**Report**

**Introduction:**

**1. Pre-processing:** It included stemming, and removal of stop words from our corpus of text.

**2. Dataset used:** For the given task, I used dataset from Kaggle

<https://www.kaggle.com/datasets/jensenbaxter/10dataset-text-document-classification>

Within this dataset there were 10 classes

business  
entertainment  
food  
graphics  
historical  
medical  
politics  
space  
sport  
technologie

**Model Used**

BERT (Bidirectional Encoder Representations from Transformers) is a deep learning model developed by Google. BERT is a pre-trained model that has been trained on a large corpus of data. It is helpful for next sentence prediction and classification.

**BERT Architecture**: BERT is a transformer-based model, which means it processes data using the self-attention mechanism. First, the WordPiece tokenizer is used to tokenize the data into the model. The text then is transferred through a number of transformer blocks after being tokenized. A feed-forward neural network layer and a self-attention layer are the two sub-layers that make up each transformer block.

The **self-attention layer** takes care of processing the tokens in the input series and giving each token a weight based on how significant it is to the other tokens. The results of the self-attention layer are then applied to a non-linear activation function by the feed-forward neural network layer.

BERT has two versions - BERT Base and BERT Large. BERT Base has 12 transformer blocks, and BERT Large has 24 transformer blocks.

**BERT for Text Classification**: Text classification is one of the many NLP jobs that BERT can be tuned for. Text input is passed into the BERT model for text classification, and the classification layer receives its input from the output of the last transformer block that corresponds to the [CLS] token. A straightforward feed-forward neural network that maps the BERT output to the desired number of classes usually serves as the classification layer.

**Fine-tunning the model**

To fine-tune BERT for text classification, a labelled dataset is required. The labelled dataset is used to train the classification layer, and the pre-trained BERT weights are frozen during this process. In our case the Class column was our target column and Document was the training column. To tune BERT, for the model we have used Hugging face Bert and then create functional layers of BERT layer, hidden layer, dropout layer and output layer. Then, we fitted our dataset to the model.

**Evaluation Metrics**

To evaluate the model, I have used precision, recall and accuracy.

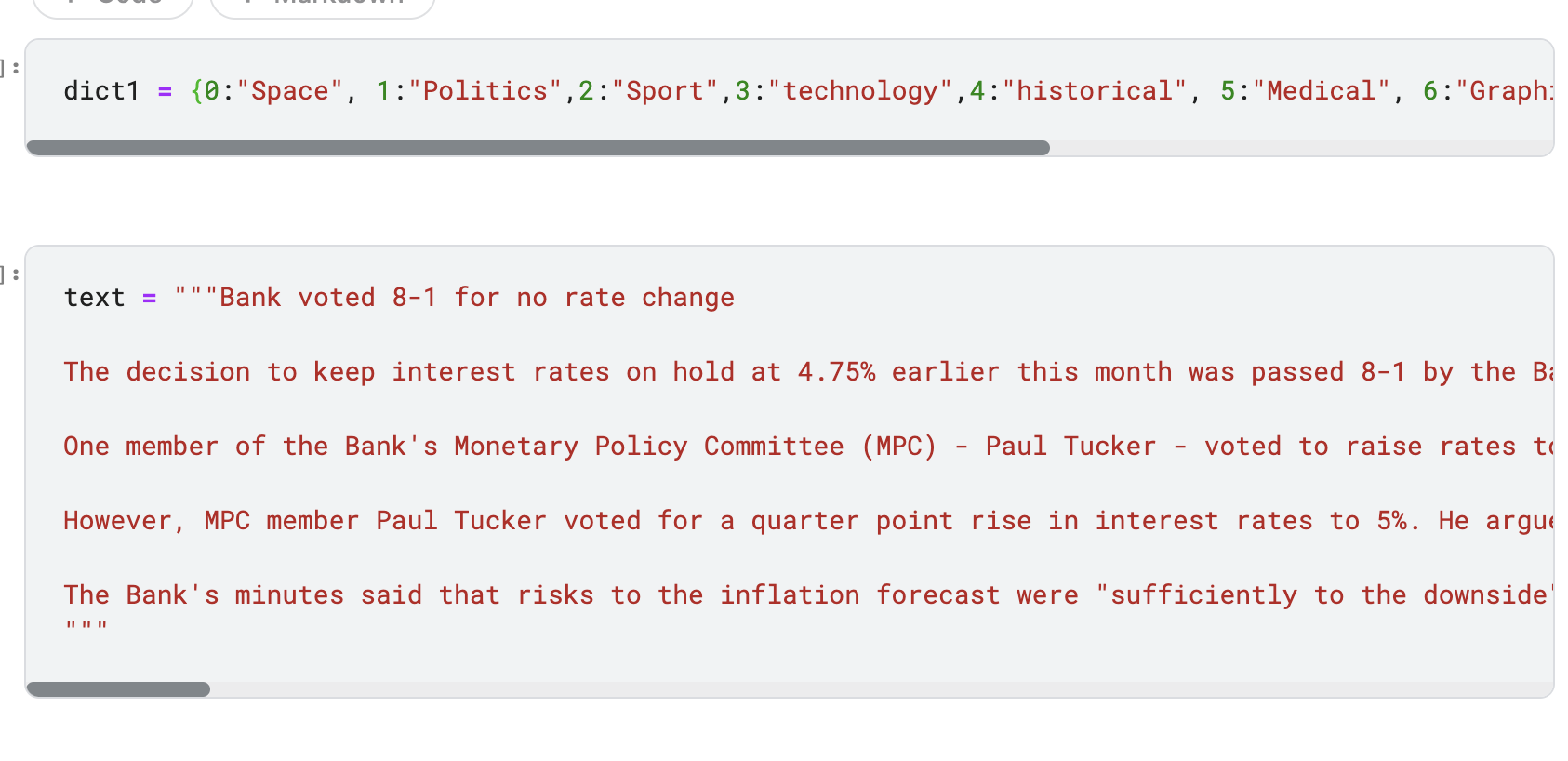
(in %) for validation

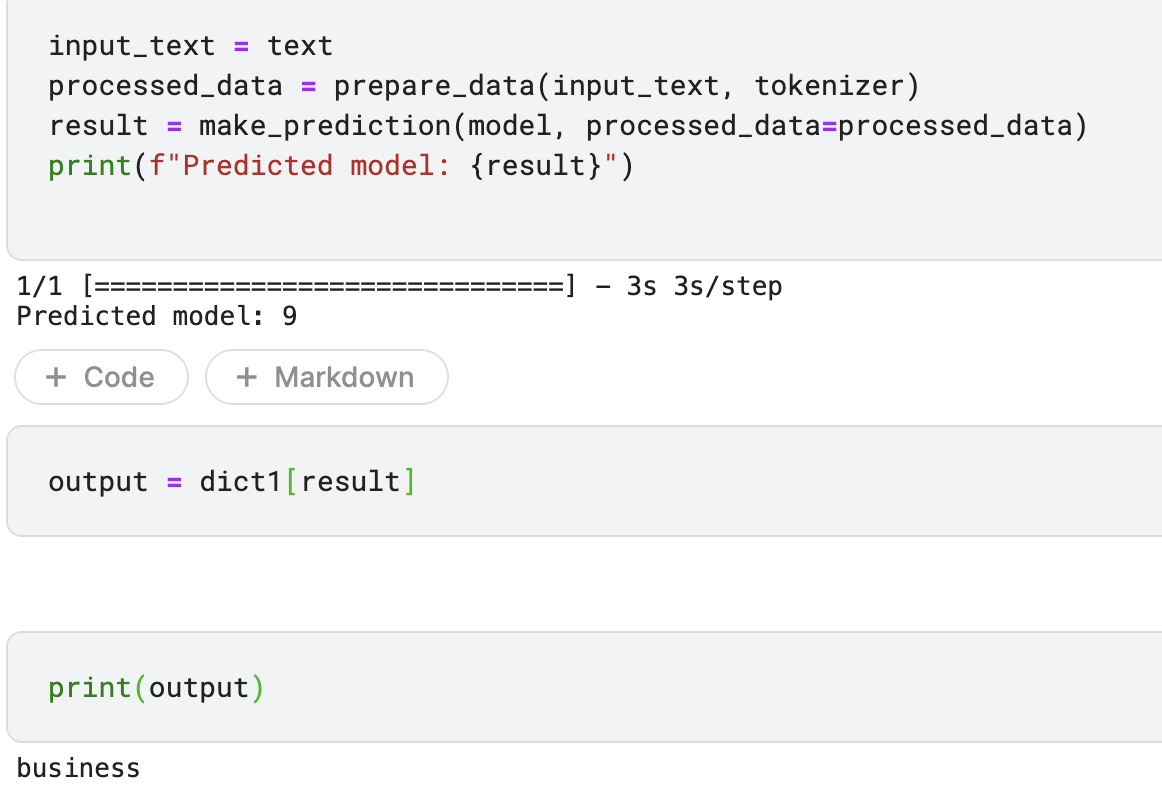
Recall: 97.12

Precision:99.02

Accuracy: 97.12

**Sample Prediction**





**Explanation of the prediction:**

Before passing the text to the model, we need to convert it into tokens that are accepted by BERT.

In the function prepare\_data(input\_text, tokenizer), we provide the text which is then converted into tokens using tokenizer.encode\_plus(). This function takes the input\_text and max\_length as parameters and returns a token from which we can obtain input\_ids and attention masks. We then convert input\_ids and attention masks into float tensors. We now have what we need to pass to our model.

We then pass this processed\_data into make\_prediction(), which generates the output using model.predict(processed\_data).

Since the result obtained from the above predict() method is an integer from 0 to 9, we need to convert it into its respective category, such as business, sports, etc.

**Performance:**

The accuracy, precision, and recall values are excellent, indicating that the model has been trained reasonably well due to the way it is working. On validation dataset the model gave 98% accuracy.

**Improvements:**

1. To improve the fine tuning of model more, we need to provide a bigger corpus of data.
2. Reducing the overfitting of the model is very crucial, we need to apply more techniques to reduce the overfitting of the model.

My kaggle notebook: https://www.kaggle.com/vanshpundirflt/model-text-classification