Introduction:

After a semester exploring various topics in Natural Language Processing (NLP), we were asked to do a final project applying NLP and data processing to text or images. This project focuses on predicting movie revenue using a combination of structured and unstructured features, including reviews, crew information, genres etc. from datasets sourced from Kaggle.

Our project progresses from a baseline model to increasingly complex architectures, concluding in an advanced neural network with BERT embeddings. The report examines how textual features and NLP techniques enhance predictive accuracy.

Our hypothesis during this project was that as model complexity increases and more advanced NLP techniques are applied, the predictions will improve. Advanced models, particularly those using NLP embeddings, are expected to capture richer contextual and semantic information, leading to better predictions.

This report outlines the data integration, preprocessing, model architectures, and results, focusing on the impact of model complexity and NLP techniques on the performance.

Data: Features and Preparation

To build an accurate and effective model, we utilized a dataset of 'imdb_movies' taken from Kaggle.com website. This dataset contains entries, each representing a movie. The dataset is rich with features that provide a complete view of each movie. The dataset consists of 12 columns and 10,179 rows, with the following features:

- names: The name of the movie.
- date_x: The release date of the movie (format: MM/DD/YYYY).
- score: A numerical rating or score for the movie (float).
- genre: A list of the movie's genres, separated by commas.
- overview: A brief description of the movie.
- crew: Key members of the movie's cast and crew, including actors and their roles.
- orig_title: The original title of the movie, which may differ for international releases.
- status: The release status of the movie.
- orig_lang: The original language of the movie.
- budget_x: The production budget of the movie (in USD, float).
- revenue: The revenue generated by the movie (in USD, float).
- country: The country associated with the movie's release.

In this dataset, there are a few textual features but only one feature provided valuable free-text descriptions: the overview, summarizing the storyline of each movie. Initially, this feature was considered in the models for use in predicting the movie revenue. However, through experimentation, it was evident that the 'overview' feature posed challenges for the models in predicting revenue. The general language used in movie overviews made it difficult to establish meaningful correlations. When text embeddings were created - the mathematical representations of the text, the 'overview' feature failed to provide useful signals, resulting in limited impact on the model's predictions.

Due to this reason, we decided to add another feature of free-text data that will be more helpful for the model's predictions.
After searching for more data, we chose the "48,000+ movies dataset" also from Kaggle.com website and combined this dataset with our existing one, integrating its review column to improve our feature set and provide more valuable free-text feature. This combination, conducted a new table with the same features as before but added a new feature:

- review: User or critic reviews of the movie (containing 5,583 non-null entries)

The review column was much more helpful. Reviews are full of opinions and feelings, like whether people loved or hated a movie. These emotions are often tied to how well a movie does financially. In the advanced models the embeddings captured this sentiment and provided strong connections to the revenue data. This made a noticeable difference in improving the model's accuracy.
In the end, we focused on review text instead of the overview, as reviews provided deeper insights and proved to be more effective for predicting movie revenues. To ensure more accurate results, we chose to use only the rows where the review column is not null. This decision allowed us to focus on the more informative entries, which led to better results, as the reviews offered valuable insights and stronger correlations with movie revenue. The final dataset used was a combination of the two datasets, including all 5,583 rows - a significant amount of data for building and evaluating the models' effectiveness and accuracy.
The dataset contains a wide range of values in the revenue column, indicating significant variability in the data. This variability means the data may contain extreme outliers, which can cause biased analyses and impact the model's performance. To focus on the central range of the data and to be able to identify the trends more accurately, we apply the IQR method to remove extreme outliers, ensuring that the analysis is driven by the majority of the data rather than being disproportionately influenced by a few extreme values.
The IQR (Interquartile Range) method is an approach used to detect and handle outliers in a dataset. It works by focusing on the middle 50% of the data, which is considered the most representative of the dataset. Applying this approach doesn't change the data a lot and removes a relatively small amount of data. It provides a clearer picture of trends and patterns with more accurate results. To keep consistency between the models this method was applied in the beginning of each model, this way all the models work on the same data set.

```python
# Prepare target variable and remove outliers
print(f'Number of lines before applying IQR method: {len(df)}')
y = df['revenue']
Q1 = y.quantile(0.25)
Q3 = y.quantile(0.75)
IQR = Q3 - Q1
mask = (y >= (Q1 - 1.5 * IQR)) & (y <= (Q3 + 1.5 * IQR))
df_cleaned = df[mask]
y = df_cleaned['revenue']
X = df_cleaned.drop('revenue', axis=1)
print(f'Number of lines after applying IQR method: {len(df_cleaned)}')
```

The code snippet where the IQR method occurs

```
Number of lines before applying IQR method: 5582
Number of lines after applying IQR method: 5182
```

A screenshot indicating the number of rows left after IQR method is applied

All the models in the project use the same core features for prediction, ensuring consistency across the models. Additionally, by maintaining uniformity in the features, it becomes easier to observe the improvements between the models based on the

preprocessing techniques which differ according to the complexity of the model being used.

Model 1- Baseline Model: Predicting with the Average Revenue

In this baseline model, we aim to create the simplest possible prediction mechanism for estimating movie revenues. This model serves as a starting point for comparison against more complex models we will talk about later in the article.

Data Preparation:
The dataset is loaded from a CSV file (imdb_movies.csv) containing information about movies, including their revenue. To ensure data integrity, rows with missing revenue values are removed, as the target variable (revenue) is critical for both training and evaluation. It was not necessary to process all the features because this model only used the revenue column for the prediction.

Train-Test Split:

- Training Set: 80% -Used to compute the baseline prediction.
- Test Set: 20% -Used to evaluate the model's performance.

Model Architecture:

The core idea of the baseline prediction in this model is to provide a simple, straightforward approach to predict the movie revenue using a constant value - specifically, the average revenue from the training dataset. This strategy doesn't rely on any features of individual movies assuming that the most reasonable prediction for any movie is simply the average revenue observed in the training data.

Evaluation:

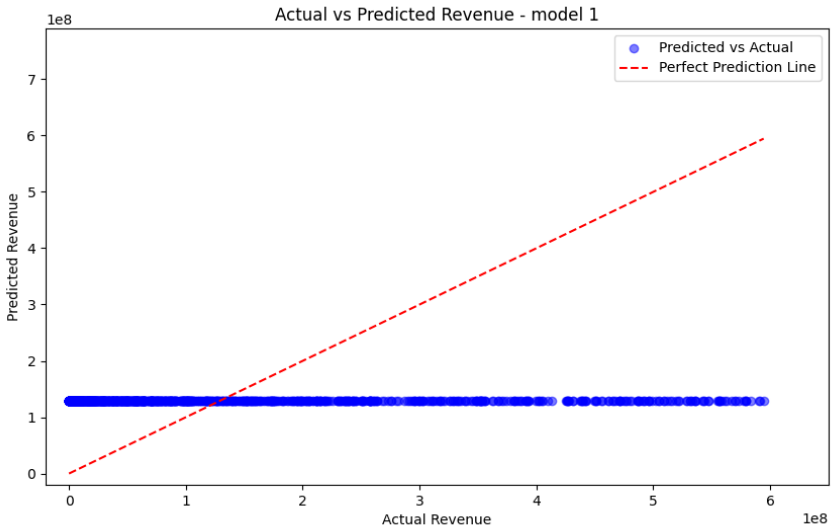
The metrics computed for the model evaluation are the MAE, MSE and RMSE. The results obtained from running the code are:

Model 1 - Baseline	
Mean Absolute Error (MAE)	$ 121,433,652.00
Mean Squared Error (MSE)	$ 23,676,641,432,614,836.00
Root Mean Squared Error (RMSE)	$ 153,872,159.38

In summary, the baseline model offers a simple yet essential benchmark by predicting the average revenue for all movies, against which more sophisticated models can be compared. While computationally simple, the baseline model is limited by its inability to adapt to movie-specific characteristics.
More advanced models can leverage additional features and algorithms to enhance predictive accuracy and capture the complex factors influencing movie revenue. The next models show the improvement in the prediction of the revenue as the models themselves improve from one to the other.

```
Average Revenue is $129,338,439.36
Evaluation Metrics:
Mean Absolute Error (MAE):        $121,433,652.00
Mean Squared Error (MSE):         $23,676,641,432,614,836.00
Root Mean Squared Error (RMSE):   $153,872,159.38
```

A screenshot of the evaluation metrics of Model 1



The graph of actual revenue Vs. predicted revenue of Model 1

Model 2- Regression Model: Predicting based on linear regression

This model improves upon the baseline by using linear regression to predict the revenue. Here, the model integrates multiple feature types such as text, numerical, categorical, and date data. The goal is to predict movie revenues more accurately by considering a diverse range of factors that could influence revenue generation and not only by using the average of the train data's revenues.

Data Preparation:
The dataset is cleaned and processed to handle missing values, outliers, and various feature types:

- Text Features: Text columns (e.g., movie names, reviews, crew details) are cleaned, combined, and converted into numerical features using TF-IDF vectorization with 1- and 2-grams to capture patterns.
- Numerical Features: Features like 'score' and 'budget' are imputed for missing values and normalized using a Power Transformer.
- Categorical Features: Categorical data (e.g., genre, country) is one-hot encoded, with missing values replaced by 'missing.'
- Date Feature: Release dates are split into components (day, month, year, weekday) with missing values filled using the median value.
- Outliers in the target variable (revenue) are removed using the IQR method, ensuring a cleaner dataset for training.

Train-Test Split:

- Training Set: 80% - Used to compute the linear regression prediction.
- Test Set: 20% - Used to evaluate the model's performance.

Model Architecture:

The model is a linear regression modified to ensure non-negative predictions, as movie revenues cannot be negative. This ensures more realistic predictions. Traditional linear regression can produce negative values, so the predictions are adjusted using 'np.maximum(predictions, 0)'. This function compares each predicted value with 0 and returns the maximum value, effectively replacing any negative predictions with 0.
All preprocessing steps are integrated into a single pipeline, which allows the model to handle transformations automatically during training and prediction, ensuring consistency and ease of use.

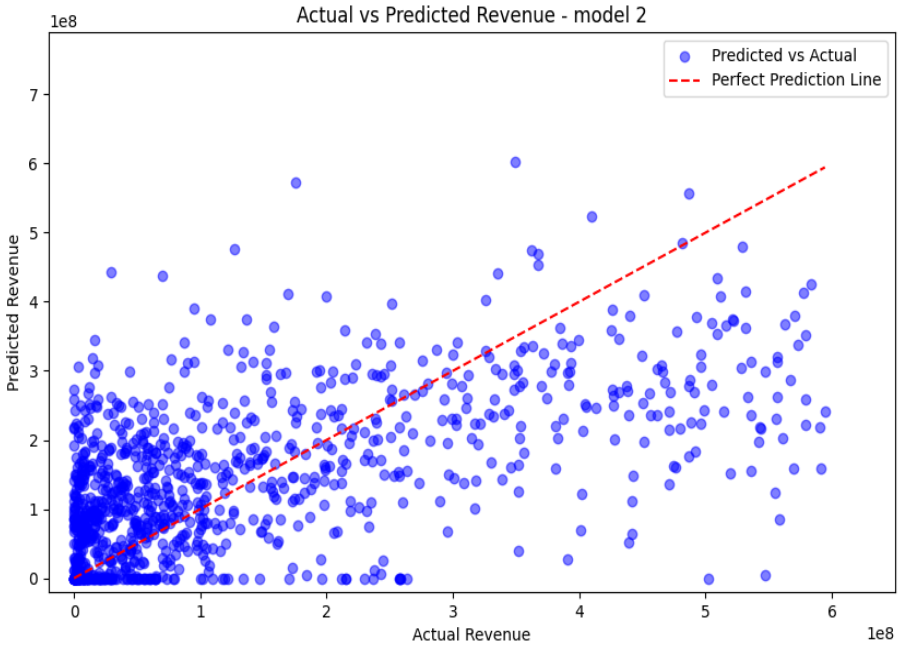
Evaluation:

The linear regression is applied to predict the revenues. The results obtained:

Model 2 – Linear Regression	
Mean Absolute Error (MAE)	$ 92,968,624.58
Mean Squared Error (MSE)	$ 16,097,876,478,672,158.00
Root Mean Squared Error (RMSE)	$ 126,877,407.28

```
Evaluation Metrics:
Mean Absolute Error (MAE):        $92,968,624.58
Mean Squared Error (MSE):         $16,097,876,478,672,158.00
Root Mean Squared Error (RMSE):   $126,877,407.28
```

A screenshot of the evaluation metrics of Model 2



The graph of actual revenue Vs. predicted revenue of Model 2

Elaboration about the NLP in this model:

Text Features: movie names, reviews, and crew details
These features are processed using TfidfVectorizer, transforming text into numerical features based on TF-IDF. Key parameters include:
max_features = 500: Limits vocabulary to the top 500 words/n-grams, optimizing dimensionality and performance. The number 500 was picked after trying different variations for the max_features amount and this gave the best results.
ngram_range = (1, 2): Captures unigrams and bigrams for better context and word significance.
min_df = 2: Excludes rare words/n-grams appearing in fewer than 2 documents.
max_df = 0.95: Filters overly common words/n-grams in more than 95% of documents.

This TF-IDF process converts each movie's text into structured numerical vectors (of the size of the TF-IDF vocabulary) enabling the model to extract insights from textual features.

Categorical Features: genre, language, country, status
Categorical data refers to features that represent discrete categories rather than continuous values. In this model these features are encoded using One-Hot Encoding converting categories into a numerical format usable by the model.

By combining more advanced text processing and categorical encoding and using all the features, this model beats the baseline, handling diverse features effectively for more accurate revenue predictions.

Model 3- Basic Fully Connected Neural Network

The third model is a Basic Fully Connected Neural Network designed to predict movie revenues using textual, categorical, and numerical features. It builds on the regression model, using the advanced capabilities of neural networks to improve predictive accuracy.

Data Preparation:

The preprocessing of the data in this model was mostly similar to the second model's preprocessing provided:

- Date Features: Release date was converted into year, month, day, and weekday components, with missing values filled using the median.
- Numerical Features: Missing values in budget_x and score were imputed with the median, and all numerical features were normalized using StandardScaler.
- Categorical Features: Rare categories (less than 1% frequency) in columns like country and orig_lang were grouped as "Other," and the other features were encoded by one-hot encoding. Multi-label binarization (where each sample can belong to multiple categories simultaneously) was applied to the genre column to convert genres into a multi-hot encoded format.
- Text Features: The textual features like review and names were vectorized using TF-IDF with a maximum of 500 features.

- Outliers in the revenue data were removed using the IQR method and the revenue was normalized using 'StandardScaler' to ensure the target variable was on the same scale as the features.

Normalization of features and target variables is crucial because neural networks are sensitive to input and output scales. Without normalization, discrepancies between feature and target scales can lead to inaccurate predictions, slower training, or instability. Ensuring similar scales allows the model to learn relationships more effectively and enhances training efficiency and stability.

Train-Test Split:

- Training Set: 70%
- Validation: 15%
- Test Set: 15% - Used to evaluate the model's performance.

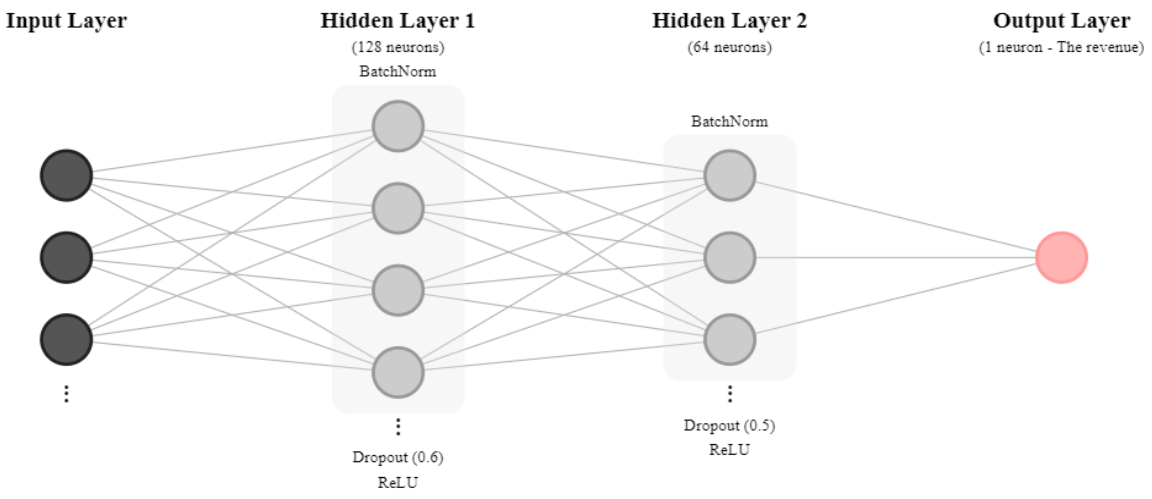
Model Architecture:

The neural network consists of:

1. Input Layer: Accepts the normalized feature vector (input size matches the number of features).
2. Hidden Layers:
    - First layer: 128 neurons with Batch Normalization and Dropout (60%).
    - Second layer: 64 neurons with Batch Normalization and Dropout (50%).
3. Output Layer: A single neuron for revenue prediction.
4. Activation Function: ReLU was used in the hidden layers to introduce non-linearity.
5. Initialization: Xavier initialization was applied to the weights for better convergence.

Batch Normalization normalizes layer activations during training, ensuring stable input distributions and improving gradient flow. This leads to faster, more stable learning and helps the model handle diverse feature distributions.
Dropout randomly deactivates neurons (60% in the first layer, 50% in the second), encouraging the network to learn more generalized and robust representations (forcing the network to not rely on specific neurons) while reducing overfitting.
Xavier initialization ensures weights are scaled to maintain uniform variance of activations across layers, preventing exploding or vanishing gradients and enabling effective learning from the first epoch.

Together, Batch Normalization, Dropout, and Xavier initialization create a stable, efficient, and robust architecture. The ReLU activation ties these components together by introducing the non-linear transformations necessary for the model to capture complex patterns. These elements together enable the network to learn effectively and make accurate predictions on the dataset.



An illustration of the Neural Network of Model 3

Training Process:

The model's training process incorporates several techniques to optimize learning, improve generalization, and avoid overfitting:

- Optimizer: The Adam optimizer is used for its adaptive learning rate properties, adjusting step sizes based on past gradients for efficient training.
- Learning Rate Scheduler: A 'ReduceLROnPlateau' scheduler monitors the validation loss and reduces the learning rate when it stops improving. This prevents the model from missing the optimal solution and ensures it continues to learn effectively as it gets closer to convergence.
- Early Stopping: Training stops if validation loss doesn't improve for 20 epochs, preventing overfitting and avoiding learning noise.
- Batch Training: Mini-batches (size = 32) balance memory efficiency and convergence speed, enabling techniques like Batch Normalization for stable learning.
- Gradient Clipping: To handle the risk of exploding gradients, gradient clipping is applied, ensuring that gradients remain within a predefined range.
- Validation: A separate validation set checks performance, guiding hyperparameter adjustments and determining training duration.
- Loss Function: The Mean Squared Error (MSE) loss function is used because it emphasizes larger errors. The model is trained to minimize this loss by iteratively adjusting weights during the training.

```
# Set up optimizer, loss function, and scheduler
optimizer = torch.optim.Adam(model.parameters(), lr=0.001, weight_decay=1e-2)
torch.nn.utils.clip_grad_norm_(model.parameters(), max_norm=1.0)
criterion = nn.MSELoss()
scheduler = optim.lr_scheduler.ReduceLROnPlateau(optimizer, mode: 'min', patience=5, factor=0.1)
```

The code snippet where the optimizer, gradient clipping, the loss function and the learning rate scheduler are defined.

The training process is designed to optimize the model's learning efficiency and generalization capabilities. Adam provides adaptive updates, the scheduler fine-tunes training, and early stopping, gradient clipping, and Batch Normalization ensure stability. Together, they enable the model to converge effectively and deliver accurate predictions.

Evaluation:

The model was evaluated using Mean Absolute Error (MAE), Mean Squared Error (MSE) and Root Mean Squared Error (RMSE). The model triggered the early stopping mechanism and stopped after 69 out of 150 epochs.

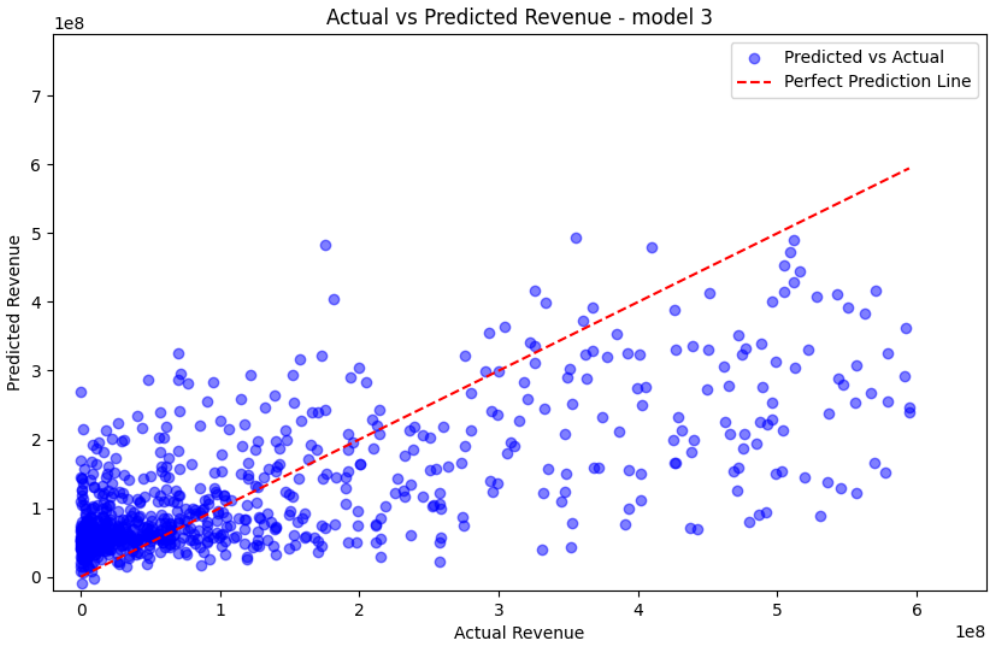
Model 3 – Simple Neural Network	
Mean Absolute Error (MAE)	$ 81,014,284.61
Mean Squared Error (MSE)	$ 13,285,544,772,123,966.00
Root Mean Squared Error (RMSE)	$ 115,262,937.55

This model introduces several key improvements over Model 2, primarily focused on enhancements in the architecture and training strategy, rather than data preparation. These architectural and training improvements provide significant advantages over Model 2, enabling the network to better capture the underlying patterns in the data and make more accurate predictions.

```
Evaluation Metrics:
Mean Absolute Error (MAE):        $81,014,284.61
Mean Squared Error (MSE):         $13,285,544,772,123,966.00
Root Mean Squared Error (RMSE):   $115,262,937.55
```

A screenshot of the evaluation metrics of Model 3

The graph of training and validation loss of Model 3 over the 150 epochs with early stopping



The graph of actual revenue Vs. predicted revenue of Model 3

Elaboration about the NLP in this model:

To ensure consistency and highlight improvements through architecture changes, the preprocessing for this model remains almost identical to the previous one.

Text Features: movie names, reviews, and crew details
Text data is processed using TfidfVectorizer (as in Model 2). Two distinct vectorizers are applied to crew and review columns to emphasize their unique information.

The key parameters are unchanged:
max_features = 500, ngram_range = (1, 2), min_df = 2, max_df = 0.95.

Categorical Features: genre, language, country, status
One-hot encoding is used for country and original language. Rare categories (occurring in less than 1% of data) are grouped as "Other."
The status feature is label-encoded for numerical representation.
The genre feature is multi-hot encoded, capturing all genres associated with a movie, preserving its inherent multi-class nature.

This model achieved more accurate revenue predictions by combining advanced preprocessing techniques with a neural network specifically designed to effectively handle the dataset's complexity, all while maintaining a preprocessing approach mostly consistent with the previous model.

Model 4 - Advanced Neural Network Using Bert

Model 4 represents the most advanced approach in this project, using a deep neural network architecture enhanced with BERT embeddings for textual data. This model was also designed to integrate a variety of features - textual, numerical, and categorical - while addressing challenges such as overfitting and effective feature utilization.

Data Preparation:

To ensure consistency across models, outliers in the revenue data were systematically removed using the IQR method.
Numerical, Categorical and Date Features were treated similarly to the previous model. For the Numerical features, budget and score, the missing values were filled with the median of the relevant column. The Categorical features like language and country were again processed to group rare categories into an "Other" category and One-hot encoding was applied on them. Finally, the date features were converted into multiple numerical features: year, month, day, and weekday as done in the previous models.

Text features: Unlike the previous models that utilized traditional TF-IDF vectorization, Model 4 used BERT embeddings to capture semantic and contextual meanings from the textual data. This approach allowed the model to get deeper insights from text, which proved to be very helpful for enhancing prediction accuracy.
Key textual features processed in this model were movie names, crew details, genres and most importantly the reviews. A detailed explanation of how these features were processed will be provided later in this section of the report.

Train-Test Split:

- Training Set: 70%
- Validation: 15%
- Test Set: 15% of the data, reserved for evaluating final model performance.

Model Architecture:

The architecture of this advanced neural network is a significant improvement in complexity and capability compared to previous models, incorporating sophisticated design principles to enhance performance and generalization.

- Layers and Connections: The model consists of five fully connected layers with progressively smaller sizes (1024 → 512 → 256 → 128 → 1). Each layer connects not only to the next but also to non-adjacent layers, improving information flow and reusing key features learned earlier.
- Regularization techniques: To prevent overfitting and steady training, the following techniques were implemented:
    - Layer Normalization is applied after each layer to stabilize gradients and ensure consistent data distribution.
    - Activation Function: The GELU (Gaussian Error Linear Unit) activation function is used in this model. Unlike ReLU, which applies a hard cutoff at zero, GELU softly activates neurons based on the input value's proximity to zero, balancing linearity and non-linearity. This makes it particularly effective for text data,

enhancing the model's ability to process BERT embeddings and capture complex relationships, ultimately improving predictive performance.
    - Drop out: A 30% rate randomly deactivates neurons during training, promoting generalized learning and reducing overfitting.
    - Orthogonal Initialization: Weights were initialized using an orthogonal strategy, ensuring that gradient flow remains stable and balanced. This way it avoids issues like exploding or vanishing gradients, which can hold back training efficiency. It ensures that each layer contributes unique, independent information to the learning process, ultimately improving the model's predictive performance.
- Optimizations: The model was trained using advanced optimization strategies to provide better performance and convergence:
    - Adam Optimizer: Provides adaptive learning rate adjustments with weight decay to prevent overfitting.
    - Cosine Annealing Scheduler: Dynamically adjusts the learning rate, enabling faster initial learning and fine-tuning later.
    - Gradient Clipping: Keeps gradients within a certain predefined range, ensuring stability during training.

Training Process:

Training this advanced neural network required a combination of modern techniques to optimize learning, generalization, and convergence, some of which were also used in the previous model.

- Early Stopping: Training stopped if validation loss failed to improve for 20 epochs, preventing overfitting and avoiding learning noise.
- Batch Training: Data was processed in mini-batches of size 32, balancing memory efficiency and training speed. After testing batch sizes (32, 64, 128), 32 was found to be the optimal one.
- Training Epochs: The model was set for 150 epochs, with early stopping ensuring efficient resource use by stopping the training when validation performance stabilized.
- Dynamic Learning Rate Adjustment: The 'cosine annealing scheduler' reduced the learning rate as validation loss improved, allowing fine-tuning in later training stages.
- Loss Functions: A hybrid loss function was used to balance sensitivity to both small and large prediction errors: Mean Squared Error that ensures that the model prioritizes minimizing large errors, which have a higher impact on revenue predictions and Huber Loss which provides robustness to outliers by penalizing large deviations less aggressively than MSE.

Evaluation:

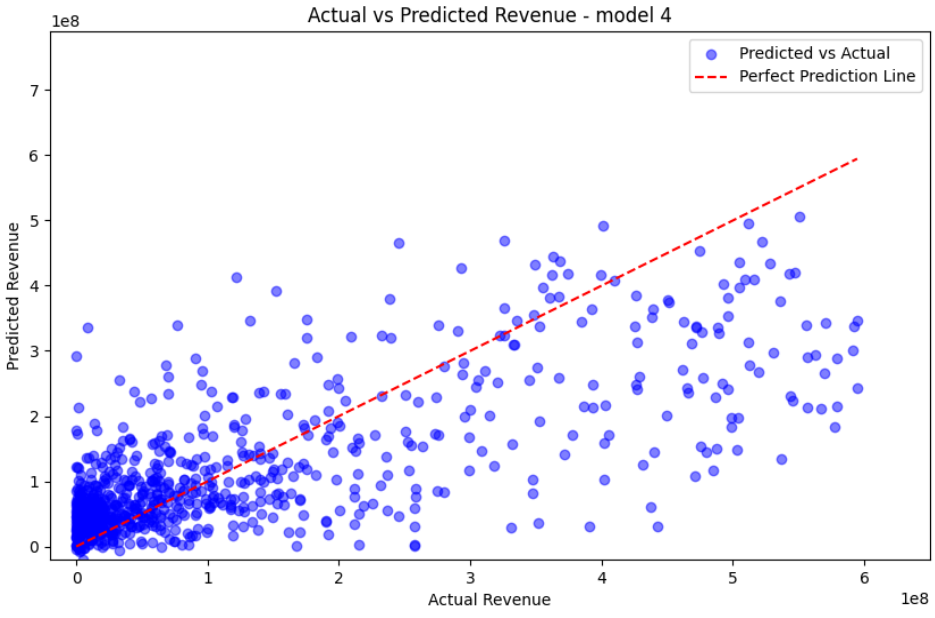
The model's performance was evaluated using the same metrics applied to previous models. The model triggered the early stopping mechanism and stopped after 142 out of 150 epochs.

Model 4 – Advanced Neural Network
Mean Absolute Error (MAE)
Mean Squared Error (MSE)
Root Mean Squared Error (RMSE)

```
Evaluation Metrics:
Mean Absolute Error (MAE):        $73,535,891.10
Mean Squared Error (MSE):         $11,611,688,000,320,616.00
Root Mean Squared Error (RMSE):   $107,757,542.66
```

A screenshot of the evaluation metrics of Model 4

The graph of training and validation loss of Model 4 over the 150 epochs with early stopping



The graph of actual revenue Vs. predicted revenue of Model 4

Elaboration about the NLP in this model:

This model enhances text processing by replacing TF-IDF vectorization with BERT embeddings, which capture rich semantic and contextual relationships in the text, enabling the model to:

- Understand sentiments in reviews correlated with movie revenues.
- Incorporate contextual information about movie names, genres, and crew, providing deeper predictive signals. The use of BERT ensured that the model found and used deeper, context-aware representations, indicating a significant improvement in handling textual data.

Text Features: the key textual features contained:

1. **Reviews:** Reviews provide sentiment and opinion data strongly correlated with movie revenue. Positive reviews often indicate strong audience reception, driving financial success. BERT embeddings enable the model to capture nuanced sentiments (positive, neutral, or negative) and complex patterns like themes of criticism or compliments.
2. **Movie Names:** Capturing key information about branding and recognition that could influence the revenue.
3. **Crew:** Focuses on key contributors like directors or lead actors, limiting processing to the top ten significant contributors to capture the most critical information.
4. **Genre**: Genres were processed to provide semantic embeddings that capture relationships between themes (e.g., action, comedy, drama). For example, genre popularity trends might influence revenue. While genres are short phrases (e.g., Action, Drama), BERT embeddings capture relationships and similarities between genres. It finds the needed connections based on these patterns.

How BERT works? There are three main parts in this model's processing pipeline:

1. Bidirectional Contextual Understanding: BERT reads the entire input sequence at once, considering the context of each word both before and after it. This bidirectional approach helps BERT interpret words accurately based on their context, (for example, "bat" like the animal or like what is used in baseball).
2. Tokenization: BERT tokenizes text into subword units, ensuring even rare or unknown words are meaningfully represented. For example, "unbelievably" is tokenized as ["un", "##believ", "##ably"], enabling effective handling of complex or unfamiliar words.
3. Embedding Generation: Each token is mapped to a high-dimensional vector, which encodes the Semantic meaning (e.g., similarity between tokens) and the Positional information (where the token appears in the sequence). These embeddings capture both the word's meaning and its contextual usage which help us to get the most accurate and nuanced representations of the text.

The output of BERT depends on how it is used. In this model the Pooled Output (CLS Token) output version is used. The [CLS] token ("classification token") is a special token added to the beginning of every input sequence in BERT. It provides a "summary" representation for the entire sequence, thus, very helpful for this model that makes predictions based on the overall meaning of the input. This output summarizes the

entire input sequence. After getting the representation, it is fed into the subsequent neural network layers to incorporate textual information into the predictive model.

Examples to how BERT works for the features in this Model:

For the “review” feature:
the phrase "a breathtaking masterpiece" would be encoded with strong positive sentiment, while "a complete disaster" would convey negativity.

For the “name” feature:
"The Avengers" indicates a superhero action movie, potentially associated with high revenue. Also, movie series can be identified, which will provide additional information based on the name that will cause closeness in the embeddings.

For the “crew” feature:
The presence of well-known directors (Christopher Nolan) or actors (Leonardo DiCaprio). Or even just associations between names and movie success based on patterns learned from the dataset during the training.

For the “genre” feature:
Action and Adventure may appear closer in embedding space than Drama and Horror. In addition, combined genres like Comedy and Romance are treated contextually, allowing BERT to understand genre combinations better than one-hot encoding.

Categorical Features:
Categorical variables like language, status, and country were transformed using one-hot encoding, representing each category as a binary vector, similar to Model 3. One-hot encoding was chosen over BERT embeddings for these features due to its simplicity, efficiency, and adequacy for representing this type of data meaningfully, given the model's requirements and the nature of the data.

To summarize this part, this model leverages BERT embeddings to process textual data like reviews, movie names, genres, and crew details, enabling deeper understanding of sentiments and contextual relationships that TF-IDF cannot capture. While complex text features were processed using BERT, simpler categorical features used one-hot encoding for efficiency. This combination of BERT embeddings, advanced neural architecture, and robust training strategies resulted in the model achieving much better performance compared to the previous approaches.

Results:

FINAL RESULTS	
Mean Absolute Error (MAE)	
Model 1 – Baseline	$ 121,433,652.96
Model 2 – Linear Regression	$ 92,968,624.58
Model 3 – Simple Neural Network	$ 81,014,284.61
Model 4 – Advanced Neural Network	$ 73,535,891.10

Mean Squared Error (MSE)	
Model 1 – Baseline	$ 23,676,641,432,614,836.00
Model 2 – Linear Regression	$ 16,097,876,478,672,158.00
Model 3 – Simple Neural Network	$ 13,285,544,772,123,966.00
Model 4 – Advanced Neural Network	$ 11,611,688,000,320,616.00

Root Mean Squared Error (RMSE)	
Model 1 – Baseline	$ 153,872,159.38
Model 2 – Linear Regression	$ 126,877,407.28
Model 3 – Simple Neural Network	$ 115,262,937.55
Model 4 – Advanced Neural Network	$ 107,757,542.66

From the baseline to the final model, the improvement was:

- A 39.44% reduction in Mean Absolute Error
- A 50.96% reduction in Mean Squared Error
- A 29.97% reduction in Root Mean Squared Error

Additionally, looking at the graphs with the blue dots presenting the 'actual revenue Vs. predicted revenue' for each model, the points are getting closer and closer to the red line that symbolizes the optimal situation.

Discussion :

The results of our project show that increasing model complexity and using more advanced NLP techniques significantly improved predictive accuracy. While the baseline model provided a simple reference, it struggled to utilize the data effectively. The regression model, with structured feature processing and text-based insights, achieved better results.

Neural network models further enhanced accuracy by capturing complex relationships within the data and the use of BERT embeddings in the advanced neural network demonstrated the power NLP techniques, enabling deeper semantic and contextual insights from textual features like reviews, genres etc.

Throughout the project, we faced unexpected results, such as overfitting and varying levels of feature influence. This required experimentation, including:

- Replacing the less effective 'overview' feature with the more impactful 'review' feature.
- Running various TF-IDF parameters to identify the optimal settings for processing textual data.
- Adjusting batch sizes and the number of layers in the neural networks to improve performance.
- Tuning dropout percentages to address overfitting.

Our hypothesis that more complex models would yield better results was supported.

Future research directions could further enhance this work by incorporating more features like, for instance, temporal trends and seasonal patterns in movie releases, exploring other transformer-based language models other than BERT and maybe developing methods that combine multiple model architectures. These extensions would provide additional depth to the current analysis while possibly finding new insights into movie revenue prediction.

Conclusions:

This project successfully demonstrated the impact of increasing model complexity and advanced NLP techniques on predictive performance, validating our initial hypothesis. The iterative adjustments and data-driven refinements we made were essential to overcoming challenges and achieving meaningful results.

Data:
https://www.kaggle.com/datasets/ashpalsingh1525/imdb-movies-dataset/data
https://www.kaggle.com/datasets/yashgupta24/48000-movies-dataset
Git:
https://github.com/NoaFishman/NLP_Project.git