**Multiclass Text Classification using BERT**

What is multiclass classification?

It is a supervised ML task model which predicts one label out of three or more possible classes unlike binary classification which has only two classes

1. Problem type

Given an input, the model assigns it to one of N classes

1. Algorithms used
2. Logistic Regression – One vs Rest or Softmax function for multiclass
3. Naïve Bayes
4. Support Vector Machined (SVM) – one vs one or one vs rest
5. Decision Trees and Random Forest
6. Neural Networks using softmax activation in output layer
7. Evaluation Metrics
8. Accuracy ( if classes are balanced )
9. Precision, Recall, F-1 Score ( Per class or weighted average)
10. Confusion Matrix ( for misclassifications )

**Multiclass Classification in NLP**

In NLP multiclass classification is used for

1. Sentiment Analysis
2. Topic Classification
3. Intent Detection
4. Spam Detection

**Multiclass Classification Using BERT**

What is BERT?

BERT mean Bidirectional Encoder Representation from Transformers. It is an NLP model based on **transformer architecture** designed to understand context of words in a sentence bidirectionally i.e from left to right and right to left

What is a Transformer?

A transformer is a type of neural network architecture that excels at processing and understanding sequences of data, like text.

It has

* Self-Attention Mechanism - Allows each word to weigh the importance of all other words in the sentences
* Computes Query (Q), Key (K), Value (V) matrices to determine attention scores.
* 

A Softmax classifier is a multi-class classifier that is only used as an output layer of a neural network that outputs a probability distribution over all possible classes

(<https://www.youtube.com/watch?v=8ah-qhvaQqU>)

(<https://www.geeksforgeeks.org/what-is-softmax-classifier/>)

* With context to BERT it Helps BERT understand context bidirectionally (unlike unidirectional models like GPT(**A GPT** (Generative Pre-trained Transformer) model is a type of large language model (LLM) that uses deep learning to generate human-like text) (<https://www.geeksforgeeks.org/introduction-to-generative-pre-trained-transformer-gpt/>)
* Multi-Head Attention- Runs multiple self-attention layers in parallel ("heads") then Concatenates outputs and linearly transforms them.
* Positional Encoding - Since Transformers don’t process words sequentially (unlike RNNs), they add **positional embeddings** to retain word order information.
* Layer Normalization and Residual Connections - **Residual connections** (skip connections) help avoid vanishing gradients. **Layer normalization** stabilizes training.

BERT uses Transformer Architecture which uses self attention mechanism from Transformer model to weigh the importance of different words in a sentence. This enables parallel processing making it efficient for large scale training

Pre-Training and Fine-Tuning:

BERT is pre trained on a massive corpus using two tasks

1. Masked Language Modeling (MLM) - Randomly masks 15% of words and predicts them.
2. Next Sentence Prediction(NSP) – Predicts if two sentences follow each other

It is fine tuned for downstream tasks like sentiment analysis, NER etc.

Applications of BERT:

* Text Classification (e.g., spam detection, sentiment analysis)
* Named Entity Recognition (NER)
* Question Answering (e.g., SQuAD dataset)
* Text Summarization
* Machine Translation
* Chatbots & Virtual Assistants

This project takes data from SciKit - Scikit-learn is an open source machine learning library that supports supervised and unsupervised learning. It also provides various tools for model fitting, data preprocessing, model selection, model evaluation, and many other utilities.

Data Preprocessing is done using SpaCy which is a NLP library. The data is converted into tokens and cleaned.

The text is the vectorized using BERT using BERT Tokenizer converting the training and testing split into tokens.

It is then structured for batching which is then again converted into train and test set

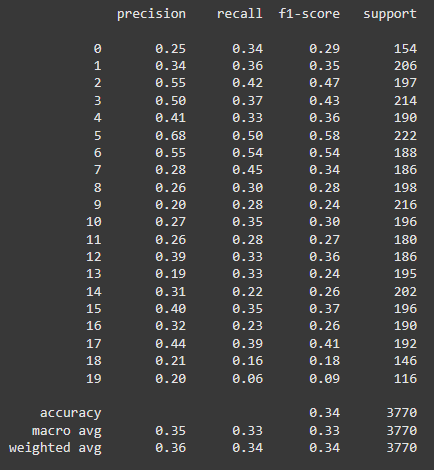
The sets are then loaded into DistilBERT transformer for sequence classification

After giving steps and training arguments to initialize the trainer API

The training took around 1 hour for 5 epochs with the results as

{'eval\_loss': 2.236729860305786, 'eval\_accuracy': 0.3376657824933687, 'eval\_runtime': 57.2868, 'eval\_samples\_per\_second': 65.809, 'eval\_steps\_per\_second': 8.239, 'epoch': 5.0}

This gave non impressive results for the classification into the defined categories



The overall accuracy is 34% which is very low.

The model needs more training, resources and better parameters for a better performance. The training takes a lot of time when using a BERT encoder

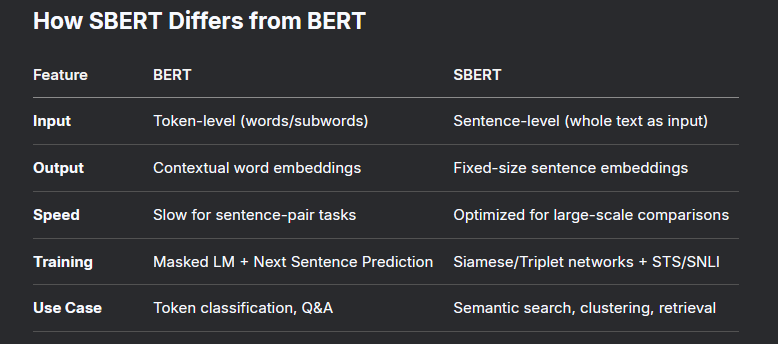
**Using SBERT-SHAP-XGBoost Stack**

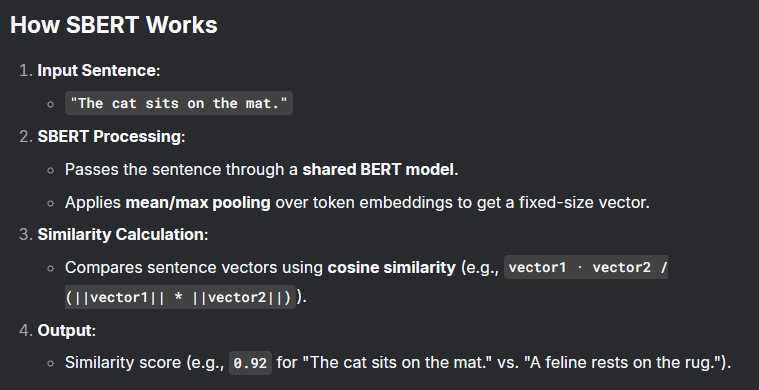
The steps are mirrored to the DistilBERT model but more optimized for speed and interpretability

**SBERT**

SBERT (Sentence-BERT) is a modification of the BERT model designed to efficiently generate semantically meaningful **sentence embeddings**. Unlike vanilla BERT, which processes sentences in pairs (slow for large-scale tasks), SBERT produces fixed-sized sentence vectors that can be compared using cosine similarity, enabling fast semantic search, clustering, and similarity tasks.

It is almost a 100% faster than the base BERT model





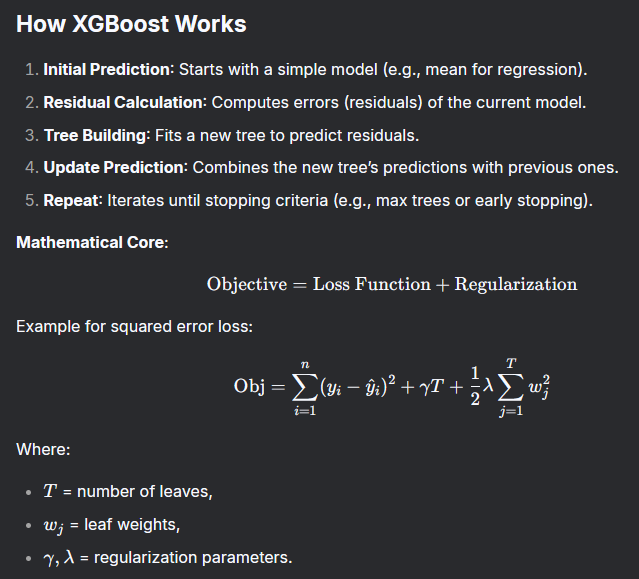
**XGBoost** (eXtreme Gradient Boosting)

It is a highly optimized, scalable machine learning algorithm based on gradient boosting, designed for speed and performance. It dominates structured/tabular data problems.

XGBoost uses decision trees as its base learners combining them sequentially to improve the model’s performance. Each new tree is trained to correct the errors made by the previous tree and this process is called boosting.

It has built-in parallel processing to train models on large datasets quickly. XGBoost also supports customizations allowing users to adjust model parameters to optimize performance based on the specific problem. Its used for

Regression, classification (binary/multiclass), ranking tasks. Custom loss functions.



**SHAP**

SHAP (SHapley Additive exPlanations) is a game theory-based method used to explain the output of machine learning models. It assigns each feature an importance value for a specific prediction

