AutoML Implementation

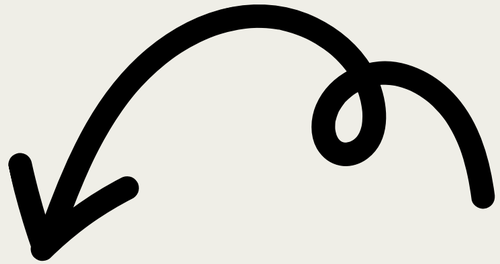
MULTI-LABEL TEXT CLASSIFICATION- TPOT &BERT

Janani Harshatha J
janani1473@gmail.com

- Multi-label Classification: A type of classification problem where each instance can belong to multiple classes simultaneously.
- TPOT: An automated machine learning library that optimizes machine learning pipelines using genetic programming.
- BERT: A transformer-based model that provides state-of-the-art performance on a wide range of NLP tasks.

WORKFLOW

- Data Preprocessing
- Feature Extraction
- Tokenization
- Extracting BERT embeddings
- TPOT Configuration
- Model Training and Evaluation
- Wrapping TPOT in a MultiOutputClassifier.
- Evaluating the model on the test data



FEEDBACK COLLECTION

FEEDBACK COLLECTION:

COMPONENTS:

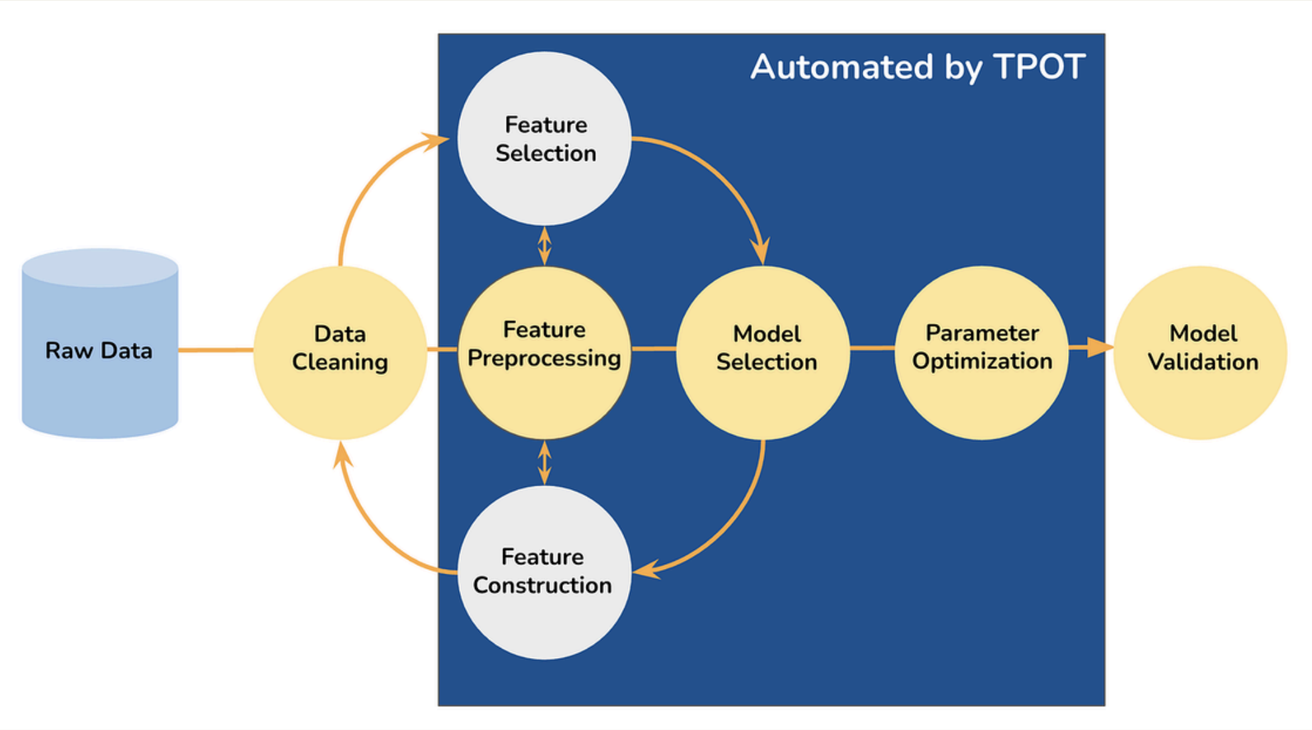
- UPLOAD DATA POINTS
- UPLOAD FILES
- PREDICT AND CORRECT THE OUTPUT USING THE TRAINED MODEL

At regular intervals of time the feedback data is pre-processed and is used for re-training the model. Only the feedback data is used to re-train because of computational complexity. Once in a while the feedback data combined with the original data can be used to train the model.

BENEFITS OF USING FEEDBACK DATA FOR RETRAINING

1. Improved Accuracy: Feedback data helps the model learn from its mistakes, leading to enhanced accuracy.
2. Adaptation to Real-World Scenarios: User feedback ensures the model adapts to the specific nuances and variations in real-world data.
3. Continuous Learning: Incorporating feedback allows the model to continuously learn and improve over time.
4. User-Centric Improvements: Focusing on user feedback prioritizes improvements that are most relevant to users.
5. Efficient Use of Resources: Retraining with feedback data can be more resource-efficient than using the entire dataset.
6. Quality Control: Feedback data often includes corrections, helping identify and rectify systematic errors in the model.

TPOT : MULTI-LABEL CLASSIFICATION



TPOT (Tree-based Pipeline Optimization Tool) is an automated machine learning (AutoML) tool that optimizes machine learning pipelines using genetic programming. While TPOT is primarily designed for single-label classification and regression tasks, it can be adapted for multi-label classification with some modifications.

HOW IS TPOT USED HERE:

To use TPOT for multi-label classification, you first need to prepare your dataset where the target variable is a binary matrix indicating the presence or absence of each label. TPOT, which is designed for single-label tasks, can be adapted by including a multi-label classifier such as OneVsRestClassifier or MultiOutputClassifier in the pipeline. These classifiers transform the multi-label problem into multiple binary classification tasks, training a separate classifier for each label.

We define a custom configuration for TPOT that includes these multi-label classifiers, allowing TPOT to explore different combinations of preprocessing steps and classifiers to find the best-performing pipeline. During the optimization process, TPOT uses genetic programming to evaluate various pipelines and optimize their performance.

Finally, you evaluate the optimized pipeline using appropriate multi-label metrics such as Hamming loss, or accuracy to ensure it performs well on the multi-label classification task.

HOW TPOT WORKS:

- TPOT uses genetic algorithms, a concept inspired by natural selection (like evolution). It starts with many random pipelines (models) and keeps improving them over generations.
- Pipeline: A pipeline is a complete process that involves data pre-processing (like cleaning and scaling data) and applying a machine learning model.

Steps:

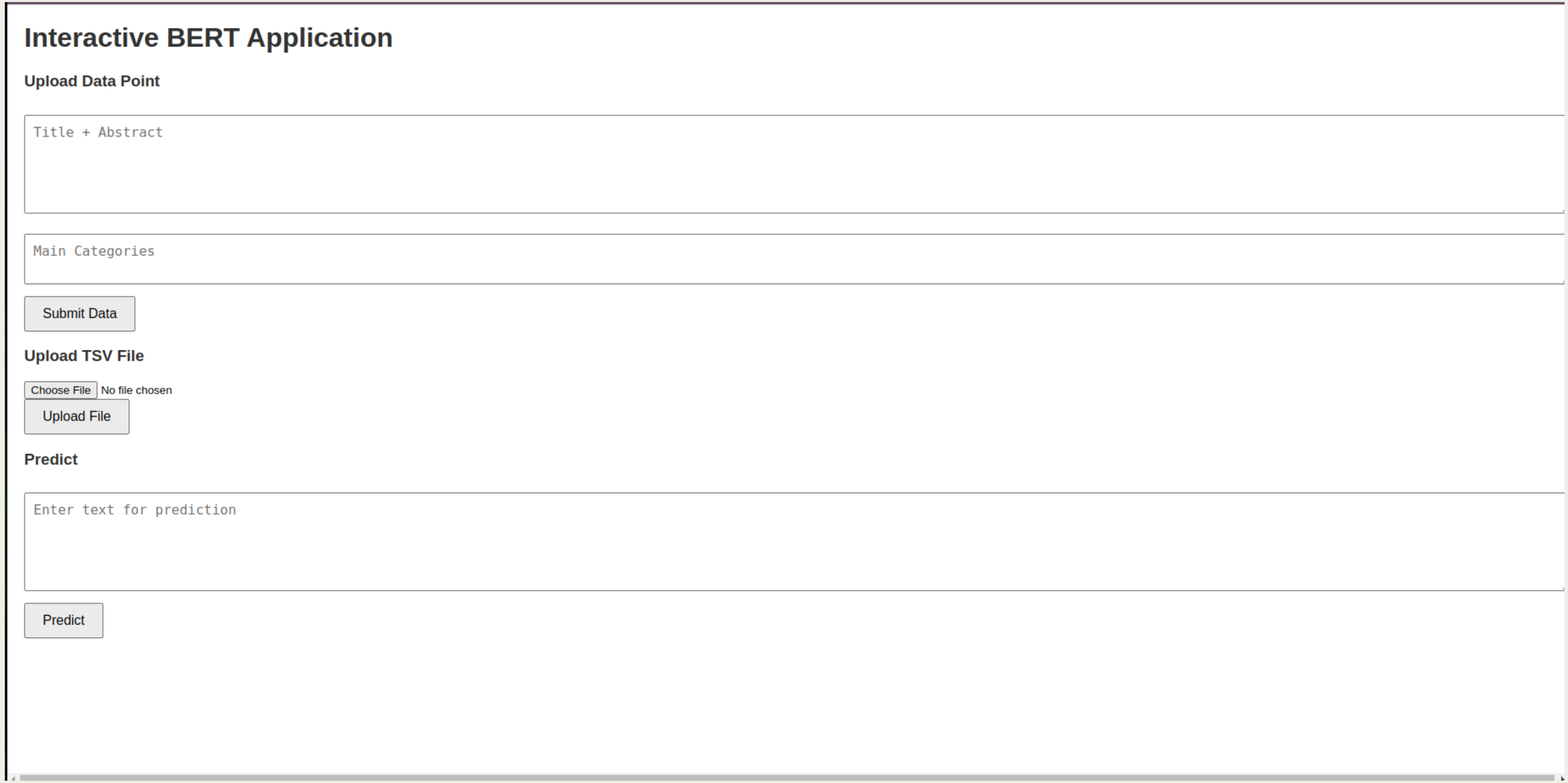
Step 1: TPOT generates random pipelines. These pipelines consist of different combinations of machine learning models and data preprocessing steps.

Step 2: It evaluates these pipelines on the training data to see how well they perform.

Step 3: It keeps the best-performing pipelines and modifies them slightly (like in evolution), mixing and matching parts of different pipelines (models and preprocessing steps) to create new ones.

Step 4: It repeats this process for many generations, improving the pipeline each time until it finds the best solution.

FEEDBACK FORM:



COMPENENTS:

/upload_data: Collects user-uploaded text and categories from the form. The data is saved to a TSV file using the save_to_tsv function.

/upload_tsv: Allows users to upload a TSV file containing text and categories. This function validates the file, ensuring it has the expected columns and is not empty.

/predict: This route accepts input text, tokenizes it using the BERT tokenizer, and feeds it to the BERT model. The model returns a prediction (class label), which is displayed on the webpage. The prediction is mapped to one of eight categories (e.g., astro-ph, cs, etc.). /feedback: Collects feedback from the user. It handles two types of feedback: If the prediction is correct, it saves the input text and the predicted output. If the prediction is incorrect, it saves the correct output provided by the user.

CONCLUSION:

By implementing a continuous learning framework, you can ensure that your model remains up-to-date and relevant over time. This involves collecting and storing feedback data, preprocessing the data, periodically retraining the model with TPOT, evaluating the model, deploying the updated model, and continuously monitoring the performance of the deployed model. This feedback loop ensures that the model continuously improves based on user feedback and new data

THANK YOU !