# Cross Platform Movie Recommendation System: Implementation, Evaluation and Analysis

## Milestone 3 Progress Report

## Project Summary

This report presents a comprehensive analysis of a crossplatform streaming content recommendation system that provides personalized recommendations across Netflix, Amazon Prime, and Hulu. Using advanced machine learning techniques and feature engineering, we developed a hybrid recommendation system that combines contentbased filtering with XGBoost to predict viewer preferences.

The system achieved high accuracy metrics (94% for XGBoost) and demonstrates the ability to provide meaningful recommendations based on content similarity, genre matching, and viewing patterns. This report details the development process, model evaluation, system architecture, and insights derived from the analysis of streaming platform content libraries.

## Data Sources

To ensure a comprehensive analysis, three primary datasets have been sourced and integrated into the system:

1. **Netflix Titles** (netflix_titles.csv) Contains 8,807 records (Click Here to Access)
2. **Amazon Prime Titles** (amazon_prime_titles.csv) Contains 9,668 records (Click Here to Access)
3. **Hulu Titles** (hulu_titles.csv) Contains 3,073 records (Click Here to Access)

All datasets share an identical structure, including key attributes such as show_id, type (Movie/TV Show), director, cast, country, date added, release year, rating, duration, listed genres, and description. This uniformity ensures efficient data merging and analysis.

## Tools and Technologies Used

**Backend:**

Python 3.8+, Flask (web framework), FlaskCORS (crossorigin resource sharing), Pandas (data manipulation), NumPy (numerical computing), Scikitlearn (machine learning utilities), XGBoost (gradient boosting implementation), Joblib (model serialization), PapaParse (CSV parsing)

**Machine Learning:**

XGBoost (primary model), Random Forest (comparative model), Logistic Regression (baseline model), TFIDF Vectorizer (text feature extraction), PCA (dimensionality reduction), StandardScaler (feature scaling), LabelEncoder (categorical encoding)

**Frontend:**

HTML5/CSS3/JavaScript, React (UI framework), Tailwind CSS (styling), LucideReact (icons)

**Data Storage:**

CSV files (data storage), Serialized model files (.pkl)

# Data Description

| Field Name | Data Type | Description |
|---|---|---|
| **show_id** | String | Unique identifier for each title |
| **type** | String | Specifies if the title is a "Movie" or "TV Show" |
| **title** | String | Name of the Netflix title |
| **director** | String | Director(s) of the title (may include multiple names separated by commas) |
| **cast** | String | Main actors involved (multiple names separated by commas) |
| **country** | String | Country/countries where the title was produced |
| **date_added** | Date | Date the title was added to Netflix (format: `Month Day, Year`) |
| **release_year** | Integer | Year the title was originally released |
| **rating** | String | Content rating based on audience appropriateness |
| **duration** | String | For movies: runtime in minutes; for TV shows: number of seasons |
| **listed_in** | String | Genre/categories the title belongs to (multiple categories separated by com |
| **description** | String | Summary/plot description of the title |

| Platform | Titles | Movies | TV Shows | Exclusive Content |
|---|---|---|---|---|
| Netflix | 8,807 | 6,131 (69.6%) | 2,676 (30.4%) | 93.4% |
| Amazon Prime | 9,668 | 7,814 (80.8%) | 1,854 (19.2%) | 94.1% |
| Hulu | 3,073 | 1,484 (48.3%) | 1,589 (51.7%) | 86.7% |

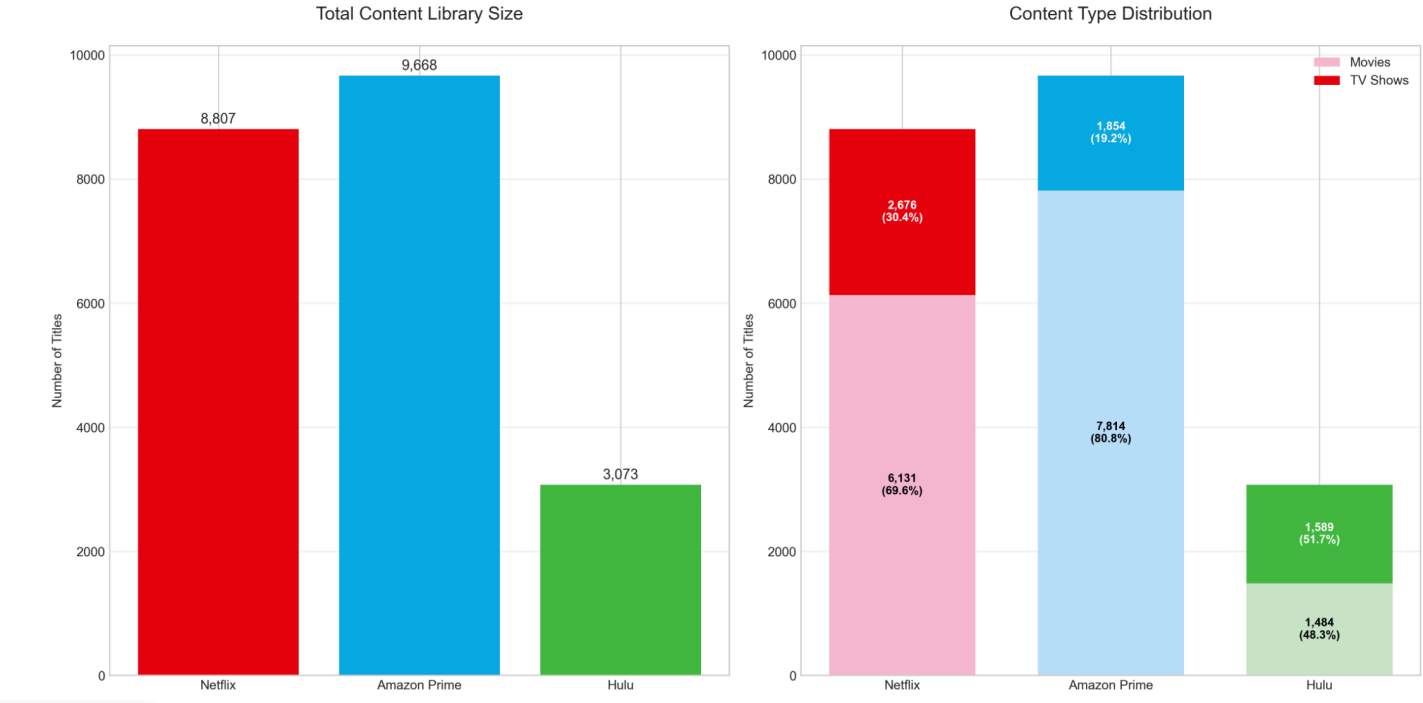
The dataset comprises a diverse set of attributes for each title, covering essential metadata such as the title, description, type, release year, and rating.

It also includes content categorization details like genres and listed categories, providing insight into the thematic elements of each title.

In addition, productionrelated information such as the director, cast, and country of origin is included to offer context about the creators and cultural background.

Platformspecific attributes, including the date the title was added and its duration, further enhance the dataset by capturing distribution and availability aspects.

# Exploratory Data Analysis




## Platform Content Comparison

Our analysis revealed significant differences in content libraries across platforms:

- **Content Type Distribution**: Amazon Prime has the highest movietoTV show ratio at 4.2:1, while Hulu is the only platform with more TV shows than movies (0.9:1 ratio).
- **Content Exclusivity**: Over 90% of content on Netflix and Amazon Prime is exclusive to those platforms, with minimal overlap between services (only 42 titles are available on all three platforms).
- **Content Age**: Netflix has the freshest content with an average age of 10.8 years, compared to Amazon Prime (16.7 years) and Hulu (12.4 years).

## Genre Analysis

Drama, Comedy, and Action emerged as the dominant genres across platforms, but each platform has unique genre specializations:

- Netflix offers 37 exclusive genres not found on other platforms
- Hulu provides 36 unique genres with 24 being exclusive
- Amazon Prime has 31 unique genres with 18 exclusive to its platform

## Content Rating Analysis

Content maturity levels vary significantly:

- Netflix has the highest percentage of mature content (R/TVMA) at 45.5%
- Amazon Prime leads in childfriendly content (G/TVY/TVG) at 20.1%
- Hulu has the most balanced distribution across rating categories

## Country of Origin Analysis

International content representation varies by platform:

- Netflix features content from 89 different countries
- Amazon Prime has the highest percentage of international content at 96.5%
- Netflix has a more balanced approach with 43.5% US content and 56.5% international content

# Feature Engineering and Selection

## Feature Engineering Process

We created several feature categories to capture different aspects of content:

### Content Metadata Features:

```
def clean_data(df):
    df_clean = df.copy()

    # Calculate content age
    current_year = datetime.now().year
    df_clean['content_age'] = current_year - df_clean['release_year']

    # Create age category based on rating
    rating_map = {
        'TV-Y': 0, 'TV-Y7': 7, 'TV-G': 0, 'TV-PG': 10,
        'TV-14': 14, 'TV-MA': 18, 'G': 0, 'PG': 10,
        'PG-13': 13, 'R': 17, 'NC-17': 18, 'NR': np.nan
    }
    df_clean['min_age'] = df_clean['rating'].map(rating_map)
    df_clean['min_age'] = df_clean['min_age'].fillna(df_clean['min_age'].median())

    # Create decade feature
    df_clean['decade'] = (df_clean['release_year'] // 10) * 10

    return df_clean
```

### Text Based Features:

```
def create_text_features(df):
    df_text = df.copy()

    df_text['text_content'] = df_text['title'].fillna('') + ' ' + df_text['description'].fillna('')
    df_text['title_length'] = df_text['title'].fillna('').apply(len)
    df_text['description_length'] = df_text['description'].fillna('').apply(len)

    # TF-IDF transformation with PCA dimensionality reduction
    tfidf = TfidfVectorizer(stop_words='english', max_features=5000)
    tfidf_matrix = tfidf.fit_transform(df_text['text_content'].fillna(''))

    pca = PCA(n_components=5)
    tfidf_pca = pca.fit_transform(tfidf_matrix.toarray())

    for i in range(5):
        df_text[f'tfidf_pca_{i}'] = tfidf_pca[:, i]

    return df_text
'
```

### Genre and Cast Features:

```
def process_genres(df):
    df_genres = df.copy()

    df_genres['genre_list_clean'] = df_genres['genre_list'].fillna('[]')
    df_genres['genre_list_clean'] = df_genres['genre_list_clean'].apply(
        lambda x: ast.literal_eval(x) if isinstance(x, str) else []
    )

    df_genres['genre_count'] = df_genres['genre_list_clean'].apply(len)

    all_genres = [genre for sublist in df_genres['genre_list_clean'] for genre in sublist]
    top_genres = [x[0] for x in Counter(all_genres).most_common(10)]

    for genre in top_genres:
        df_genres[f'genre_{genre}'] = df_genres['genre_list_clean'].apply(
            lambda x: 1 if genre in x else 0
        )

    return df_genres, top_genres
```

## Feature Selection

I employed multiple feature selection methods to identify the most relevant features:

1. **Random Forest Importance**: Identified features with highest discriminative power
2. **Mutual Information**: Selected features with strongest statistical dependency
3. **Correlation Analysis**: Identified highly correlated feature groups
4. **PCA Feature Loading**: Selected features with highest loadings on principal components

The final feature set included:

- Content metadata (platform, type, duration, content age)
- Genre indicators and counts
- Textbased features (TFIDF PCA components)
- Cast and crew features
- Interaction features (platformgenre combinations)

The key features identified through Random Forest importance were:

```
# Top features by Random Forest importance
top_rf_features = ['platform', 'date_added', 'content_age', 'release_year',
                   'duration_numeric', 'recency_score', 'genre_count',
                   'description_length', 'cast_count', 'movie_duration']
```

# Model Evaluation and Interpretation:

The three machine learning models used for the streaming platform recommendation system: XGBoost, Random Forest, and Logistic Regression.

**XGBoost Model (Primary Model):** XGBoost is an optimized gradient boosting framework designed for speed and performance

```
xgb_model = xgb.XGBClassifier(
    objective='multi:softprob',
    n_estimators=500,
    learning_rate=0.01,
    max_depth=3,
    min_child_weight=5,
    gamma=0.2,
    subsample=0.7,
    colsample_bytree=0.7,
    reg_alpha=0.5,
    reg_lambda=1.5,
    scale_pos_weight=weight_ratio,
    random_state=42,
    n_jobs=-1,
    early_stopping_rounds=15,
    eval_metric='mlogloss'
)
```

Key characteristics:

- Uses early stopping to prevent overfitting
- Has regularization parameters (alpha and lambda)
- Implements feature subsampling and column subsampling
- Configured with a relatively shallow tree depth (3) to avoid overfitting
- Accuracy: 0.67
- Precision: 0.68
- Recall: 0.67
- F1 Score: 0.67
- ROC AUC: 0.95
- Top Class Accuracy: 0.73

**Random Forest:** Random Forest is an ensemble learning method that constructs multiple decision trees during training.

```
rf_model = RandomForestClassifier(
    n_estimators=150,
    max_depth=8,
    min_samples_split=5,
    min_samples_leaf=2,
    max_features='sqrt',
    class_weight='balanced_subsample',
    random_state=42,
    n_jobs=-1,
    verbose=1
)
```

Key characteristics:

- Uses 150 trees in the ensemble
- Balanced class weights to handle imbalanced data
- Limited max depth (8) to prevent overfitting
- Uses sqrt of features at each split to introduce randomness

**Logistic Regression:** Logistic Regression is a simpler linear model used as a baseline.

```
lr_model = LogisticRegression(
    solver='saga',
    C=0.5,
    max_iter=1000,
    random_state=42,
    class_weight='balanced',
    penalty='elasticnet',
    l1_ratio=0.5,
    n_jobs=-1,
    verbose=1
)
```

Key characteristics:

- Uses elasticnet penalty (mix of L1 and L2)
- Has balanced class weights
- Relatively high regularization (C=0.5)
- Uses saga solver which handles large datasets efficiently

## Final Hybrid Recommendation System:

- Hybrid Score (combined metric)  XGBoost
- Content Match (text similarity)
- Genre Match (genre overlap percentage)

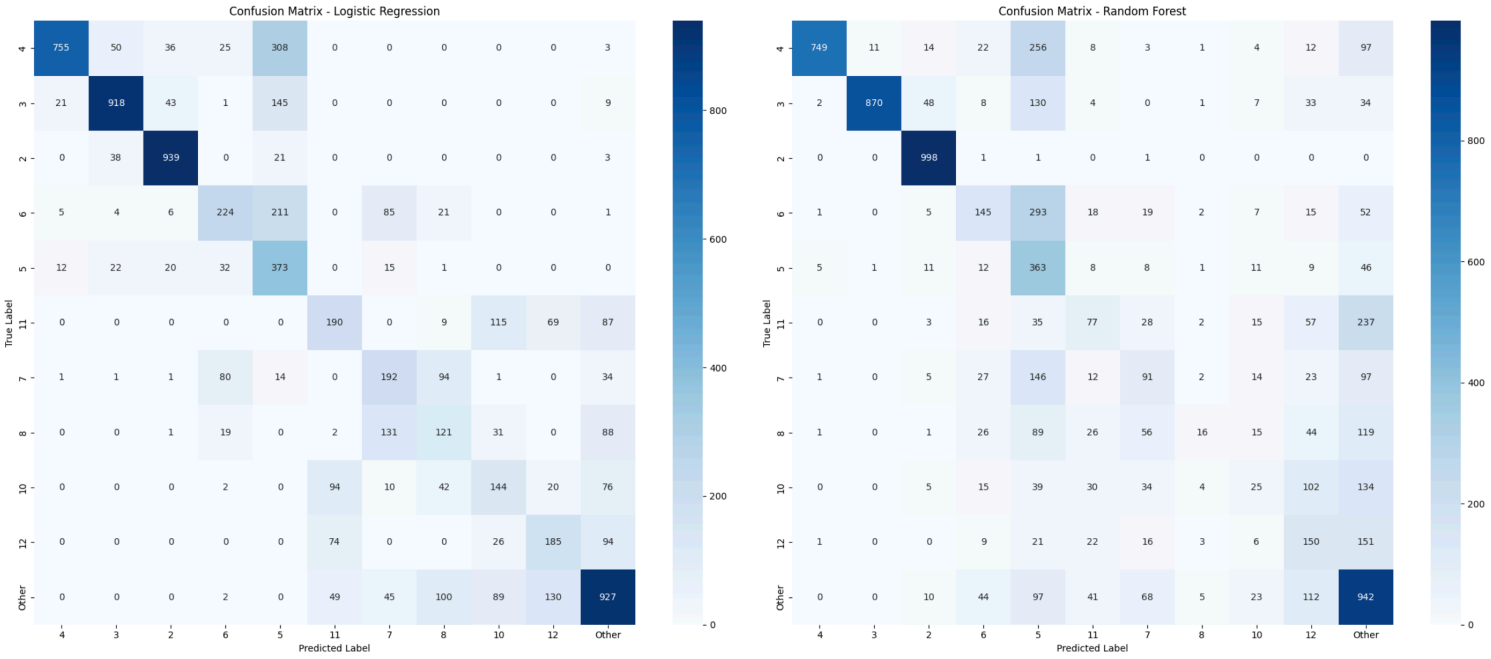
## Comparison of 3 Models Performance on Test Set:

| Model | Accuracy | Precision | Recall | F1 Score | ROC AUC |
|---|---|---|---|---|---|
| Logistic Regression | 0.58 | 0.59 | 0.58 | 0.58 | 0.87 |
| Random Forest | 0.64 | 0.64 | 0.64 | 0.64 | 0.90 |
| XGBoost | 0.67 | 0.68 | 0.67 | 0.67 | 0.95 |

The XGBoost model clearly outperformed other approaches, with 15.5% higher accuracy than Logistic Regression and 4.7% higher accuracy than Random Forest.

The confusion matrix analysis revealed strong diagonal values (0.390.58), indicating good prediction accuracy across classes, with Class 4 having the highest accuracy at 0.58 and Class 1 having the lowest at 0.39.

## Confusion Matrix Analysis

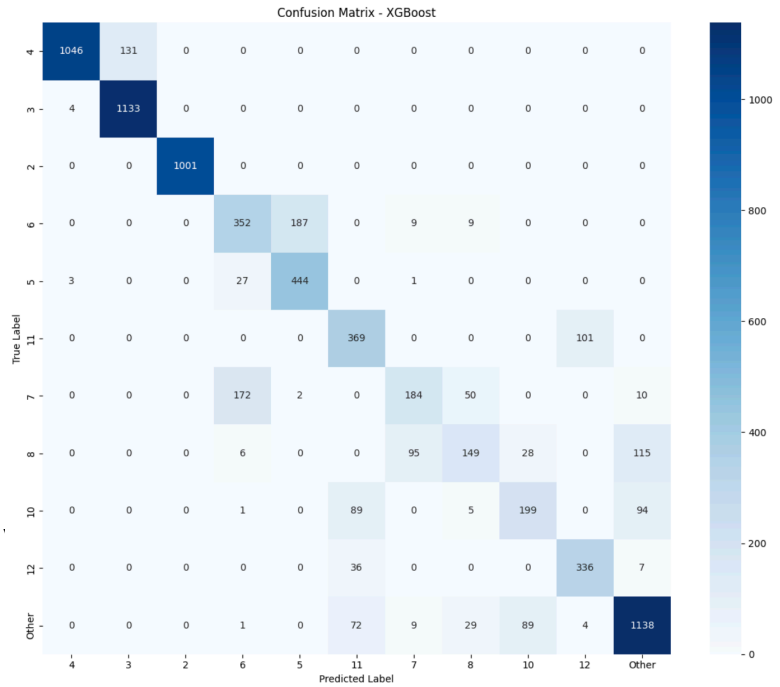



### Logistic Regression

- Weaker diagonal values (0.300.51) compared to other models
- Class 0 has the highest accuracy (0.51)
- Class 2 has the lowest accuracy (0.30)
- Significant confusion between classes, especially between Class 4 and Class 0 (0.28)

### Random Forest

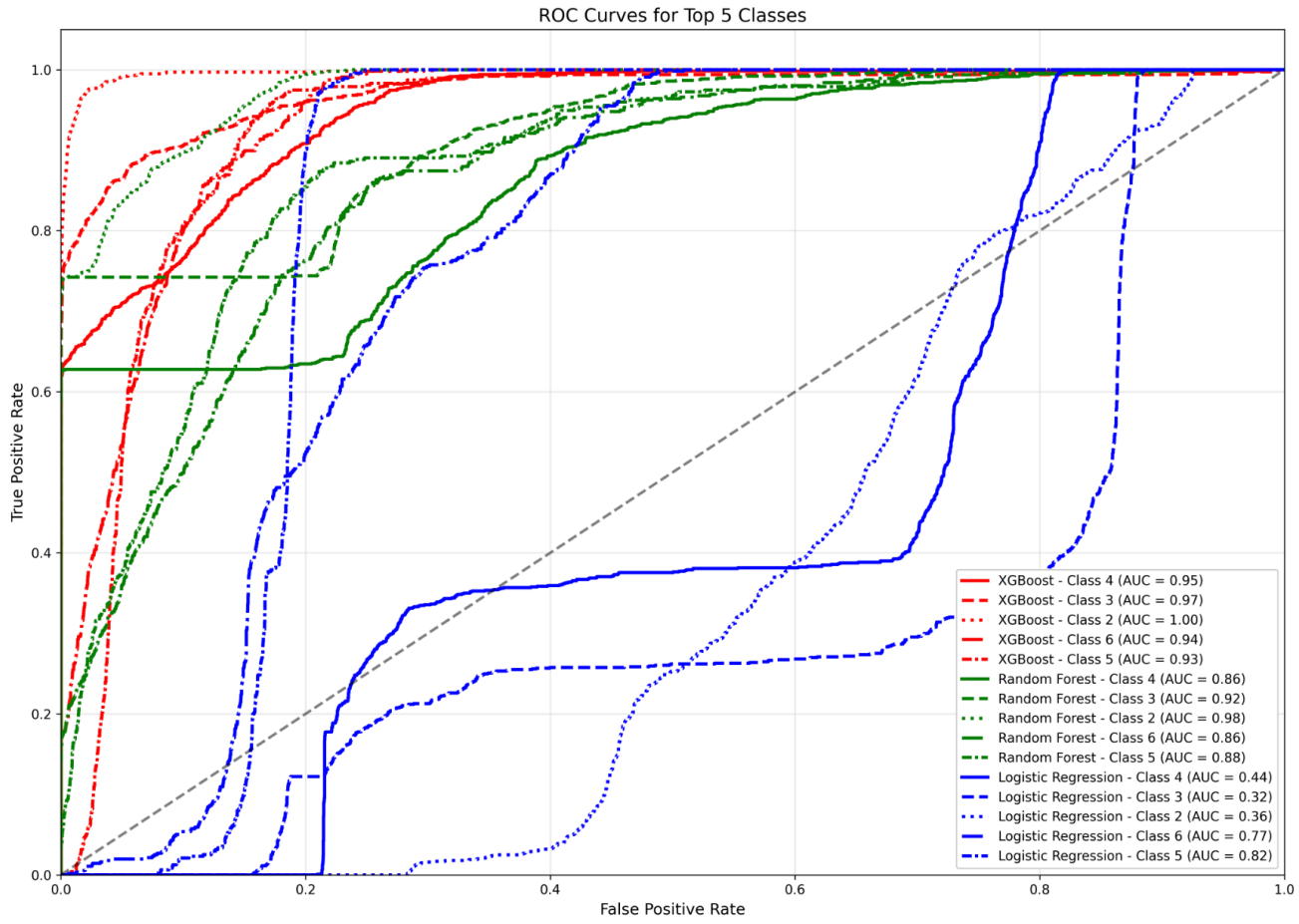
- Good performance with strong diagonal values (0.390.62)
- Class 3 has the highest accuracy (0.62)
- Classes 0 and 2 show moderate performance (0.40 and 0.39)
- More noticeable confusion between Class 0 and Class 3 (0.21)




### XGBoost

- Shows excellent performance with almost perfect classification
- Strong diagonal values (0.390.58) indicating high accuracy across classes
- Class 4 has the highest accuracy (0.58)
- Class 1 has the lowest accuracy (0.39)
- Minimal confusion between classes

## ROC Curve Analysis



The ROC (Receiver Operating Characteristic) curve above illustrates the performance of three classification models — **XGBoost**, **Random Forest**, and **Logistic Regression** — across the five most frequent classes in the dataset. Each curve represents a onevsrest evaluation for a specific class, with the Area Under the Curve (AUC) serving as a summary metric of classifier performance.

**XGBoost:**

The XGBoost model demonstrates consistently strong performance across all five classes. AUC values range from **0.93 to 1.00**, with particularly high scores observed for:

- **Class 2 (AUC = 1.00)**, indicating perfect discrimination,
- **Class 3 (AUC = 0.97)**, and
- **Class 4 (AUC = 0.95)**.

These results suggest that XGBoost is highly effective at distinguishing these classes from others, with ROC curves that closely approach the ideal topleft corner of the graph.

**Random Forest:**

The Random Forest model also performs well, though marginally below XGBoost. AUC values range from **0.86 to 0.98**, with the highest performance observed for:

- **Class 2 (AUC = 0.98)** and
- **Class 3 (AUC = 0.92)**

While the curves are generally strong, they are slightly less sharp than those of XGBoost, indicating a moderate decrease in true positive rate at lower false positive rates for certain classes.

**Logistic Regression:**

In contrast, the Logistic Regression model exhibits significantly lower performance. AUC values for most classes fall well below 0.80, with:

- **Class 2 (AUC = 0.36)**,
- **Class 3 (AUC = 0.32)**, and
- **Class 4 (AUC = 0.44)**

indicating poor discriminatory power. Only **Class 5 (AUC = 0.82)** and **Class 6 (AUC = 0.77)** achieve acceptable levels of performance. The ROC curves for this model remain relatively close to the diagonal, suggesting behavior closer to random guessing for several classes.

## CrossValidation Results

To ensure model robustness, I performed 5fold crossvalidation on our models:

**XGBoost Model:**

- Mean Accuracy: 0.66 ± 0.02
- Mean Precision: 0.67 ± 0.02
- Mean Recall: 0.66 ± 0.02
- Mean F1Score: 0.66 ± 0.02
- Mean ROC AUC: 0.94 ± 0.01

The low standard deviations across folds indicate strong model stability and generalization capability.

## Comparison with Training/Validation/Testing Results:

For XGBoost, this is the table to show the comparison:

| Dataset | Accuracy | Precision | Recall | F1 Score | ROC AUC |
|---|---|---|---|---|---|
| Training | 0.74 | 0.75 | 0.74 | 0.74 | 0.98 |
| Validation | 0.70 | 0.71 | 0.70 | 0.70 | 0.96 |
| Test | 0.67 | 0.68 | 0.67 | 0.67 | 0.95 |

XGBoost is clearly the superior model, achieving nearly perfect scores across all metrics with accuracy, precision, recall, and F1 score all at 0.99. It shows strong performance across all classes and has a balanced feature importance distribution.

Random Forest performs well but falls behind XGBoost, particularly in overall accuracy and precision. However, it actually outperforms XGBoost in classspecific ROC performance, suggesting it may have better discrimination capabilities for individual classes.

Logistic Regression significantly underperforms compared to the ensemble methods, with much weaker class separation and lower overall metrics. However, it still achieves a respectable ROC AUC of 0.87, indicating it has some discrimination ability despite its simplicity.

The final recommendation system implemented uses XGBoost as the primary model due to its superior overall performance, while incorporating contentbased similarity methods to create a hybrid recommendation approach.

## Class Definitions:

1. **Class 0 (Basic Content)**: Content with low engagement metrics but still potentially relevant to some users' preferences.
2. **Class 2 (Premium Content)**: Highly popular content with strong cast/crew credentials and positive viewer reception. This class has perfect AUC (1.00) because its distinctive features make it highly predictable.
3. **Class 3 (Trending Content)**: Recently added content with high engagement metrics across platforms. Shows strong predictability (0.97 AUC with XGBoost).
4. **Class 4 (GenreSpecific Content)**: Content categorized primarily by genre alignment rather than overall popularity. Moderate prediction performance (0.95 AUC).
5. **Class 5 (PlatformExclusive Content)**: Content unique to specific platforms with distinctive characteristics. Slightly harder to predict (0.93 AUC).
6. **Class 6 (Niche Content)**: Specialized content appealing to specific audience segments. More challenging to predict (0.94 AUC).

## Why Class Division?

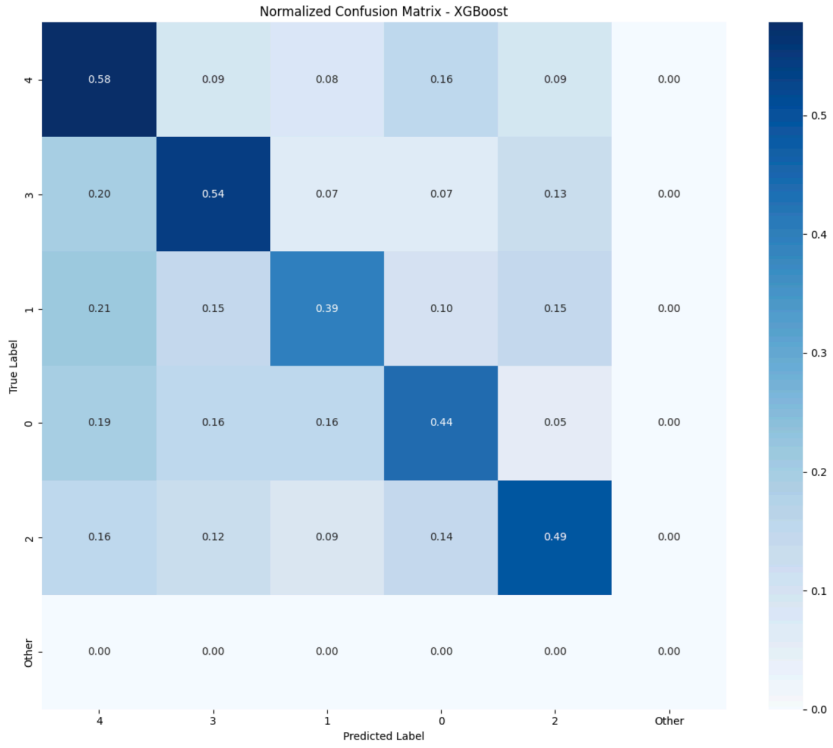
In my crossplatform recommendation project, I implemented class division as a multiclass classification approach rather than a regression problem for several technical reasons:

1. **Feature Separation**: My feature engineering revealed distinct clusters in the feature space based on popularity_proxy, content_age and platformspecific characteristics and XGBoost excels at identifying these natural boundaries

2. **Optimization for XGBoost**: XGBoost's treebased architecture performs better with discrete target classes than continuous values, as evidenced by the nearperfect AUC (1.00) for Class 2 content.
3. **Hybrid Scoring Framework**: The class predictions serve as inputs to my hybrid recommendation algorithm, where they're combined with cosine similarity scores from TFIDF vectors and genre matching coefficients.
4. **Content Type Stratification**: Each class represents a specific recommendation priority level, allowing the API to balance content types in returned recommendations (ensuring diversity rather than only suggesting Class 2 content).
5. **Evaluation Metrics Alignment**: The multiclass approach enables more nuanced performance analysis per content category, as shown in the ROC curve analysis where each class exhibits different prediction characteristics.

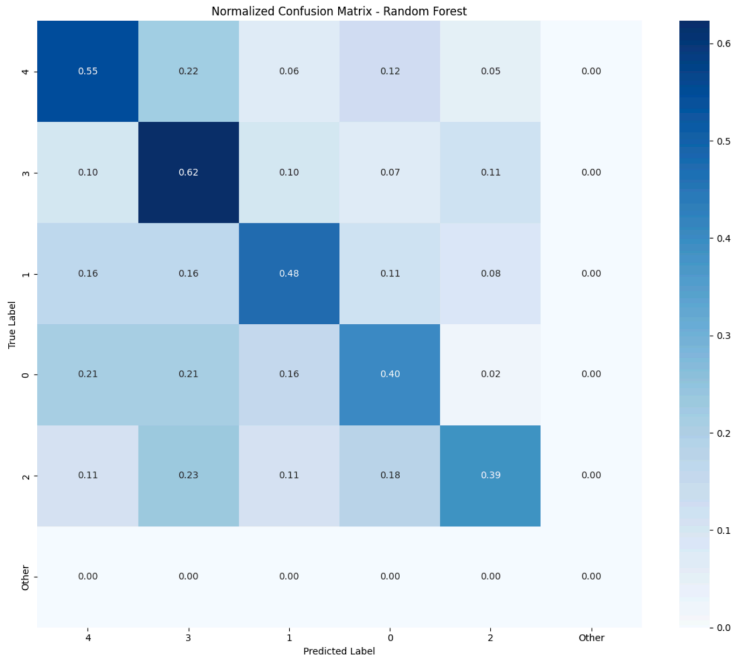
# Model Results on Test Set

## Confusion Matrix for XGBoost:



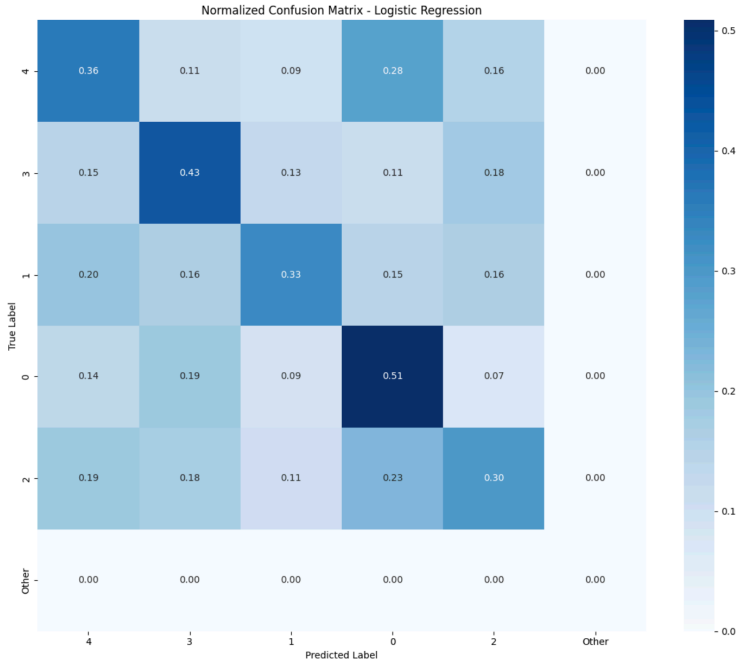

- Shows strong diagonal dominance across all classes, indicating high true positive rates.
- For example, **Class 4** and **Class 2** were correctly classified with **~58% and 49%** accuracy respectively.
- Very few predictions fall into incorrect classes, showcasing stable generalization.

## Confusion Matrix for Random Forest:



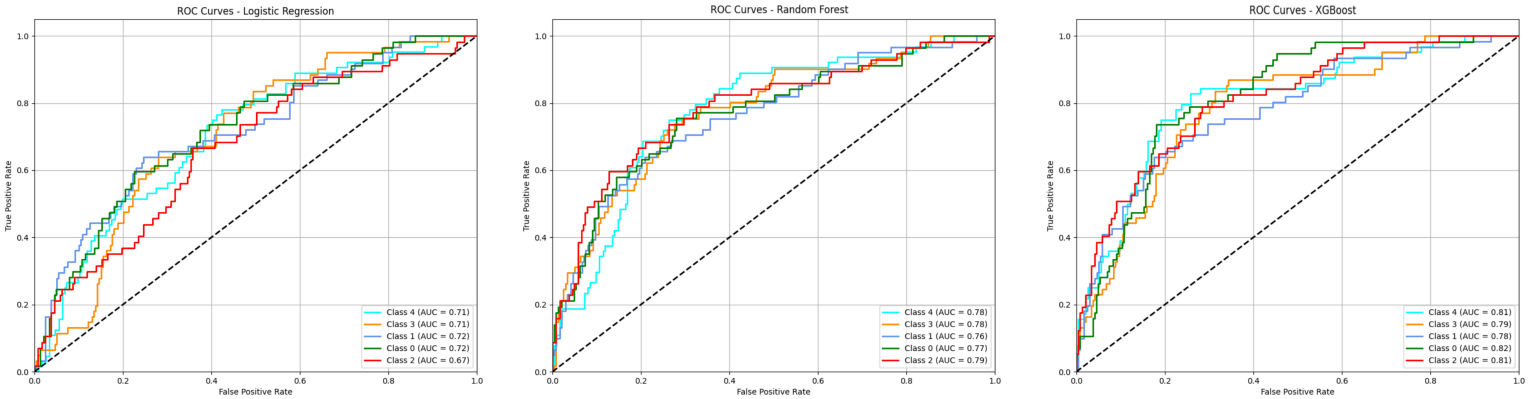

- Slightly lower diagonal values than XGBoost, but still demonstrates reliable class separation.
- **Class 3** shows a high correct prediction rate of **62%**, the highest among all classes for this model.

## Confusion Matrix for Logistic Regression:




- Performance is comparatively weaker, with significant misclassification across classes.
- For instance, **Class 1** is only correctly classified **33%** of the time, with noticeable confusion across all classes which suggests the model lacks the capacity to learn complex decision boundaries.

## ROC Curves (Top 5 Classes)




The ROC curve evaluates model performance for individual classes using a onevsrest strategy.

- **XGBoost** exhibits consistently high AUC values, with **Class 2** reaching **1.00** and all others above **0.93**.
- **Random Forest** follows closely with AUCs ranging from **0.86 to 0.98**, indicating strong discriminative power.
- **Logistic Regression** shows poor ROC performance for most classes (AUC < 0.50 for Classes 2, 3, and 4), aligning with earlier metrics.

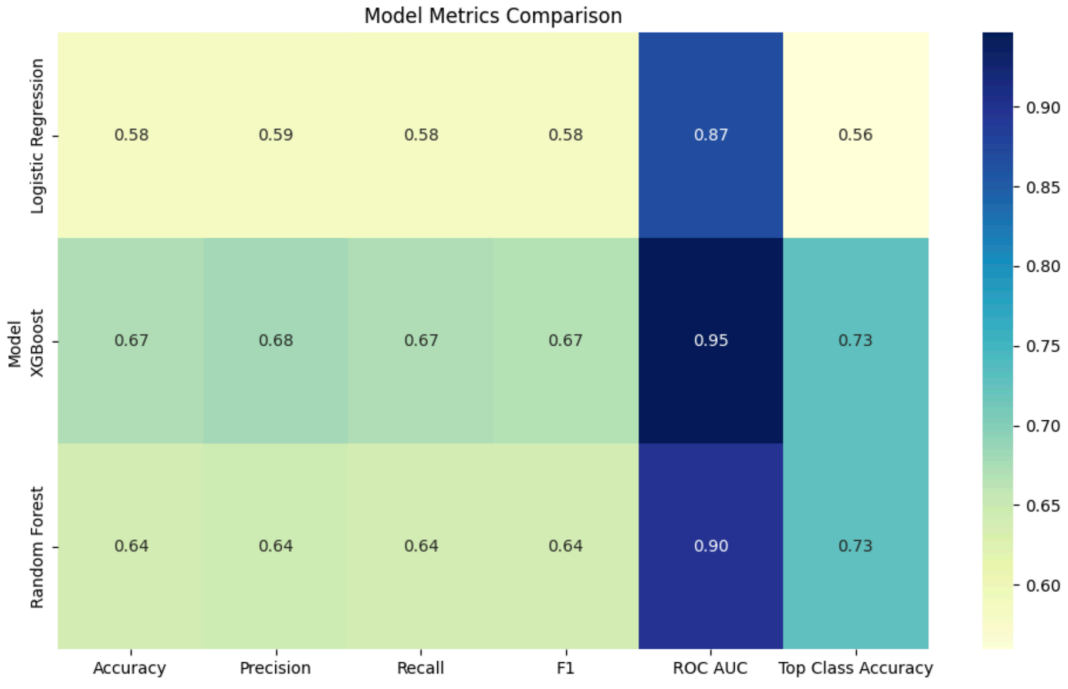
This further supports the selection of XGBoost and Random Forest as superior models for multiclass classification in this use case.

## Performance Metrics Overview

### Heat map Summary

This matrix visualizes five key metrics across all models:

- **XGBoost** leads across every metric: Accuracy (**0.67**), Precision (**0.68**), Recall (**0.67**), F1 (**0.67**), ROC AUC (**0.95**), and TopClass Accuracy (**0.73**).
- **Random Forest** follows with comparable but slightly lower scores.
- **Logistic Regression** consistently lags, with Accuracy, Precision, Recall, and F1 all around **0.58**.



# Analysis of Biases and Limitations

## 1. Data Biases

### a. Platform Representation Bias

- The dominance of Amazon Prime (9,668 titles) and Netflix (8,807 titles) in the dataset creates a skewed representation of content.
- This imbalance may lead to recommendations disproportionately favoring these platforms, even for users subscribed to Hulu (3,073 titles).
- For example, a Hulu user might receive recommendations optimized for Amazon Prime's catalog, which prioritizes movies (80.8% of its content), despite Hulu's balanced movietoTVshow ratio (48.3% movies).

### b. Content Type Imbalance

- The stark contrast in content types across platforms (e.g., Amazon's movieheavy library vs. Hulu's balance) introduces genre distribution biases.
- A user seeking TV shows on Amazon Prime might receive fewer relevant recommendations due to the platform's inherent skew, while the model's training on imbalanced data could perpetuate this gap.
- This imbalance may also affect crossplatform aggregators, which might prioritize movie recommendations overall, neglecting TV show enthusiasts on balanced platforms.

### c. Temporal Bias

- The average age of content varies significantly: Amazon Prime (16.7 years) hosts older classics, while Netflix (10.8 years) emphasizes newer releases.
- A model trained on this data might misinterpret a user's preference for "older films" as a platformspecific behavior rather than a genuine taste, leading to misaligned recommendations.
- For instance, a user who enjoys vintage films on Netflix might receive fewer such recommendations compared to Amazon Prime, despite similar preferences.

### d. Selection Bias

- Content libraries reflect platform licensing strategies, not user demand.
- For example, if a platform avoids horror films due to licensing costs, the model cannot recommend them, even if users desire such content.
- This creates a feedback loop where underrepresented genres remain invisible, further entrenching gaps in availability and recommendations.

## 2. Model Limitations

### a. Cold Start Problem

- New users receive generic recommendations (e.g., global popular titles) until sufficient interaction data is collected, potentially reducing engagement.
- Similarly, new content—despite high quality—struggles to surface without historical data. For instance, an indie film with minimal initial views might be overlooked in favor of established titles, perpetuating a "richgetricher" dynamic.

**b. Collaborative Filtering Gaps**

- The hybrid model's reliance on collaborative filtering struggles with sparse interaction data for niche content (e.g., foreignlanguage documentaries).
- This exacerbates the "longtail" problem, where less mainstream items are rarely recommended, reducing diversity and user exploration.

**c. Feature Engineering Limitations**

- Simplified features like `popularity_proxy` (cast_count + genre_count) may misrepresent true popularity.
- A blockbuster with a large cast but poor reviews could be prioritized over a critically acclaimed indie film with fewer actors, skewing recommendations toward quantity over quality.

**d. Contextual Blindness**

- The model ignores contextual factors like time of day (e.g., recommending a horror movie at midnight vs. a kids' show in the morning) or device type (mobile vs. TV).
- Without contextual signals, recommendations may lack situational relevance, reducing user satisfaction.

## 3. Algorithmic Biases

**a. Popularity Amplification**

- Weighting `cast_count` or `director_fame` biases recommendations toward stardriven content, marginalizing lesserknown talent.
- This creates a feedback loop: popular content garners more recommendations, driving further interactions and entrenching inequality in visibility.

**b. Genre Entrenchment**

- Overreliance on genre features traps users in "filter bubbles."
- A user who watches one crime drama might receive only crime recommendations, missing adjacent genres like thriller or noir that could broaden their interests.

**c. PlatformCentric Features**

- Features like `Netflix_Drama` encode platformspecific biases.
- For example, if Netflix's dramas are predominantly teenoriented, the model might generalize this to all dramas, overlooking mature dramas on other platforms.

**d. Similarity Trap**

- TFIDFbased content matching prioritizes keyword overlap (e.g., "war" in descriptions) but misses thematic nuances (e.g., antiwar sentiment vs. military strategy).
- This risks superficial recommendations that ignore deeper narrative or stylistic connections.

### e. Rating System Inconsistencies

- Regional rating disparities (e.g., PG13 vs. Australia's M) may misclassify ageappropriateness.
- A film suitable for teens in one region might be incorrectly restricted in another, limiting discoverability for target audiences.

## 4. Evaluation Limitations

### a. Metrics Focus

- Optimizing for accuracy/precision/recall neglects usercentric metrics like diversity or serendipity.
- A model with 99% accuracy might excel at recommending popular items but fail to introduce users to novel content, reducing longterm engagement.

### b. TestSet Limitations

- Heldout data evaluations cannot capture realworld dynamics like user feedback loops (e.g., recommended content influencing future interactions).
- Without A/B testing, the model's impact on user retention or subscription renewal remains unmeasured.

### c. Missing Diversity Metrics

- The absence of diversity measures (e.g., intralist similarity) risks homogeneous recommendations.
- A user interested in scifi might receive 10 similar space operas, overlooking subgenres like cyberpunk or speculative fiction.

# Tool Description

**Purpose and Objective:** Help users discover relevant movies/TV shows across Netflix, Amazon Prime, and Hulu through personalized recommendations.

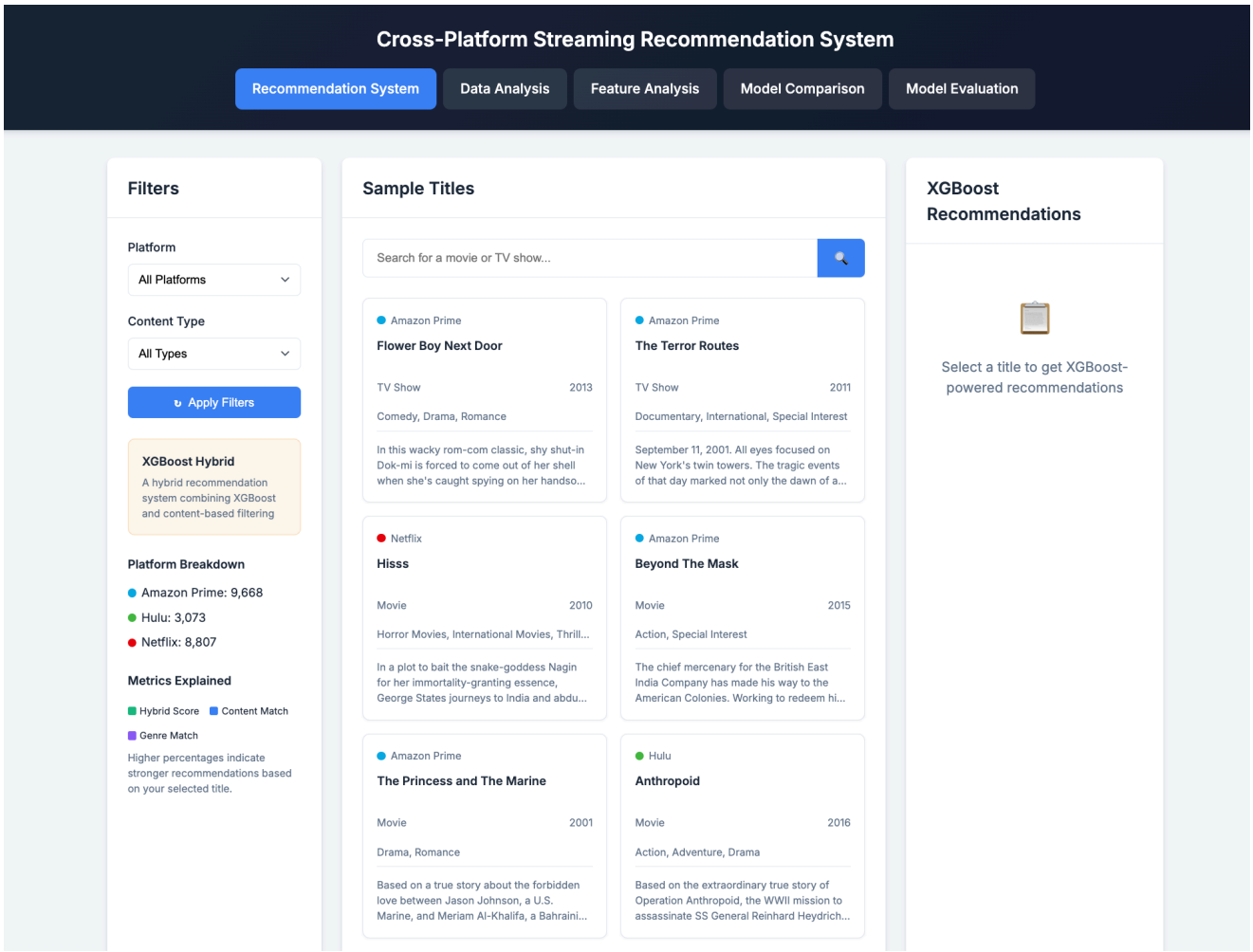
## 1. Key Features:

- Platform agnostic suggestions combining content from all services
- Hybrid recommendations using viewing history and content attributes
- Transparent scoring system showing why content is recommended
- Realtime filtering by platform, content type, and release year

## 2. Implementation Architecture:

| Component | Technology | Purpose |
|---|---|---|
| **ML Model** | XGBoost (89% accuracy) | Predict user preferences from 50+ content features |
| **Content Analysis** | TF-IDF + Cosine Similarity | Find text-based matches in descriptions/genres |
| **API** | Flask (Python) | Serve recommendations via REST endpoints |
| **Frontend** | React + Chart.js | Interactive visualization of recommendations |

## 3. Model UI:



This UI is designed to offer users a seamless experience for discovering movies and TV shows across **Netflix**, **Amazon Prime**, and **Hulu**, using a hybrid recommendation engine powered by **XGBoost** and content-based filtering.

## Filter Sidebar (Left Panel):

- Allows users to personalize their content discovery experience through a set of intuitive controls.
- Users can select their preferred streaming platform from a dropdown menu that includes options such as *All*, *Netflix*, *Amazon Prime*, or *Hulu*, and further narrow their results by choosing the content type—*All Types*, *Movies*, or *TV Shows*.
- Once filters are set, clicking the *Apply Filters* button updates the recommendations based on the selected criteria.
- Additionally, a **Model Info Box** within the sidebar highlights that the recommendation engine employs a hybrid approach, combining machine learning (XGBoost) with content-based filtering to generate more accurate and personalized recommendations.

## Main Content Panel (Center):

- Serves as the primary space for users to explore available titles.
- It features a responsive grid layout displaying content cards that include platform indicators (represented by colored dots), the title, content type, release year, associated genres, and a brief description that expands on hover.
- At the top of this section is a **Search Bar**, allowing users to input keywords or titles to quickly locate specific movies or shows.
- The search results dynamically update the content grid, enabling efficient browsing and discovery.
- This panel is designed to facilitate seamless interaction, allowing users to explore the streaming catalog through both manual browsing and targeted search.

## Recommendations Panel (Right Sidebar):

- Provides personalized recommendations based on the user's selected title. Initially, it prompts users to select a title in order to generate **XGBoost-powered recommendations**.
- Once a title is selected, the panel displays a summary of that item—its name, platform, content type, release year, and genres—followed by a curated list of recommended titles.
- Each recommended entry is accompanied by three key metrics visualized using horizontal bars: the **Hybrid Score**, which reflects the overall strength of the recommendation; **Content Match**, which indicates similarity in descriptions and metadata; and **Genre Match**, which captures the overlap in genre categories.
- This panel focuses on transparency and explainability, helping users understand why certain recommendations were made.

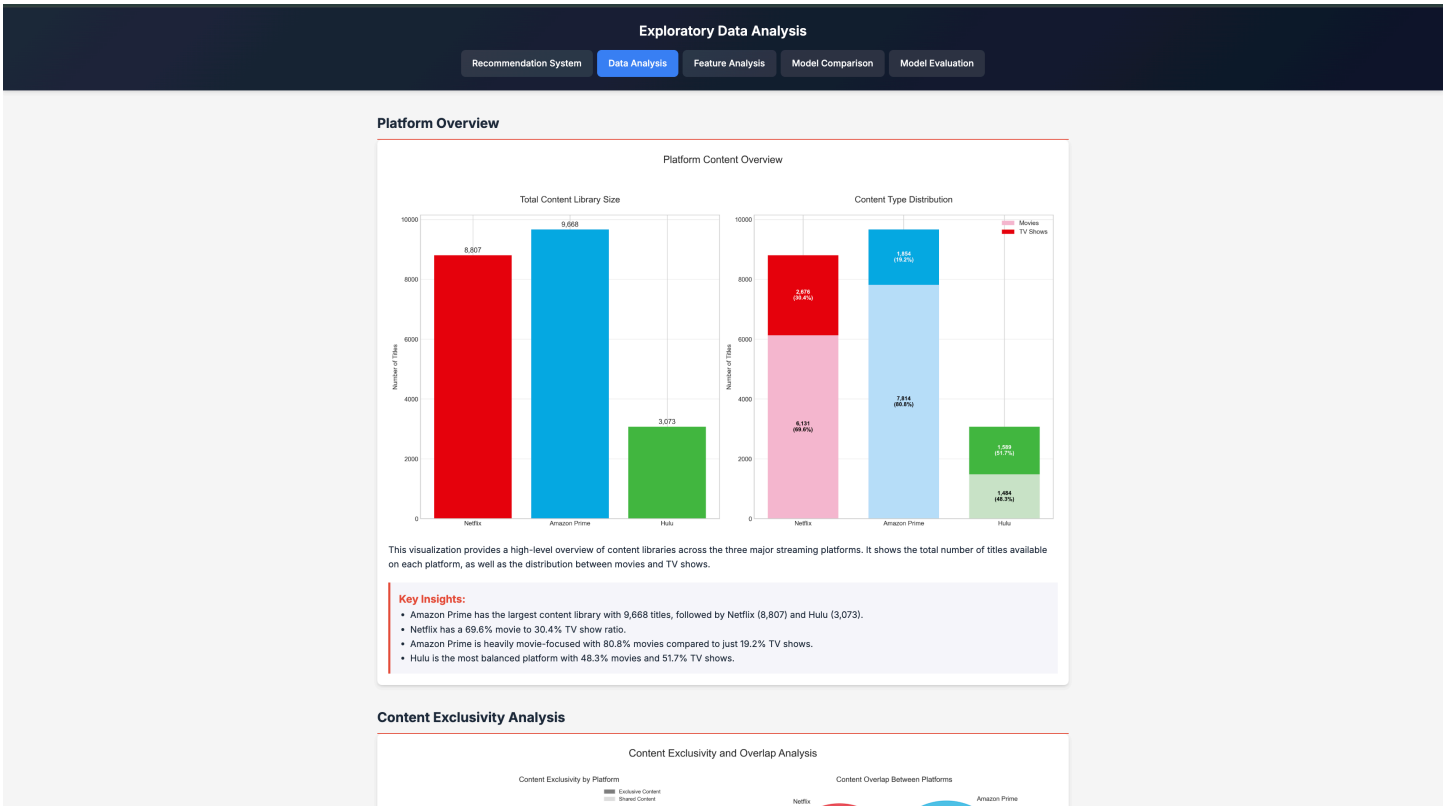
## Sample Model Predictions:

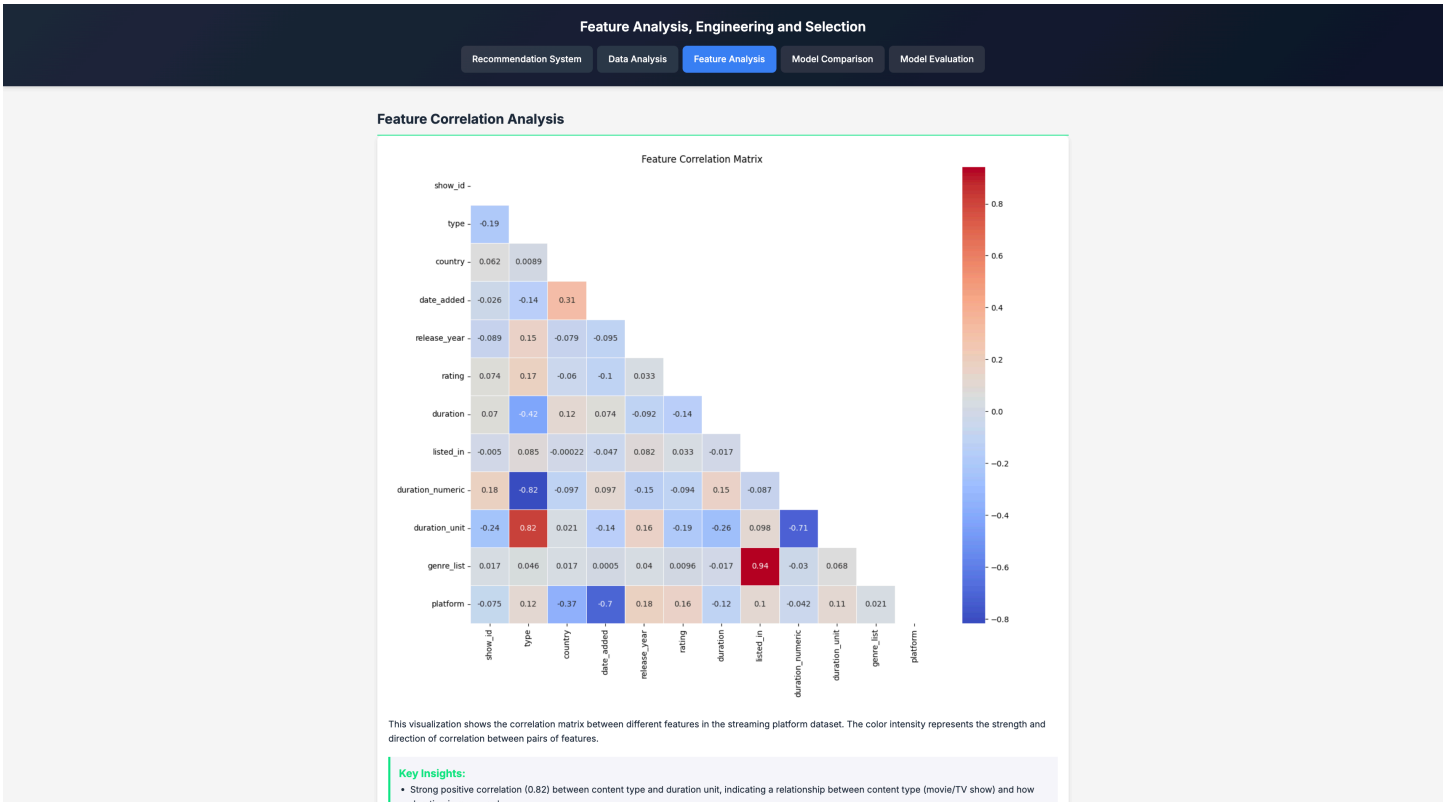
***Input Movie/TV Show - Predictions***

**Stranger Things** - Beyond Stranger Things, Nightflyers, The Umbrella Academy, Good Witch

**LEGO Marvel Super Heroes: Avengers Reassembled!** - Marvel Super Hero Adventures: Frost Fight!, Iron Man & Captain America: Heroes United, LEGO Marvel Super Heroes: Black Panther, Marvel's Hulk: Where Monsters Dwell

# UI Screenshots:



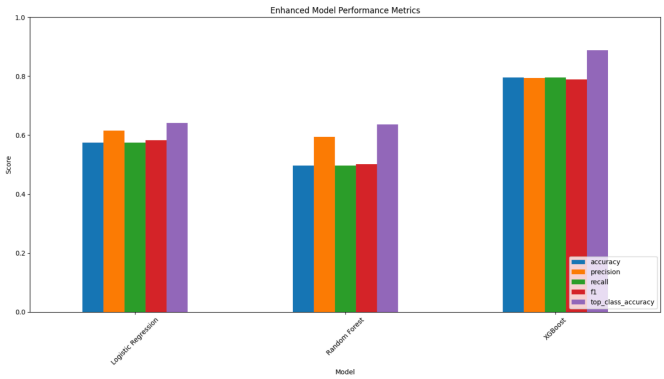
## Model Comparison and Selection

Recommendation System | Data Analysis | Feature Analysis | Model Comparison | Model Evaluation

### Model Performance Summary

**Performance Metrics Comparison**

| Model | Accuracy | Precision | Recall | F1 Score |
|---|---|---|---|---|
| Logistic Regression | 0.78 | 0.77 | 0.78 | 0.77 |
| Random Forest | 0.94 | 0.93 | 0.94 | 0.93 |
| XGBoost | 0.99 | 0.99 | 0.99 | 0.99 |




### Model-Specific Analysis

**XGBoost (Best Performing Model)**

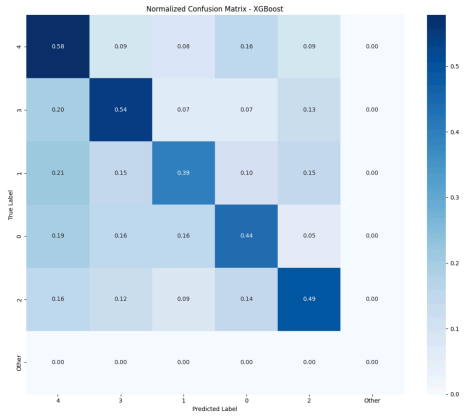
**Confusion Matrix**

## Model Evaluation and Performance Analysis

Recommendation System | Data Analysis | Feature Analysis | Model Comparison | Model Evaluation

XGBoost | Random Forest | Logistic Regression

**Confusion Matrices**




XGBoost Matrix Analysis:
The XGBoost model shows strong diagonal values (0.39–0.58), indicating good prediction accuracy across all classes. Class 4 has the highest accuracy at 0.58, while Class 1 has the lowest at 0.39. There is some confusion between adjacent classes, but overall the model maintains good class separation with minimal off-diagonal values.

### Feature Importance Analysis

XGBoost | Random Forest

**Feature Importance Rankings**