**Logistic regression for the BBC dataset**

As usual we started our implementation from loading the dataset into our working environment. Through our data loader we reads the data from a CSV file, assumes a specific format with category and sentence columns, and returns the data as a list of tuples, where each tuple represents a data point.

Later we have use a method of **calculate\_tf()** a function that calculates the term frequency of each word in the sentence. This function provides a way to transform text data into numerical representations. We used it as method to extract our feature.

Later in the evaluator class we have **evaluate\_logistic\_regression()** function applies a logistic regression model with given weights to a dataset, extracts features from sentences using the calculate\_tf() function, calculates the activation scores for each category, determines the predicted category, compares it with the actual category, and finally calculates and returns the accuracy of the predictions.

Then we have all our utility functions that we perform the main thing which includes the **clear\_data** and the **training** method. Our clear\_data function takes a dataset as input and processes each sentence by removing certain punctuation marks (.,!,?). The processed data is then returned as a list of tuples (sentence, category). It is used to preprocess the dataset and remove punctuation that may not contribute to training the algorithm.

And our training model maps the labels in the dataset to numerical values, initialize weights for each category and feature weights. And through iteration it calculates the activation score for each category based on weights and features. Update the weight using gradient ascent and return the trained weights, category and label to number mapping which are the raw material for the prediction stage.

Finally we have the main.py file where we imported and specify functions. You can run the code over three

**Naïve Bayes for BBC dataset**

Similarly we started our implementation from loading the dataset into our working environment. Through our data loader we load data from a CSV file, shuffle the data randomly, and split it into training and testing datasets based on the provided split ratio. It can be used to prepare data for training and evaluating our models.

And then through our feature extractors we provide functions to preprocess text, generate word embedding, calculate TF-IDF values, and extract a bag-of-words representation from a given dataset.

Then we implement a Naive Bayes Classifier that utilizes a provided feature extractor to extract features from texts, trains the classifier on a dataset, predicts the category of new texts, and calculates the accuracy of the classifier. It leverages probabilities and smoothing techniques to make predictions based on the occurrence of features in different categories.

We provided main.py file where you can run and test the code.