Flow Diagram

Get the Necessary Libraries

Transformers (Used in code)

Why? As it is the official library from Hugging Face with extensive support for different models and features.

Exception (If any Package is missing)

Import the Necessary Libraries
(Python Libraries)

Simpletransformers (Not Used)

Why? Using the transformers library instead of simpletransformers, Provides more control and flexibility for model fine-tuning and

LOAD the Dataset
(Hugging Face)

Converting Dataset into DataFrames

Why? It can streamline the pre-processing and exploration steps, leading to more effective data preparation.

Data Cleaning
- Removing Duplicates
- Handling Missing Values
- Data Type Conversion
- Outlier Detection and Handling
- Standardizing and Normalizing Data

Data Inspection
Examine the structure and content of the dataset.

PreProcessing
- Tokenization
- Padding and Truncation
- Special Tokens and Attention Masks

Create Training and Validation Sets
Split the dataset into training and validation sets.

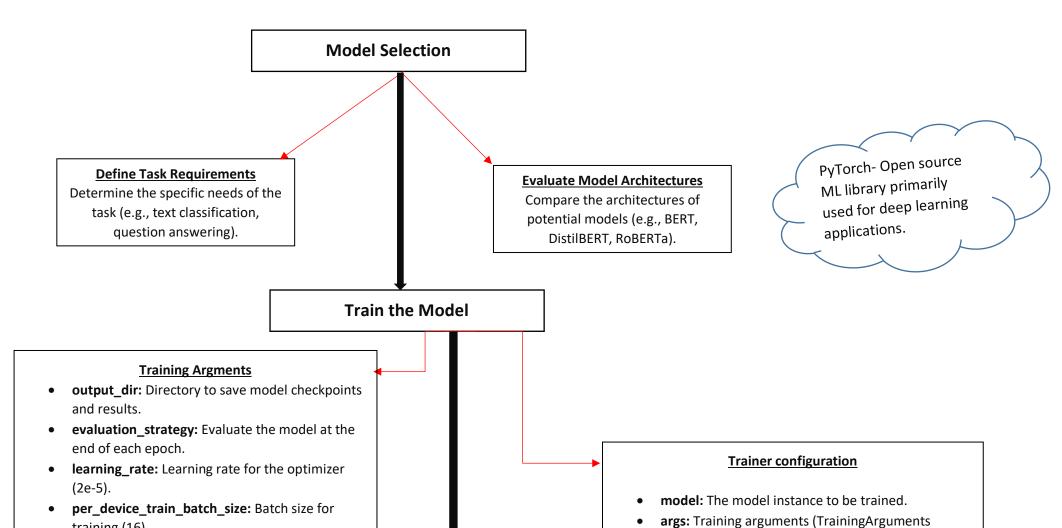**Important Pointers to remember:**

Indicators!!

Red Arrow indicates the types/methods included

Black Arrow indicates the Flow Of the model

Yellow box indicates the Reasons or any exception

Blue Cloud includes the Extra points

"Used print statements multiple times to verify that we are on the right track."

Model Selection

Define Task Requirements
Determine the specific needs of the task (e.g., text classification, question answering).

Evaluate Model Architectures
Compare the architectures of potential models (e.g., BERT, DistilBERT, RoBERTa).

PyTorch- Open source ML library primarily used for deep learning applications.

Train the Model

Training Argments

- **output_dir:** Directory to save model checkpoints and results.
- **evaluation_strategy:** Evaluate the model at the end of each epoch.
- **learning_rate:** Learning rate for the optimizer (2e-5).
- **per_device_train_batch_size:** Batch size for training (16).
- **per_device_eval_batch_size**: Batch size for evaluation (16).
- **num_train_epochs:** Number of epochs to train the model (2).
- **weight_decay:** Weight decay for regularization (0.01).
- **logging_dir:** Directory to save training logs.
- **logging_steps:** Frequency of logging training metrics (every 1000 steps).

Trainer configuration

- **model:** The model instance to be trained.
- **args:** Training arguments (TrainingArguments instance).
- **train_dataset:** Dataset to be used for training.
- **eval_dataset:** Dataset to be used for evaluation/validation.
- **tokenizer:** Tokenizer instance for encoding the data.

Evaluate the Model → Prepare Evaluation Data
→ Load the Model
→ Generate Predictions
→ Compute Metrics
→ Analyze Results

Why Evaluate?

- Measure Performance Identify
- Overfitting/Underfitting
- Compare Models
- Fine-Tune Hyperparameters
- Track Progress

Save the Model

Why Save?

- Reusability
- Deployment
- Reproducibility
- Backup
- Transfer Learning
- Performance Evaluation

Points kept in mind while designing the model:-

- Clear Objectives
- Choose Appropriate Architecture
- Understand the Data
- Evaluate Metrics
- Ensure Scalability
- Plan for Deployment

Visualization and Reporting

Bar Charts
Loss Curves
Precision-Recall Curves
Confusion Matrix

Matplotlib, pyplot >> Popular for creating static, animated and interactive Visuals.

Process:-

a. Calculating Length
b. Creating Plot/Subplot
c. Displaying the Plot

MODEL PREPARED