Documentation: Semantic Classification with DistilBERT

# 1. Workflow

The notebook implements a semantic classification pipeline using DistilBERT. Below is the step-by-step workflow:  
  
1. Importing Libraries: Required packages such as PyTorch, Hugging Face Transformers, and datasets are imported.  
2. Dataset Preparation: The dataset is loaded and tokenized using the DistilBERT tokenizer.  
3. Custom Dataset Class: A PyTorch Dataset class (`HRDataset`) is defined to handle encodings and labels.  
4. Model Initialization: DistilBERT pre-trained model with a classification head is loaded from Hugging Face.  
5. Training Setup: Optimizer, loss function, and training parameters (epochs, batch size, learning rate) are defined.  
6. Training Loop: The model is trained on the dataset with backpropagation and gradient updates.  
7. Evaluation: The model is evaluated on test/validation data to measure performance.  
8. Prediction: The trained model is used to make predictions on new input text.

# 2. Pros and Cons

## Pros

- Uses DistilBERT, which is a lightweight and faster version of BERT.  
- Pre-trained embeddings improve accuracy with minimal training data.  
- Hugging Face Transformers library simplifies implementation.  
- Easily extensible to different text classification tasks.  
- Good trade-off between speed and accuracy.

## Cons

- Training can still be slow on CPU compared to traditional ML models.  
- Requires significant memory for larger datasets.  
- Fine-tuning may overfit on small datasets if not regularized.  
- Black-box model: lacks interpretability compared to classical ML.  
- Dependency on pre-trained models; performance may degrade if domain is very different.