Unified Report

Gabriel Felip Mano Lasig (M00864474)
Jedd Madayag (M00859870)
Jettlance Rivera (M00863406)
Nouar Mahamoud (M00873504)

Word Count (last updated: 15/4/25): 6294

Table of Contents

1.1 Dataset Selected and Problem Definition

The dataset we've selected contains a list of video games that had sales greater than 100,000 copies up to the year 2017. That being said, this dataset was last updated 8 years ago as of the writing of this report. This dataset can be accessed through this link: https://www.kaggle.com/datasets/gregorut/videogamesales

To be more specific about the dataset, it features not just sales from one region but from North America, Europe, Japan and "Others", presumably other lesser-known regions. In addition, "Global Sales" would add up all the previous 4 sales. It also includes the Publisher and Platform that was available to play on when it was first released, along with the Genre in which the game is described as.

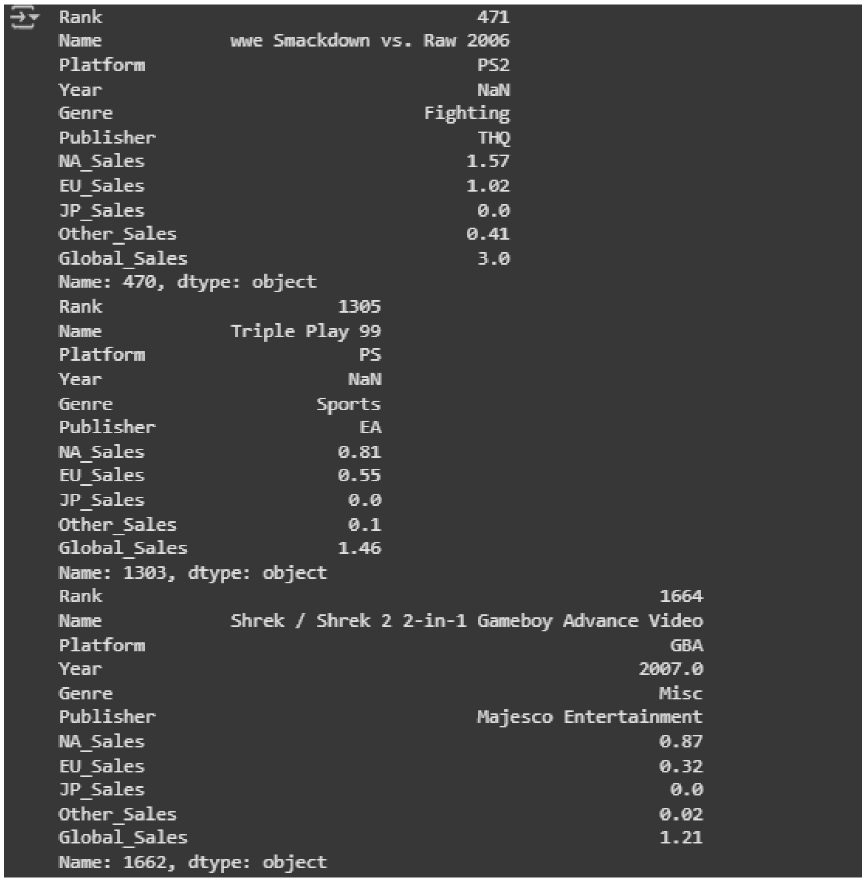
Gabriel believes that this kind of dataset can be classified either as a regression, classification or clustering problem and for various reasons. After some discussion and input from other members, we decided that we can use regression learning models as we could predict the global sales of a game using features like regional sales, the genre and publisher. We would be working with continuous numerical values. We can use classification learning models as we can determine whether a game is considered successful or not depending on the sales. This would mean working with a binary classification problem. We can uBIse clustering learning models as we could cluster games of certain genres that may be considered successful, or base it on the publisher and how well their games sell.

After some discussion, we agreed that the dataset will be a Regression problem as we will be training the model to predict the global sales of games using features such as the platform, publisher and other sales.

1.2 Data Preprocessing

Data Preprocessing is extremely important when trying to generate graphs or results. If the data quality is inconsistent or low quality, it will just result in poor usability. In our case, the dataset that our group chose had incomplete values in a couple areas. To mitigate against incomplete data, Jedd and Jett started off by reading the vgsales.csv file which is the dataset that we chose. After doing this, we checked the head, tail, info and shape of the dataframe. Doing so would allow our group to get an overview on the structure and contents of the dataframe. It also gave us an insight on any missing values in the dataframe.

After checking the missing values, we started to clean the data frame. Originally, the dataset had over 200 missing values in the "Year" column and 58 missing values in the "Publisher" column. We started off by replacing all the missing values in the year with "NAN" which stands for "Not An Number". After cleaning the "Year" column, we started to replace the "N/A" values in the "Publisher" Column. We did this by replacing all the missing values with the official publishers of those games. To execute this, we had to research the publisher of those games and with those results we used a built-in function to replace the data at column "Publisher". As a result, we were able to replace the 470th row in the "Publisher" column with a string. Once we made these changes, we had to ensure these changes were executed, so we printed the location of the data at row 470. Then we repeated this process 58 times and these were a glimpse of the results:



```
Rank                                471
Name           wwe Smackdown vs. Raw 2006
Platform                            PS2
Year                                NaN
Genre                          Fighting
Publisher                           THQ
NA_Sales                           1.57
EU_Sales                           1.02
JP_Sales                            0.0
Other_Sales                        0.41
Global_Sales                        3.0
Name: 470, dtype: object
Rank                     1305
Name           Triple Play 99
Platform                   PS
Year                      NaN
Genre                  Sports
Publisher                  EA
NA_Sales                 0.81
EU_Sales                 0.55
JP_Sales                  0.0
Other_Sales               0.1
Global_Sales             1.46
Name: 1303, dtype: object
Rank                                                   1664
Name           Shrek / Shrek 2 2-in-1 Gameboy Advance Video
Platform                                                GBA
Year                                                 2007.0
Genre                                                  Misc
Publisher                             Majesco Entertainment
NA_Sales                                               0.87
EU_Sales                                               0.32
JP_Sales                                                0.0
Other_Sales                                            0.02
Global_Sales                                           1.21
Name: 1662, dtype: object
```

Although this was a lengthy process, removing incomplete data will enable us to generate accurate visualisations giving us a clear insight in forecasting global video games sales.

1.3 Exploratory Data Analysis

Before we start training the model on predicting the global sales of games in this dataset, we first need to learn a bit more of this dataset. Meaning that, we should start with trying to find any unique insights in this dataset and come up with possible conclusions as to why those insights exist. This also means discovering patterns and trends using graphical methods. We can't fully rely on numbers as it doesn't show the full picture. Visualisation would then help in further understanding the data, making decisions that may be helpful for model development.
We can use these set of graphs (and more) to determine:

- Histograms
    - Distribution across certain values
- Heatmaps/Pairplots
    - Correlations between features

- Distplots (line or bar graphs)
    - Skewness and kurtosis

Taking a look again at potential features:

- Genre
- Publisher
- Platform

With these in mind we could answer some questions relating to these, such as:

- Which genre sells the most globally or regionally?
- What is the most popular or preferred platform?
- Which publisher do the customers prefer?

Genre:

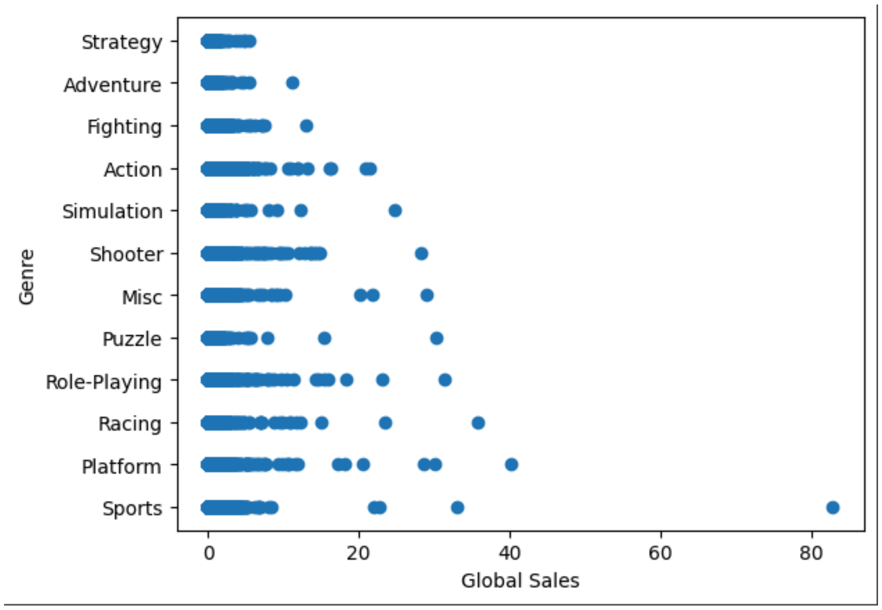
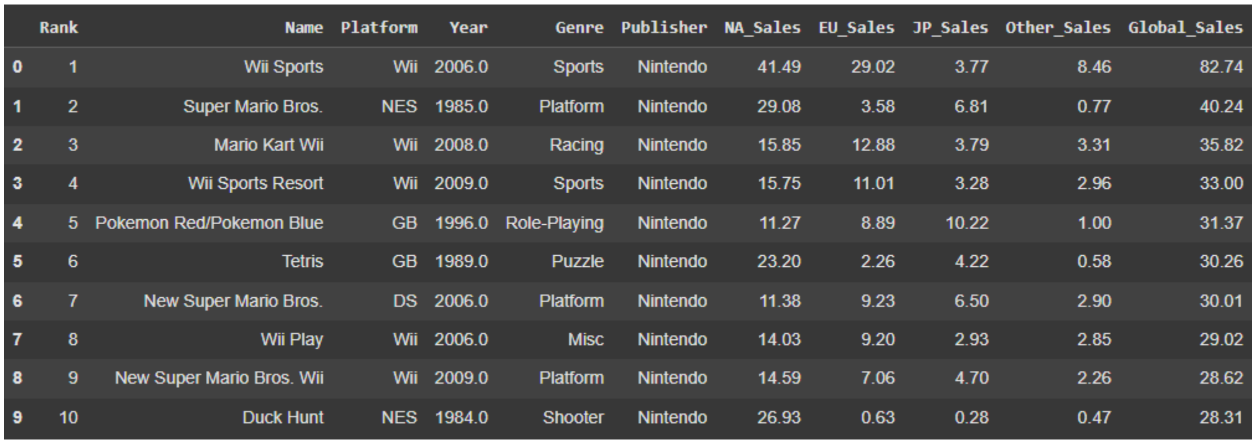


Figure 1.3.1 - number of games in each genre and its global sales

Looking at figure 1.3.2, we can see the spread of games in their respective genre and the number of sales sold globally. Generally, games under the "Strategy" genre seem to be the least popular, having on average the lowest global sales compared to other genres. Sports has one game that's an "outlier", a game that performed far beyond other genres, that being "Wii Sports" in figure 1.3.2. The genre with the games that sold well on average goes to "Shooter", looking at the density compared to others, with "Action" and "Role-playing" being next. In fact, those 3 genres, along with "Platform" seem to be the most popular genre or genre that generally sells well globally.

Publisher:

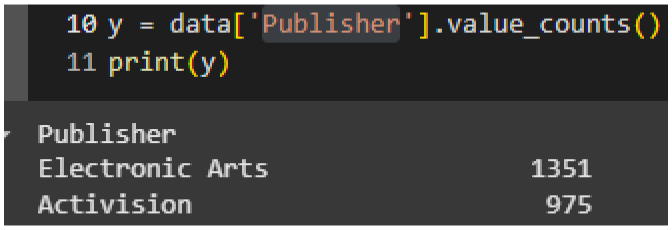
The approach here will be a little bit different. Gabriel first decided to look at the top 10 most selling games globally to see which developer/publisher seems to pull the most revenue. He believes that depending on the publisher that's tied to the sale number, it means that it's possible that the publisher that occurs on top the most is the more popular publisher/developer.



	Rank	Name	Platform	Year	Genre	Publisher	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_Sales
0	1	Wii Sports	Wii	2006.0	Sports	Nintendo	41.49	29.02	3.77	8.46	82.74
1	2	Super Mario Bros.	NES	1985.0	Platform	Nintendo	29.08	3.58	6.81	0.77	40.24
2	3	Mario Kart Wii	Wii	2008.0	Racing	Nintendo	15.85	12.88	3.79	3.31	35.82
3	4	Wii Sports Resort	Wii	2009.0	Sports	Nintendo	15.75	11.01	3.28	2.96	33.00
4	5	Pokemon Red/Pokemon Blue	GB	1996.0	Role-Playing	Nintendo	11.27	8.89	10.22	1.00	31.37
5	6	Tetris	GB	1989.0	Puzzle	Nintendo	23.20	2.26	4.22	0.58	30.26
6	7	New Super Mario Bros.	DS	2006.0	Platform	Nintendo	11.38	9.23	6.50	2.90	30.01
7	8	Wii Play	Wii	2006.0	Misc	Nintendo	14.03	9.20	2.93	2.85	29.02
8	9	New Super Mario Bros. Wii	Wii	2009.0	Platform	Nintendo	14.59	7.06	4.70	2.26	28.62
9	10	Duck Hunt	NES	1984.0	Shooter	Nintendo	26.93	0.63	0.28	0.47	28.31

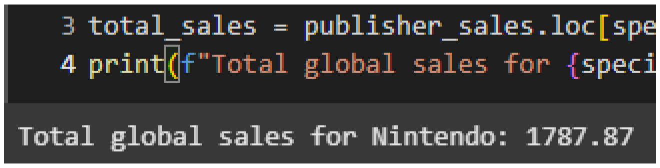
Figure 1.3.2 - top 10 ranked by Global Sales

Looking at the results in figure 1.3.3, Nintendo completely dominates the charts. In some way, this can seem understandable, with Nintendo being one of, if not, the biggest game company catered to the younger demographic. Some of its games also target those who would want to play with their family and friends, especially locally or in real life. Those games include Mario Kart and Wii Sports, with those specifically advertised to be played via multiplayer. It also doesn't help that Nintendo is a veteran company, started in the 80s and producing iconic franchises like Super Mario and Pokemon, their experience throughout the years clearly shows its positives. Another observation that was interesting was the number of games made by one publisher.

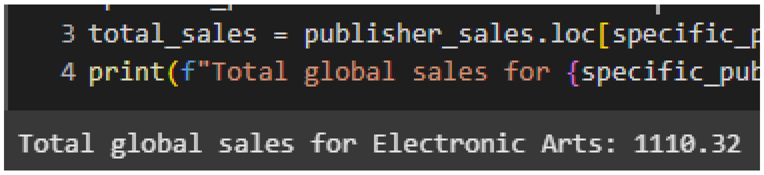


```
y = data['Publisher'].value_counts()
print(y)

Publisher
Electronic Arts                 1351
Activision                       975
```

The most number of games collected in this dataset came from Electronic Arts, and despite this their games don't do as well in sales compared to Nintendo's games.

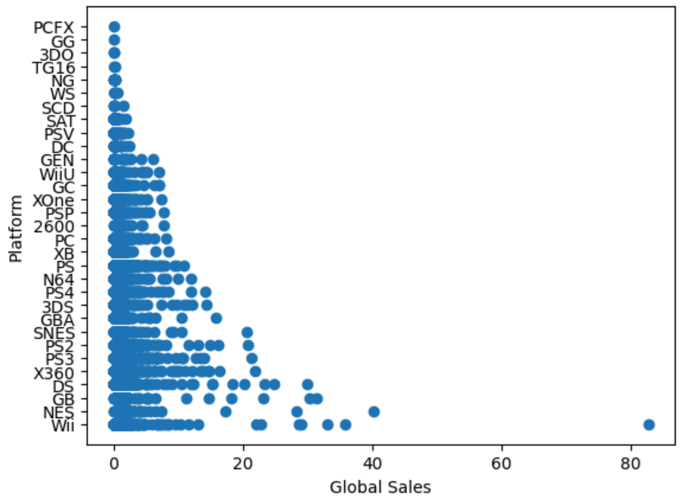


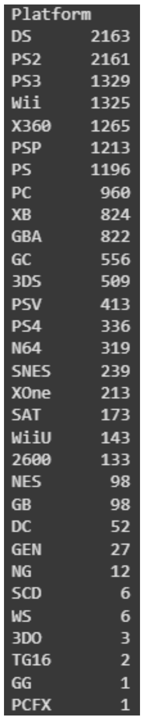
```
total_sales = publisher_sales.loc[spe
print(f"Total global sales for {speci

Total global sales for Nintendo: 1787.87
```



```
total_sales = publisher_sales.loc[specific_p
print(f"Total global sales for {specific_pub

Total global sales for Electronic Arts: 1110.32
```

Platform:





```
Platform
DS      2163
PS2     2161
PS3     1329
Wii     1325
X360    1265
PSP     1213
PS      1196
PC       960
XB       824
GBA      822
GC       556
3DS      509
PSV      413
PS4      336
N64      319
SNES     239
XOne     213
SAT      173
WiiU     143
2600     133
NES       98
GB        98
DC        52
GEN       27
NG        12
SCD        6
WS         6
3DO        3
TG16       2
GG         1
PCFX       1
```

Figure 1.3.3 - games in their respective platform and its sales and the total games in each platform

One observation in figure 1.3.3, specifically looking at the graph, the Xbox One on average made less sales globally compared to older platforms, such as the PSP (released in 2005 vs 2013) and Nintendo 3DS (released in 2010 vs 2013). Surprising considering the weight in the name in Microsoft and the previous success of the Xbox 360, seen at top 4 in figure 1.3.3. Technically, the Wii would be placed second if not for the game “Wii Sports” (see figure 1.3.1 and 1.3.2). The Xbox 360, Nintendo DS and PS3 to name have a close cluster or density in their global sales, therefore on average any game in those platforms would perform relatively well. There isn’t really a correlation in most games in a platform and the global sales in that platform in figure 1.3.3. Despite the Nintendo DS having the most games published, it still didn’t rank on top. The Nintendo Gameboy, Nintendo Entertainment System and Nintendo Wii outperformed in sales despite the lesser games. Linking to figure 1.3.2, the Nintendo again shows that the philosophy that they follow since the 80s works, with them also performing at the top.

The possible biases can vary, but it is difficult to point out every bias as it is heavily dependent on the people, what they enjoy and believe in and much more. This dataset luckily does not expose any person's personal experience or opinion of any features in here, but we can still safely assume some potential biases that may further answer our questions just from our graphical findings, later discussed in this report.

1.4 Model Development and Evaluation

Since the data is supervised, meaning we know what each data value represents, we would have to use methods for supervised data. As it is a Regression problem, we'll use the "LinearRegression" method. To check whether the model is accurate with its predictions, we'll use appropriate evaluation metrics for the appropriate problem. For regression, this would be MAE, RMSE and R^2 Score.

Quick Definitions:

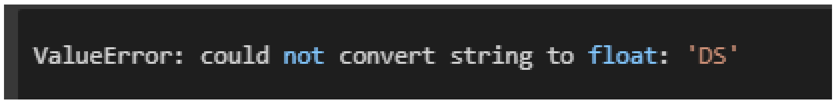
The Mean Absolute Error (MAE) is a way to measure the absolute value of difference between the actual and predicted values. Ranging from 0 to 1, this means that as the MAE is closer to 0, the learning model can accurately predict the values when compared to the actual values. Closer to 1 then means that the learning model is more likely to be inaccurate when predicting the values.

The Root Mean Squared Error is similar to the Mean Absolute Error, as it also measures the difference between the real values and the predicted values. The key difference (EUMeTrain) is that mathematically, RMSE is calculated by squaring the difference between the real and predicted values first, then finding the average. Only then would the square root of it give the final result. RMSE is used typically to emphasise larger errors found as the model trains and tests, useful to see if any points in the dataset affect the accuracy of the model and how to treat it. Similar to the MAE, if the RMSE is closer to 0, this means that the learning model can more accurately predict the values in a dataset. Closer to 1 means it is more likely to be inaccurate when predicting the values. Another thing to note is that the difference of RMSE and MAE indicate the variance of the individual difference between values. Meaning the higher the difference, the greater the inaccuracy and spread.

The R^2 Score measures how "good" of a fit a model is or the variance found in the dependent variable that can generally be explained by the independent variable. This means that the higher the R^2 Score, or generally when it's closer to 1, it means that the model is very fitted and there's low variance or difference in the data. This means the opposite when the Score is closer to 0.

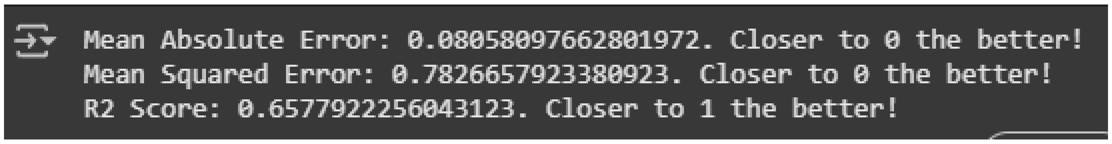
Findings:

In this dataset the dependent variable or variable we will be testing is "Global _Sales" and the independent variables that we will use to manipulate the dependent variable are "Platform", “Genre”, “Publisher”, ”NA_Sales”, ”EU_Sales”, “JP_Sales”, “Other_Sales”, determined by Gabriel. It’s good to note that when applying the independent variable, that all the features listed here should be in a ‘float’ data type, as the regression model only accepts that data type for learning. The sales features abide by this rule but not for “Platform”, “Genre” and “Publisher” as they are ‘string’ data types. Trying to fit the model gives this error:



```
ValueError: could not convert string to float: 'DS'
```

Gabriel was able to solve this using SkLearn’s “OneHotEncoder”. This encoder learns the categorical features first, then we transform that categorical features to numerical representations. Then the independent variable can be updated to use the newly changed features.
Because of this, we can now start on fitting and training the model. Here are the results, using a test size of 30%:



```
Mean Absolute Error: 0.08058097662801972. Closer to 0 the better!
Mean Squared Error: 0.7826657923380923. Closer to 0 the better!
R2 Score: 0.6577922256043123. Closer to 1 the better!
```

Looking back at the definitions for each, we can see it can be particularly accurate when looking at it individually (MAE), but when it comes to predicting at a much larger scope (RMSE), it can get inaccurate. This means that at some points, it can closely predict the value of a game’s “Global_Sales” but at some point during the process, it misses the mark and inaccurately predicts. Furthermore, the R² Score suggests a somewhat fitted model with some variance between its predictions and real values, further proving the hypothesis of missing the mark.

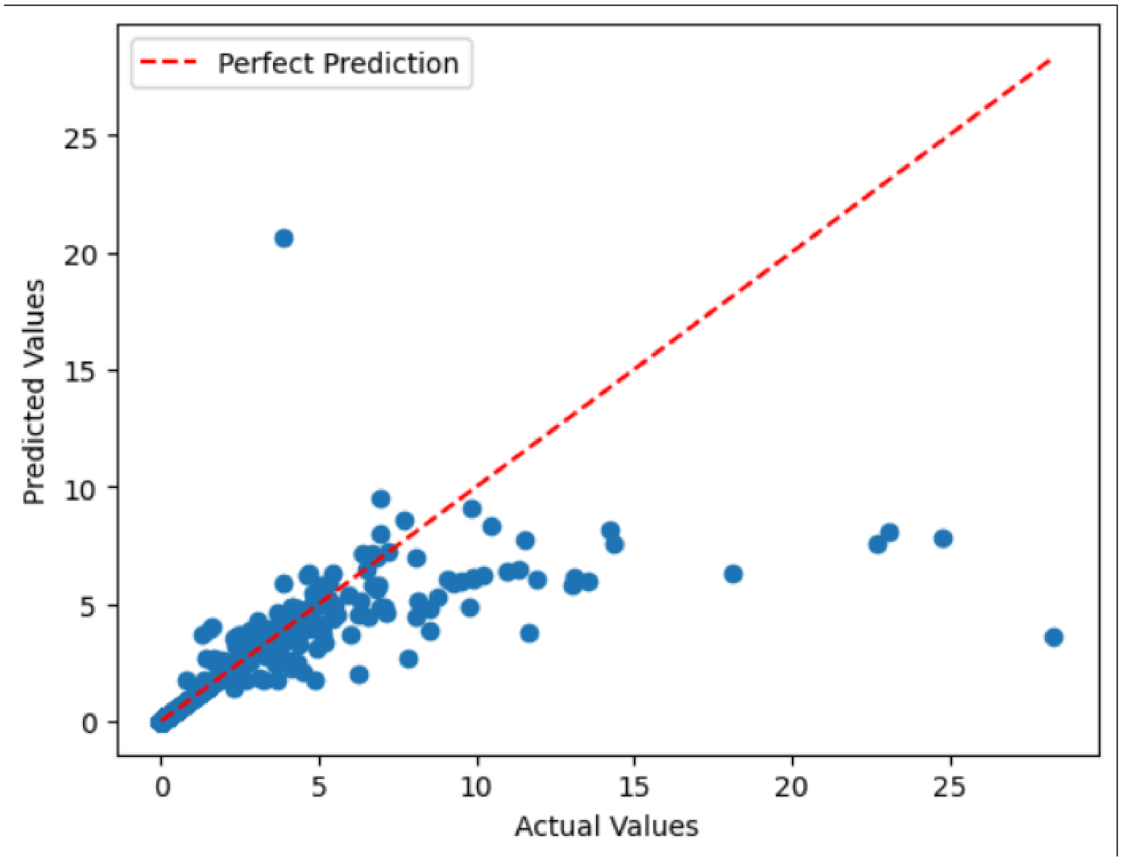
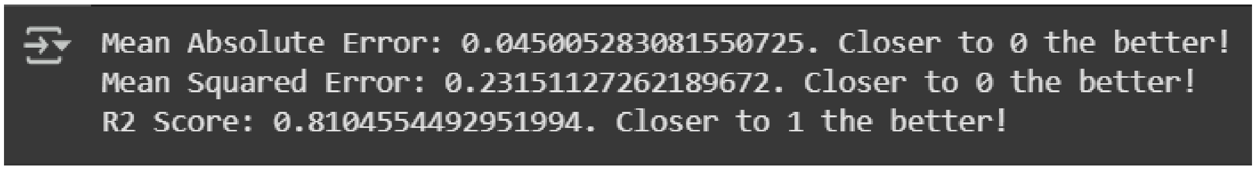


Figure 1.4.1 - Graphical Visualisation of the learning model's predictions against actual values with a test size of 30%.

A better way of understanding the results of the model's predictions is through graphs. Using a scatter line graph, we can see the spread of the model's predictions to the actual values. Looking at Figure 1.4.1 above, we can see initially a good fit, but later on it starts to stray away from the perfect prediction. Whilst it is expected, it becomes apparent that the model predicts far from the actual value. The initial cluster of great predictions proves the MAE's close to zero score, whereas the latter part is proven by the RMSE's score.

After running it again, this time with a smaller test size (20% from 30%), these are the results:



```
Mean Absolute Error: 0.045005283081550725. Closer to 0 the better!
Mean Squared Error: 0.23151127262189672. Closer to 0 the better!
R2 Score: 0.8104554492951994. Closer to 1 the better!
```

It got a much better result compared to using a much larger test size. This makes sense as the size of data it trains from is larger than last time, meaning the model was able to learn more patterns.

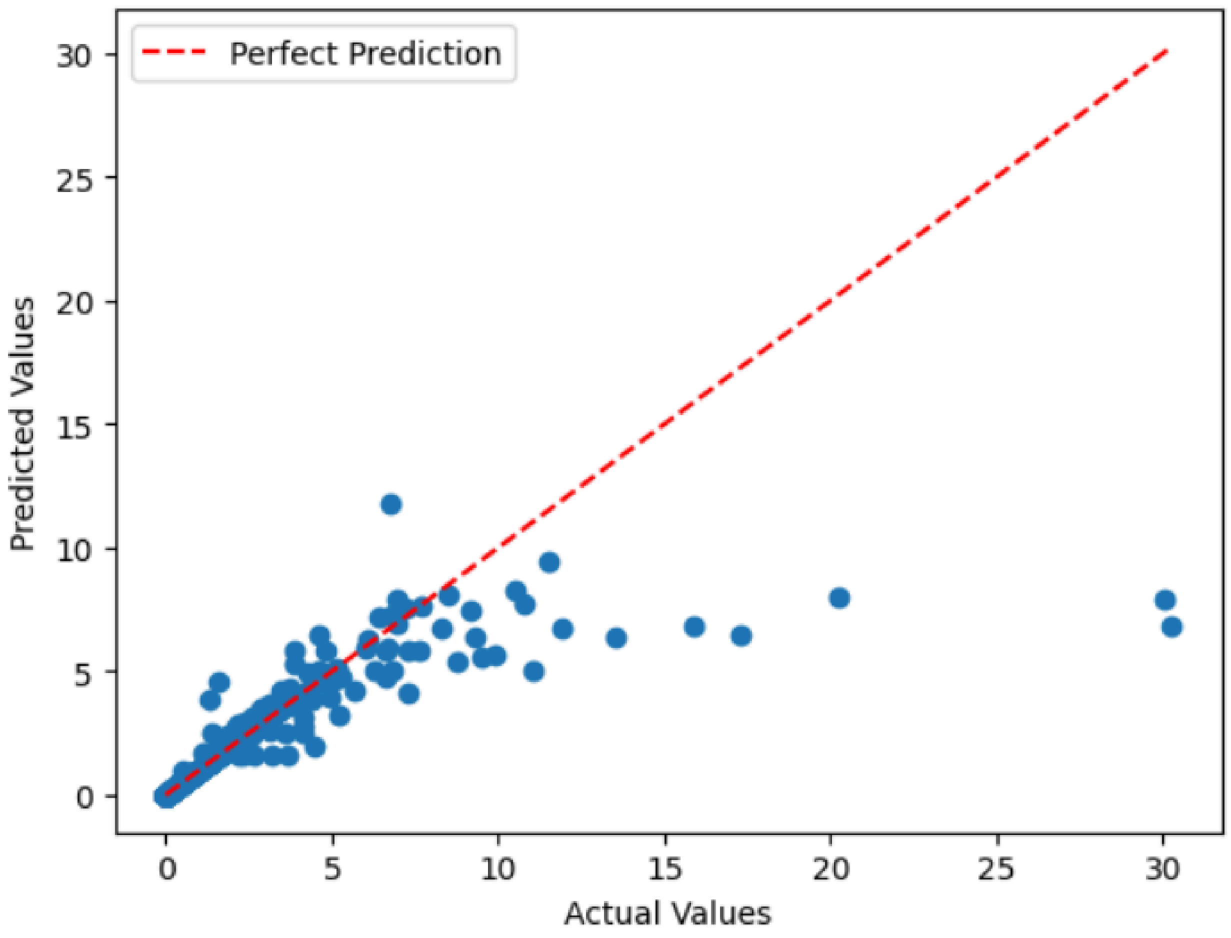


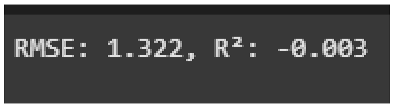
Figure 1.4.2 - Graphical Visualisation of the learning model's predictions against actual values with a larger test size.

Looking at Figures 1.4.1 and 1.4.2, there are much less outliers or data points that are further away from the perfect prediction. There is also a much tighter fit meaning that the variance of values is smaller. This is further proven with the R^2 Score being higher than the initial testing. The MAE score backs up the observation that the point in which the model begins to predict values far from the actual values is further than in the first test, where in Figure 1.4.1 it begins to stray at just under 10, whereas in Figure 1.4.2 it strays in-between 5 and 10 in the x-axis.

Using a different model for Comparison:

This model was trained on data that was not scaled or normalised. Supposedly, scaling or normalising the data may affect the performance of these measures.Various online forums gave mixed opinions on this ((Nguyen) and (davidparks21)), with (davidparks21) saying that depending on the scaling but here is what we learnt. The MAE and RMSE can be affected if the data was normalised. This could be due to how the numerical data is scaled between each other. In this online forum (stuffingmybrain), data would be normalised or scaled depending on the variance. In our case however, we believe that the numbers aren't egregiously scaled such

that the range is too large. It already takes into account that it is measured in millions (i.e., 86 for a game's sale means it made 86 million).
According to this article by (IBM), as the number of predictors increases, the input-output relationship changes. This can be solved using Regularisation. Using Lasso regression, we can compare the MAE, RMSE and R² Score and see if regularisation improved our results.
Here's the results using Lasso Regression:



```
RMSE: 1.322, R²: -0.003
```

The R2 Score has vastly improved, meaning that there is little to no variance between the predicted and real values. However, the RMSE has slightly worsened. Considering the distance from 1, they're both the same at about 0.32, but this doesn't distract from the fact that with a different model, it can still poorly perform when taking into account the difference of the predicted and real values in some parts of the process. Regularisation hasn't proved enough to be a solution to the poor results from earlier, but it has proved that because of there being more than 2 predictors, that it can affect the prediction results, seen from using a different learning model.

1.5 Ethical Considerations

In developing and deploying machine learning models, addressing ethical considerations is paramount to ensuring fairness, transparency, and accountability. This section highlights potential biases and fairness issues within the dataset and model, and proposes practical strategies to mitigate these ethical challenges. The analysis is based on the dataset and steps shown in the provided screenshots, which include the video game sales dataset '***vgsales.csv***' and the associated data preprocessing, exploratory data analysis (EDA), and model development steps.

Potential Biases

1. Dataset Biases:

- **Representation Bias**: The dataset contains video game sales data up to 2017, with features such as '***Platform***', '***Genre***', '***Publisher***', and regional sales ('***NA_Sales***', '***EU_Sales***', '***JP_Sales***', '***Other_Sales***'). However, the dataset may not equally represent all platforms, genres, or publishers. For example, certain genres like "Action" or platforms like "PS4" may be over-represented, leading to biased predictions for under-represented categories such as "Puzzle" games or less popular platforms. This is evident in the exploratory data analysis (EDA) where certain genres and platforms dominate the dataset.

- **Historical Bias**: The sales data reflects historical market trends favouring certain regions (e.g., North America or Europe) over others, as seen in the regional sales columns (‘***NA_Sales***’, ***‘EU_Sales’***, etc.). This could lead to a model that underestimates the potential success of games in under-represented regions like Japan or other parts of the world. For instance, the model may prioritize games that historically performed well in North America, marginalizing games that are popular in other regions.

- **Label Bias**: The sales figures in the dataset may be influenced by external factors such as marketing budgets or regional preferences, which could introduce bias into the model's predictions. For example, games with higher marketing budgets may have higher sales, regardless of their quality or appeal, as seen in the ‘***Global_Sales***’ column.

2. Model Biases:

- **Algorithmic Bias**: The model, as shown in the screenshots, uses features like ***‘Platform’***, ‘***Genre***’, and ‘***Publisher***’ to predict ‘***Global_Sales***’. However, the model may disproportionately favour certain genres, platforms, or publishers due to their prevalence in the dataset. For example, if the dataset contains more data for "Action" games, the model may predict higher sales for "Action" games, even if other genres like "Puzzle" or "Simulation" have the potential to perform well.

- **Feedback Loops**: If the model is deployed in a real-world scenario, its predictions could influence future sales, potentially reinforcing existing biases in the gaming industry. For instance, if the model predicts low sales for indie games (due to their under-representation in the dataset), publishers may be less likely to invest in them, further marginalizing indie developers.

3. Deployment Biases:

- **Regional Disparities**: The model may not account for cultural or regional differences in gaming preferences, leading to inaccurate or unfair predictions for certain markets. For example, a game that is popular in Japan (as seen in the `JP_Sales` column) may not perform as well in North America, and vice versa. The model, trained on global sales data, may fail to capture these regional nuances.

- **Accessibility**: The model's predictions may not be equally beneficial to all stakeholders, particularly smaller publishers or indie developers who may not have the resources to compete with larger companies. This is evident in the dataset, where major publishers like "Nintendo" or "Electronic Arts" dominate the sales figures, potentially marginalizing smaller publishers.

Practical Strategies for Mitigating Ethical Challenges

1. Dataset Auditing and Preprocessing:

- **Conduct a thorough audit of the dataset**: We could start by analyzing the distribution of data across different categories to identify any over-represented or under-represented groups. For example, if certain genres or platforms dominate the dataset, we might need to balance the data to ensure fair representation.

- **Balance the dataset**: One way to address representation bias is by using techniques like oversampling under-represented genres or platforms. For instance, if indie games are, we could generate synthetic data or oversample existing data points to create a more balanced dataset.

- **Normalize and scale the data**: As shown in the screenshots, we could use techniques like min-max scaling or z-score normalization to ensure that the model does not disproportionately favour games with higher sales figures. This helps in creating a level playing field for all games, regardless of their sales scale.

2. Bias Detection and Mitigation in Models:

- **Implement fairness-aware algorithms**: We could use algorithms that explicitly account for fairness constraints during training. For example, adversarial biasing can help reduce bias by ensuring that the model's predictions are fair across different groups.

- **Evaluate the model using fairness metrics**: In addition to traditional metrics like MAE and R^2 Score, we could incorporate fairness metrics such as demographic parity or equalized odds. This would help us ensure that the model does not disproportionately favour certain genres, platforms, or publishers.

3. Transparency and Explainability:

- **Use explainability tools**: Tools like SHAP or LIME can help us understand how the model makes decisions. By providing insights into the model's decision-making process, we can identify and address potential sources of bias.

- **Document the model's limitations**: We could create a model card that outlines the model's performance, limitations, and potential biases. This ensures transparency and helps stakeholders understand the model's behaviour and limitations.

4. Inclusive Design and Stakeholder Engagement:

- **Involve diverse stakeholders**: We could engage with representatives from smaller publishers and indie developers to gather their feedback on the model's predictions. This ensures that the model addresses the needs and concerns of all stakeholders, not just the dominant ones.

- **Conduct impact assessments**: Before deploying the model, we could analyze how its predictions might affect different stakeholders. This helps in identifying any potential negative consequences and addressing them proactively.

5. Continuous Monitoring and Feedback:

- **Establish ongoing monitoring mechanisms**: We could set up a system to regularly monitor the model's performance and fairness in real-world settings. This involves collecting data on the model's predictions and comparing them to actual outcomes.

- **Create feedback loops**: We could develop a feedback mechanism where users can report any issues or biases they encounter. This ensures that the model can be continuously improved based on real-world feedback.

By addressing these ethical considerations, we can build a model that is not only technically robust but also socially responsible and equitable. This approach fosters trust, ensures fairness, and minimises harm, ultimately contributing to the responsible use of AI technologies in the gaming industry.

2.1 Text Dataset Selection and Preprocessing

Goal and Dataset Selection:

The goal of this section of this report is to create a neural network that complements with a text-based dataset. More specifically, with a text-based dataset, we will carry out tasks in which it requires processing and outputting text or strings. As for the goal, Gabriel has decided that the task that we'll solve is to determine whether a given sentence is considered "positive" or "negative". Theoretically, if used throughout the web this neural network can be deemed useful for summarisation and quick and ease of understanding of certain texts. In recent years, online e-commerce companies like Amazon created machine learning models to summarise user reviews, making it easier for users to decide whether to buy something or not. Others created tools to look at the price history of items and comparison of similar items, further helping in the decision process. Whilst our model wouldn't be as sophisticated, this task can be a good starter in learning how and why to develop a machine learning model.

The dataset we selected is IMDB Movie Reviews. This dataset can be accessed through this link:
https://www.kaggle.com/datasets/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews?resource=download (Mass et al. 142-150).
Gabriel and Jedd decided on this dataset as it can be used for "Binary Sentimental Classification", knowing whether a movie is likable or not, based on user reviews and knowing if they're positive and negative. This dataset contains 2 columns making it a simple dataset, the review and its sentiment, to use for analysis and testing. Also, we checked the sentiment count of the dataset and we concluded that it is balanced - with 25000 positive reviews and 25000 negative reviews.

Preprocessing:

The process of preprocessing the data can go some ways: removing numbers or punctuation in text, tokenizing words in a text or removing "stopwords" in text for better summarisation. This is done to clean data, ensuring that the data the model will be trained with isn't inaccurate or inconsistent, and overall making it an easier process for it to learn. Especially for tokenization, where a model can benefit from as it aids in finding relationships or trends between words.

Taking a look at a sample data from our dataset:

"A wonderful little production. <br /><br />The filming technique is very unassuming- very old-time-BBC fashion and gives a comforting, and sometimes discomforting, sense of realism to the entire piece…"

We can see some things that we have to clean out, before we can start on submitting this as input for our model. Things such as 'html tags' and special characters like dashes and periods should be processed through and removed. This is because they are unnecessary characters that could affect the training process of our model. This is the case for every single review, therefore we use a ‘for loop’ to iterate through each review and apply the necessary filters until we get a “cleaned” review. This can be done by importing ‘re’ and a tokenizer, to remove the ‘html tags’, special characters and stop words (for example, ‘and’, ‘the’). Next is to equalise the spacing between words then join them together so that we can append it to a new array, storing the “cleaned” reviews. Here are the results (compare this to the one above):

“*A wonderful little production The filming technique unassuming old time BBC fashion gives comforting sometimes discomforting sense realism entire piece The actors extremely well chosen Michael Sheen got polari voices pat You truly see seamless editing guided references Williams diary entries well worth watching terrificly written…*”

2.2 Deep Learning Model Implementation

Designing the Model:

We’ll be using Tensorflow to create the neural network. This is because of it being beginner-friendly, and easier to understand when it comes to building the model. It also means that with its open and large library, we have access to various models meaning we can compare the performance of each, to see which is better if we wanted to. We’ll be using two different models, LTSM (Long Short-Term Memory) and Naive Bayes. We’ll mainly be focusing on LTSM and how it’s developed. The reason for the second model is as initially explained above, we’ll compare performances and results to see which is better, along with seeing how to improve the LTSM performance whilst minimising the effects on accuracy if we can.

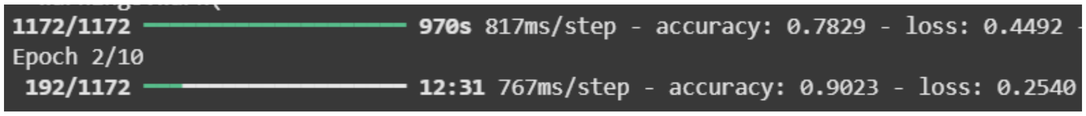
Long Short-Term Memory Model

LTSM is split into three different layers, the Embedding, Hidden and the Output Layer. The Embedding layer is essentially the input, and that input is our preprocessed reviews from our dataset. The Output layer should only contain one node, and that node should tell us whether an inputted review is a ‘positive’ or ‘negative’ review. The Hidden Layer is where the inputs are being processed.
The Embedding is simple, giving it 10000 input nodes, then linking it to 128 output nodes. Seeing as there are that many and we have around 30000 data points for training (initially set the ‘test_size’ to 0.25, so 75% left is used for training), this means that 3 ‘epochs’ would be sufficient to get through all of the available data.
The Hidden Layer contains 3 individual layers of LSTM. Each of these layers contain 128 output nodes which will lead to the other hidden layers, with the same number of input nodes.
The Output Layer only contains 1 node, which is the resulting node after being processed through the Hidden Layers. Its activation is set to “Sigmoid”, meaning that the result will always return a value of 0 or 1.

Training the Model:

Long Short-Term Memory Model

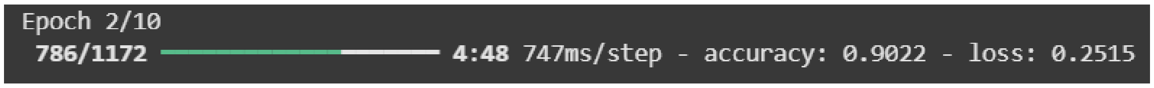
Using Tensorflow, the way to train the model with our data is by feeding the data and letting it loop and "remember". Called 'epochs', the model loops through the neural network a set number of times and in each iteration, the accuracy slowly increases and the "loss", which measures how much of the useful data is lost during training, decreases.



```
1172/1172 ━━━━━━━━━━━━━━━━━━━━ 970s 817ms/step - accuracy: 0.7829 - loss: 0.4492
Epoch 2/10
 192/1172 ━━━━━━━━━━━━━━━━━━━━ 12:31 767ms/step - accuracy: 0.9023 - loss: 0.2540
```

Figure 2.2.1 - screenshot of results between "epoch" 1 and 2

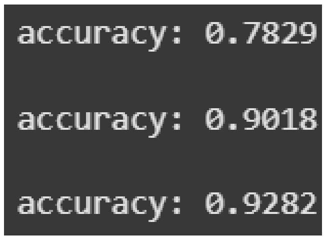
Depending on the size of the data we give it, also called the 'batch_size' in Python, the time it takes for the model to finish one epoch, or cycle, varies. In the initial testing of this, we set the 'batch_size' to 32, typically a standard number to set it. However, as seen on the screenshot above, it would take around 10 minutes to complete one training cycle. Again, to be expected with the amount of data we're handling plus the batch sizes we're feeding to the model. Eventually, the accuracy and loss percentages come at a stand-still, with it not increasing and decreasing respectively in each step of the cycle. This means that we can reduce some things to better reduce the training time whilst not affecting the accuracy.



```
Epoch 2/10
 786/1172 ━━━━━━━━━━━━━━━━━━━━ 4:48 747ms/step - accuracy: 0.9022 - loss: 0.2515
```

Figure 2.2.2 - screenshot of results of "epoch" 2

Comparing Figure 2.2.2 to Figure 2.2.1, focusing on the second 'epoch' we can see that the accuracy and loss percentages haven't shifted much at all within that time frame. In fact, the accuracy decreases despite not losing as much data as each step comes and goes. Going into the third 'epoch' the increase in accuracy was not as big of a jump as we thought. First 'epoch' ended at around 78%, the second 'epoch' ended at 90% and the third 'epoch' ended at 93%. The difference in each cycle decreases, therefore if it continues till the tenth 'epoch', then Gabriel believes that change in increased accuracy plateaus around 3-4 'epochs' and we won't get much of a better performance after that.



```
accuracy: 0.7829

accuracy: 0.9018

accuracy: 0.9282
```

Figure 2.2.3 - Results from epochs 1 to 3, showing a decrease in accuracy increase

Naive Bayes

The way to build and train this model is not too dissimilar to what we did in the first part of the coursework. Some extra steps had to be made, such as vectorizing our training data, and vector fitting the testing data, in order for the model to work. This is because without this, the model reads our data as a 1D array when it actually expects a 2D array.

Vectorizing our data solves this problem. As for what data we're using, we are assigning our reviews under the 'x' and the sentiments, positive or negative under 'y'. Before this, the positives and negatives were replaced with 1's and 0's respectively for consistency. After all this, we can now fit and train the model, which then leads to evaluating both models.

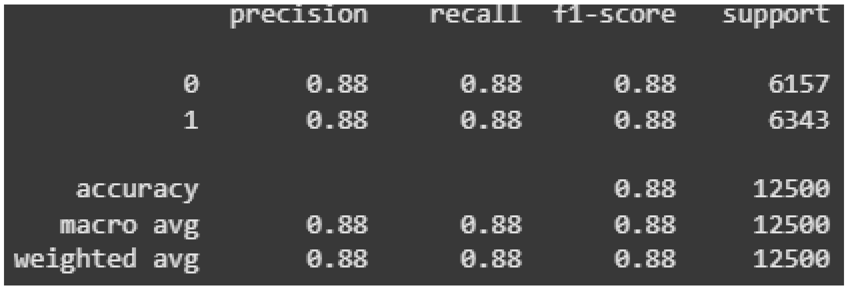
2.3 Evaluation and Insights

Insights:

To see how we can determine the performance of either model, we'll be focusing on the Accuracy, Precision, Recall scores, Classification Report and Confusion Matrix for both models. The Confusion Matrix details four important points: True Positives, False Positives, True Negatives and False Negatives. This is important to see these, especially in an essentially binary classification model, as we get to see where the models went wrong and hypothesise why and how to solve this.

LTSM Performance

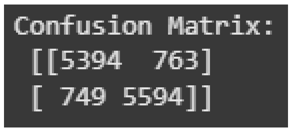
Looking at the performance of the LTSM model:
Accuracy Score -> 0.87904
Classification Report:



```
              precision    recall  f1-score   support

           0       0.88      0.88      0.88      6157
           1       0.88      0.88      0.88      6343

    accuracy                           0.88     12500
   macro avg       0.88      0.88      0.88     12500
weighted avg       0.88      0.88      0.88     12500
```

Figure 2.3.1 - LTSM Classification Report

Confusion Matrix:

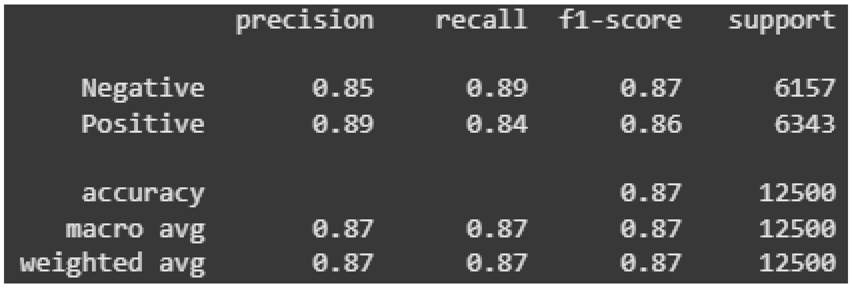


```
Confusion Matrix:
 [[5394  763]
 [ 749 5594]]
```

Figure 2.3.2 - LTSM Confusion Matrix

Naive Bayes Performance

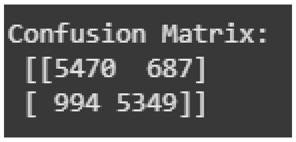
Looking at the performance of the Naive Bayes model:
Accuracy Score -> 0.86552
Classification Report:



```
              precision    recall  f1-score   support

    Negative       0.85      0.89      0.87      6157
    Positive       0.89      0.84      0.86      6343

    accuracy                           0.87     12500
   macro avg       0.87      0.87      0.87     12500
weighted avg       0.87      0.87      0.87     12500
```

Figure 2.3.4 - Naive Bayes Classification Report

Confusion Matrix:



```
Confusion Matrix:
 [[5470  687]
 [ 994 5349]]
```

Figure 2.3.5 - Naive Bayes Confusion Matrix

Evaluation:

Strengths and Limitations

Comparing the Classification Report from LTSM (Figure 2.3.1) and Naive Bayes (Figure 2.3.4), the results show they're both similar across all measures of performance, with minimal differences ranging from 0.1 to as low as 0.4. This means that both models perform similarly when it comes to correctly classifying what's positive or negative during the test.

However, when look at both model's Confusion Matrix (Figures 2.3.2 and 2.3.6), we can see that the LTSM model made less incorrect classifications, with the LTSM model making 1512 incorrect conclusions whereas the Naive Bayes model making 1681 incorrect conclusions, 169 more. Especially when we look at the False Negatives, LTSM can better identify when a review is negative, whereas the Naive Bayes model can slightly better identify when a review is positive.

Generally, the group agrees that either one of the models would be good enough for a task like classifying whether a review is positive or negative. However, the group believes that if there is a scenario where a user would specifically want to know whether a movie is good or not depending on the overall reception from online reviews then a user can pick the LTSM model to see if their initial opinion of a movie being potentially bad is backed up by the reviews, as this model is more accurate in finding negative reviews, or the user can pick the Naive Bayes model to see if their initial opinion of a movie being potentially good is backed up by the reviews, as this model is more accurate in finding positive reviews, and vice versa.

Areas for Improvement

Linking to Figure 2.2.3, there are various ways we could improve the training portion. After some discussion, the group decided these are the ones that can be make some adjustments:

1. Batch_size -> 128 up from 32
2. Epochs -> 3 down from 10
3. Num_words
4. max_length

There are other ways of improving the performance, such as using a much more powerful GPU/TPU for machine learning training. In fact, with a better GPU/TPU, we'd be able to run the training with more 'epochs', even if the "plateau point" was made initially. Despite that, it would improve the performance, and even if the improvement was small, it would still be a better alternative to other models like Naive Bayes. Had we been able to create a learning curve graph for the LTSM model, we may have been able to either prove or disprove the initial hypothesis. However as it stands we do not have access to such a resource as we are stuck with Google Colab. Either way, either one or more of these attributes can be adjusted to hopefully change the process of training the model.

In terms of training the model, decreasing the num_words may improve the performance of the model because it can reduce noise which can be extremely useful for this dataset as it contains a lot of vocabulary and text. Therefore, reducing the number of words allows the model to focus on the most important and frequently occurring words, potentially improving its performance by reducing the noise and removing anything redundant in the model. Moreover, limiting the max_length of the model can improve a few things. One of these being better memory management. Because this dataset contains a lot of long text, the model will require more memory and power to process. Ultimately, controlling the memory usage of the model increases the performance of the model and provides more efficient processing times.

References:

1. Google Colab: MachineLearning/AI.ipynb
2. EUMeTrain. “Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE).” *Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE)*, 2020, https://resources.eumetrain.org/data/4/451/english/msg/ver_cont_var/uos3/uos3_ko1.htm#:~:text=The%20MAE%20is%20a%20linear,weighted%20equally%20in%20the%20average.&text=The%20RMSE%20is%20a%20quadratic,in%20both%20of%20the%20references. Accessed 9 3 2025.
3. Nguyen, Duc. “Pre-processing data: Normalizing data labels in regression?” *machine learning - Pre-processing data: Normalizing data labels in regression?*, 2016, https://stackoverflow.com/questions/36540745/pre-processing-data-normalizing-data-labels-in-regression. Accessed 11 2 2025.
4. davidparks21. “Is it valuable to normalize/rescale labels in neural network regression?” *normalization - Is it valuable to normalize/rescale labels in neural network regression?*, 2017, https://datascience.stackexchange.com/questions/22776/is-it-valuable-to-normalize-rescale-labels-in-neural-network-regression. Accessed 11 2 2025.
5. stuffingmybrain. “Why should we normalize our data? Are there any situations in which we *won't* want to normalize?” *Why should we normalize our data? Are there any situations in which we *won't* want to normalize?*, 2022, https://www.reddit.com/r/datascience/comments/v4u6e4/why_should_we_normalize_our_data_are_there_any/. Accessed 11 2 2025.
6. IBM. “What Is Regularization?” *What is Regularization? | IBM*, 2023, https://www.ibm.com/think/topics/regularization. Accessed 11 2 2025.
7. Mass, Andrew L., et al. “Learning Word Vectors for Sentiment Analysis.” *Proceedings of the 49th Annual Meeting of the Association for Compuational Linguistics: Human Language Technologies*, vol. 1, no. 1, 2011, 142--150. *Learning Word Vectors for Sentiment Analysis*, http://www.aclweb.org/anthology/P11-1015.