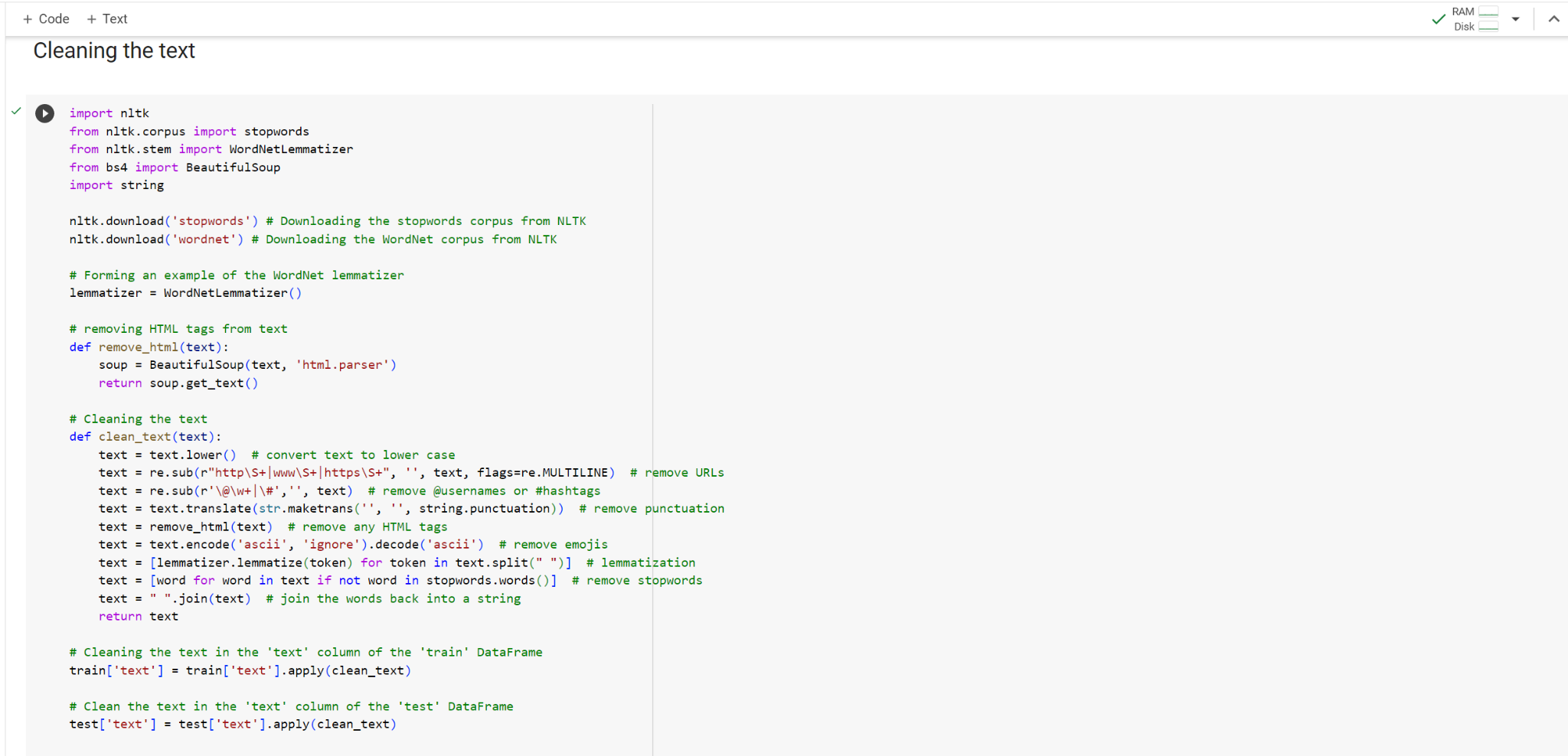
**BUDT737 - BIG DATA AND ARTIFICIAL INTELLIGENCE**

**Project - Natural Language Processing with Disaster Tweets**

**Code Used: Introduction to NLP with TensorFlow and spaCy**

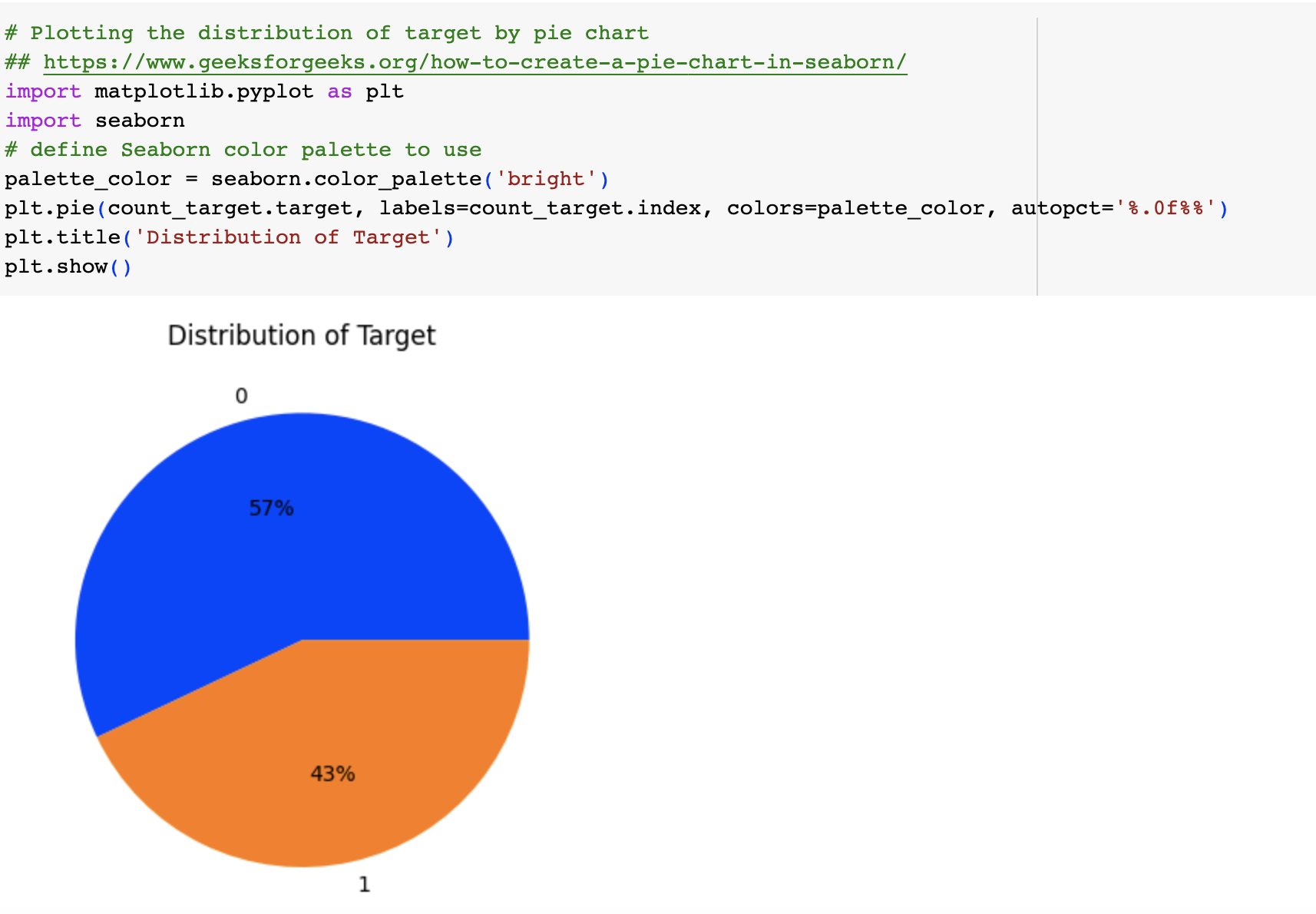
**Link:** [**https://www.kaggle.com/code/olemagnushiback/introduction-to-nlp-with-tensorflow-and-spacy?scriptVersionId=126979046**](https://www.kaggle.com/code/olemagnushiback/introduction-to-nlp-with-tensorflow-and-spacy?scriptVersionId=126979046)

**Data Cleaning And Preprocessing :**

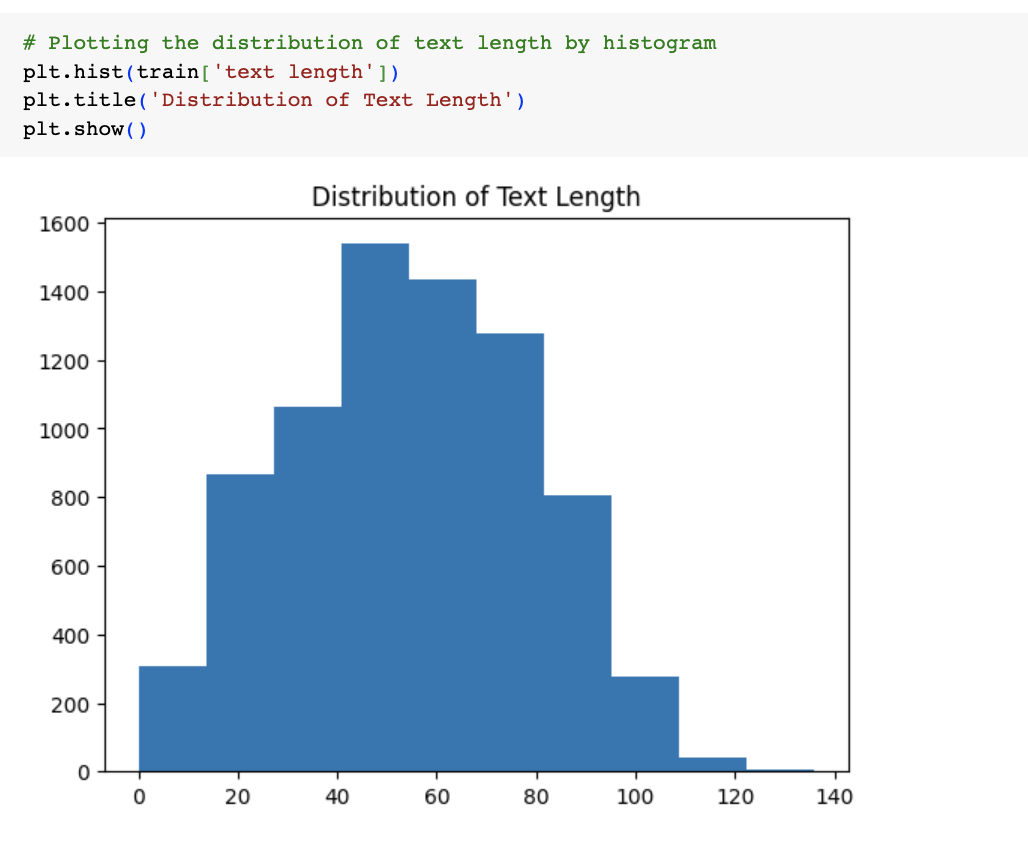


**Exploratory Data Analysis (EDA)**

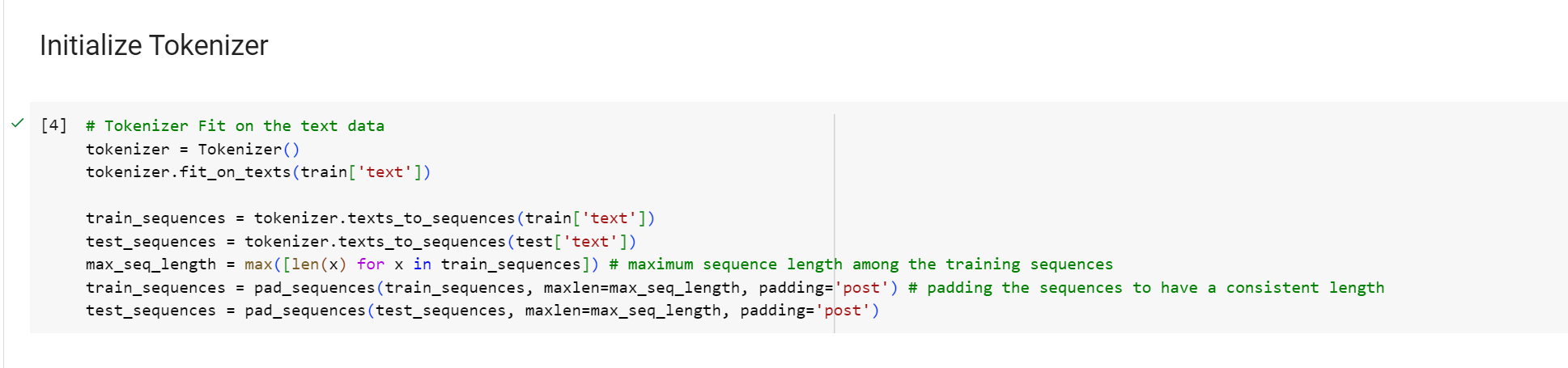
1. **Pie Chart of Distribution of Targets**

****

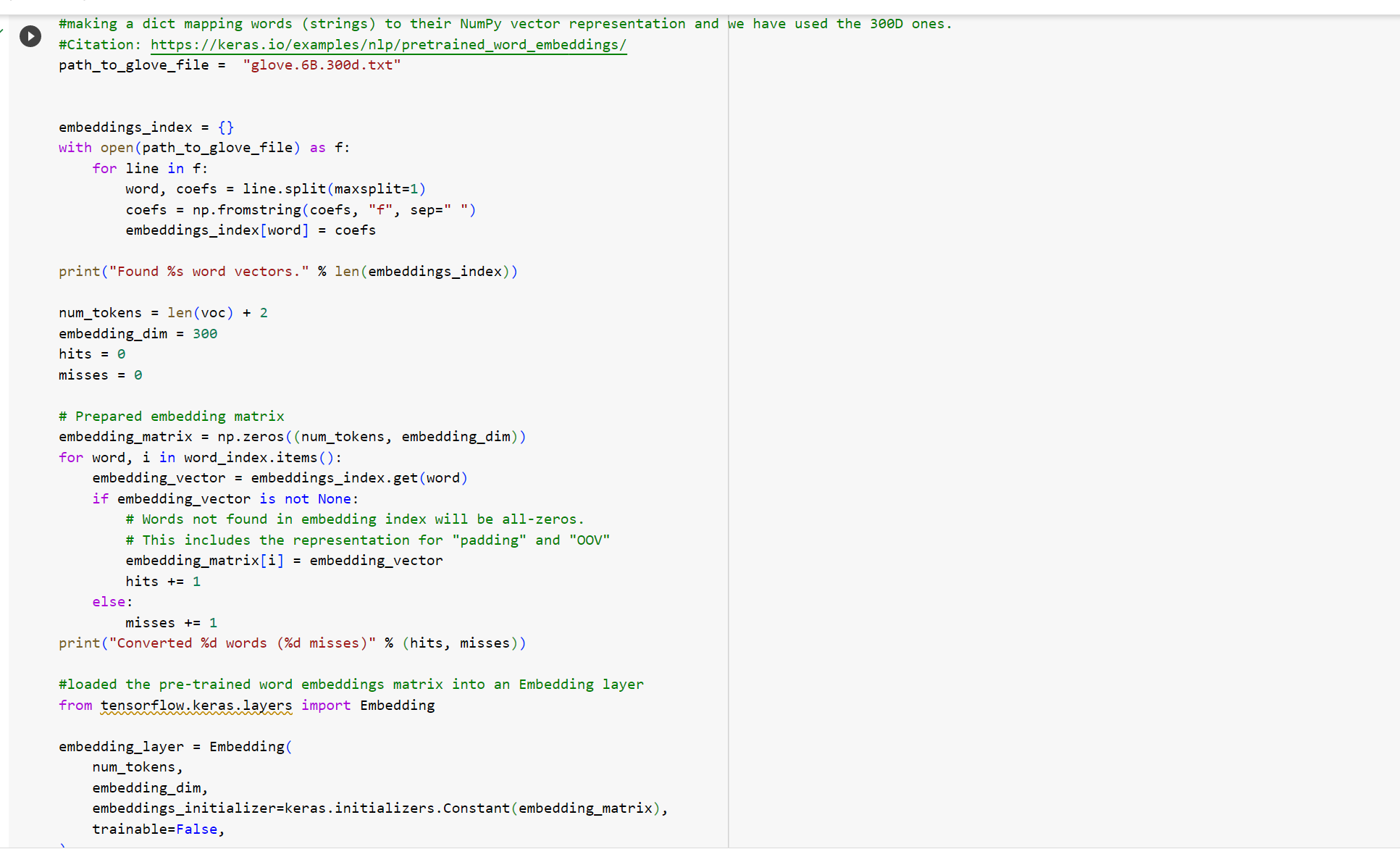
1. **Distribution of text length**

****

**Method 1: Tokenization and Padding (1st Model)**

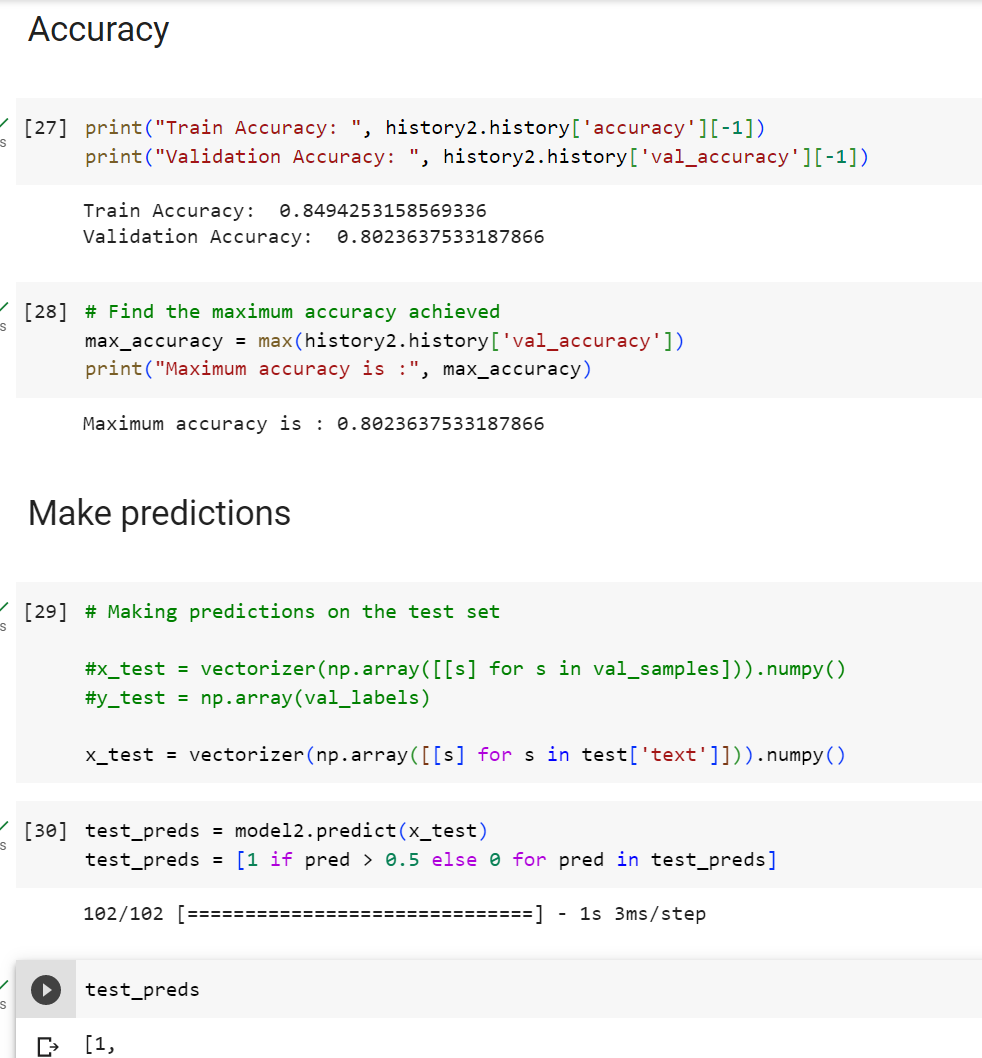


**Method 2: Pre- Trained word embeddings (2nd Model)**

****

**Training the Model and Predictions on the test data:**

****

****

**Methods used:**

During the course of our project, we endeavored to explore and implement three distinct machine learning frameworks, namely **Keras, PyTorch** and **Random Forest Classifier** in order to enhance the accuracy and optimize the performance of our models.

Below, we provide a description of the three methods employed:

**Keras:** This framework was utilized as one of our approaches in developing the models. We leveraged the functionalities and abstractions offered by Keras to facilitate the construction and training of our machine-learning models, aiming to achieve improved accuracy and performance.

**Our Model Interpretation and Analysis:**

Initially, we developed an initial base model for experimentation purposes. We incorporated various enhancements, including modifications to layers, learning rates, batch sizes, epochs, and other relevant parameters, with the aim of maximizing its accuracy. The model creation process involved several steps, such as data and text preprocessing, tokenization, padding, model creation, and training. Upon completion of these steps, we proceeded to generate visualizations for analysis and evaluation.

Subsequently, we introduced global embedding methods into our model, which led to the development of Model B. Notably, this approach yielded a significant improvement in accuracy, achieving a nearly 80% accuracy rate.

**GloVe:** To process the text into numerical data, we need word embedding, and in our case, we have used a technique called glove embedding. As our methodology, we used pre-trained word embedding from the Keras site and followed the instructions. We initially created a vocab index and vectorized text. Then we loaded pre-trained word embedding, prepared embedding matrices, and loaded those into the layers.

**PyTorch:** As part of our efforts, we also employed PyTorch, another prominent machine learning library, to explore alternative techniques and methodologies for enhancing our models. By harnessing the capabilities and flexibility provided by PyTorch, we aimed to further optimize the accuracy and performance of our models.

As a part of our experiment and process of improving accuracy, we created a base model for pytorch but the accuracy was not going higher than 0.73.

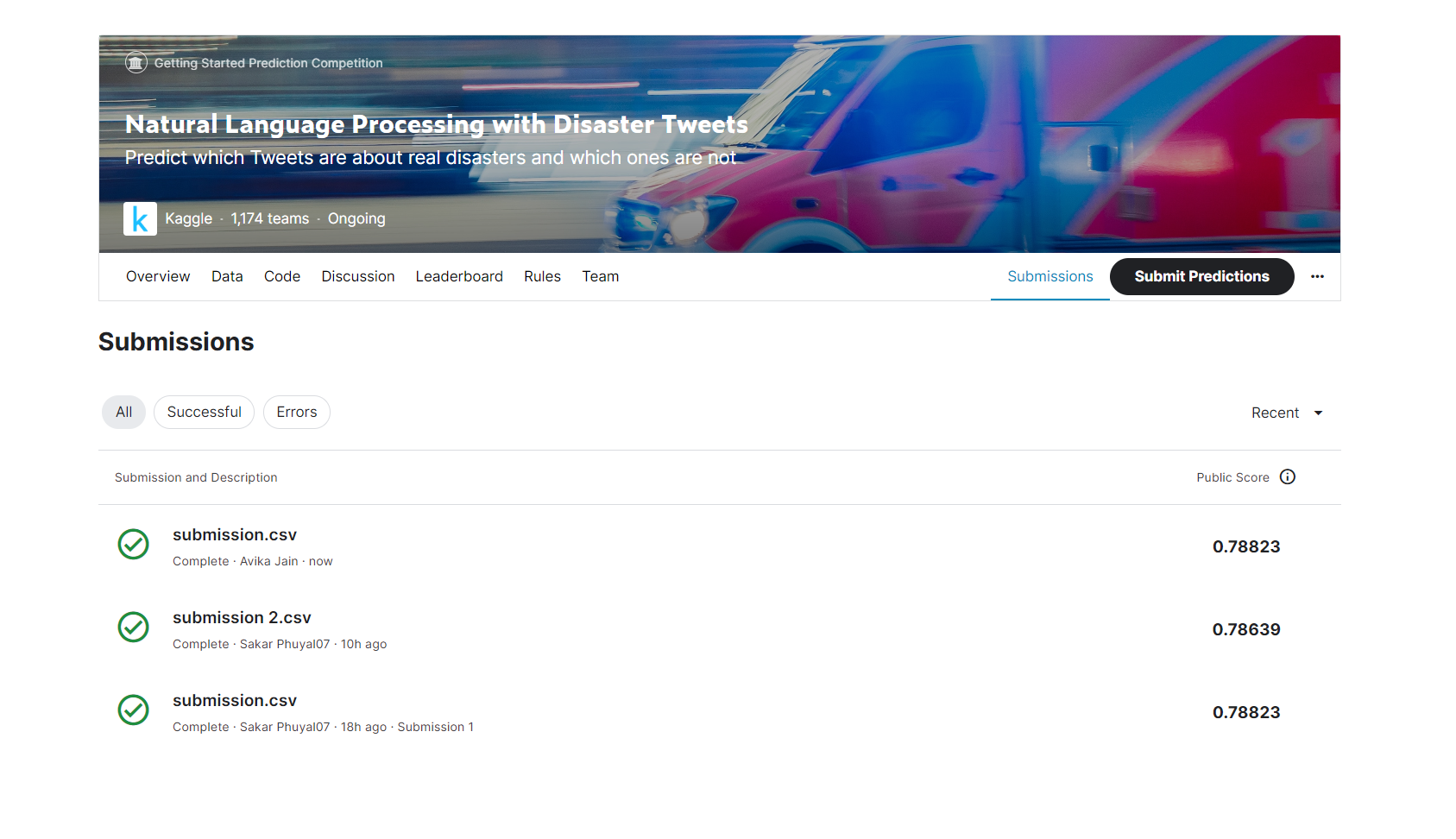
**Random Forest:**

For the previous two methods, we used Pytorch and Keras, and to further enhance and improve accuracy, we worked on Random Forest Classifier. Random Forest Classifier is an estimator that uses decision trees to improve predictive accuracy. Initially, the model's accuracy was around 74, and I was unable to improve significantly. It was just a slight improvement. So, we decided to stick with the Keras Model.

**Following extensive experimental analysis and interpretation, we focused on incorporating global embedding in our Keras model, which resulted in achieving the highest accuracy of 0.8 (refer to google colab file) . Consequently, we have decided to use this model as our final submission.**

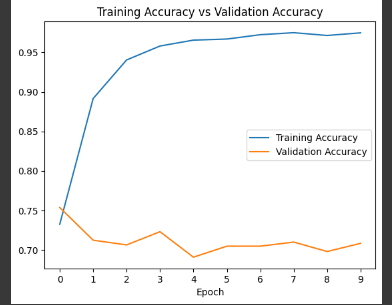
**Score from Kaggle.com:**

We made three submissions of our code to Kaggle, resulting in accuracies of 78.8% 78.6% and 78.8% respectively.



**Results and reports:**

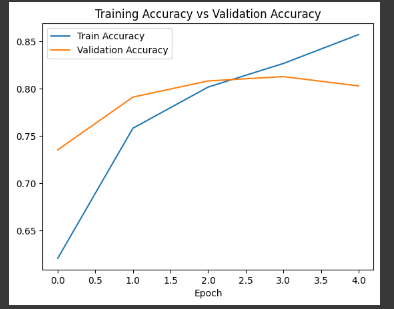
The initial model, also known as the base model, yielded an accuracy of 70.8 percent, showcasing a notable improvement compared to the original model's accuracy of 64 percent. To visualize the performance of the model, we generated a graph plotting the training accuracy against the validation accuracy, resulting in the following outcomes.



The validation accuracy for the initial epochs displayed the highest values; however, as the number of epochs increased, the slope of the validation accuracy graph gradually flattened and eventually declined.

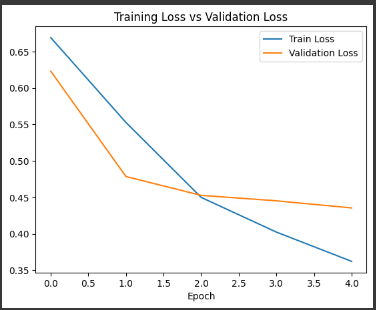
Consequently, we devised an alternative model, referred to as Model 2. By leveraging pre-trained word embeddings, we constructed a Keras model with a train-validation split of 80% and 20% respectively.

Upon executing the model in Google Colab, we achieved a validation accuracy of 80.2%, marking a significant improvement compared to Model 1. This enhancement amounted to a noteworthy 12% increase from the base case, which was deemed highly satisfactory by the entire team. Consequently, we deemed this model as the final selection.



We generated the graph displaying the relationship between Training accuracy and Validation accuracy. From the graph, it is evident that the validation accuracy initially rises and reaches a value of 0.8, after which it levels off, whereas the training accuracy continues to increase.

Additionally, we created another graph illustrating the correlation between Train loss and Validation loss, yielding the subsequent outcomes.



As it is clearly visible, we were able to minimize the validation loss upto the value of 0.43 which is again a decent number.

**References and Citations-**

*Keras. Using pre-trained word embeddings.* [*https://keras.io/examples/nlp/pretrained\_word\_embeddings/*](https://keras.io/examples/nlp/pretrained_word_embeddings/)

*Kaggle. Introduction to NLP with TensorFlow and spaCy.* [*https://www.kaggle.com/code/olemagnushiback/introduction-to-nlp-with-tensorflow-and-spacy?scriptVersionId=126979046*](https://www.kaggle.com/code/olemagnushiback/introduction-to-nlp-with-tensorflow-and-spacy?scriptVersionId=126979046)

*Chollet, F., et al. (2015). Keras: The Python deep learning library. Retrieved from* [*https://keras.io/*](https://keras.io/)

*Matthew Mayo. Tokenization and Text Data Preparation with TensorFlow & Keras. Retrieved from:* [*https://www.kdnuggets.com/2020/03/tensorflow-keras-tokenization-text-data-prep.html*](https://www.kdnuggets.com/2020/03/tensorflow-keras-tokenization-text-data-prep.html)

*Abadi, M., et al. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Retrieved from* [*https://www.tensorflow.org/*](https://www.tensorflow.org/)

*Bird, S., et al. (2009). Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit. Retrieved from* [*https://www.nltk.org/*](https://www.nltk.org/)

*Python Software Foundation. (n.d.). re - Regular expression operations. Retrieved from* [*https://docs.python.org/3/library/re.html*](https://docs.python.org/3/library/re.html)

*Matplotlib Development Team. (2021). Matplotlib: Visualization with Python. Retrieved from* [*https://matplotlib.org/*](https://matplotlib.org/)

*BeautifulSoup Documentation. (2021). BeautifulSoup Documentation. Retrieved from* [*https://www.crummy.com/software/BeautifulSoup/bs4/doc/*](https://www.crummy.com/software/BeautifulSoup/bs4/doc/)

*Count NaN Values in Pandas DataFrame. Retrieved from* [*https://sparkbyexamples.com/pandas/count-nan-values-in-pandas/*](https://sparkbyexamples.com/pandas/count-nan-values-in-pandas/)

*pandas.DataFrame.value\_counts. Retrieved from* [*https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.value\_counts.html#pandas.DataFrame.value\_counts*](https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.value_counts.html#pandas.DataFrame.value_counts)

*Vinamrayadav. How to Create a Pie Chart in Seaborn? Retrieved from* [*https://www.geeksforgeeks.org/how-to-create-a-pie-chart-in-seaborn/*](https://www.geeksforgeeks.org/how-to-create-a-pie-chart-in-seaborn/)

*Kartikaybhutani. Python | Pandas Series.str.len(). Retrieved from* [*https://www.geeksforgeeks.org/python-pandas-series-str-len/*](https://www.geeksforgeeks.org/python-pandas-series-str-len/)

*Matplotlib: Visualization with Python. Retrieved from* [*https://matplotlib.org/*](https://matplotlib.org/)

*Skrg141. How to Adjust Number of Ticks in Seaborn Plots?. Retrieved from* [*https://www.geeksforgeeks.org/how-to-adjust-number-of-ticks-in-seaborn-plots/*](https://www.geeksforgeeks.org/how-to-adjust-number-of-ticks-in-seaborn-plots/)

*Fchollet. (2020). Using pre-trained word embeddings. Retrieved from* [*https://keras.io/examples/nlp/pretrained\_word\_embeddings/*](https://keras.io/examples/nlp/pretrained_word_embeddings/)