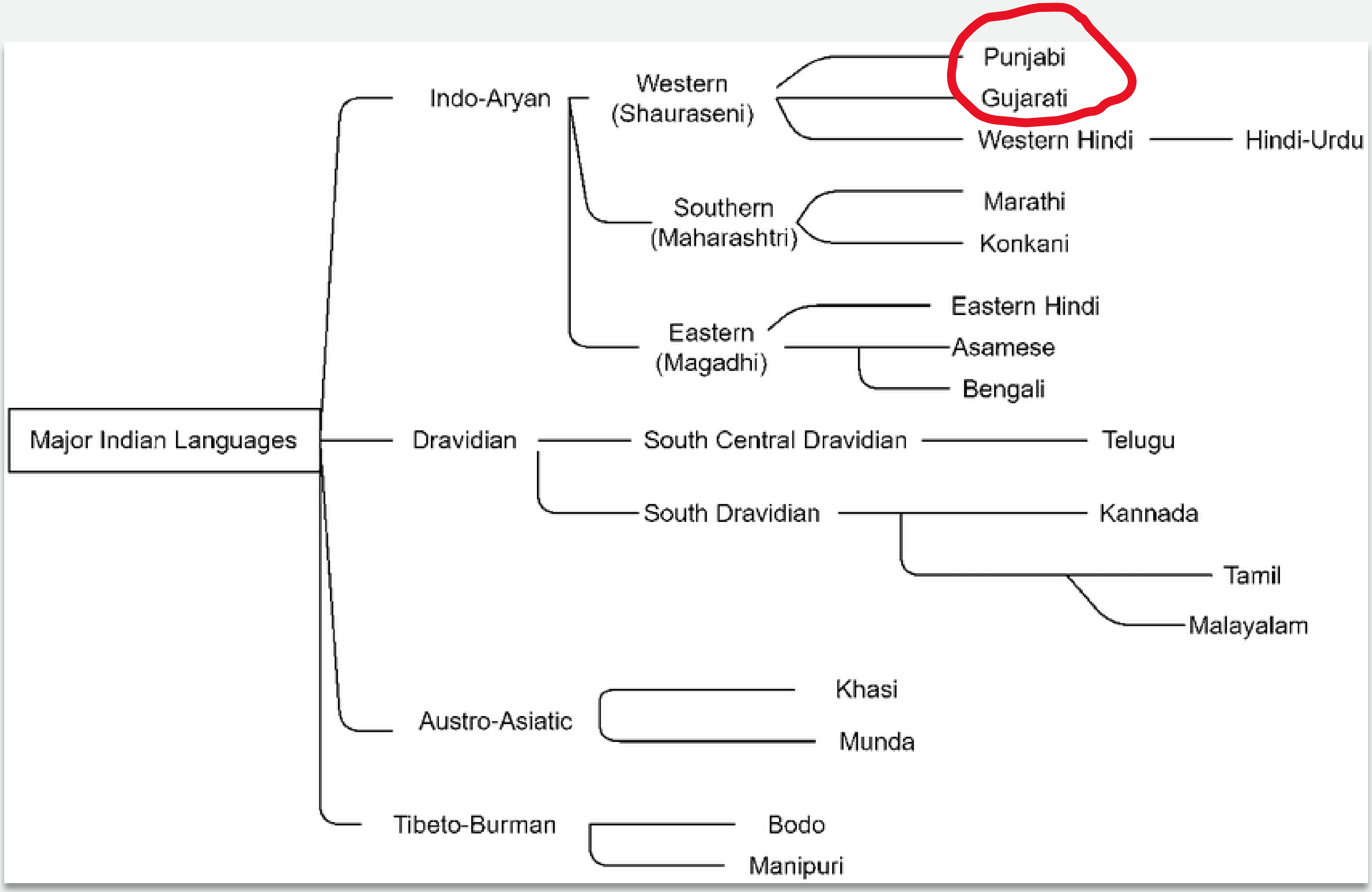
True Label	Bengali	Gujarati	Hindi	Kannada	Malayalam	Marathi	Punjabi	Tamil	Telugu	Urdu
Bengali	129	1	0	1	0	2	0	0	0	3
Gujarati	2	65	2	0	0	0	71	0	1	1
Hindi	0	0	126	0	0	1	0	0	0	0
Kannada	3	0	2	128	2	0	0	2	0	1
Malayalam	0	0	0	3	150	0	0	0	1	0
Marathi	4	0	1	0	0	139	0	0	0	3
Punjabi	0	69	1	0	0	0	70	0	0	0
Tamil	3	0	1	0	1	0	1	132	1	0
Telugu	2	0	0	0	0	0	0	0	123	0
Urdu	3	1	2	0	0	5	0	1	3	137

Confusion Matrix



	precision	recall	f1-score	support
0	0.88	0.95	0.91	136
1	0.48	0.46	0.47	142
2	0.93	0.99	0.96	127
3	0.97	0.93	0.95	138
4	0.98	0.97	0.98	154
5	0.95	0.95	0.95	147
6	0.49	0.50	0.50	140
7	0.98	0.95	0.96	139
8	0.95	0.98	0.97	125
9	0.94	0.90	0.92	152

Classification Report



Tree diagram to illustrate the language closeness of major Indian languages

Dewangan, Shubham & Alva, Shreya & Joshi, Nitish & Bhattacharyya, Pushpak. (2021). Experience of neural machine translation between Indian languages. Machine Translation. 35. 10.1007/s10590-021-09263-3.

Acknowledgements

Thank You!

Ms. Angel Griffin (MSDA Associate Director)

Dr. Ermelinda DeLaVina (MSDA Director)

Dr. Dvijesh Shastri (Capstone Mentor)

All respectful MSDA Faculties!

Demo

Github Repo: https://github.com/sonalgan/IndiSpeak