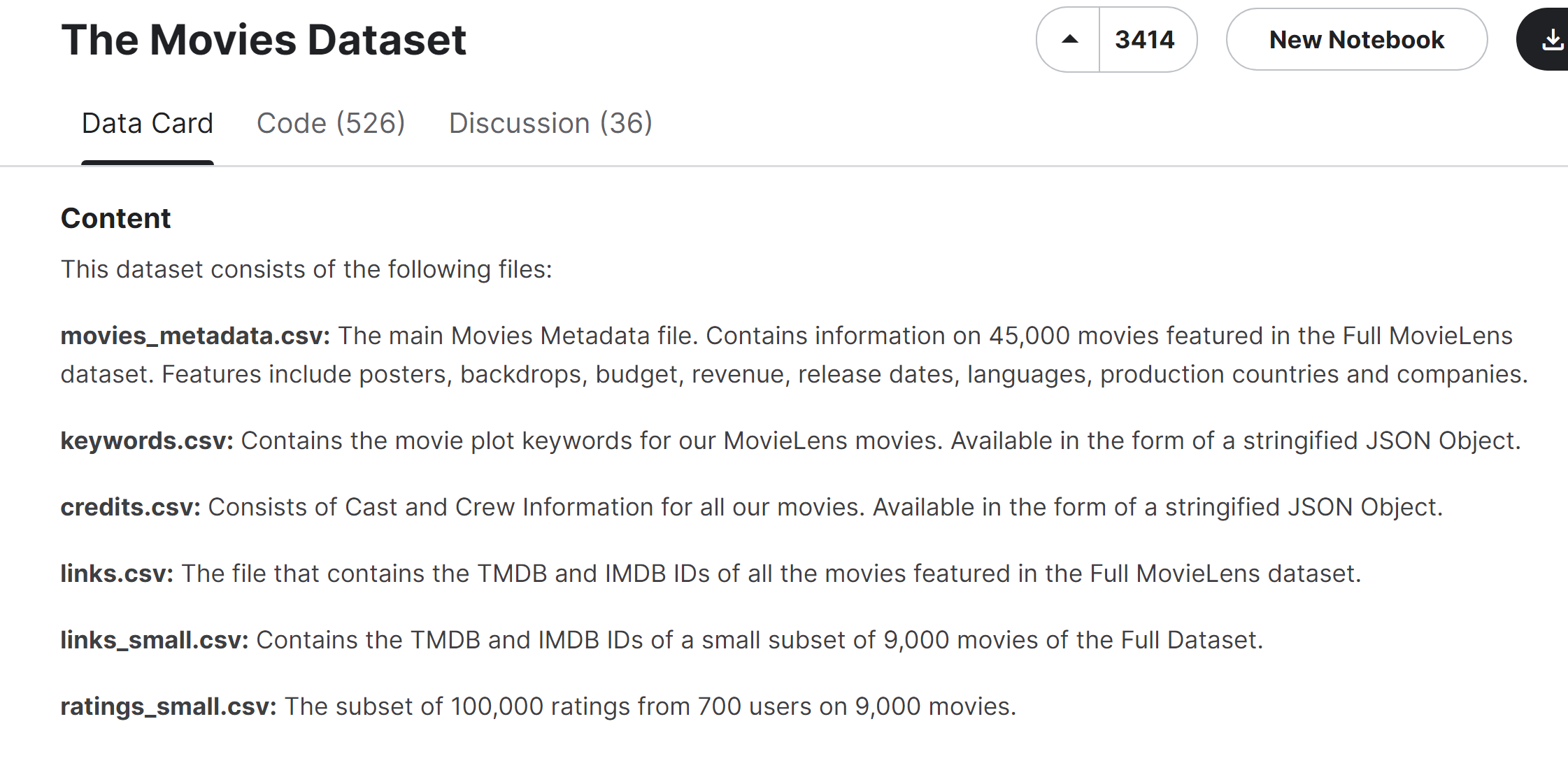
# Problem Description

I am a movie enthusiast with a particular interest in sci-fi, fantasy, thriller, horror, and mystery genres. However, I am selective in my movie choices as not all genres are equally appealing to me. I am also a workaholic and value my time. Therefore, I find it challenging to allocate 2-3 hours of my schedule to watch a movie that does not captivate me. Additionally, the quality of movies has increased significantly due to the rise of OTT platforms, and as a result, many movies fail to meet my expectations. This is especially true in the post-COVID era where we have become accustomed to high-quality content.

Despite extensive research, I was unable to find a suitable dataset for my project. I explored several popular datasets such as [Kaggle](https://www.kaggle.com/datasets/rounakbanik/the-movies-dataset), [MovieLens](https://grouplens.org/datasets/movielens/latest/), [HydraMovies](https://data.world/iliketurtles/movie-dataset), and [IMDB](https://developer.imdb.com/non-commercial-datasets/). While some of these datasets offered attributes like tag\_line, full\_summary, and plot\_keywords, none of them contained detailed storylines. Additionally, many of these datasets had missing data for fields like budget and revenue, which are crucial for making an automated guess on whether the movie is good or bad.



The Movies Dataset contains plot keyword information, but not the plot or the storyline details (Source: [The Movies Dataset (kaggle.com)](https://www.kaggle.com/datasets/rounakbanik/the-movies-dataset))

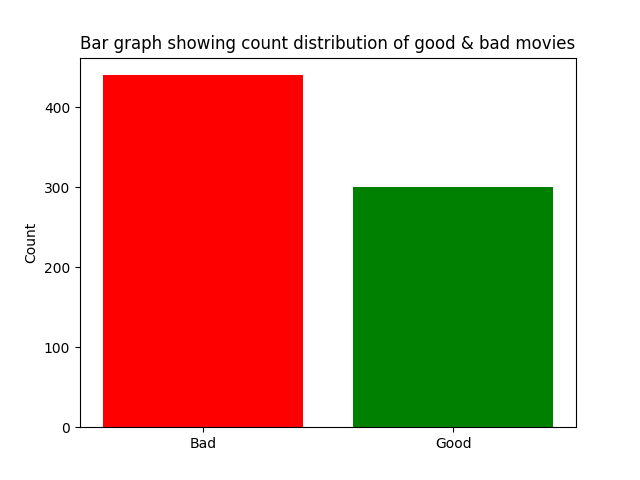
Therefore, I decided to build an application to classify movies as good and bad using storyline as the data. This is different than previous approaches as it considers the plot details instead of keywords or brief summary. The movie story plays a more important role in determining the desirability of a movie than other features like plot keywords and genre which are derived. I have also tested by including budget as a feature.

# Data Collection

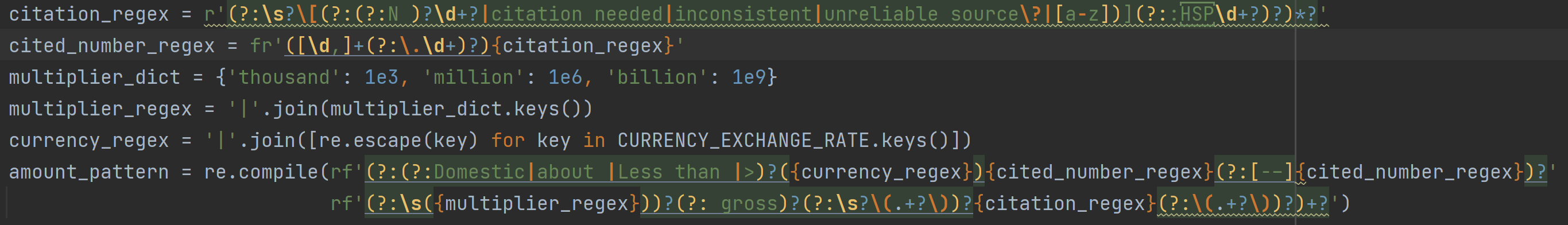
Because of the problems in the aforementioned datasets, I created a scraping script to get high-quality storyline, budget and box-office collection data from Wikipedia. As the Wikipedia movie article HTML code has no well-defined format and is intricate, the development of the scraper was the most time consuming part of the project. The scraping script gets the list of movies to scrape from a movie metadata file downloaded from [The Movies Dataset (kaggle.com)](https://www.kaggle.com/datasets/rounakbanik/the-movies-dataset?select=movies_metadata.csv). This file is also used by the scraper when it fails to determine a valid Wikipedia URL for the movie or if it fails to extract the features from the retrieved Wikipedia page. In these situations, it uses the attributes of the same name from this source metadata file as fallback values.

The scraper uses the budget and revenue fields to calculate the labels. If either of the two values are missing, it labels the movie as bad. This is partly true because these movies tend to be the ones made on a small-scale and intended for very niche category of audiences. Therefore, there is a high chance that the movie wouldn’t be appealing to me and the general audience. In case both the values are present, it will label the movie as good only in case the following holds-

In other words, a movie is labelled good if it has more than 25% profit margin. Below is the chart showing the distribution of good and bad movies-

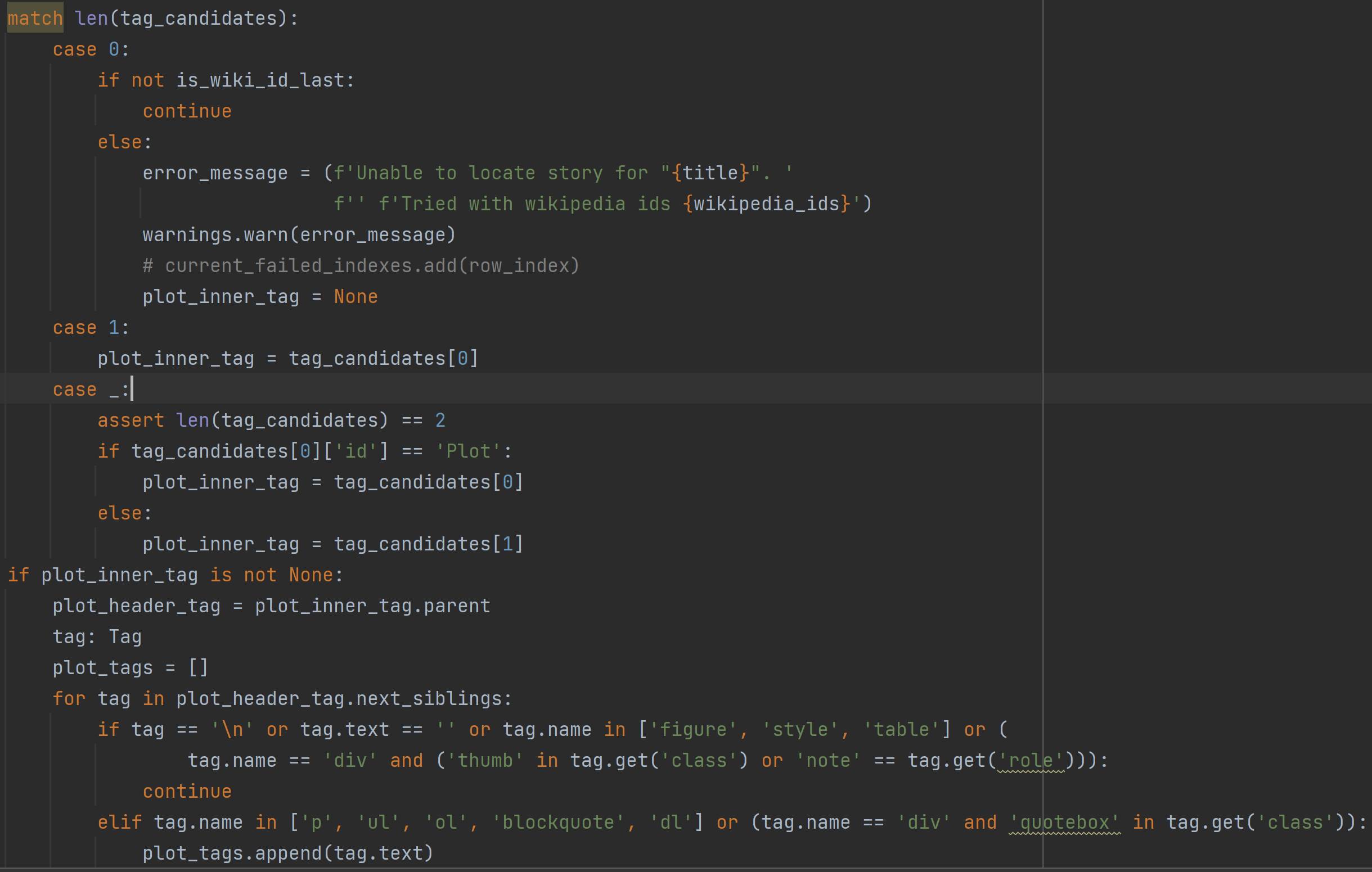


Budget and revenue data were missing for 53.13% and 52.05% records, respectively, in the source metadata file. These number were brought down to 30.54% and 16.18%, respectively, in the scraped data file. The following complex piece of code made it possible-



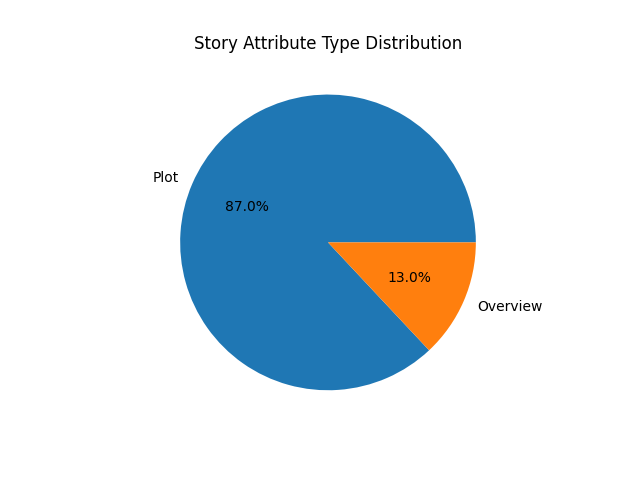
The Wikipedia amount pattern required for parsing budget and revenue is so dynamic and complex that it itself requires dynamic runtime generation

It was also not straightforward to get the plot attribute because the entire page code in HTML is flat and not hierarchical & sectioned like the page display. The scrapper uses complex logical conditions to determine the start and the end tag containing the plot. This is complicated further by the HTML code of other miscellaneous sections that might occur in between.

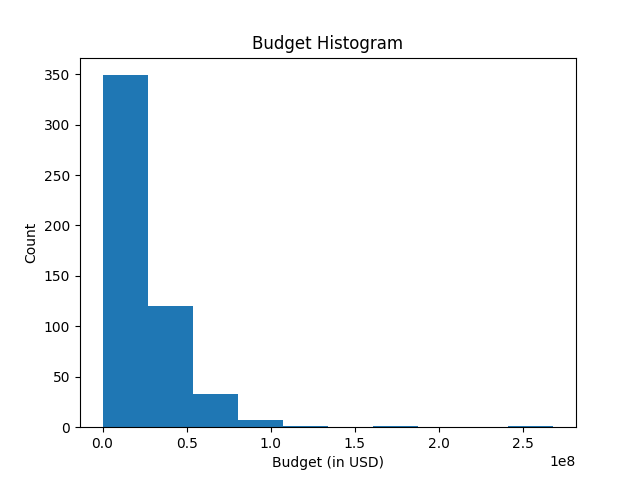


A part of the section of code that parses the plot

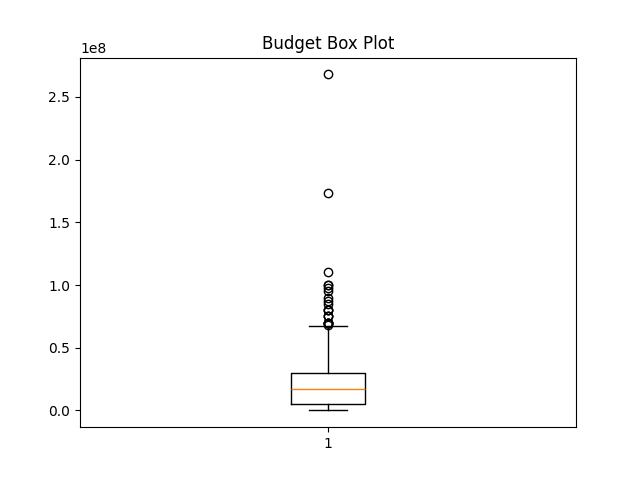
If the script fails to capture the plot from the page, or if no valid Wikipedia URL is found, the scraper uses the ‘overview’ attribute from the source metadata file as the value of the story attribute. The below pie chart illustrates how frequently this has occurred-



The above pie chart shows the percentage distribution for the type of data contained in the story attribute

I had also trained with budget as an additional feature as low-budget films tend to not be appealing. Below are the charts showing the data distribution of movies based on their budget.

The X-axis shows the budget in multiples of 100,000,000 USD. The Y-axis shows the count of movies occurring in a budget range. Budget ranges span 50,000,000 USD and start from 0.



The above budget histogram and box & whisker plot indicate that films with budgets somewhere above 75,000,000 are outliers. However, I cannot remove the high-budget films as they are the main points of interest. The ideal approach here would be to get more training data and then filter out the low-budget films. Since not all low-budget movies are bad, a better indicator would be whether the film was released internationally.

The budget charts give no indications of the outliers which are on the other side of the value spectrum i.e. budget values that are too small.

# The Models

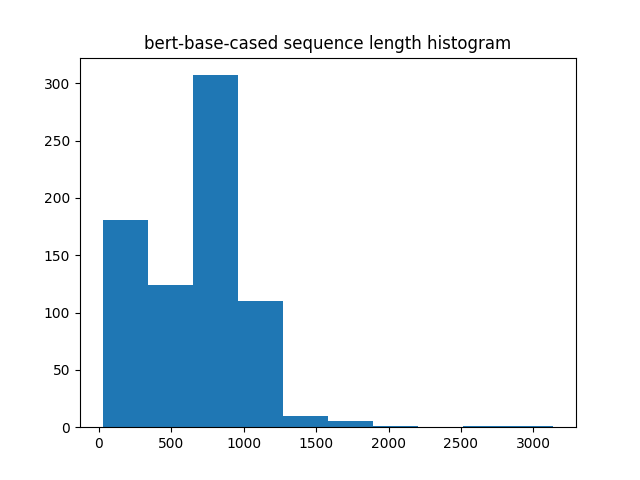
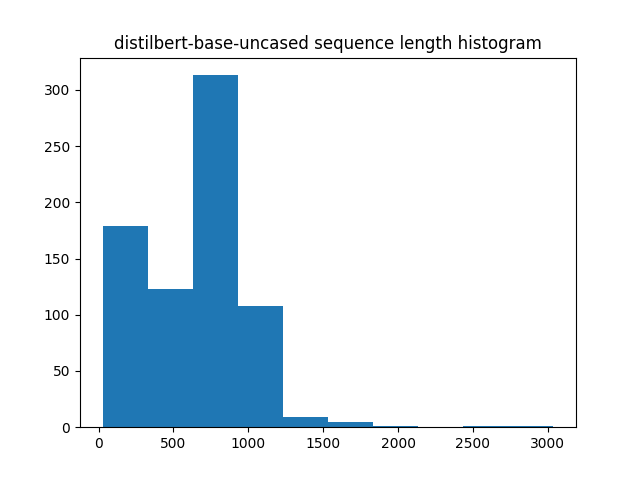
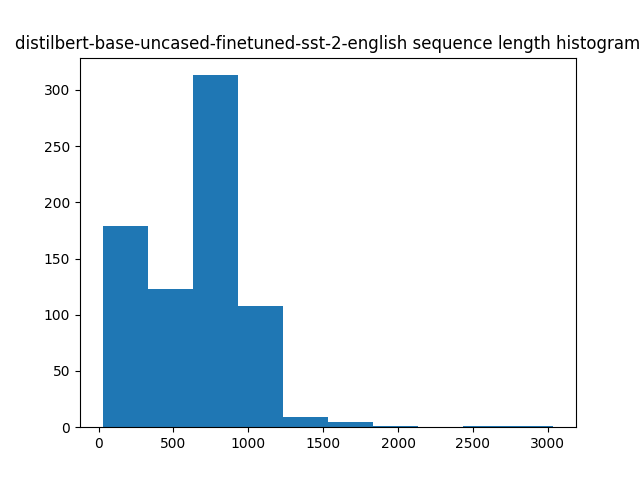
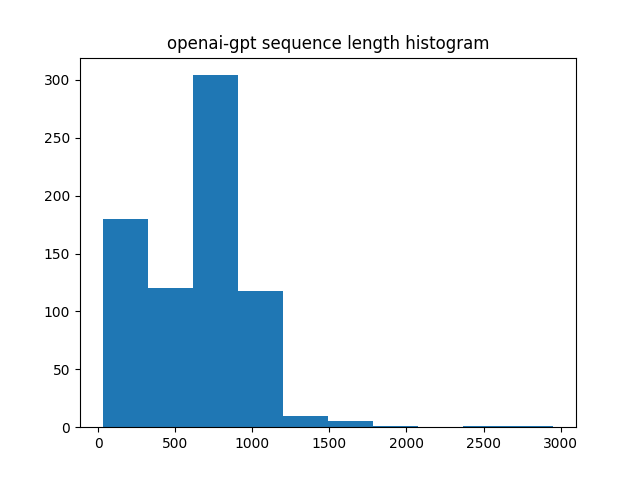
To get the best results I have used the following transformer based pretrained models-

1. OpenAI’s GPT1- GPT1 stands for Generative Pre-trained Transformer. It is a natural language processing (NLP) model developed by OpenAI in 2018 which has 117 million parameters. It was trained on a combination of two datasets: the Common Crawl, a massive dataset of web pages with billions of words, and the BookCorpus dataset, a collection of over 11,000 books on a variety of genres. I was able to train it but could not run the evaluation script on it due heavy memory requirements.
2. Google’s BERT BASE cased- BERT stands for Bidirectional Encoder Representations from Transformers. It is a deep learning language model that was introduced by researchers at Google in 2018. The model has 110M parameters in the BASE version. It was trained on a massive dataset that includes the Common Crawl, a dataset of web pages with billions of words, and the BookCorpus dataset, a collection of over 11,000 books on a variety of genres.
3. HuggingFace’s DistilBERT BASE uncased- DistilBERT base model is a smaller and faster version of the BERT base model. It was introduced by Hugging Face in 2019. DistilBERT was trained on a combination of two datasets: the Common Crawl, a massive dataset of web pages with billions of words, and the Wikipedia dataset, a collection of articles from Wikipedia. It has 67M parameters.
4. HuggingFace’s DistilBERT BASE uncased finetuned on SS2- This is the best performing model. SST-2 is a dataset for sentiment analysis that consists of 11,855 single sentences extracted from movie reviews.

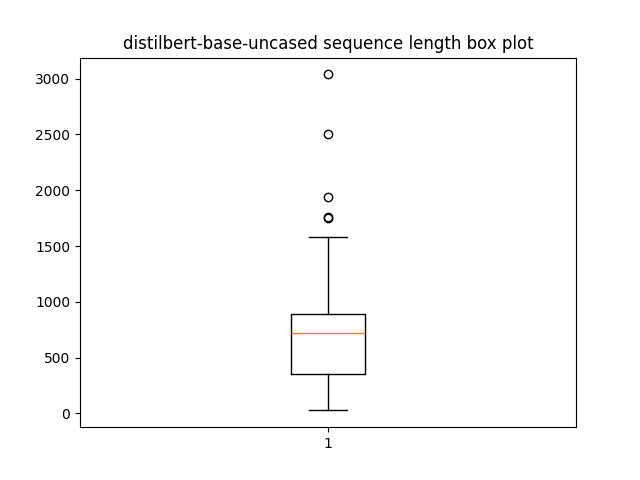
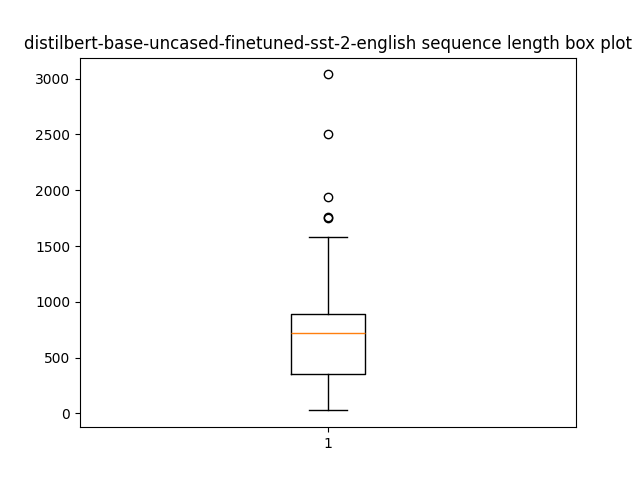
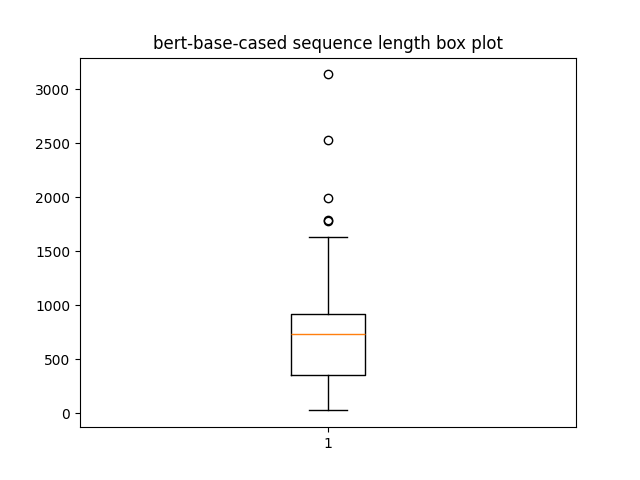
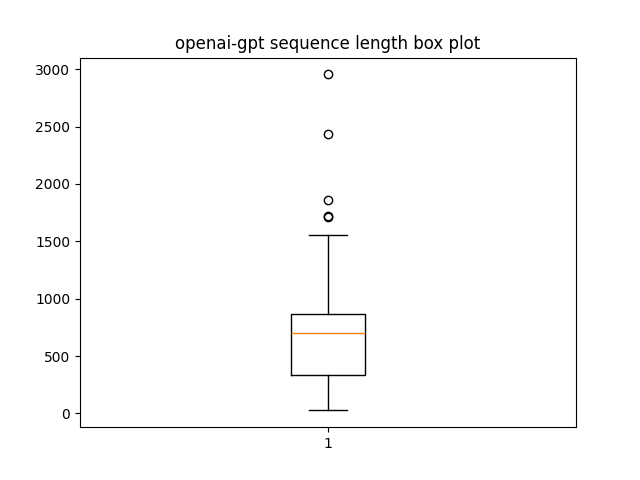
In the above ‘cased’ or ‘uncased’ represents whether the model is case sensitive or not.

Finetuning pre-trained model is the de-facto approach for NLP tasks. However, the problem with these models is the quadratic performance cost inherent in the attention layers of the transformer architecture. Therefore, these models assume a very conservative limit on the size of their input. The input is sequence of integers representing tokens in the sentence. Tokens are smallest parts of a sentence that can mean something. For all models, 512 is the upper bound for the input token count. I had tried using the smallest variant of GPT-2 for its higher upper bound of 1024, however it failed to run with an out-of-memory error.

Below are charts for the token length distribution for the story attribute. Note that the tokenization process varies from model to model. But in this case they are almost same.

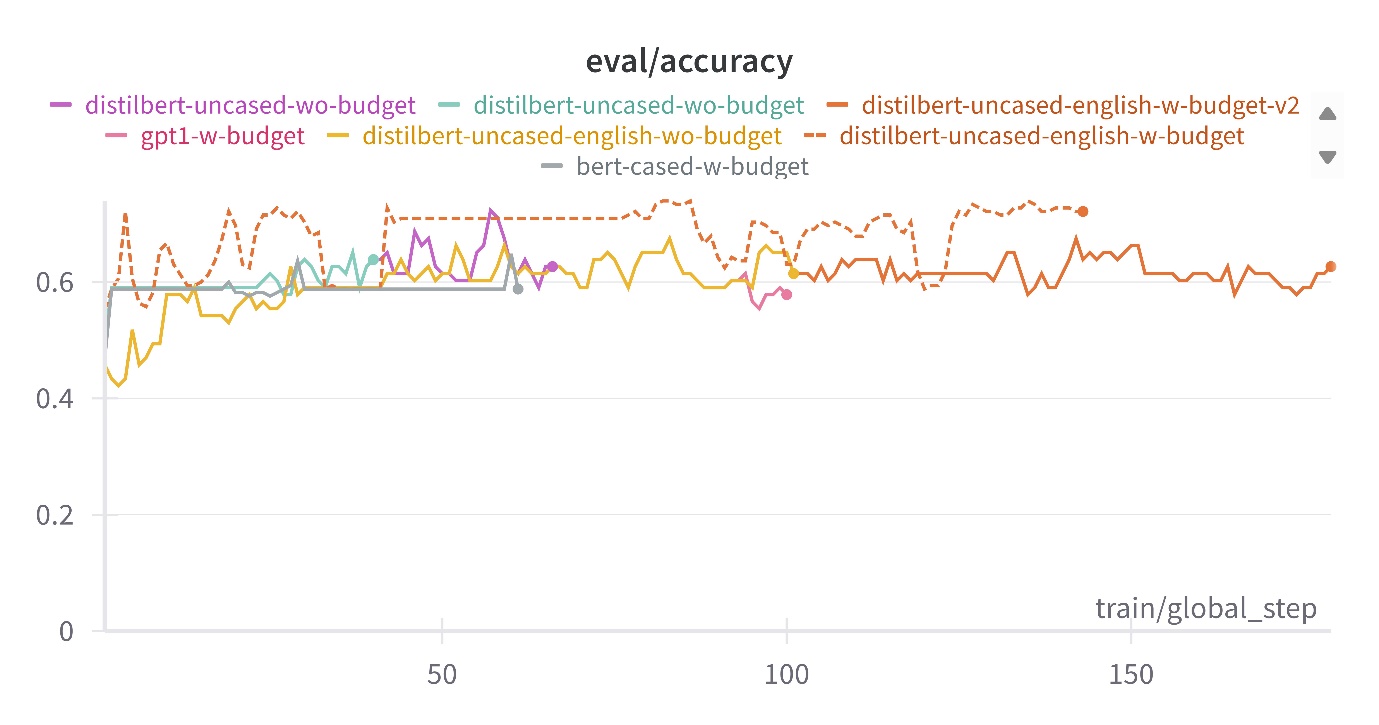
 

Even though these appear to be the same they vary slightly. Notice that for the first two DistillBERT models the longest bar is longer than other models. The 1000-1250 bar of GPT is the longest.

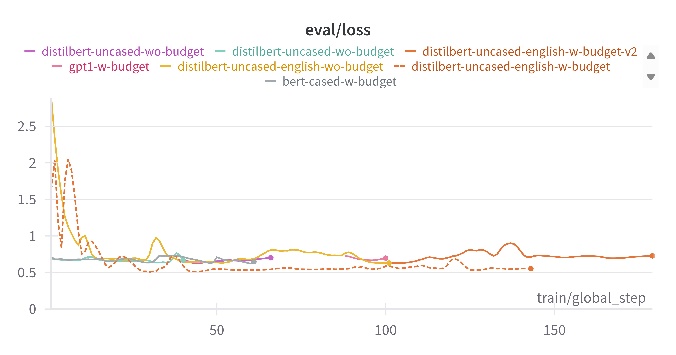
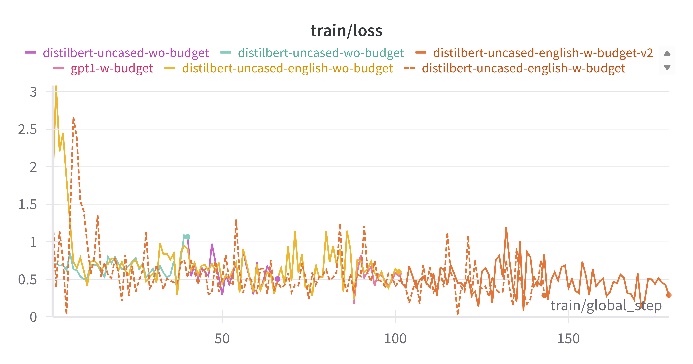
  

# Training

The training script resumes fine-tuning the models from the last saved checkpoint if any. It crashed frequently due to wandb side cache memory leak issue. WandB is the service that I am using for creating my training process charts.



Validation Accuracy: DistillBERT SST-2 finetuned with budget as additional feature performed the best. BERT with budget showed no improvement. Maybe it is because it is very hard to train. Similar start was seen from DistillBERT without budget which started picking up pace, however, it kept crashing due to memory issues so I couldn’t train much. DistillBERT SST-2 finetuned w/wo budget was very fast. This allowed it to complete 100-150 steps in a single run.

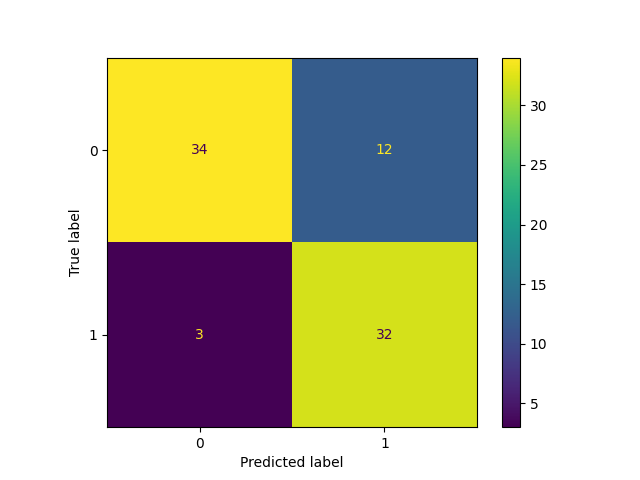
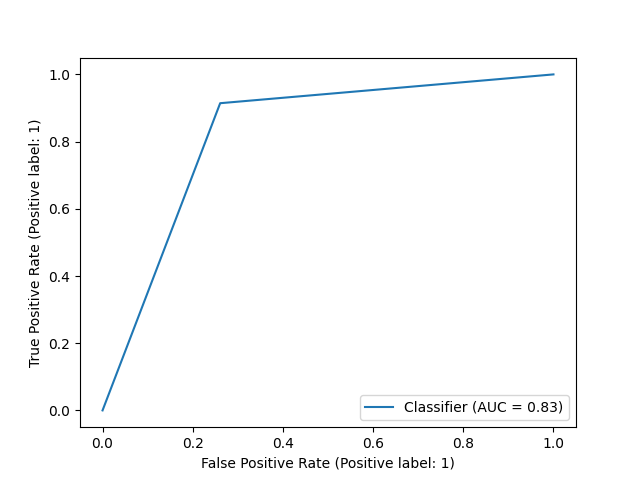


Learning curve. As expected, the validation loss is slightly higher. Although gradually, the training loss is improving. However, validation loss seemed to hit a plateau. More training would be required to determine whether the model starts to overfit. The aforementioned trends seem to be applicable to all models.

# Experimental Results

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Model** | **AUC** | **Accuracy** | **Precision** | **Recall** | **F1 score** |
| BERT(Budget) | 0.57 | 0.62 | 0.67 | 0.23 | 0.34 |
| DistillBERT | 0.63 | 0.58 | 0.51 | 0.97 | 0.67 |
| DistillBERT SST-2 | 0.69 | 0.69 | 0.63 | 0.68 | 0.66 |
| DistillBERT SST-2 (Budget) | 0.83 | 0.81 | 0.73 | 0.91 | 0.81 |

Given below are the ROC curve and confusion matrix for DistillBERT uncased SST-2 finetuned, the best performing model, all done on unseen test data. Metrics and charts for other models are not included here for brevity. They can be found in the evaluation\_results directory in the project, or can be generated using test\_predictor.py script.



# Missclassification Analysis for DistillBERT SST-2 (Budget)

The Shadow (1994 Film)- This was misclassified as hit with strong confidence by the model (78.7). It was also purported to be a hit, however, it [failed at the box office](https://en.wikipedia.org/wiki/The_Shadow_(1994_film)) most probably due to competition.

The Pharaoh’s Army- Misclassified as a hit. This was labelled a flop because revenue data was unavailable. However, this has got an IMDB rating of 6.6/10 and 63% users liked it on Google reviews.

Dead Man Walking- Misclassified as flop. This filmed earned more than 7 times its budget and has strong rating from critic sites. This got misclassified most probably because the model was pretrained for sentiment analysis and [plotline](https://en.wikipedia.org/wiki/Dead_Man_Walking_(film)) is very serious one. Also as per [Wikipedia](https://en.wikipedia.org/wiki/Dead_Man_Walking_(film)), the film’s success is also attributed to excellent performances by the cast.