1.Introduction

In the ever-evolving landscape of artificial intelligence, the relentless advancement of AI technologies has captured global attention. With each passing day, distinguishing between human-generated content and that produced by AI becomes an increasingly intricate task. Text creation stands out as a prime example.

In response to this challenge, we endeavored to equip our model to navigate and address the nuances of discerning a Large Language Model (LLM) from a text crafted by a human. This report delves into our strategic approach to meet this challenge head-on, shedding light on the measures taken to provide a reliable framework for identifying AI-generated content consistently.

Leveraging various libraries such as TensorFlow, pandas, and transformers, our code encompasses data preprocessing, exploratory data analysis, and the training of a recurrent neural network (RNN) model. The model is designed to learn patterns distinguishing between human-written and machine-generated text.

Throughout the code, various visualizations, such as bar plots and histograms, are employed to provide a comprehensive understanding of the dataset's characteristics. In conclusion we evaluated the model's accuracy and loss, along with the generation of submission files for further evaluation.

2.Detect AI Generated Text

a. Dataset Description

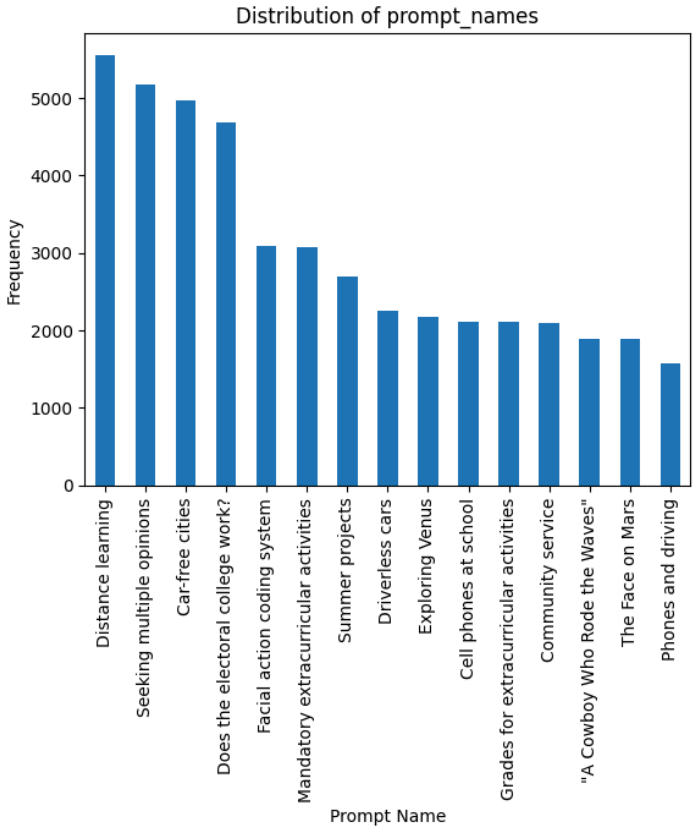
In order to train our model, we employed the "DAIGT V2 Train Dataset". This dataset encompasses 44,867 unique entries.

Note: After analyzing the data we added a collection of essays generated from a GPT-2 pretrained model that we will discuss in the next paragraph.

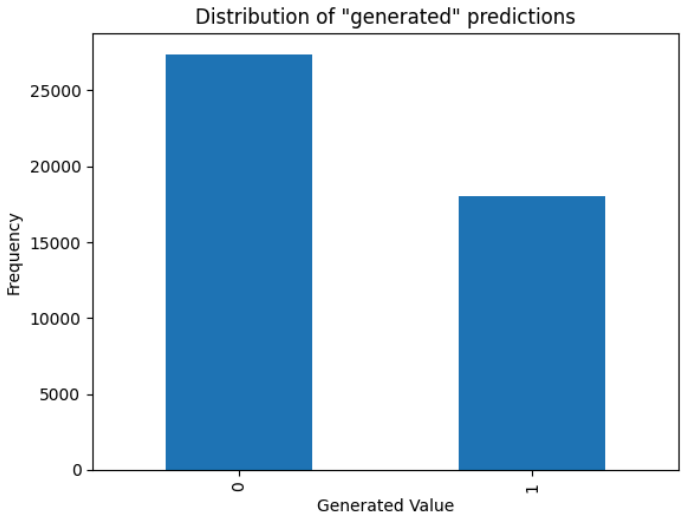
	text	generated	prompt_name	source	RDizzl3_seven	id
0	Phones\n\nModern humans today are always on th...	0	Phones and driving	persuade_corpus	False	khlwbu66
1	This essay will explain if drivers should or s...	0	Phones and driving	persuade_corpus	False	zj2ua25q
2	Driving while the use of cellular devices\n\nT...	0	Phones and driving	persuade_corpus	False	mu94agmf
3	Phones & Driving\n\nDrivers should not be able...	0	Phones and driving	persuade_corpus	False	j2rce830
4	Cell Phone Operation While Driving\n\nThe abil...	0	Phones and driving	persuade_corpus	False	ynbe4k9v
...	...	...	...	...	...	...
44863	Dear Senator,\n\nI am writing to you today to ...	1	Does the electoral college work?	kingki19_palm	True	fvb2rfri
44864	Dear Senator,\n\nI am writing to you today to ...	1	Does the electoral college work?	kingki19_palm	True	uggyhduo
44865	Dear Senator,\n\nI am writing to you today to ...	1	Does the electoral college work?	kingki19_palm	True	s017hzcp
44866	Dear Senator,\n\nI am writing to you today to ...	1	Does the electoral college work?	kingki19_palm	True	il1b5rf4
44867	Dear Senator,\n\nI am writing to you today to ...	1	Does the electoral college work?	kingki19_palm	True	80alw5xl

Each entry consists of various informative columns:

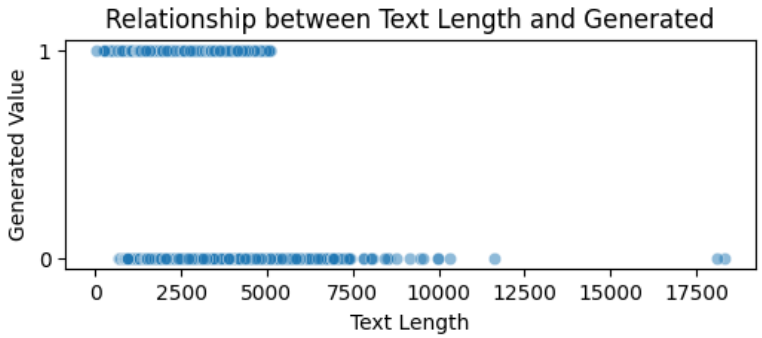
- The Text Column contains the actual essay text, serving as the primary input for our model.
- In the Generated Column, values of 0 represent human-generated texts, while values of 1 denote AI-generated texts.
- The Prompt Name Column identifies the original topic or prompt associated with the text generation.
- The Source Column indicates the authorship or source of the respective text.
- The final column, labeled "RDizzl3_seven," informs us whether the current essay prompt is derived from the original seven prompts specified in the competition.



A notable observation is the extensive variety of prompt names within the dataset, revealing a diverse spectrum. Interestingly, among these names, four distinct values exhibit a higher prevalence compared to others, underscoring their notable frequency in the dataset.



The graph indicates a notable inclination towards text created by human authors, highlighting a clear bias in favor of human-generated content over machine-generated text.



This graph illustrates the correlation between the length of texts and their respective generation types. Specifically, it reveals that machine-generated texts generally adhere to a length spectrum spanning from 0 to 5000 characters. In contrast, texts crafted by human authors exhibit instances where this length range is surpassed.

b. Methods

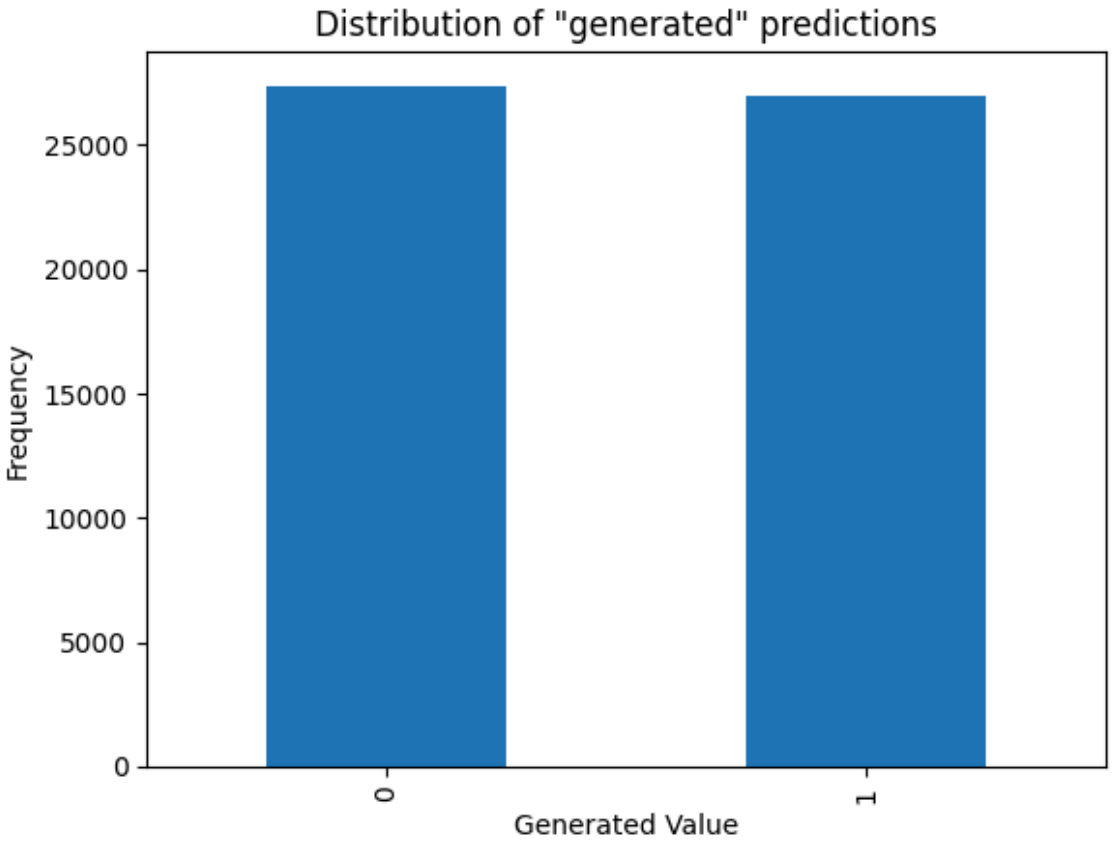
i. Preprocessing

Preprocessing textual data is a crucial element in preparing data for the training of machine learning models. In the context of this report, several stages were adopted to organize and transform training and testing data using the TensorFlow library. Initially, the training data was loaded from 1 CSV file using the pandas library, after that we renamed the column “label” to “generated” in order to make the dataset similar to the original one of the competition.

Due to the fact that human-written texts were more than the AI generated ones, as we said before, we choose to use a model called GPT-2 to generate more or less 9000 more texts to add in the training set.

We used GPT2LMHeadModel and GPT2Tokenizer using the Transformers library of Hugging Face for the language model and to preprocess the input. Then, in order to generate text, the model tokenizes a given prompt, generates text based on the tokenized input using a GPT-2 model utilizing an attention mask and enabling sampling, and in the end decodes the generated token sequence back into readable text.

After the generation, the new texts are saved into a csv file that we could reuse without repeating the entire process.



Using this process we can see now that the dataset is balanced.

After that, to optimize the training process, the data was then divided into fixed-size batches (**128**) and shuffled with a buffer of size **15000**, ensuring greater variability during learning.

Subsequent to our initial steps, a pivotal phase in our process involved the creation of a TensorFlow TextVectorization object, configured to accommodate a maximum of **1000** tokens. This object was seamlessly aligned with our training data through the application of a mapping function, primarily involving the 'text' attribute. It is crucial to highlight that the implementation of text vectorization serves as a fundamental bridge, translating textual information into a numerical representation that is critical for the purposes of training machine learning models.

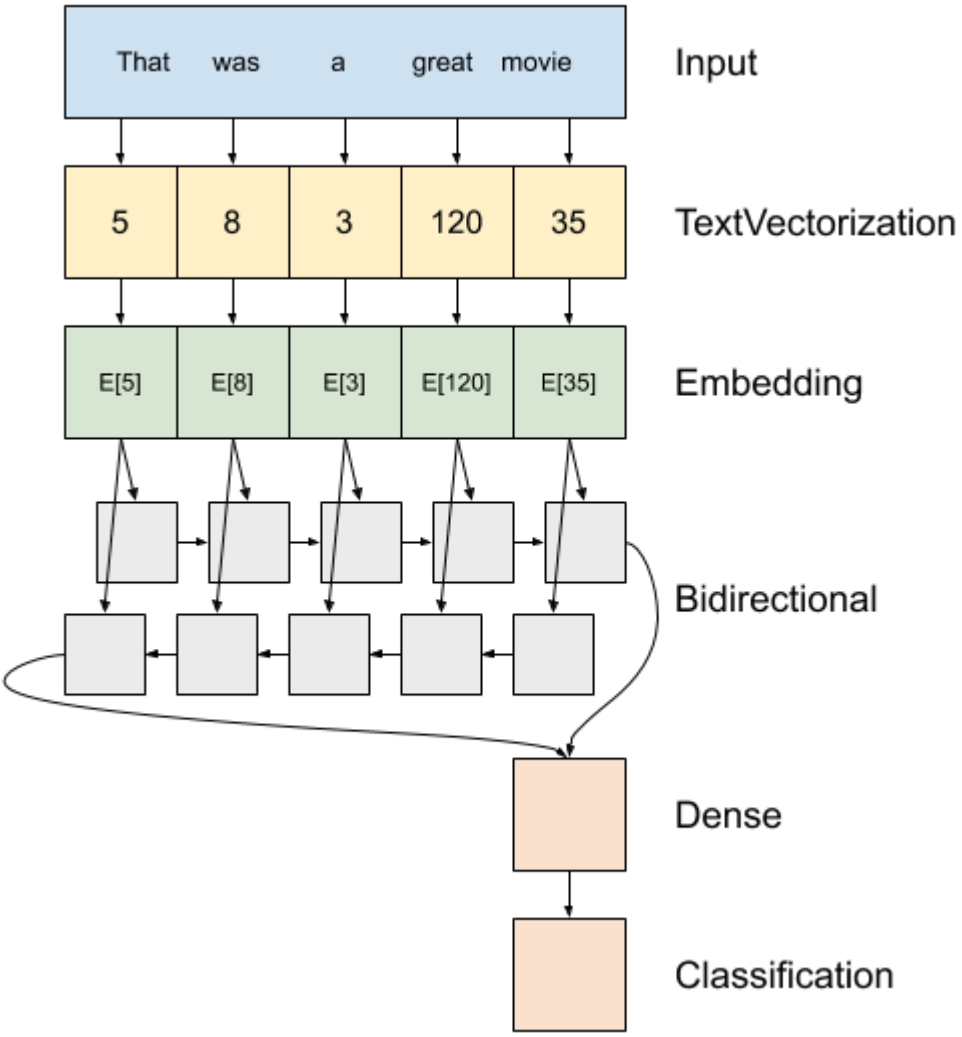
Expanding on this crucial step, the TextVectorization object operates as a dynamic mechanism, allowing us to effectively capture the underlying patterns and structures present in the training data. Through its mapping function, this object not only transforms raw text into a format digestible by machine learning algorithms but also facilitates the extraction of intricate linguistic features.

We have also incorporated an embedding layer into our model architecture. This layer, configured with a vocabulary size of **1000** words and an embedding size of **16**, plays a crucial role in our quest to encapsulate semantic relationships between words. By embedding text in a continuous vector space, we aim to improve the model's ability to understand complex linguistic nuances, thus strengthening its learning capabilities.

In essence, this strategic incorporation of an embedding layer serves as a tool to foster a better understanding of the semantic context within the text, thus elevating the overall performance and adaptability of our machine learning model.

ii. Architectures

At the core of our model architecture is a recurrent neural network, which is particularly effective in this type of problem. Unlike traditional feedforward neural networks, where information flows in one direction only, RNNs have connections that form a directed cycle, allowing them to capture temporal dependencies and patterns in sequences. In particular, each step in the sequence involves processing the current input along with information from the previous step, effectively creating a form of memory. We structured the model as a linear stack of layers, each contributing to the intricate process of extracting meaningful information from textual input.



The fundamental layer in our model is a configured instance of the TextVectorization layer, which acts as an encoder. This layer, having been pre-adapted to the training data, acts as a gateway to convert the textual input into a numeric format, thus laying the foundation for subsequent processing.

Following the crucial role of the encoder, we integrate an embedding layer into the model architecture. This layer is responsible for mapping the encoded text onto a continuous vector space, giving it a dimensionality of **16**. Specifically, this embedding layer is configured to mask zero values, effectively handling sequences of variable length.

Venturing further into the model's intricacies, **5** bidirectional Long Short-Term Memory (LSTM) layers are stacked. An LSTM Layer is a specialized type of recurrent neural network (RNN) layer, which enables the model to capture and retain long-term dependencies in sequential data through its gated structure. The bidirectional nature of these LSTM layers empowers the model to capture dependencies within input

sequences in both forward and backward directions, providing a holistic understanding of the sequential patterns at play.

Each LSTM layer comprises **8** neurons, ensuring a nuanced representation of the sequential data.

To delve even deeper into the nuances of complex relationships, a densely connected layer with **16** units and a Rectified Linear Unit (ReLU) activation function is seamlessly incorporated. As a precautionary measure against overfitting during training, a dropout layer with a rate of **0.5** is introduced, enhancing the model's generalization capabilities.

The model's final touch involves a dense layer with a single unit and a sigmoid activation function, designed explicitly for binary classification. This layer produces an output probability, offering insights into the likelihood of the input belonging to the positive class.

In terms of compilation, the model adopts binary cross-entropy as the loss function, the Adam optimizer with a learning rate of 1e-4, and accuracy as the chosen evaluation metric. This meticulously crafted architecture is purpose-built for tasks such as sentiment analysis or binary classification of sequential text data, exemplifying its adaptability and efficacy in extracting meaningful information from sequential textual inputs.

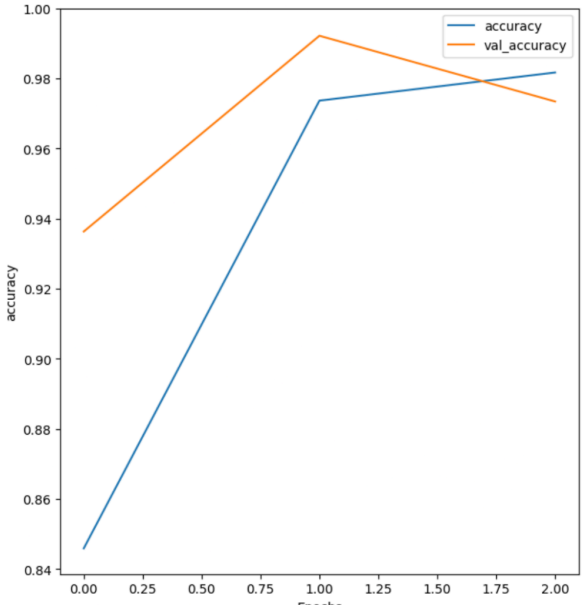
iii. Training and Experiments

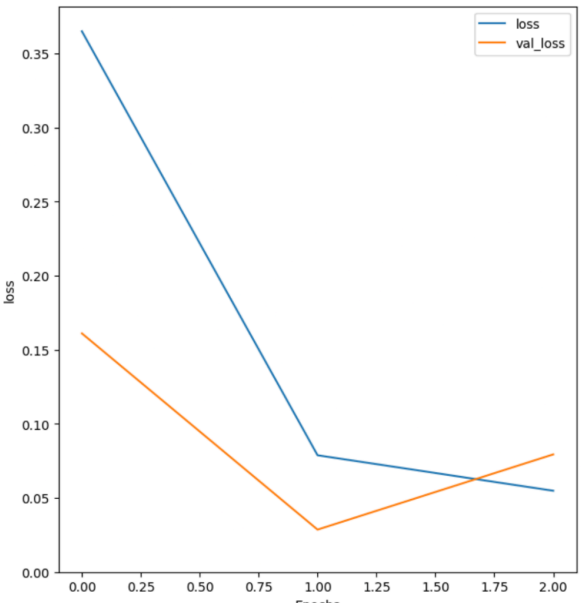
To train the model we have established 80% of the dataset as a training set. As a first attempt with a TextVectorization layer (encoder), followed by an Embedding layer that maps the encoded text into a 64-dimensional continuous vector space while effectively handling variable-length sequences with zero-value masking.

Subsequently, we stacked two Bidirectional Long Short-Term Memory (LSTM) to capture intricate sequential patterns. The first LSTM layer of 64 neurons returns sequences, providing a richer representation of dependencies, while the second LSTM layer composed of 32 neurons condenses this information into a final representation.

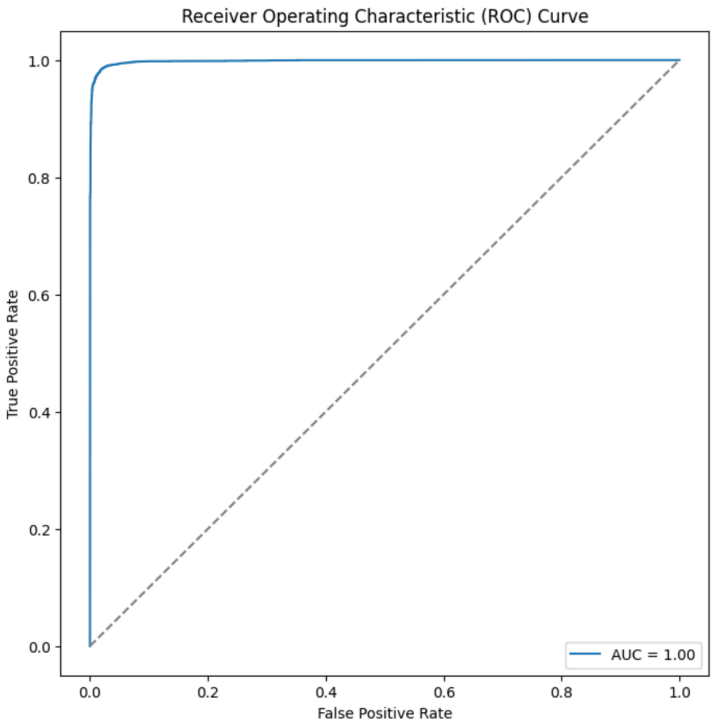
A Dense layer with 64 units and a Rectified Linear Unit (ReLU) activation function is introduced to capture complex relationships. The architecture concludes with a Dense layer with a single unit and a Sigmoid activation function. It utilizes the binary cross-entropy loss function, which is well-suited for binary classification tasks. Then we chose Adam as optimizer, with a learning rate of 1e-4, providing efficient updates to the model weights during training, while the evaluation metric employed during training is accuracy. We trained the model over 3 epochs, with a number of validation steps of 20. We got relatively high scores of accuracy, starting from a value of 0.84 in the first epoch to 0.98 of the last epoch. However, in the competition, our model

achieved a score of 0.828, highlighting the opportunity to enhance its generalization capabilities.

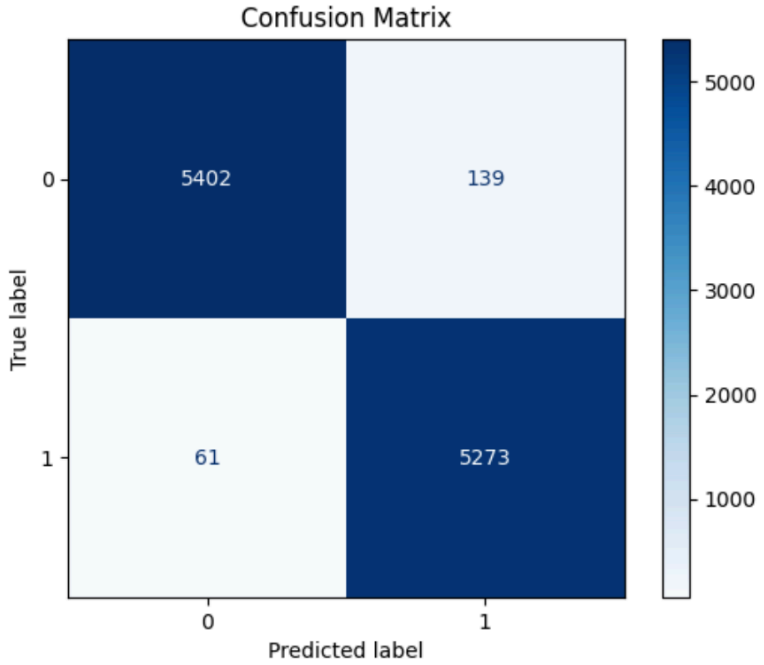




Our initial training results indicated a performance drop in the validation set compared to the training set, which raised suspicions of overfitting. Overfitting hampers a model's ability to generalize, impacting its performance on new, unseen data.

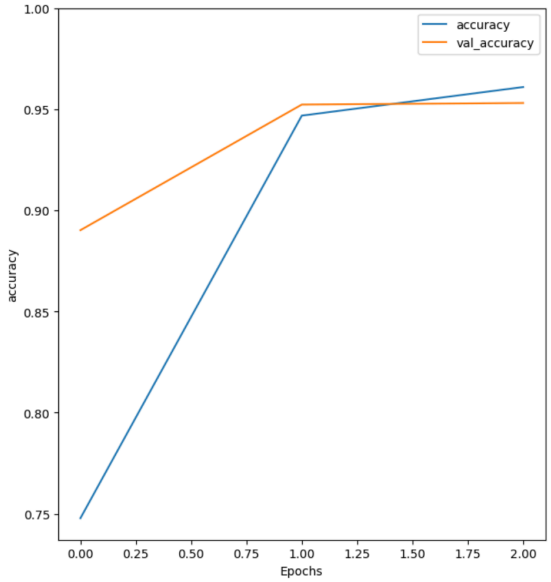


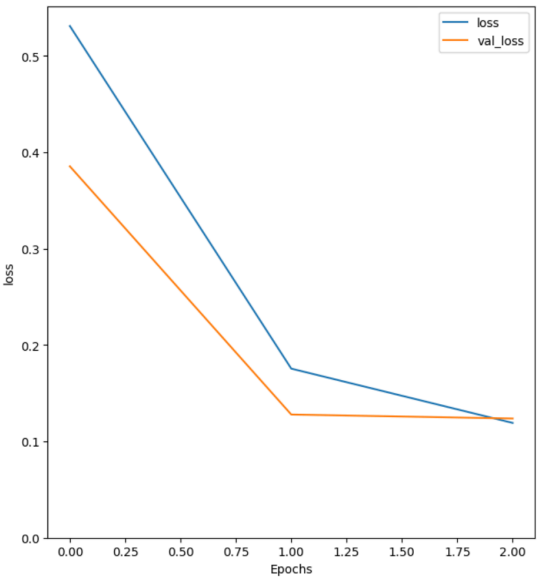
The model demonstrates strong performance.

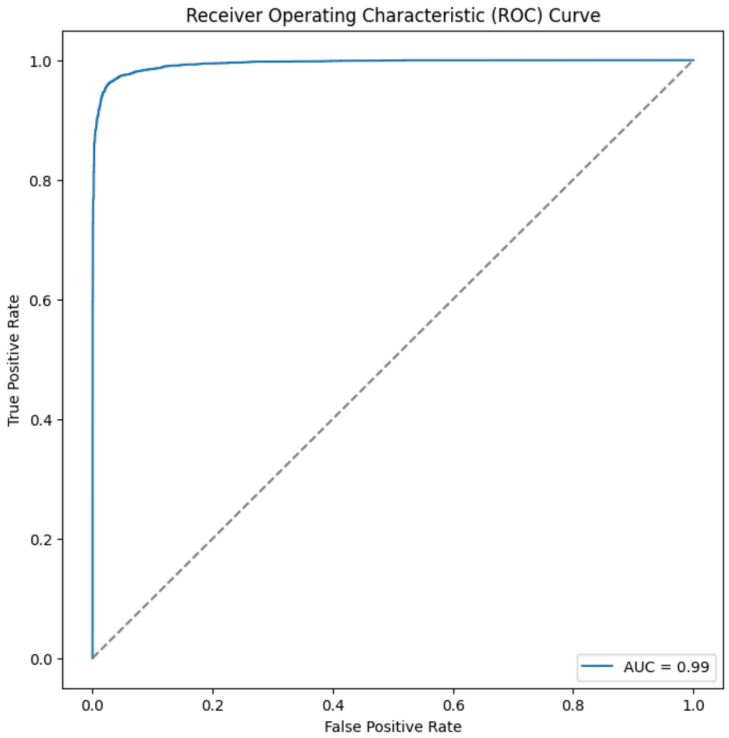


The model exhibits strong performance in terms of accuracy, precision, and true positive rate. However, the presence of false negatives suggests room for improvement in identifying positive instances

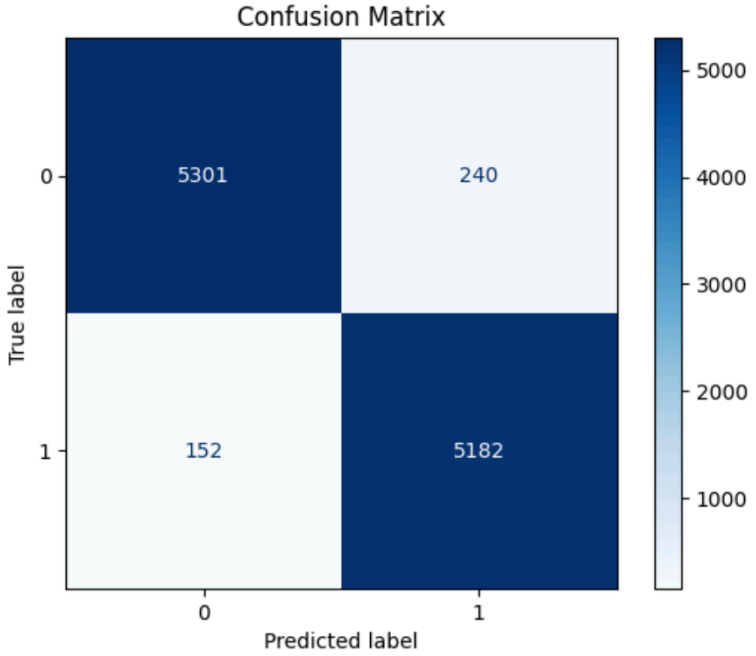
To try to mitigate the problem we introduced a Dropout layer with a dropout rate of 0.5 immediately after the Dense layer. The Dropout layer randomly drops a specified percentage of connections during training, acting as a regularization technique to prevent the model from relying too heavily on specific neurons. In addition to the Dropout layer, we modified the values of the Bidirectional layers in the model architecture. Bidirectional layers process input data in both forward and backward directions, providing a more comprehensive understanding of sequential patterns. The number of neurons in these two Bidirectional layers was set to 16, 16 respectively.







The model demonstrates strong performance.



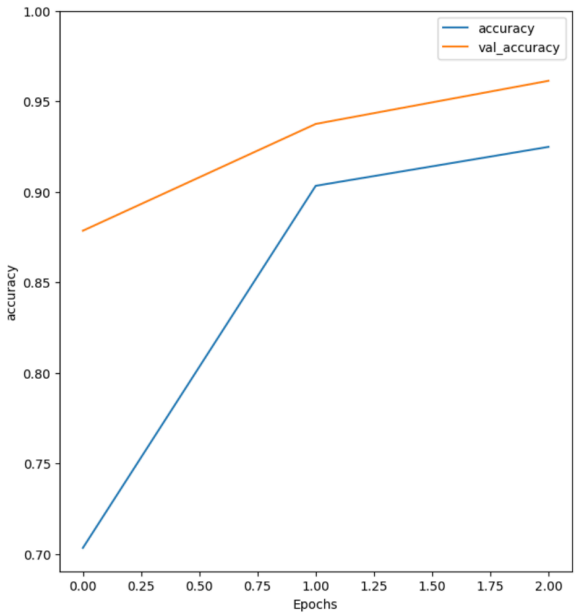
The model exhibits strong performance in terms of accuracy, precision, and true positive rate. Similar to the previous matrix, the presence of false negatives suggests room for improvement in identifying positive instances.

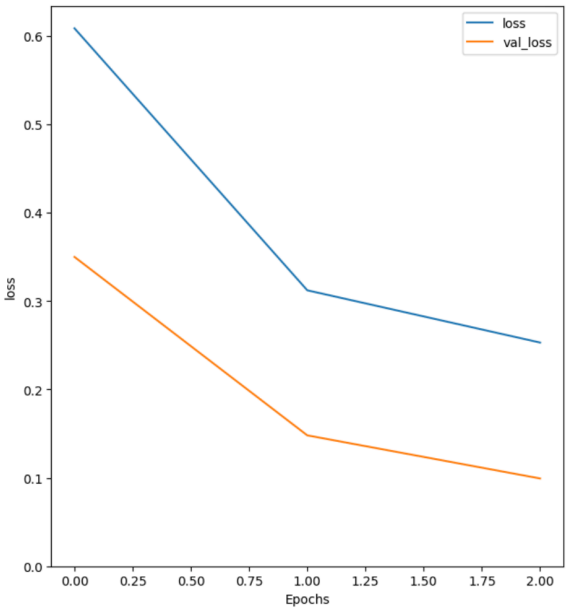
Despite achieving high accuracy values in the range of 0.74 to 0.96 during our experiments, it's noteworthy that the competition score obtained was: 0.79.

3.Results

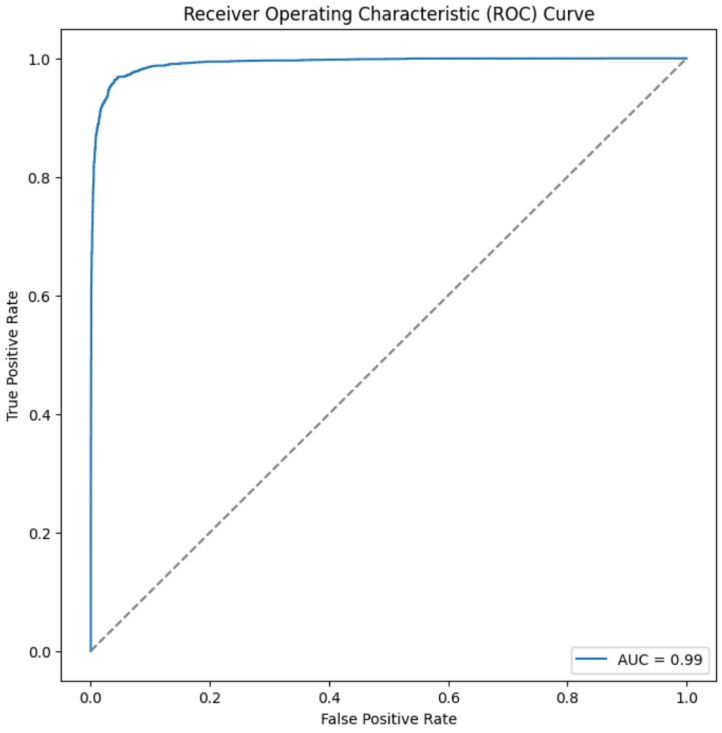
a. Final Results

In a further attempt to combat overfitting, we experimented with the depth and configuration of Long Short-Term Memory (LSTM) layers. We introduced 5 LSTM layers, each with 8 neurons, to assess their impact on model performance.

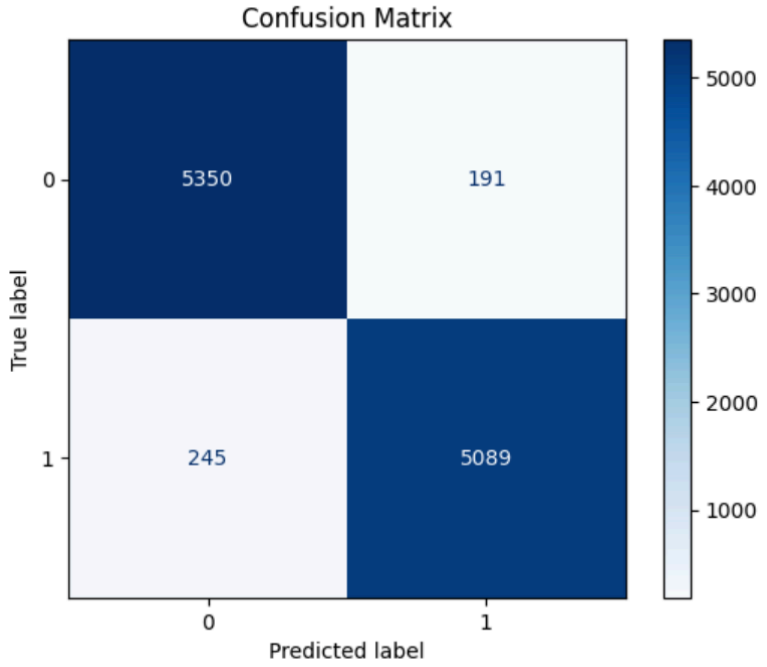




The model demonstrates significant accuracy improvement from the first epoch to the third epoch in both training and validation phases. The training loss steadily decreases across epochs, indicating that the model is effectively learning from the dataset. Validation loss also follows a similar trend.



The model demonstrates strong performance.



In summary, the model exhibits strong performance in terms of accuracy, precision, and true positive rate.

In this scenario, we observed commendable accuracy values ranging from 0.70 at the start to 0.92 towards the end. However, the actual competition score obtained was 0.837, indicating no improvement after the last modifications.

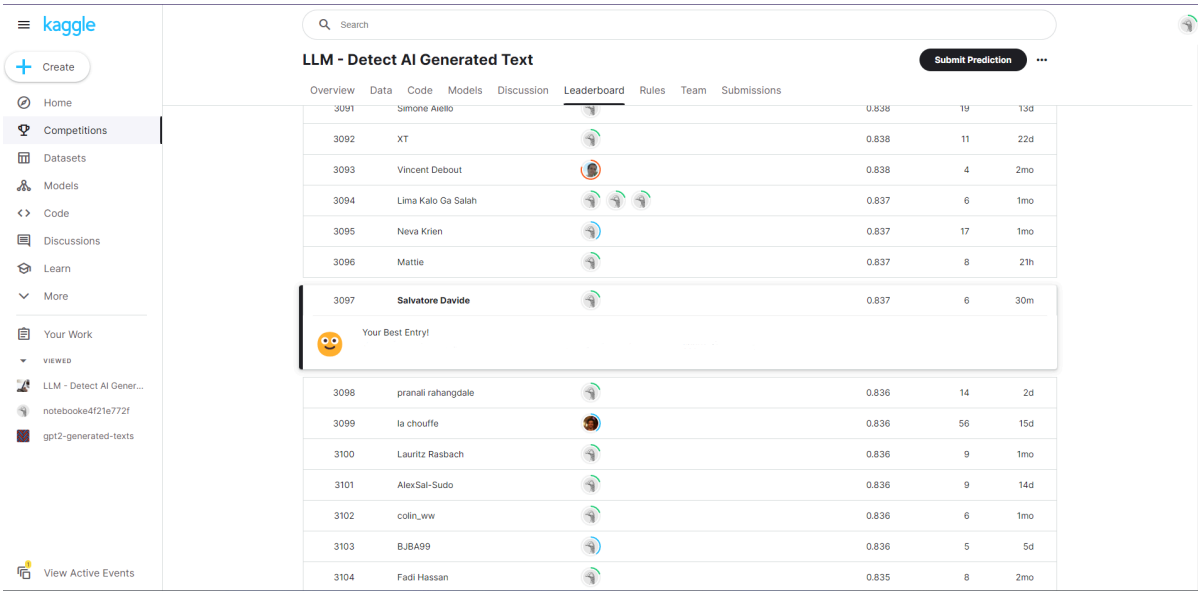
Overfitting Mitigation: The model's performance on the validation set suggests effective generalization, as indicated by a relatively high accuracy and low loss.

The training time per epoch remains consistent, indicating a stable and efficient training process.

We can say that the RNN model exhibits promising learning capabilities, achieving high accuracy and effectively generalizing to new data. The validation accuracy suggests that the model is not overfitting to the training set.

These results are encouraging and indicate the potential effectiveness of the trained model in making predictions on unseen data. Further evaluation on real-world test data will provide a comprehensive understanding of the model's capabilities.

b. Proof of the Challenge Participation



4. Conclusions

In this project, we embarked on the challenging task of distinguishing between human-generated content and text produced by Large Language Models (LLMs) in the realm of artificial intelligence.

Our strategic approach involved conducting exploratory data analysis, and training a recurrent neural network (RNN) model. This model was specifically designed to recognize patterns distinguishing between human and machine-generated text.

GPT-2 model generated approximately 9000 AI-generated texts to balance the dataset.

Our model featured a TextVectorization layer acting as an encoder, an embedding layer, bidirectional LSTM layers capturing sequential patterns, a dense layer for complex relationships, and a final layer for binary classification.
The model aimed to understand intricate linguistic features and semantic relationships between words.

Different model configurations were experimented with, including varying LSTM layer depths and neuron counts. Overfitting was addressed with the addition of a Dropout layer.

Exploring alternative architectures and fine-tuning hyperparameters could enhance model performance.

In conclusion, our approach successfully navigated the challenges of differentiating human and AI-generated text, and the model exhibited promising capabilities. The dynamic nature of the AI landscape calls for continuous exploration and improvement in AI model development.