



Module  
**BDM- 3014 -Introduction to Artificial Intelligence  
(DSMM Group 3)**

**A Report on Anime Recommendation System**

Submitted By

**Anil Poudel**

**Bipin Pandey**

**Koshish Aryal**

**Punam Bhattarai**

**Shweta Laljibhai Thummar**

**Suresh Mahat**

**Uttam Tamang**

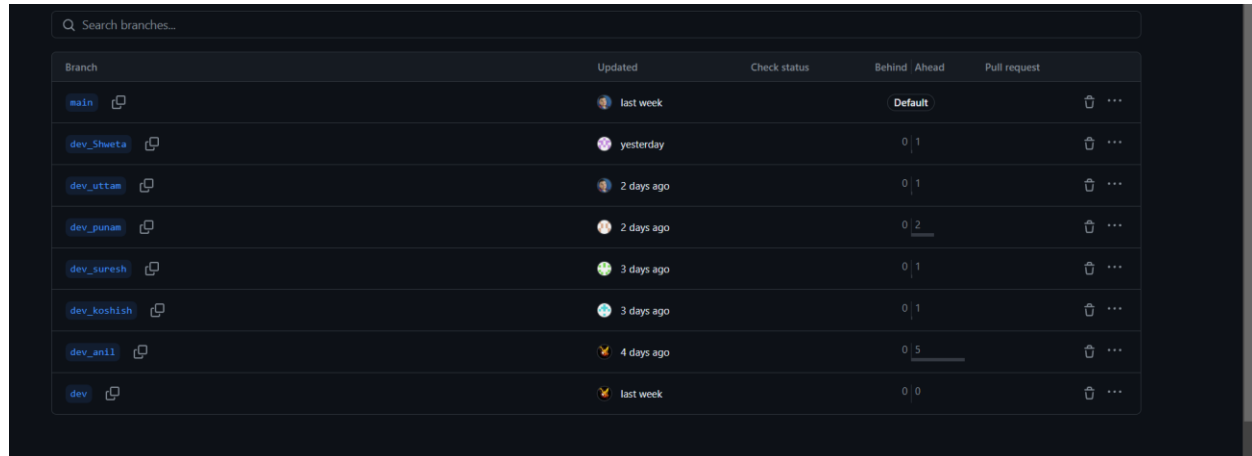
# Introduction

The purpose of the project is to develop a robust system that could help anime viewers to choose the anime wisely. It helps to recommend a anime to the user based on popularity, ranking, score and episodes and so on based on the sentiment and the user preferences. Here, users will able to choose their requirement based on their needs.

The project is maintained in GitHub with all the team members as collaborators. Here, all individuals will make the changes in the their code based on their part of job and commit the task in their respective branch which will later joined on the main branch. Each pull request are reviewed by another member and verified and then approved.

## Git Branch Management

To assure quality code and maintainability, we used a simple gitflow. During the initial phase, two branches are created: "main" and "dev." The “feature” branch contains new features that are added and integrated for the project, and it is later merged with the develop branch where each individual worked on. Release branch is created when we are ready to test our product and merged later into main and develop after performing fixes and testing. Furthermore, we use git best practices for meaningful commit messages, proper naming standards, and semantic versioning for tags and models.

A screenshot of the GitHub repository's 'Branches' page. At the top is a search bar labeled 'Search branches...'. Below it is a table with columns: 'Branch', 'Updated', 'Check status', 'Behind / Ahead', and 'Pull request'. The table lists several branches: 'main' (Default, updated last week), 'dev\_ShwetA' (updated yesterday, 0 | 1 ahead), 'dev\_uttam' (updated 2 days ago, 0 | 1 ahead), 'dev\_punam' (updated 2 days ago, 0 | 2 ahead), 'dev\_suresh' (updated 3 days ago, 0 | 1 ahead), 'dev\_koshish' (updated 3 days ago, 0 | 1 ahead), 'dev\_anil' (updated 4 days ago, 0 | 5 ahead), and 'dev' (updated last week, 0 | 0 ahead). Each row has a trash icon and three dots for actions.

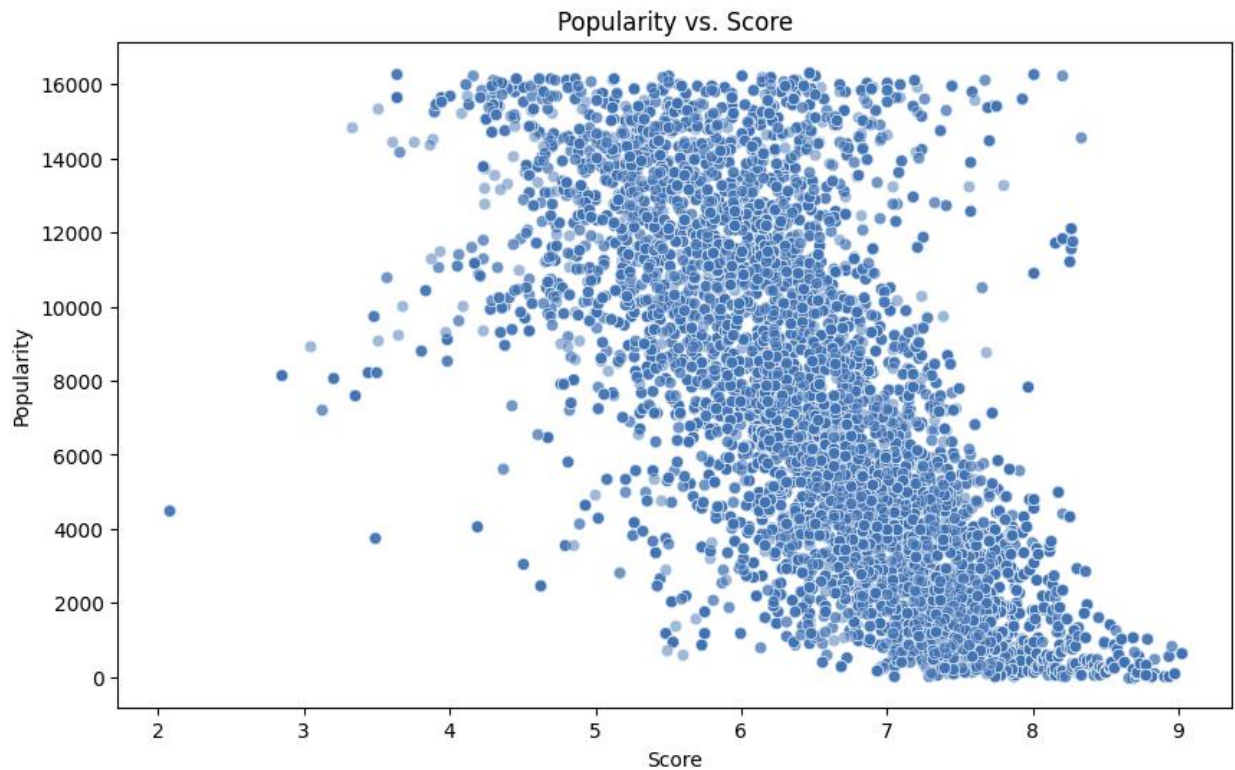
Branch	Updated	Check status	Behind / Ahead	Pull request
main	last week		Default	🗑️ ...
dev_ShwetA	yesterday		0   1	🗑️ ...
dev_uttam	2 days ago		0   1	🗑️ ...
dev_punam	2 days ago		0   2	🗑️ ...
dev_suresh	3 days ago		0   1	🗑️ ...
dev_koshish	3 days ago		0   1	🗑️ ...
dev_anil	4 days ago		0   5	🗑️ ...
dev	last week		0   0	🗑️ ...

## Branch Structure

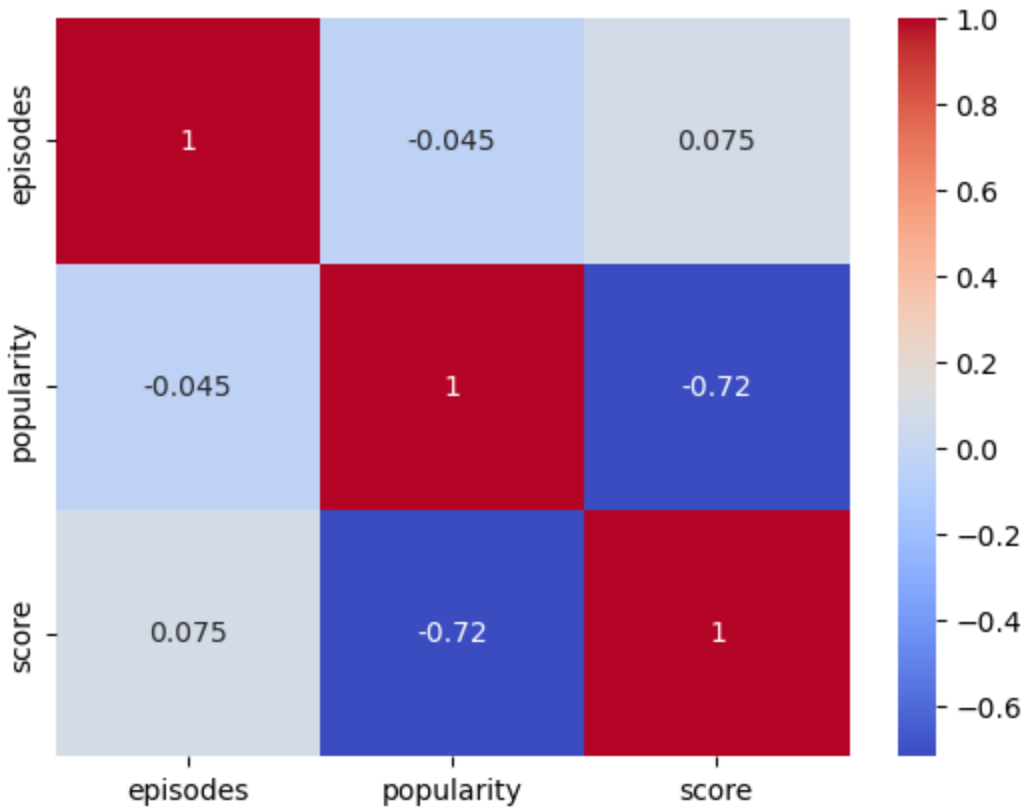
Identifying and collecting data relevant to the machine learning task was performed by using the data source Kaggle where the data was held. The data was visualized using charts like box plots,

histograms, and scatter charts to analyze the data. Checking the missing data and decide to use an appropriate handling strategy (e.g., imputation, removal). The data was cleaned using different metrics like removing the null values, and duplicate values, and filling the null value using valid metrics. The extreme values that might distort the model's learning were removed once it was found out. Once all the processing are done data is visualized using the possible visualizing mechanism and explained it uses.

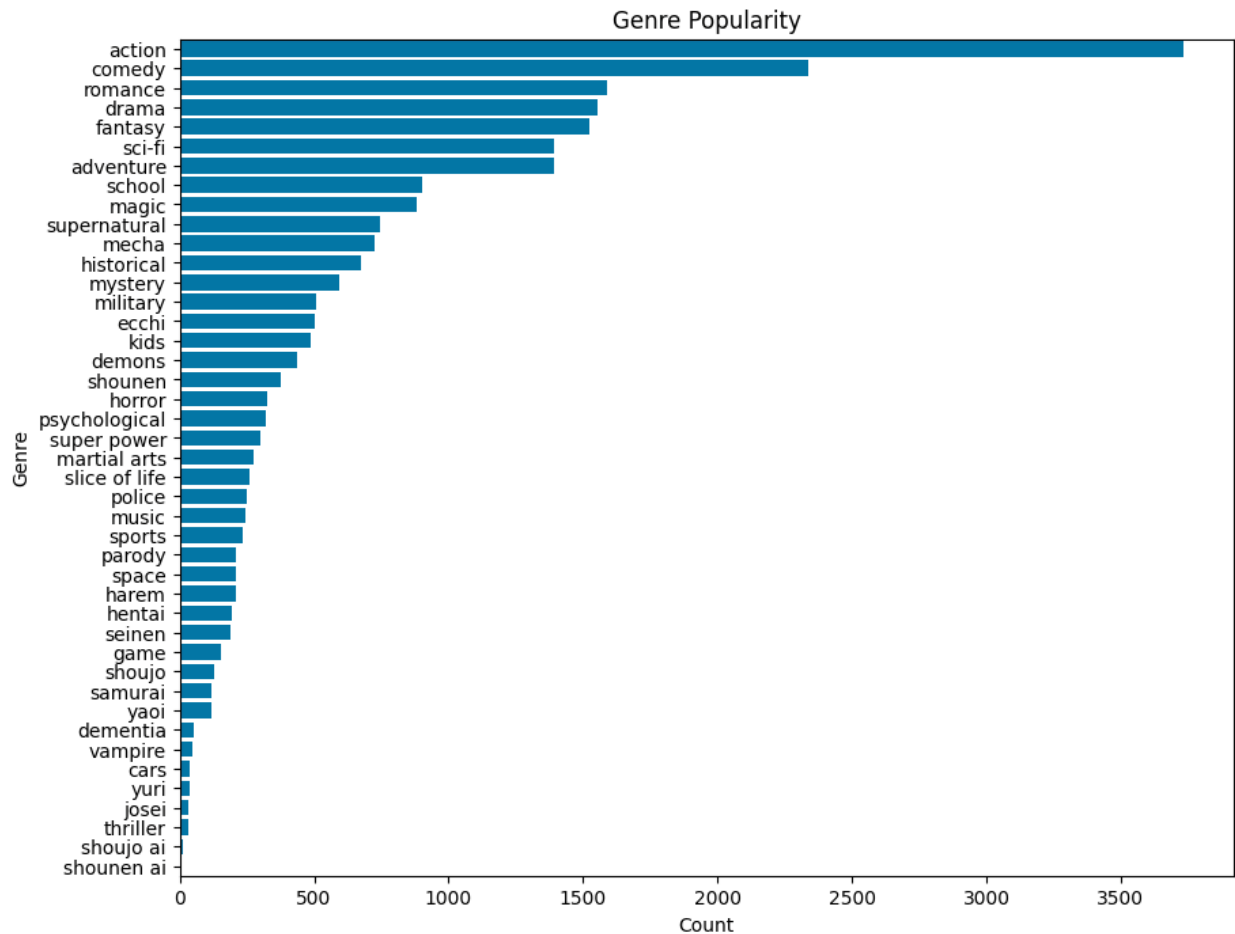
Some Visualization to understand the Data:



Popularity vs. Score: Scatter plots showcased a broad range of popularity scores across different anime, with no clear linear relationship between popularity scores and user ratings, suggesting diverse user preferences.



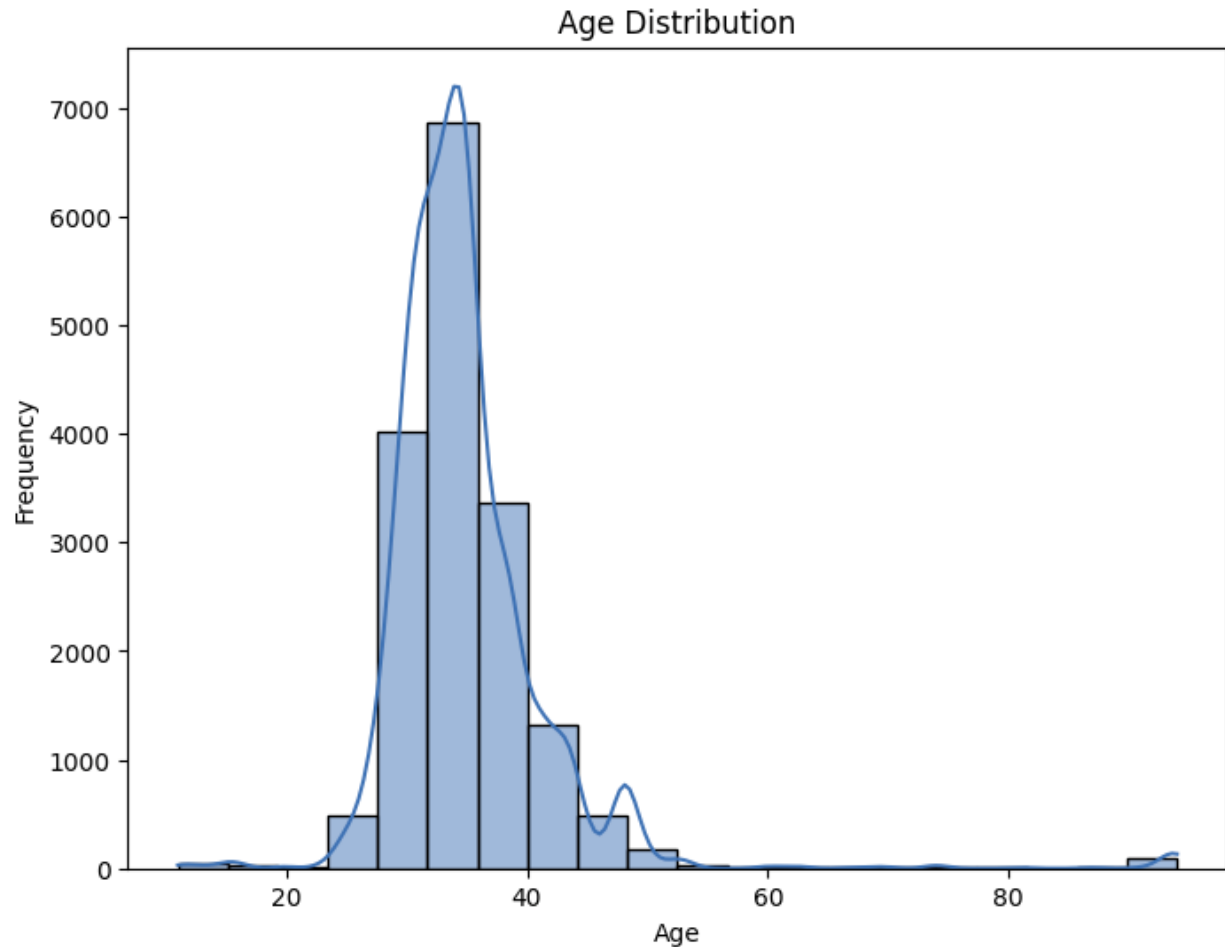
The **heatmap** shows the relationship between the score, popularity and episodes. This helped to see the correlation between three data and choose the which model to be used to train the data.



This **bar chart** is a visualization of the popularity of various genres. Some insights from the bar chart:

- **Top Genres:** Action, comedy, and romance are highly popular, indicating strong audience interest.
- **Moderate Popularity:** Genres like drama, fantasy, and sci-fi also have significant presence but are less dominant.
- **Less Popular Genres:** Niche genres such as yaoi, dementia, vampire, cars, and yuri are less favored within the dataset.
- **Diverse Interests:** The dataset showcases a wide range of genre preferences.
- **Focus Suggestions:** Content creators may benefit from focusing on popular genres to maximize audience reach.

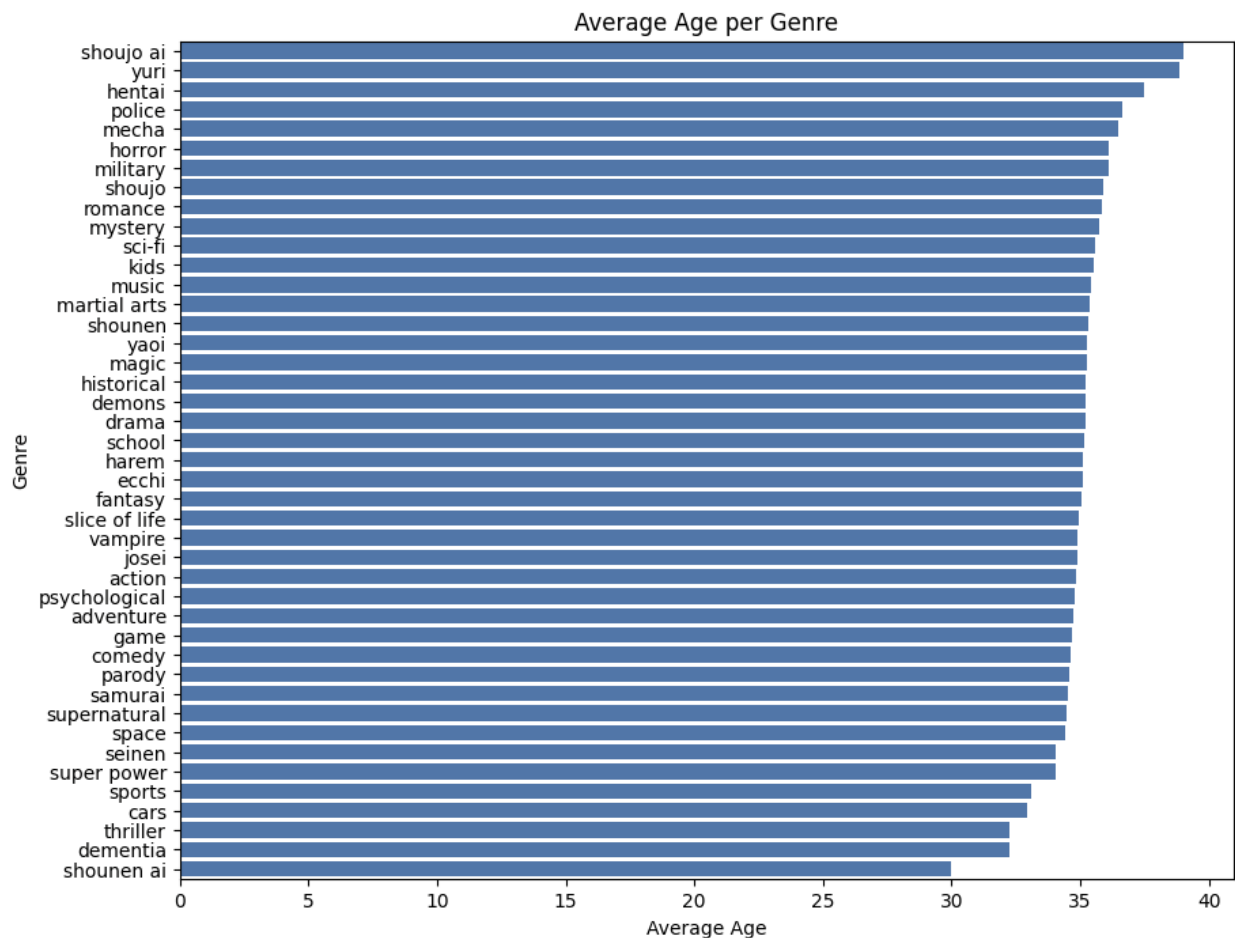
This visualization provides insights into genre preferences, aiding decisions related to content creation or marketing strategies.



The **histogram** provides the following insights:

- **Dominant Age Group:** The majority of age in the dataset are in their 20s, indicating a primary age group of anime viewers or characters.
- **Decline in Older Age Groups:** There is a noticeable decrease in frequency as age increases, with very few subjects in their 60s or older.
- **Distribution Characteristics:** The distribution is unimodal, with one main peak, and slightly right-skewed, suggesting a gradual decrease in frequency as age increases beyond the peak.
- **Concentration of Data:** There's a concentration of subjects around the peak, indicating a common age range in the dataset.
- **Kernel Density Estimate (KDE):** The KDE curve suggests a somewhat normal distribution with a slight right skew, reinforcing the prevalence of younger ages.

Overall, the histogram highlights a predominantly young demographic in the dataset, with fewer subjects as age increases, and a notable concentration in the 20s age range. It also shows niche genres that might attract smaller but dedicated groups of viewers.



This **bar chart** offers these insights:

- **Youthful Genres:** Genres like "shounen ai," "dementia," and "sports" attract younger audiences, with lower average ages.
- **Mature Audiences:** "Shoujo ai" and "yuri" appeal to older viewers, having higher average ages.
- **Broad Appeal:** Genres like "mystery" and "romance" draw in a wide range of ages, as seen from their balanced average ages.
- **Age Trends:** Generally, the average age rises from youth-oriented to adult-oriented genres, indicating different age preferences across genres.
- **Content Strategy:** This data can help creators target specific age groups, for instance, focusing on "shoujo ai" and "yuri" for older viewers and youth-centric genres for younger audiences.

This visualization aids in understanding the age demographics of genre preferences, assisting in content development and marketing strategies.

**PCA (Principal Component Analysis):**

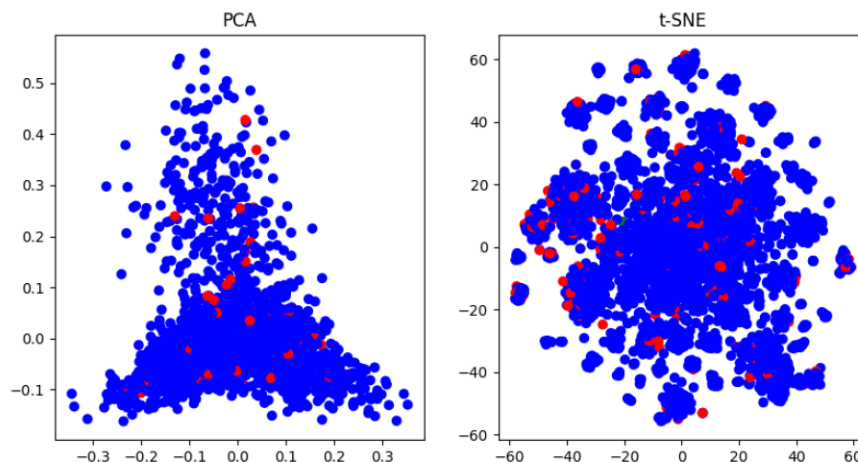


PCA is a linear dimensionality reduction technique that seeks to project high-dimensional data onto a lower-dimensional subspace while preserving the variance in the data as much as possible.

In below visualization, it displays the distribution of reviews in a lower-dimensional space (2D) obtained through PCA, with each point color-coded based on its sentiment score.

`plt.subplot(1, 2, 1)`: This line creates a subplot grid of 1 row and 2 columns, and selects the first subplot for the subsequent plot.

It generates a scatter plot where each point corresponds to a review. The x and y coordinates of each point are taken from the first and second principal components (`pca_result[:, 0]` and `pca_result[:, 1]`). The color of each point is determined by the sentiment score of the corresponding review, which is mapped to colors using a colormap (`color_map`).



### **tSNE (t-distributed Stochastic Neighbor Embedding):**

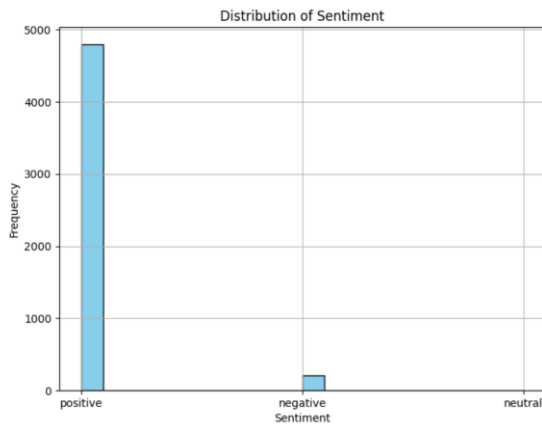
t-SNE is a non-linear dimensionality reduction technique that focuses on preserving the local structure of the data, particularly the relationships between nearby points in high-dimensional space.

In above visualization, it provides a two-dimensional representation of the data obtained through t-SNE, where each point represents a review, and the color of the points indicates their sentiment scores.

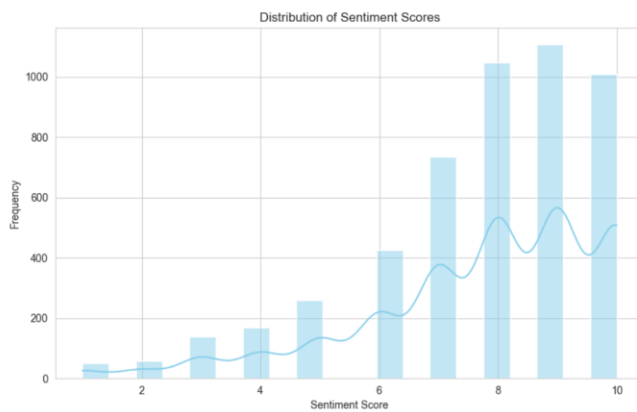
It generates a scatter plot where each point corresponds to a review. The x and y coordinates of each point are taken from the first and second components of the t-SNE result (`tsne_result[:, 0]` and `tsne_result[:, 1]`). The color of each point is determined by the sentiment score of the corresponding review, which is mapped to colors using a colormap (`color_map`).

### **Histogram :**

Below histogram provides an overview of how different sentiment categories are distributed across the reviews in the dataset, showing the frequency of each sentiment category.



Below histogram provides an overview of how sentiment scores are distributed across the reviews in the dataset, showing the frequency of different sentiment scores.



### Boxplot :

Below visualization provides a summary of the distribution of sentiment scores, including the median, quartiles, and any outliers present in the data.



## Approach 1:

### Pre Processing of the data for Sentiment Analysis:

For **sentiment analysis**, before computers can analyze text data, it needs some cleaning and organization. We start by fixing typos, slang, and abbreviations, and removing unnecessary things like punctuation and numbers. Converting everything to lowercase ensures consistency. Then, we break the text down into individual words. Finally, we remove common words that don't hold much meaning and simplify words to their root form to capture similar ideas. This preprocessing step helps us prepare the text data for analysis by making it cleaner, more organized, and easier for computers to work with.

To dive deeper into the meaning of the text by analyzing its sentiment. We use libraries like NLTK or TextBlob, which act like sentiment dictionaries, to understand if the text expresses positive, negative, or neutral emotions. Based on the overall sentiment of the words used, we assign a sentiment score (positive, negative, or neutral) to each review, summarizing the overall feeling of the text. This allows us to categorize reviews based on their emotional tone.

We have leverage techniques like TF-IDF vectorizer to capture the importance of words within the context of the reviews. This allows us to represent the text data numerically, making it suitable for further analysis. Furthermore, we used word embedding techniques like Word2Vec to represent each word as a vector in a high-dimensional space, capturing semantic relationships between words. Finally, dimensionality reduction techniques like PCA or t-SNE can be employed to visualize the data in a lower-dimensional space, helping us explore potential patterns or groupings within the reviews based on the extracted features. These techniques provide a richer understanding of the text data beyond just sentiment scores.

We build a text classification model to automatically predict sentiment based on the reviewed text. The model learns from a training dataset and is evaluated on a separate testing set using metrics like accuracy, precision, recall, and F1-score. These metrics help us understand how well the model captures sentiment and guide us in refining it for optimal performance.

Topic modeling techniques like Latent Dirichlet Allocation (LDA) can unveil hidden themes within the reviews. Visualizing these topics using word clouds or heatmaps provides a clearer understanding of the key themes discussed. Aspect analysis takes this a step further, allowing us to associate sentiment scores with specific topics. This helps identify which aspects of the reviewed subject are driving positive or negative sentiment. By following these steps, text analysis transforms raw data into meaningful insights.

This analysis aims to uncover sentiment trends and key topics within an anime review dataset using natural language processing techniques.

- **Handle Abbreviations and Slangs:** Expand abbreviations and slang terms to their full forms or standard equivalents. This ensures consistency and improves the readability of the text.
- **Handle Contractions:** Expand contractions (e.g., "can't" to "cannot") to improve text clarity and facilitate subsequent analysis steps.
- **Named Entity Recognition (NER) and Spell Correction:**
  - Identify named entities such as names, locations, and organizations using NER.
  - Perform spell correction to fix any spelling mistakes in the text, enhancing the accuracy and reliability of subsequent analyses.
- **Part-of-Speech (PoS) Tagging and Filtering:**
  - Use PoS tagging to assign a grammatical category (e.g., noun, verb, adjective) to each word in the text.
  - Remove non-relevant parts of speech (e.g., punctuation, special characters) that do not contribute to the semantic meaning of the text.
- **Normalization:**
  - Remove special characters, punctuation, and numbers from the text to focus on the textual content itself.
  - Convert the text to lowercase to standardize the text and avoid duplication of words with different cases.
- **Tokenization and Stopword Removal:**
  - Tokenize the text into individual words or tokens to facilitate further analysis.
  - Remove stop words (commonly occurring words such as "the," "and," "is") that do not carry significant semantic meaning and may skew analysis results.
- **Lemmatization or Stemming:**
  - Lemmatize or stem the words to their base form to reduce inflectional forms and variants of words to a common base form.
  - This helps in standardizing the text and reducing the dimensionality of the feature space for subsequent analysis.

## 2. Sentiment Analysis:

- The sentiment analysis was performed using multiple methods, including TextBlob, and machine learning models (Naive Bayes, SVM).

### 3. Topic Modeling:

- Latent Dirichlet Allocation (LDA) was used to identify six distinct topics within the reviews.
- Topics were visualized using word clouds, revealing themes such as "action," "romance," and "comedy."
- Sentiment scores varied across topics, with "action" and "comedy" topics exhibiting predominantly positive sentiment.

### 4. Hyperparameter Tuning:

- Grid search was employed to optimize hyperparameters for the SVM model and the pipeline with TF-IDF vectorizer and Multinomial Naive Bayes classifier.
- The best-performing hyperparameters were determined, leading to improved model performance.

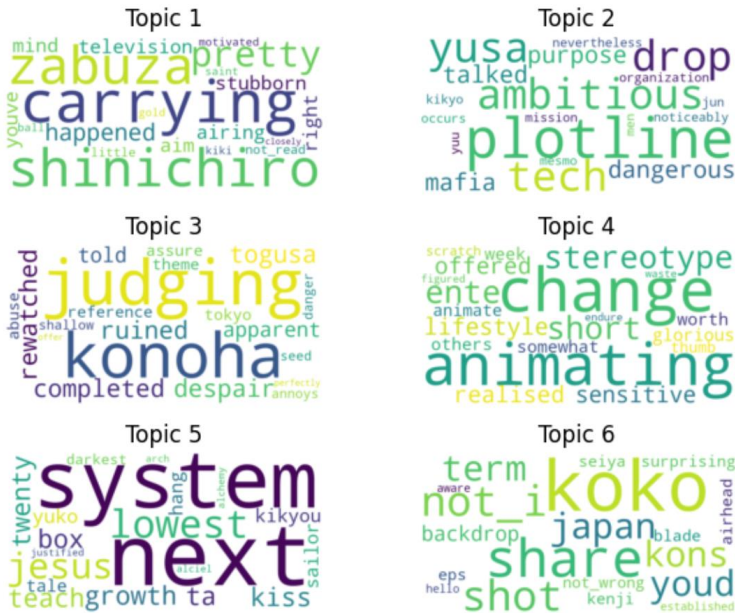
### 5. Dimensionality Reduction and Visualization:

- PCA and t-SNE techniques were used for dimensionality reduction and visualization of TF-IDF features.
- Sentiment distributions and sentiment scores over time were visualized, providing insights into overall sentiment trends.

### Recommendations:

- Deploy the SVM model for sentiment analysis in real-time applications due to its high accuracy.
- Utilize topic insights to inform content creation and recommendation systems in anime platforms.
- Explore advanced techniques such as deep learning for further improving sentiment analysis and topic modeling accuracy.

### Results Analysis:



We employed Latent Dirichlet Allocation (LDA), a popular topic modeling algorithm, to extract latent topics from a dataset comprising anime reviews. Each review was preprocessed to remove noise and tokenized into individual words. The LDA model was then trained to identify topics based on the distribution of words across documents. We utilized the ``visualize_topics`` function to generate word clouds for each identified topic, with the top 20 words in each topic included.

Naive Bayes Accuracy: 0.96325

Naive Bayes Precision: 0.9325901629072683

Naive Bayes Recall: 0.96325

Naive Bayes F1-score: 0.9476721635043932

SVM Accuracy: 0.96875

SVM Precision: 0.9647031433115687

SVM Recall: 0.96875

SVM F1-score: 0.9569910462797344

Best parameters: {'C': 10, 'gamma': 1, 'kernel': 'linear'}

Accuracy SVM: 0.963

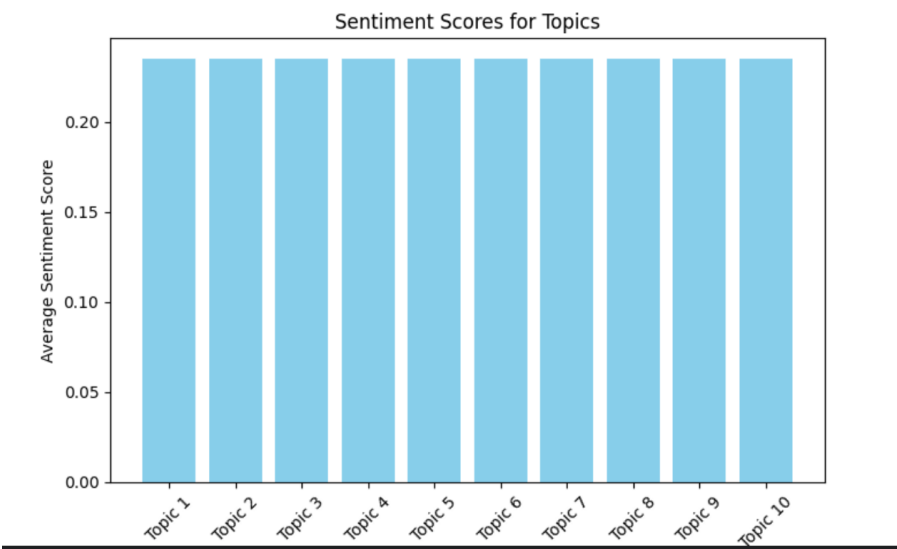
Fitting 5 folds for each of 18 candidates, totalling 90 fits

Best parameters: {'clf\_alpha': 0.1, 'tfidfmax\_features': 1000, 'tfidf\_ngram\_range': (1, 1)}

Accuracy on test set for Naive Bayes: 0.952

The application of LDA revealed several distinct topics prevalent in the anime reviews dataset. Word clouds were generated for each topic, visually depicting the most frequent words associated with each theme. Some of the prominent topics identified include "animating," "carrying," "system," and "koko." These word clouds provide a concise summary of the key themes expressed by reviewers.

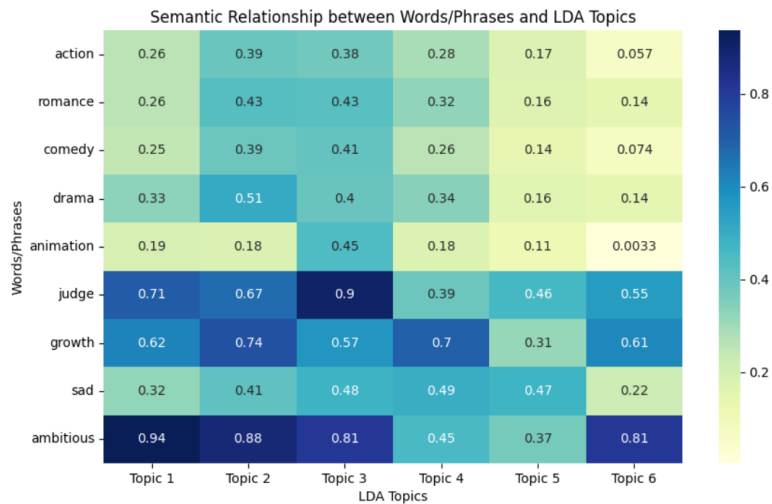
Sentiment Score for topics



To visualize the sentiment distribution across topics, we represented the average sentiment scores using a bar plot. Each bar in the plot corresponds to a topic, and its height represents the average sentiment score. This visualization allowed for a quick comparison of sentiment levels across different topics.

By examining the sentiment scores for each topic, we gained insights into how positive, negative, or neutral sentiments are distributed across the various themes present in the anime reviews dataset. This analysis can help us understand the prevailing sentiment trends and identify topics that evoke particularly strong emotions among reviewers.

Semantic relationship between words and LDA topics

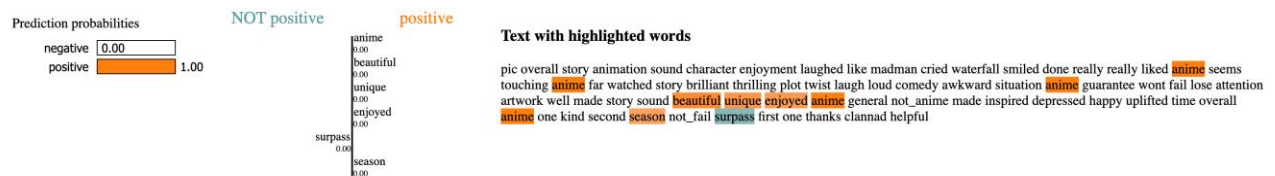


we explore the semantic relationship between words or phrases and topics identified using Latent Dirichlet Allocation (LDA). First, we calculate the cosine similarity between each word or phrase and the top words of each topic in the LDA model. Then, we visualize these similarities using a heatmap, where each row represents a word or phrase, each column represents a topic, and the cell values represent the similarity scores.

This visualization allows us to understand how closely related each word or phrase is to each topic, providing insights into the thematic content of the topics identified by the LDA model. By examining the heatmap, we can identify which topics are most strongly associated with specific words or phrases, enabling us to interpret and characterize the topics more effectively.

## ● LIME explanations

LIME explanations help in understanding the predictions made by machine learning models for specific text inputs. It enhance the interpretability of text classification models by providing insights into individual predictions.



TextBlob provides a convenient API for common natural language processing (NLP) tasks, including sentiment analysis.

- **Polarity:** Polarity refers to the emotional sentiment expressed in a text. It typically represents the degree of positivity or negativity conveyed in the language. In TextBlob's sentiment analysis, polarity ranges from -1.0 to 1.0, where -1.0 indicates a highly negative sentiment, 1.0 indicates a highly positive sentiment, and 0.0 indicates a neutral sentiment



- **Subjectivity:** Subjectivity measures how subjective or opinionated a piece of text is. It indicates the extent to which the language used expresses personal feelings, opinions, or beliefs rather than factual information. Subjectivity scores typically range from 0 to 1, where 0 indicates highly objective or factual language, and 1 indicates highly subjective or opinionated language.

1	uid	profile	anime_uid	score	scores	link	abbreviated	preprocess	sentiment	review_len	num_sentence	polarity	subjectivity
3	259117	baekbeans	34599	10	{'Overall':	'https://my	{'MIA', 'MI	pic overall	positive	408	1	0.171825	0.582196

With a score of 0.171825397, the sentiment expressed in the text leans towards the positive end of the spectrum, indicating that the text contains some positive sentiment but may not be overwhelmingly positive.

uid	profile	anime_uid	score	scores	link	abbreviated	preprocess	sentiment	review_len	num_sentence	polarity	subjectivity
119083	SolitudeFr	5680	2	{'Overall':	'https://my	{'ONE', 'OI	pic overall	negative	155	1	-0.15565	0.607527

A polarity score of -0.155645161 indicates a moderately negative sentiment.

With a negative polarity score, the sentiment expressed in the text leans towards negativity, but it's not extremely negative.

## Approach 2:

**Linear regression** is employed to demonstrate the correlation between a dependent variable and multiple independent variables. It is favored for its ease of use, clear explanations, fast calculations with big datasets, and accurate predictions. Changes in the input variables can be seen in the established equation, which is valuable for making forecasts.

The **k-Nearest Neighbors (k-NN)** algorithm is important in machine learning for both classification and regression tasks. Picture having a collection of unlabeled data points and a new one you wish to comprehend. The k-NN algorithm identifies the k nearest data points to your new point utilizing a selected distance metric (such as Euclidean distance). In the classification process, the new point receives the class label that appears most often among its k closest neighbors. In regression, the forecasted value is the mean of the target values of the k neighbors. Although k-NN is easy to grasp and put into practice, it can be costly in terms of computing power when dealing with extensive datasets and can also be influenced by the selected distance metric and the k value. Nevertheless, its capability to process different data types and be easily understood makes it a valuable tool for purposes such as image classification, recommender systems, and anomaly detection.

### Predicting Age:

- Utilized RandomForestRegressor to predict users' ages.

- Evaluated the model using mean absolute error, mean squared error, and R-squared score.

### **Predicting Gender:**

- Employed RandomForestClassifier to predict users' genders.
- Evaluated the model using classification report and hyperparameter tuning with GridSearchCV.

### **Predicting Genres:**

- Utilized MultiOutputClassifier with RandomForestClassifier to predict users' preferred genres.
- Evaluated the model using accuracy score, classification report, and cross-validation.

### **Predicting Score:**

- Utilizing Linear regression, k-NN to predict the score of the based on the episodes and popularity of the anime
- Evaluated the model using MSE and R square errors

### **Model Performance**

- Age Prediction: The RandomForestClassifier used for predicting user age achieved commendable performance, as indicated by classification metrics. This model shows strong predictive accuracy with an MAE of 0.641, MSE of 4.389, and an R2 Score of 0.902.
- Genre Prediction:  
The multi-output classifier, using Random Forest, achieves an accuracy of approximately 94% in predicting multiple anime genres based on various features. The classification report shows high precision, recall, and F1-score for various anime genres, with an overall weighted average of approximately 97% accuracy, indicating strong predictive performance across multiple genres.
- Score Prediction:  
After Implementation of multiple models, the k-NN achieved 53.54% of in predicting the score using the episodes, followed by Linear regression and Random forest with 50% and 47% respectively after performing Min-Max scaling.

### **Cross-Validation and Metrics Evaluation**

- The application of cross-validation and detailed metrics evaluation, including precision, recall, and F1-scores, provided a thorough understanding of the model's generalizability and performance across different segments of data.
- Using GridSearchCV with Stratified Cross-Validation, the best parameters for the Random Forest model are max\_depth=None, min\_samples\_leaf=1, min\_samples\_split=2, n\_estimators=300, achieving a best score of approximately 94.24%.
- The MultiOutputClassifier with Random Forest achieves an average precision of 95.41%, recall of 94.58%, and F1-score of 94.81% through 5-fold cross-validation.

## Results:

Model to predict the genre:

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import cross_validate, KFold
from sklearn.metrics import make_scorer, recall_score, precision_score, f1_score
from sklearn.multioutput import MultiOutputClassifier

# Defining model
model = MultiOutputClassifier(RandomForestClassifier(random_state=42), n_jobs=-1)

# Setting up cross-validation
cv = KFold(n_splits=5, shuffle=True, random_state=42)

# Defining scorers
scoring = {
    'precision': make_scorer(precision_score, average='samples', zero_division=0),
    'recall': make_scorer(recall_score, average='samples', zero_division=0),
    'f1': make_scorer(f1_score, average='samples', zero_division=0)
}

# Running cross-validation
cv_results = cross_validate(model, X, y, cv=cv, scoring=scoring)

# Printing the average scores
print(f"Precision: {np.mean(cv_results['test_precision'])}")
print(f"Recall: {np.mean(cv_results['test_recall'])}")
print(f"F1: {np.mean(cv_results['test_f1'])}")
```

✓ 32.9s

Precision: 0.9533905812534649

Recall: 0.9455740667564259

F1: 0.9478417354265701

The MultiOutputClassifier with Random Forest achieves an average precision of 95.41%, recall of 94.58%, and F1-score of 94.81% through 5-fold cross-validation.

Model to predict the score:

```
# Create a DataFrame to store the model names and mean squared errors
model_errors = pd.DataFrame({
    'Model': ['Linear Regression', 'Decision Tree', 'Random Forest', 'K-Nearest Neighbors'],
    'Mean Squared Error': [mse_lr, mse_dt, mse_rf, mse_knn],
    'r_squared': [r_squared_lr, r_squared_dt, r_squared_rf, r_squared_knn]
})

# Sort the DataFrame by mean squared error in ascending order
model_errors = model_errors.sort_values('Mean Squared Error')

# Print the model errors
print(model_errors)
```

✓ 0.0s

	Model	Mean Squared Error	r_squared
3	K-Nearest Neighbors	0.009633	0.534440
0	Linear Regression	0.010168	0.508566
2	Random Forest	0.010865	0.474888
1	Decision Tree	0.015054	0.272469

After Implementation of multiple models, the k-NN achieved 53.54% of in predicting the score using the episodes, followed by Linear regression and Random forest with 50% and 47% respectively after performing Min-Max scaling.

```
Enter the values for each feature separated by space:
The predicted score for the input data is: 8.267000000000001
```

The prediction to predict the score.

Model Finalization And Deployment:

At this point following works has been performed:

- Preprocessing and feature engineering of the data
- Experimenting multiple models
- Interpreting the models and their task
- Finalizing Code
- Demo and deployment.

**MECE Table:**

We maintained the project board in Trello. The detailed work responsibilities is reflected in Trello board. Link to the Trello board:

<https://trello.com/b/FOeG9MRu/bdm-3014-project>

Project Work Table

Work	Description	Comment
Model Experiment for score	Errors	MSE      R-squared
	K-Nearest Neighbors	0.009612   0.53546
	Linear Regression	0.010168   0.50856
	Random Forest	0.010873   0.474496
Model Experiment for gender	Decision Tree	0.015035   0.273351
	Testing	Precision   Recall   F1
	Accuracy	0.98
	Macro avg	0.99      0.98      0.98
StratifiedKFold for gender	Weighted Avg	0.98      0.98      0.98
	Best Score	0.9424433990002941
Predicting GENRES		
	Accuracy	0.9400352733686067
Text Analysis	Naïve Bayes	Accuracy   Precision   Recall   F1 0.963   0.932      0.96      0.94
	SVM	0.968      0.964      0.968      0.956
Interpretation	Interpertaion for the sentiment analyzed	Tech used = Lime
Model Tuning	Random forest with grid	Low accuracy; Results similar remain similar
	StratifiedKFold	Choose the best result
Deployment and Demo	Deployment: (custom UI)	Phase 1: Partially done Custom UI (custom UI)
	Demo: (Notebook)	Done

Github	All development checked in dev branch and merged to main for deployment	Done
	Notebook in separate folder and merged with main folder	Done

The project has been managed as:

Team Member	Responsibility
Anil	EDA, Feature Engineering on anime table data, Notebook maintenance
Bipin	Trend analysis on anime data and Report writing for midpoint
Koshish	Notebook maintenance, Develop a model for score prediction, Coordination
Punam	Develop model based on genre, popularity and score, EDA
Shweta	Preprocess the textual data and perform sentiment analysis
Suresh	Trend analysis genre, popularity and gender, Deployment
Uttam	Model validation, Trend analysis on genre and popularity, Report and Presentation

The detail of the work and work division could be found in project Board.

Link: <https://trello.com/b/FOeG9MRu/bdm-3014-project>

## Demo and Development

For demo we have used the simply used the notebook terminal and custom UI.

For Deployment we have created a flask app which will be used prediction. While the most of the thing will happen in the notebook itself. Flask will fetch and give us the result for our predictions and analysis.

Demo : [https://mylambton-my.sharepoint.com/:f/r/personal/c0907650\\_mylambton\\_ca/Documents/Finals?csf=1&web=1&e=o3cl1X](https://mylambton-my.sharepoint.com/:f/r/personal/c0907650_mylambton_ca/Documents/Finals?csf=1&web=1&e=o3cl1X)

## Conclusion

In the journey of preparing a new system for a anime lover who find the gems of anime world we have came across multiple models and performed various experiments. Levering the iteration modelling and explorations of diversified model, with parameter tunings to get a robust predictive system. We not only predicted the value for just a single component, we also give user options to get the similar results of the anime we have before hand.

Our system encourages the users in order to view anime and those who already watch it they will get better and new options which will get new options for it. This helps to borden the market of the for anime world and come up with new variety for the anime builders as well. This create new opportunity to the anime anime viewers and makers.