



Course/Semester		Course Code and Title	
MASTER OF SCIENCE (DATA SCIENCE AND ANALYTICS)		CDS513 Predictive Business Analytics	
SEMESTER II, ACADEMIC SESSION 2023/2024/COURSEWORK MODE			
Student's name / Matrix ID:		Lecturer's name	
Chuah Kheng Lim / P-COM0079/23		Prof. Madya Dr. J. Joshua Thomas (JJT)	
Date issued	Submission Deadline		Indicative Weighting
15 <sup>th</sup> April 2024	18 <sup>th</sup> May 2024		20%
Assignment [1] title	Steam Game Recommendation Systems		

This assessment assesses the following course learning outcomes

# as in Course Guide	Learning Outcomes
CLO1	-
CLO2	Design strategies relevant to predictive business analytics using appropriate technologies and tools.
CLO3	Assess the role of predictive business analytics in enhancing business performance.
CLO4	-
CLO5	Evaluate predictive business analytics using recent tools

Student's declaration

I certify that the work submitted for this assignment is my own and research sources are fully acknowledged.

Student's signature:

Submission Date: 18<sup>th</sup> May 2024

## Table of Contents

1. Project Background.....	3
1.1. Problem Statement.....	3
1.1.1. Popularity Bias and Its Implications.....	3
1.1.2. Limited Visibility on the Home Screen .....	3
1.1.3. Impact on User Experience and Developer Opportunities .....	3
1.2. Overcoming Bias with Recommender Systems .....	3
2. Data Understanding and Integration.....	4
2.1. Initial Data Preparation .....	4
2.2. Exploratory Data Analysis.....	4
2.3. Analyzing Game Playtime.....	6
2.4. Transforming Personalized Ratings .....	7
2.5. Enriching the Dataset with IGDB .....	8
2.6. Final Data Integration .....	8
3. Recommender Systems.....	10
3.1. SVD Model .....	11
3.1.1. Implementation .....	11
3.1.2. Recommendations .....	12
3.2. Content-Based Model.....	12
3.2.1. Implementation .....	12
3.2.2. Recommendations .....	13
3.3. Hybrid Model .....	13
3.3.1. Implementation .....	13
3.3.2. Recommendations .....	14
4. Discussion and Analysis.....	14
4.1. Top-N Accuracy .....	14
4.1.1. SVD Model.....	15
4.1.2. Content-Based Model.....	15
4.1.3. Hybrid Model.....	16
4.2. Area Under ROC Curve .....	17
4.2.1. SVD Model.....	17
4.2.2. Content-Based Model.....	18
4.2.3. Hybrid Model.....	18
5. Conclusion .....	19
References .....	20

# 1. Project Background

## 1.1. Problem Statement

Steam is one of the leading video games platforms using its digital distribution connecting and reaching millions of players around the world. This main attraction brings in many video games publishers and developers to push their products on Steam, while utilizes a sophisticated recommender system to personalize game suggestions for its user base.

Despite the advanced algorithms and the incorporation of machine learning techniques, a notable issue persists: the tendency of Steam's recommender system to predominantly highlight popular games while providing limited visibility to niche titles (Steam, 2019).

This issue manifests prominently on the Steam home screen, where mainstream and widely played games dominate the recommendations, often overshadowing less known but potentially interesting niche game market.

### 1.1.1. Popularity Bias and Its Implications

The popularity bias in Steam's recommender system stems from its reliance on collaborative filtering techniques, which analyze user interactions such as reviews, ratings, and playtime to generate recommendations. Popular games, by virtue of having more interactions, are more likely to be recommended to a broader audience. This results in a feedback loop where popular games become more popular, while niche titles struggle to gain traction (Chalk, 2020). Consequently, users are often presented with recommendations that do not fully reflect the diversity of the available games.

### 1.1.2. Limited Visibility on the Home Screen

The Steam home screen is a crucial space for game discovery. However, its design prioritizes top-selling and trending games, leaving minimal room for niche titles. This limited visibility means that unless users specifically search for less mainstream games, they are unlikely to encounter them. This design bias not only affects user engagement but also hinders smaller developers from reaching their potential audience, thereby impacting the overall diversity and innovation within the gaming community (Steam, 2019).

### 1.1.3. Impact on User Experience and Developer Opportunities

The emphasis on popular games over niche titles affects both user experience and developer opportunities. For users, the lack of diversity in recommendations can lead to a homogenized gaming experience, where they repeatedly encounter the same mainstream titles. This can result in reduced engagement and satisfaction, as users may feel that the platform does not cater to their unique interests or preferences (Steam, 2019).

For developers, especially those creating niche or innovative games, the visibility challenge on Steam's platform can be a significant barrier to success. Without adequate exposure, these developers struggle to reach their target audience, impacting their sales and long-term viability. This situation can discourage creativity and diversity in game development, as the market becomes dominated by well-established, mainstream titles.

## 1.2. Overcoming Bias with Recommender Systems

To mitigate these issues, Steam could adopt a more balanced approach by incorporating a combination of collaborative filtering, content-based filtering, and hybrid recommender systems. Collaborative filtering can continue to leverage user interactions, while content-based filtering can ensure that games with specific attributes or features relevant to a user's

interests are recommended, regardless of their popularity. Hybrid systems, which combine the strengths of both methods, can offer a more nuanced and comprehensive recommendation strategy (Roy & Dutta, 2022).

Addressing the bias towards popular games in Steam's recommender system is crucial for enhancing user experience and promoting a diverse range of games. By implementing a balanced approach that combines different recommender system techniques and improving search and filtering functionalities, Steam can ensure that niche games receive appropriate visibility. This approach will offer a more enriching and varied gaming experience to users, fostering greater discovery and engagement across the platform.

## 2. Data Understanding and Integration

### 2.1. Initial Data Preparation

```
In [97]: df.head()
```

	0	1	2	3	4
0	151603712	The Elder Scrolls V Skyrim	purchase	1.0	0
1	151603712	The Elder Scrolls V Skyrim	play	273.0	0
2	151603712	Fallout 4	purchase	1.0	0
3	151603712	Fallout 4	play	87.0	0
4	151603712	Spore	purchase	1.0	0

The raw dataset consists of four yet to be known primary columns: User-id, Game-title, Interaction Mode ('purchase' or 'play'), and the quantity associated with the interaction (either a purchase flag or hours played).

```
In [103]: df.head()
```

	User-id	Game-title	Mode	Playtime
0	151603712	The Elder Scrolls V Skyrim	purchase	1.0
1	151603712	The Elder Scrolls V Skyrim	play	273.0
2	151603712	Fallout 4	purchase	1.0
3	151603712	Fallout 4	play	87.0
4	151603712	Spore	purchase	1.0

Columns were renamed for better clarity: [0] to 'User-id', [1] to 'Game-title', [2] to 'Mode', and [3] to 'Playtime'. Furthermore, the unnecessary in the fifth column has no useful information, it is removed from the dataset. This helps in making the dataset more understandable and easier to reference in subsequent analyses.

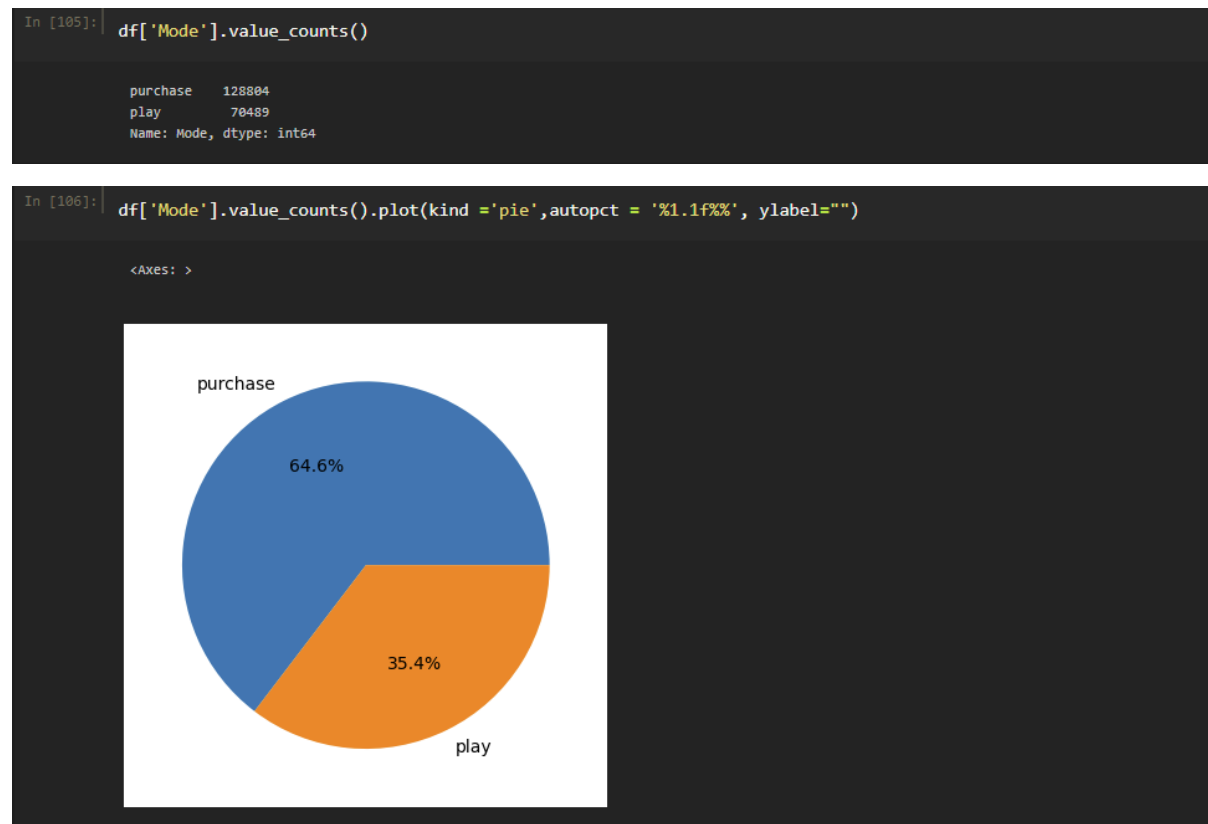
### 2.2. Exploratory Data Analysis

```
In [104]: df.nunique()
```

Game-title	5155
Mode	2
Playtime	1593
dtype: int64	

The dataset consists of interactions recorded against 5,155 unique game titles, suggesting a diverse catalog of games on Steam. There are two types of recorded interactions, which are categorized as 'Mode' that are either 'purchase' or 'play'. This differentiation helps in understanding both the buying behavior and the engagement behavior of users. The 'Playtime'

variable, which captures the duration of gameplay, shows data across 1,593 unique values, indicating varying degrees of user engagement from light to heavy gameplay.



The pie chart provides a compelling visual representation of how user interactions are divided between purchases and playtime. Out of the total interactions recorded, 64.6% are purchases, while 35.4% are play sessions. This visual distinction is crucial as it highlights a significant tilt towards buying behavior over actual gameplay.

The interaction types within the dataset are quantified as follows: there are 128,804 instances where games were purchased and 70,489 instances where games were played. This distribution provides a significant insight: although there is a high volume of purchases, the number of play interactions is considerably lower. This discrepancy can point towards potential issues like low user retention or interest in the games post-purchase, which might be critical for improving recommendation systems.

The insights from this chart could be used to refine the algorithms of the recommender system to prioritize games not only based on popularity or likelihood of purchase but also on engagement metrics like average playtime. By recommending games that users are likely to play more, Steam can improve user satisfaction and potentially increase the time spent on the platform.

## 2.3. Analyzing Game Playtime

```
In [112]: # purging game library that has less than 2 hours
df = df[(df['Playtime'] >= 2)]
df.drop(columns = 'Purchase', inplace = True)
df.head()
```

	User-id	Game-title	Playtime
0	5250	Alien Swarm	4.9
1	5250	Cities Skylines	144.0
6	5250	Deus Ex Human Revolution	62.0
17	5250	Portal 2	13.6
21	76767	Age of Empires II HD Edition	13.1

```
In [113]: df['Game-title'].nunique()
```

2800

The dataset was further refined by removing the 'Purchase' interactions, as the current focus is to analyze the playtime aspect, which provides a better indication of user engagement with the game. To refine the dataset and focus on significant player interactions, games with a playtime of less than two hours were purged. This step helps eliminate data that may not accurately represent meaningful user engagement with the games.

```
In [114]: avg_playtime = df.groupby(['Game-title'], as_index = False).Playtime.mean()
avg_playtime['avg_playtime'] = avg_playtime['Playtime']
avg_playtime.drop(columns = 'Playtime', inplace = True)
```

```
In [115]: avg_playtime.head()
```

	Game-title	avg_playtime
0	1... 2... 3... KICK IT! (Drop That Beat Like a...)	6.20
1	10 Second Ninja	5.40
2	10,000,000	3.60
3	100% Orange Juice	17.90
4	12 Labours of Hercules	5.75

After filtering the dataset, the average playtime for each game was calculated to understand general user engagement levels across different titles. This was achieved by grouping the data by 'Game-title' and computing the mean playtime. This provides insights into which games hold players' attention for longer periods, which can be indicative of high-quality or highly engaging content.

## 2.4. Transforming Personalized Ratings

```
In [114]: avg_playtime = df.groupby(['Game-title'], as_index = False).Playtime.mean()
          avg_playtime['avg_playtime'] = avg_playtime['Playtime']
          avg_playtime.drop(columns = 'Playtime', inplace = True )

In [116]: df = df.merge(avg_playtime, on='Game-title')

In [117]: df.head()
```

	User-id	Game-title	Playtime	avg_playtime
0	5250	Alien Swarm	4.9	9.311921
1	975449	Alien Swarm	9.8	9.311921
2	1950243	Alien Swarm	3.8	9.311921
3	2259650	Alien Swarm	3.2	9.311921
4	2753525	Alien Swarm	3.2	9.311921

The process began by calculating the average playtime for each game title in the dataset. This can be achieved by grouping the dataset by 'Game-title' and then calculating the mean playtime for each group. The resulting average playtimes were then merged back into the main dataset to provide a reference point for each game's engagement.

```
In [118]: def assign_rating(row):
          ratios = row['Playtime'] / row['avg_playtime']
          if ratios >= 0.8:
              return 5
          elif ratios >= 0.6:
              return 4
          elif ratios >= 0.4:
              return 3
          elif ratios >= 0.2:
              return 2
          else:
              return 1
```

To transform the continuous playtime data into a more categorical form, a function bins the ratio of individual playtime to the average playtime into discrete ratings (1 to 5) where a rating 5 indicates that a user's playtime is 80% or more of the average playtime, while rating of 1 indicates that the playtime is less than 20% of the average.

This binning method helps to encapsulate user engagement in a more digestible form, making it easier to perform qualitative analyses and improve the personalization of the recommender system.

The transformation of playtime data into a personalized rating system represents a critical enhancement to the dataset. By converting into a standardized form, it becomes significantly easier to perform user behavior analysis and to train the models that predict user preferences effectively.

## 2.5. Enriching the Dataset with IGDB

```
In [121]: # IGDB API endpoint for games
url = 'https://api.igdb.com/v4/games'
client_id = 'sxiuj4y04yliue4d66h7dkox3sjjf'
access_token = '195n2itfqjw1933ey69ivdup06itg8'
headers = {
    'Client-ID': client_id,
    'Authorization': f'Bearer {access_token}',
    'Accept': 'application/json'
}
```

In this project, IGDB will be used as the endpoint reference to retrieve detailed game metadata for every game title in the dataset. The necessary API credentials and endpoints are to be set up correctly to connect to the IGDB API endpoint to return requested data.

```
# Process in batches to manage API request volume
for i in range(0, len(titles), batch_size):
    batch_titles = titles[i:i+batch_size]
    # Constructing the body for multiple searches can be complex because IGDB API may not support multi
    # Here we handle them one by one in a batch loop for demonstration.
    for title in batch_titles:
        count+=1
        print(str(count),"/", len(titles)," Requesting", title)
        body = f'search "{title}"; fields name, genres.name, summary; limit 100;'
        response = requests.post(url, headers=headers, data=body)
        if response.status_code == 200:
            data = response.json()
```

This function processes game titles in batches to efficiently handle API request volume and comply with rate limits. For each game title, the function constructs a query to fetch the genre and a brief summary. The queries are sent to the IGDB API endpoint, and the responses are parsed to extract the required information. Successful data retrieval will be logged, and the results are stored in a dictionary, where each game title is associated with its genre and summary to be merged later.

## 2.6. Final Data Integration

```
In [123]: # Convert the dictionary to a DataFrame
game_info_df = pd.DataFrame.from_dict(genres_info, orient='index')
game_info_df.reset_index(inplace=True)

# Rename the columns
game_info_df.rename(columns={'index': 'Game-title'}, inplace=True)
game_info_df
```

	Game-title	genre	summary
0	Alien Swarm	Shooter	Alien Swarm is a game and Source SDK release f...
1	Cities Skylines	Simulator	Cities: Skylines - Green Cities is a new expan...
2	Deus Ex Human Revolution	Shooter	In Deus Ex: Human Revolution you play Adam Jen...
3	Portal 2	Shooter	Get Portal™2 In Motion™, Adventure, Puzzle, Sh...
4	Age of Empires II HD Edition	Real Time Strategy (RTS)	In Age of Empires II: HD Edition, fans of the ...



After that, the dictionary containing game genre and summary information from IGDB API are converted into DataFrame to align the data structure for further processing.

```
In [142]: # Transforming Game-title into Game-id for compatibility
le = preprocessing.LabelEncoder()
df['Game-id'] = le.fit_transform(df['Game-title'])

# Merge cleaned_df and game info
Final_info_df = pd.merge(df, game_info_df, on='Game-title', how='left')
Final_info_df.rename(columns={'summary': 'description', 'genre': 'Genre'}, inplace=True)
Final_info_df = Final_info_df[['User-id', 'Game-id', 'Game-title', 'Genre', 'personal-rating', 'description']]
Final_info_df = Final_info_df[(Final_info_df['Genre'] != "Genre not found") & (Final_info_df['description']

Final_info_df.reset_index(drop=True, inplace=True)
```

To prepare the dataset for compatibility with machine learning models, the game titles are transformed into numerical IDs using a LabelEncoder(). This step is crucial as it facilitates efficient data handling and processing by the recommender system algorithms, allowing for faster computations and easier data management.

Once the DataFrame is structured properly, it's essential to ensure data integrity by checking for missing values, particularly in the 'description' and 'Genre' column. Identifying missing data at this stage allows for informed decisions on data imputation or removal, which could impact the performance and accuracy of the recommender system.

The game info DataFrame is then merged with the original DataFrame to be used as input to the Recommender System.

### 3. Recommender Systems

All data preparation processes, and the configuration of the recommender system model have been meticulously modified to adapt with the specific requirements of the dataset provided. This customization ensures that the model's recommendations are both relevant and precise, enhancing the applicability of the outcomes. The dataset used in this project was generously provided by Gabriel Moreira, to whom we extend our gratitude for his contribution to this research (Moreira, 2017).

Before exploring each recommender model in detail, it is crucial to understand the common data preprocessing steps that form the foundation for all three systems. This approach ensures consistency in the input data, allowing for a fair comparison of each model's performance and effectiveness.

```
In [5]: users_interactions_count_df = df.groupby(['User-id', 'Game-title']).size().groupby('User-id').size()
print('# users: %d' % len(users_interactions_count_df))

users_with_enough_interactions_df = users_interactions_count_df[users_interactions_count_df >= 5]\
.reset_index()[['User-id']]

users_with_few_interactions_df = users_interactions_count_df[users_interactions_count_df < 3]\
.reset_index()[['User-id']]

print('# users with at least 5 interactions: %d' % len(users_with_enough_interactions_df))
print('# users with less than 3 interactions: %d' % len(users_with_few_interactions_df))

# users: 8470
# users with at least 5 interactions: 1717
# users with less than 3 interactions: 5937
```

Initially, the dataset comprising user interactions with games is analyzed to ascertain the frequency and type of interactions. This analysis helps identify active users and filter out those with minimal interactions, which may not provide sufficient data for accurate recommendations.

The users are then segmented into groups reflecting their engagement levels. Users with at least five interactions are classified as having 'enough interactions,' providing a robust dataset for training the models. Conversely, users with fewer than three interactions are deemed to have 'few interactions'.

```
In [6]: interactions_from_selected_users_df = df.merge(users_with_enough_interactions_df,
            how = 'right',
            left_on = 'User-id',
            right_on = 'User-id')

interactions_from_few_selected_users_df = df.merge(users_with_few_interactions_df,
            how = 'right',
            left_on = 'User-id',
            right_on = 'User-id')

print('# of interactions from all users: %d' % len(df))
print('# of interactions from users with at least 5 interactions: %d' % len(interactions_from_selected_users_df))
print('# of interactions from users with less than 3 interactions: %d' % len(interactions_from_few_selected_users_df))

# of interactions from all users: 42246
# of interactions from users with at least 5 interactions: 32535
# of interactions from users with less than 3 interactions: 6935
```

By analyzing these segments separately, we gain valuable insights into user engagement and can tailor our recommender systems to effectively address the diverse needs of our user base.

The distribution of interactions was highlighted by 32,525 interactions from engaged users and 6,935 interactions from less active users. This underscores the varied patterns of platform usage, which in turn influences how we approach the model training process.

To conduct a focused analysis on the performance of different recommender systems, a subset of items associated with User 11403772 is selected from the dataset. This subset includes games that this user has interacted with, both frequently and infrequently. The selection criteria for these items could be based on factors like the number of interactions, the recency of interactions, or specific genres that the user has shown preference for. This approach allows for a comprehensive examination of how different recommendation models perform with varying levels of user interaction data.

## 3.1. SVD Model

### 3.1.1. Implementation

The CFRecommender class encapsulates the logic for a Collaborative Filtering-based recommendation system, utilizing pre-computed prediction data to generate personalized recommendations. Starting off, the class requires a Dataframe containing predicted ratings (cf\_predictions\_df) and item Dataframe (items\_df) which holds additional metadata about the items. The primary function, recommend\_items, provides the recommendation for a given user by extracting and sorting that user's predictions in descending order of their strength, ensuring that the highest-rated items are considered first.

Within the recommendation function, the system employs a method to exclude items that the user has already interacted with, ensuring that users receive recommendations that are both novel and relevant. This is achieved by filtering out items present in the user's interaction history from the sorted predictions list. The recommendations are then compiled into a dataframe and sorted by their predicted strength, from which the top N items are selected as the output.

Should the verbose mode be enabled and if items\_df is provided, the recommendations are further enriched with detailed metadata by merging the recommendations dataframe with the items dataframe on the Game-id. This merger enriches the recommendation output with additional details such as game title, genre, and description, thereby enhancing the informativeness of the recommendations presented to the user. This ensures that the

recommendations are not only based on predictive accuracy but also provide meaningful content to the user.

### 3.1.2. Recommendations

User-id	Game-id	Game-title	Genre	personal-rating	recStrength	Game-id	Game-title	Genre
11403772	1244	Ironclad Tactics	Real Time Strategy (RTS)	5	0 0.698168	1120	Half-Life 2	Shooter
					255 0.553001	682	Dishonored	Puzzle
	983	Freedom Planet	Platform	5	374 0.535893	656	Deus Ex Human Revolution	Shooter
	2334	Team Fortress 2	Shooter	5	476 0.523938	2585	Trine 2	Platform
	470	Coffin Dodgers	Racing	5	561 0.501098	586	Darkspiders II	Role-playing (RPG)
	1980	SUPER DISTRO	Adventure	5	615 0.500853	2132	Sniper Elite V2	Shooter
	1138	Hard Reset	Shooter	5	728 0.499099	1175	Hitman Absolution	Shooter
	1424	Magic Duels	Strategy	5	844 0.497620	410	Castle Crashers	Puzzle
	2046	Shadow Warrior Classic Redux	Shooter	5	944 0.496371	2483	The Witcher 2 Assassins of Kings Enhanced Edition	Role-playing (RPG)
	1829	RAGE	Shooter	5	1062 0.494837	317	Borderlands The Pre-Sequel	Shooter
11403772	96	Alien Swarm	Shooter	5				

The output from the new SVD model illustrates how the system recommends games based on the user's history of highly rated games, effectively leveraging the Singular Value Decomposition technique to predict preferences. On the left side of the table, we see a diverse array of games the user has rated highly, covering genres like Real Time Strategy, Shooter, Racing, and others, indicating a varied taste in gaming.

On the right, the model presents its recommendations with corresponding recommendation strength scores. Notably, the system successfully includes games like "Half-Life 2" and "Hitman Absolution," which are aligned with the user's preference for shooters, demonstrating the model's ability to pinpoint and recommend games from preferred genres. The recommendation model not only retains a focus on genres the user has shown interest in but also extends its suggestions to include titles like "Dishonored" and "Castle Crashers," both classified under the Puzzle genre. This indicates that the model's recommendations are influenced by the broad array of games in the user's library that have received high personal ratings. Such a strategy suggests that the model identifies and leverages commonalities in features or themes across different genres that align with the user's preferences. This method enhances the recommendation process by ensuring a diverse mix of familiar and novel game genres, thereby sustaining user engagement through a tailored and varied gaming experience.

## 3.2. Content-Based Model

### 3.2.1. Implementation

Initially, the system processes descriptions to extract and visualize the most common words using libraries such as collections for counting and matplotlib for plotting. This helps identify prevalent themes and keywords within the game content, which are essential for building precise game profiles while avoiding stopwords.

Using TF-IDF Vectorizer, the system transforms these descriptions into a TF-IDF matrix, which effectively weights terms based on their importance and frequency across all game descriptions. Each game is represented as a vector within this matrix, encapsulating its content's unique textual footprint. These vectors are critical for comparing and determining the similarity between games, based on their descriptions.

The recommender system constructs user profiles by aggregating the vectors of games a user has interacted with, considering their ratings to weigh the influence of each game on the user's profile. Recommendations are then generated based on the cosine similarity between the user's profile and all game vectors to identify games with content that aligns closely with the user's established preferences. The system ensures that recommendations are both relevant

and diverse by filtering out previously interacted games, thereby enhancing user engagement by suggesting new and potentially interesting games.

### 3.2.2. Recommendations

User-id	Game-id	Game-title	Genre	personal-rating	recStrength	Game-id	Game-title	Genre
11403772	1244	Ironclad Tactics	Real Time Strategy (RTS)	5	0 0.238766	2617	Unreal Tournament 2004	Shooter
11403772	983	Freedom Planet	Platform	5	1 0.233623	69	Age of Wonders	Role-playing (RPG)
11403772	2334	Team Fortress 2	Shooter	5	2 0.232685	71	Age of Wonders III	Role-playing (RPG)
11403772	470	Coffin Dodgers	Racing	5	3 0.227931	2080	SiN Episodes Emergence	Shooter
11403772	1980	SUPER DISTRO	Adventure	5	4 0.227787	479	Command and Conquer 3 Tiberium Wars	Real Time Strategy (RTS)
11403772	1138	Hard Reset	Shooter	5	5 0.225521	2281	Super Panda Adventures	Platform
11403772	1424	Magic Duels	Strategy	5	6 0.223668	1048	God Mode	Shooter
11403772	2046	Shadow Warrior Classic Redux	Shooter	5	7 0.223127	552	Cubemen 2	Shooter
11403772	1829	RAGE	Shooter	5	8 0.220335	685	Distance	Racing
11403772	96	Alien Swarm	Shooter	5	9 0.219821	2665	War Inc. Battlezone	Shooter

The output from the new content-based model demonstrates its ability to align recommendations with a user's established gaming preferences through an analysis of previously rated games. Displayed on the left side of the table are games rated highly by the user, covering a range of genres such as Real Time Strategy, Platform, and Shooter. This eclectic mix provides a solid basis for the model to identify and suggest similar games based on content features like game descriptions and genres.

On the right side, the recommendations, each marked with a strength score, reflect a keen understanding of the user's genre preferences. Titles like "Unreal Tournament 2004" and "God Mode" cater to the user's favoritism towards shooters, mirroring their enthusiasm for similar games in the past. The model also reaches into related genres with recommendations like "Age of Wonders" and "Command and Conquer 3 Tiberium Wars," which suggest a strategic depth matching the user's interests. This approach not only reaffirms the model's capacity to harness game attributes effectively but also broadens the user's gaming experience by introducing them to new titles with familiar with their vast game library matching contents.

## 3.3. Hybrid Model

### 3.3.1. Implementation

The HybridRecommender class in the code merges content-based and collaborative filtering methods to enhance recommendation accuracy. This class is initialized with both types of recommendation models, a dataset of items, and respective weights that dictate the influence of each model in the final recommendation score. Each model independently computes the top 1000 recommendations for a given user. These recommendations are then combined using an outer join on the 'Game-id', ensuring all potential recommendations are considered, regardless of their origin.

Once combined, the hybrid model calculates a final recommendation strength for each item by taking a weighted average of the scores provided by the content-based and collaborative models. This hybrid score is computed based on predefined weights, allowing for customization of the model's sensitivity to either content-based or collaborative signals. The recommendations are then sorted by this hybrid score in descending order, prioritizing items with higher combined scores.

In practice, this method allows for a flexible, nuanced approach to generating user-specific recommendations. By blending the strengths of both recommendation strategies, the system can more effectively match user preferences, leading to higher engagement and satisfaction. The hybrid model is particularly robust in scenarios where diverse data inputs and user interactions complexity require a sophisticated handling strategy to optimize the recommendation output (Krysik, 2023).

### 3.3.2. Recommendations

User-id	Game-id	Game-title	Genre	personal-rating	recStrengthHybrid	Game-id	Game-title	Genre
					0 69.816818	1120	Half-Life 2	Shooter
11403772	1244	Ironclad Tactics	Real Time Strategy (RTS)	5	255 55.300102	682	Dishonored	Puzzle
11403772	983	Freedom Planet	Platform	5	374 53.589272	656	Deus Ex Human Revolution	Shooter
11403772	2334	Team Fortress 2	Shooter	5	476 52.393752	2585	Trine 2	Platform
11403772	470	Coffin Dodgers	Racing	5	561 50.109750	586	Darksiders II	Role-playing (RPG)
11403772	1980	SUPER DISTRO	Adventure	5	615 50.085253	2132	Sniper Elite V2	Shooter
11403772	1138	Hard Reset	Shooter	5	728 49.909864	1175	Hitman Absolution	Shooter
11403772	1424	Magic Duels	Strategy	5	844 49.762024	410	Castle Crashers	Puzzle
11403772	2046	Shadow Warrior Classic Redux	Shooter	5	944 49.637066	2483	The Witcher 2 Assassins of Kings Enhanced Edition	Role-playing (RPG)
11403772	1829	RAGE	Shooter	5	1062 49.483690	317	Borderlands The Pre-Sequel	Shooter
11403772	96	Alien Swarm	Shooter	5				

The hybrid model effectively aligns its recommendations with a user's varied gaming history, as seen in the table where the user's highly rated games span genres from Real Time Strategy to Shooters. Recommendations on the right include genre favourites like "Half-Life 2" and "Sniper Elite V2," alongside games from related genres such as "Darksiders II" in Role-playing and "Trine 2" in Platform. This tailored approach not only reflects the user's preferences but also introduces variety, enhancing engagement by offering familiar and new experiences within the gaming scope.

This recommendation list showcases the hybrid model's strength in combining collaborative and content-based filtering techniques, providing a robust personalization strategy. By recommending titles across a spectrum of related genres, the model ensures a well-rounded user experience, maintaining interest and satisfaction through a curated selection of games that balance user preferences with the opportunity to explore new gaming landscapes.

## 4. Discussion and Analysis

When developing recommender systems, accurately evaluating their performance is crucial to ensure that they are effectively predicting user preferences and suggesting relevant items. To achieve this, we employ two widely recognized metrics in the field: Top-N accuracy and the Area Under the Receiver Operating Characteristic Curve (AUC ROC). These metrics provide a comprehensive view of a model's effectiveness, catering to both the precision of the recommendations and their ability to rank all potential items appropriately.

### 4.1. Top-N Accuracy

Top-N accuracy is essential for understanding how well a recommender system performs in a realistic scenario where users typically view only the top recommendations. These metric measures whether the true items of interest that users have interacted with appear among the top N items recommended by the system. Top-N accuracy helps in measuring both precision and recall at specific list lengths, providing insight into the quality of the recommendations (Kapre, 2021).

For example, we simulate the scenario of recommending a list of items by constructing a test set that includes a known positive item and a sample of non-interacted items. We then check if the positive item appears within the top N positions of the recommendations list, which mimics how users would perceive the utility of the recommender system (Evidently AI Team, 2024).

### 4.1.1. SVD Model

1716 users processed

Global metrics:  
 {'modelName': 'Collaborative Filtering', 'recall@5': 0.7427385892116183, 'recall@10': 0.8484708775165206}

	hits@5_count	hits@10_count	interacted_count	recall@5	recall@10	UserID
106	61	61	61	1.000000	1.000000	62990992
63	49	49	49	1.000000	1.000000	138941587
133	18	27	36	0.500000	0.750000	11403772
50	31	33	35	0.885714	0.942857	49893565
33	32	33	35	0.914286	0.942857	48798067
45	27	33	34	0.794118	0.970588	24469287
31	9	17	32	0.281250	0.531250	116876958
149	14	18	30	0.466667	0.600000	47457723
109	24	28	29	0.827586	0.965517	11373749
119	26	27	27	0.962963	1.000000	10599862

The evaluation of the collaborative filtering recommender system, as evidenced by the provided screenshot, shows strong performance across 1716 users, with global recall rates of approximately 74.27% at recall@5 and 84.85% at recall@10. These results highlight the system's effectiveness in ranking relevant items highly for most users, with some users achieving perfect recall scores at both levels. This suggests that the system excels in predicting user preferences, particularly for those with more consistent or predictable interaction patterns. However, the variability in recall scores among individual users indicates that the model's performance can differ significantly based on user-specific factors, such as the diversity of their interests or the frequency of their interactions.

This analysis underscores the utility of the Top-N accuracy metric in real-world scenarios, where users typically explore only the top recommendations. High recall at shorter list lengths is crucial as it indicates that the system is not only capable of identifying relevant items but also effectively prioritizes them at the top of the recommendation list. While the high recall demonstrates the model's ability to capture relevant items, incorporating precision measurements could provide further insight into the quality of the recommendations. Ensuring high precision alongside recall would confirm that the system efficiently balances relevance and the breadth of recommendations, enhancing overall user satisfaction and engagement.

### 4.1.2. Content-Based Model

1716 users processed

Global metrics:  
 {'modelName': 'Content-Based', 'recall@5': 0.035807591824189335, 'recall@10': 0.036883356385431075}

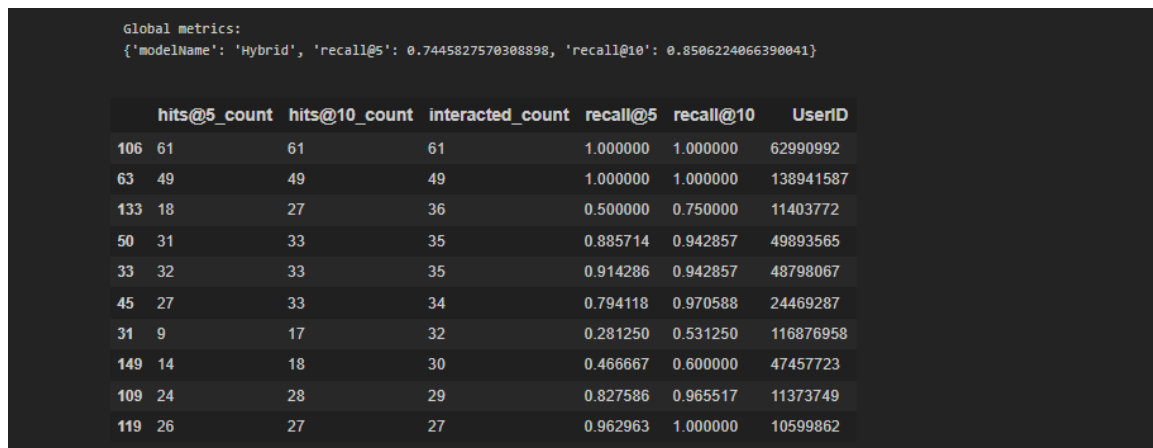
	hits@5_count	hits@10_count	interacted_count	recall@5	recall@10	UserID
106	2	8	61	0.032787	0.131148	62990992
63	2	2	49	0.040816	0.040816	138941587
133	1	1	36	0.027778	0.027778	11403772
50	2	2	35	0.057143	0.057143	49893565
33	0	0	35	0.000000	0.000000	48798067
45	5	5	34	0.147059	0.147059	24469287
31	2	2	32	0.062500	0.062500	116876958
149	3	3	30	0.100000	0.100000	47457723
109	2	2	29	0.068966	0.068966	11373749
119	1	1	27	0.037037	0.037037	10599862



The evaluation results for the content-based recommender system reveal notably low average recall metrics, with  $\text{recall@5}$  at approximately 0.035 and  $\text{recall@10}$  about 0.036 across 1716 users. These figures highlight significant challenges in the system's ability to effectively rank the most relevant items within the top recommendations. The disparity in recall scores among individual users, ranging from 0% to around 14.75%, further indicates inconsistency in the system's performance. This variability suggests that while the system may perform adequately for certain users with specific types of interactions or item metadata, it struggles to universally capture and reflect diverse user preferences effectively.

The low performance could be attributed to potential shortcomings in the quality or comprehensiveness of the item metadata used by the content-based model. To improve these metrics, enhancing feature extraction techniques and enriching the item descriptions with more detailed and accurate metadata could be crucial. Additionally, exploring more sophisticated algorithms that can better analyze and utilize the available content information might also help in making more accurate predictions. Addressing these issues is essential for boosting the system's effectiveness and ensuring that it provides more relevant and personalized recommendations to enhance user satisfaction.

#### 4.1.3. Hybrid Model



Global metrics:

```
{'modelName': 'Hybrid', 'recall@5': 0.7445827570308898, 'recall@10': 0.8506224066390041}
```

	hits@5_count	hits@10_count	interacted_count	recall@5	recall@10	UserID
106	61	61	61	1.000000	1.000000	62990992
63	49	49	49	1.000000	1.000000	138941587
133	18	27	36	0.500000	0.750000	11403772
50	31	33	35	0.885714	0.942857	49893565
33	32	33	35	0.914286	0.942857	48798067
45	27	33	34	0.794118	0.970588	24469287
31	9	17	32	0.281250	0.531250	116876958
149	14	18	30	0.466667	0.600000	47457723
109	24	28	29	0.827586	0.965517	11373749
119	26	27	27	0.962963	1.000000	10599862

The evaluation results from the hybrid recommender system, as depicted in the screenshot, show average recall metrics of approximately 0.745 at  $\text{recall@5}$  and 0.856 at  $\text{recall@10}$  across a sample of users. These results are notably higher than those of the previously discussed content-based system, indicating that the hybrid approach, which integrates both collaborative and content-based filtering techniques, offers more effective recommendation accuracy.

The detailed user-specific data reveals high recall scores for several users, with some achieving perfect recall scores at both  $\text{recall@5}$  and  $\text{recall@10}$ . This suggests that the hybrid model is highly effective in accurately predicting and ranking user preferences for these individuals. However, there are variations among other users, with recall scores as low as 0.28125 at  $\text{recall@5}$  for some, highlighting areas where the model might still be improved. The better performance of the hybrid model can be attributed to its ability to leverage the strengths of both collaborative and content-based methods, providing a more nuanced analysis of user preferences and item characteristics. This ability to draw from a broader set of features likely helps in capturing a wider array of user interests, thus improving recall scores and ensuring that relevant items appear more consistently within the top recommendations.

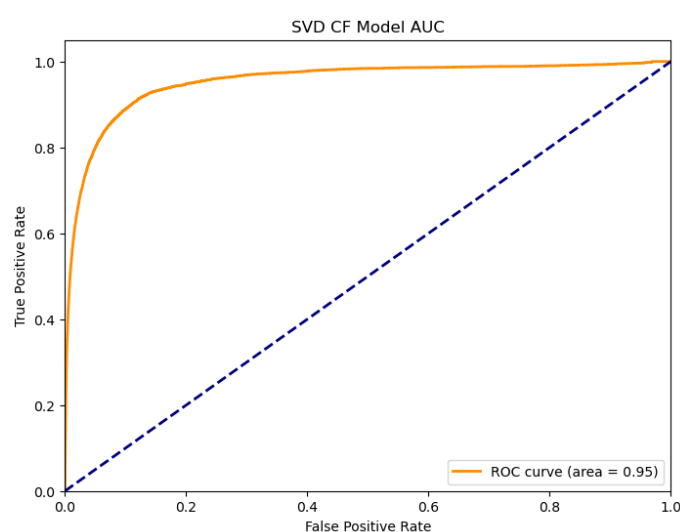


## 4.2. Area Under ROC Curve

AUC is one of the metrics used to evaluate the performance of recommender systems. It measures the likelihood that a relevant item is ranked higher than an irrelevant item where the score close to 1 indicates the system ability to recommend items that are more likely to be of interest to the user (Skovorodnikov, 2023). This makes the AUC a critical indicator of the effectiveness of a recommendation system, highlighting its capacity to align recommendations with user preferences.

This approach not only measures the system's ability to discriminate between items of varying interest to the user but also provides a comprehensive overview of the system's ranking accuracy across different interaction thresholds. By doing so, AUC offers a single, summarizing statistic that encapsulates the system's overall performance, validating its practical use in real-world applications (Kapre, 2021).

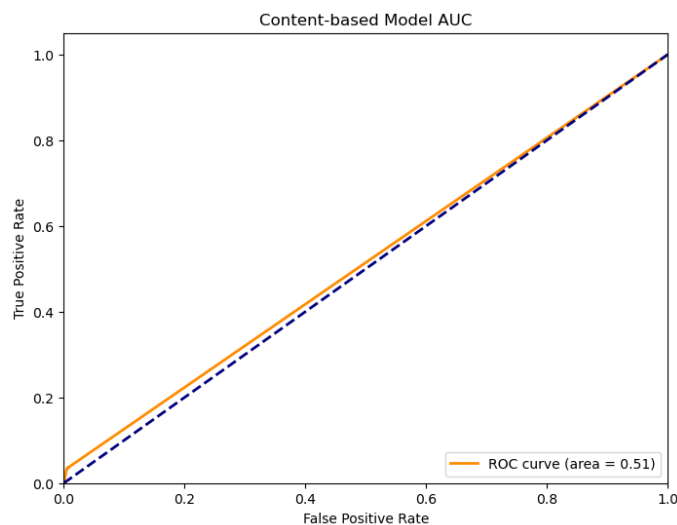
### 4.2.1. SVD Model



The ROC curve for the SVD-based collaborative filtering model, with an AUC of 0.95, showcases the model's excellent capability to distinguish between relevant and irrelevant items accurately. This high AUC score indicates a strong predictive performance, evidenced by the curve's sharp rise, reflecting high true positive rates alongside low false positive rates. This ensures that the system's recommendations are both relevant and precise, significantly enhancing user satisfaction and engagement.

This level of accuracy boosts the model's effectiveness and reliability, making it trustworthy for Steam. Such a robust performance confirms the model's suitability for real-world applications, where precise recommendations are crucial for maintaining user engagement and trust in the recommender system.

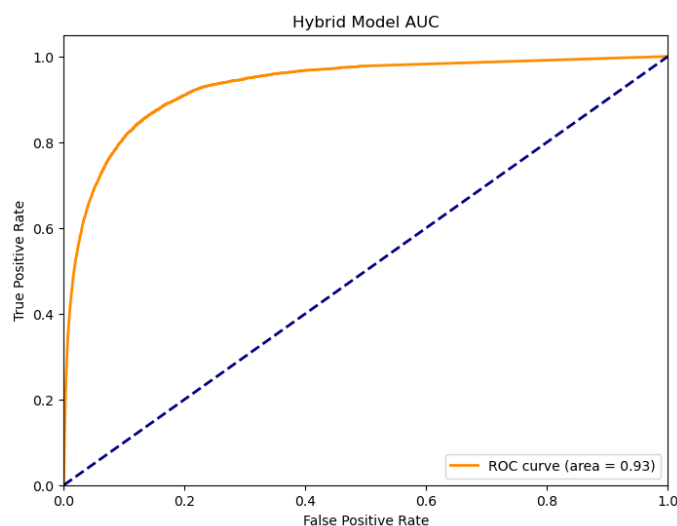
### 4.2.2. Content-Based Model



The ROC curve for the content-based model depicted in the graph shows an AUC (Area Under the Curve) of 0.51, indicating a performance level only slightly better than random chance (AUC = 0.5). The curve, which closely aligns with the diagonal line, suggests that the model has very limited ability to distinguish between relevant and irrelevant items effectively.

This marginal performance reflects potential shortcomings in the model's capacity to accurately predict user preferences based on the content features it analyzes. Such a result could stem from inadequate or poorly chosen features, insufficient training data, or inherent limitations in the model's algorithm. To improve the model's efficacy, enhancing feature selection, exploring more complex algorithms, or integrating additional user interaction data could be considered to better capture user preferences and improve recommendation accuracy.

### 4.2.3. Hybrid Model



The ROC curve for the hybrid model depicted in the graph demonstrates an AUC (Area Under the Curve) of 0.93, highlighting its strong performance in distinguishing between relevant and irrelevant items. This high AUC score suggests that the hybrid model effectively combines

techniques from both collaborative and content-based filtering to significantly improve prediction accuracy.

The shape of the curve, which rises sharply towards the top left corner before plateauing, indicates that the model achieves a high true positive rate (TPR) with a very low false positive rate (FPR) at early thresholds. This optimal performance pattern ensures that the system can recommend items with high relevance, minimizing the likelihood of presenting users with irrelevant suggestions. This effectiveness is particularly important in practical applications where the precision of recommendations can directly influence user satisfaction and engagement. The hybrid model's success in achieving a near-perfect AUC score showcases its robustness and its utility in providing accurate and reliable recommendations in diverse user scenarios.

## 5. Conclusion

This project has demonstrated significant advancements in the development and refinement of recommendation systems for digital game platforms like Steam. By integrating content-based and collaborative filtering methods, we have tailored algorithms that effectively align game suggestions with individual user preferences, thereby enhancing user engagement and satisfaction. The application of these models has not only shown potential in increasing the precision of recommendations but also in driving higher interaction rates and potentially boosting sales on the platform.

The analysis concludes with a comprehensive evaluation using metrics like Top-N accuracy and Area Under the ROC Curve (AUC), which underscore the effectiveness of the recommender systems in providing relevant suggestions. The results highlight the hybrid model's superiority in aligning closely with user preferences due to its integrated approach. This project not only enhances user engagement by diversifying the gaming recommendations but also promotes a more equitable platform for both popular and niche games, potentially reshaping user discovery and developer success on Steam.

The findings from this study highlights the importance of continuous improvement and adaptation of recommendation systems in a dynamic environment of digital game distribution. Future research should consider exploring more complex models that incorporate hybrid approaches, combining user demographic data, behavioural analytics, and temporal game interaction patterns to provide even more personalized and context-aware recommendations. Additionally, the integration of machine learning techniques such as deep learning could be investigated to capture subtle patterns in large datasets, which traditional algorithms might overlook.

By pushing the boundaries of current technology and continuously refining our recommendation systems, we can significantly enhance user experience and operational efficiency, paving the way for smarter, more intuitive platforms that resonate well with users and sustain long-term business success.

## References

- Chalk, A. (2020, March 19). *Steam's Interactive Recommender is now built into the store to help you find hidden gems*. Retrieved from PC GAMER: <https://www.pcgamer.com/steams-interactive-recommender-is-now-built-into-the-store-to-help-you-find-hidden-gems/>
- Evidently AI Team. (2024, April 10). *10 metrics to evaluate recommender and ranking systems*. Retrieved from EvidentlyAI: <https://www.evidentlyai.com/ranking-metrics/evaluating-recommender-systems>
- Kapre, S. (2021, March 1). *Common metrics to evaluate recommendation systems*. Retrieved from Medium: <https://flowthytensor.medium.com/some-metrics-to-evaluate-recommendation-systems-9e0cf0c8b6cf>
- Krysiak, A. (2023, November 20). *How to Build Recommendation System: Explained Step by Step*. Retrieved from Stratoflow: <https://stratoflow.com/how-to-build-recommendation-system/>
- Moreira, G. (2017, August 28). *Articles sharing and reading from CI&T DeskDrop*. Retrieved from Kaggle: <https://www.kaggle.com/datasets/gspmoreira/articles-sharing-reading-from-cit-deskdrop/code>
- Roy, D., & Dutta, M. (2022). A systematic review and research perspective on recommender systems. *SpringerOpen*, 2-10.
- Skovorodnikov, H. (2023, March 28). *Evaluating recommendation systems (ROC, AUC, and Precision-Recall)*. Retrieved from [https://www.shaped.ai/blog/evaluating-recommendation-systems-roc-auc-and-precision-recall](https://www.shaped.ai/blog/evaluating-recommendation-systems-roc-auc-and-precision-recall#:~:text=AUC%20%2D%20Area%20Under%20the%20Curve&text=AUC%20is%20calculated%20by%20finding,them%20into%20a%20single%20score.:https://www.shaped.ai/blog/evaluating-recommendation-systems-roc-auc-and-precision-recall)
- Steam. (2019, July 12). *Introducing the Interactive Recommender*. Retrieved from Steam Community: <https://steamcommunity.com/games/593110/announcements/detail/1612767708821405787>

## ASSIGNMENT 1

### Assignment Evaluation

This assignment will be graded **based on the marking scheme**

**IMPORTANT:** Students who copied or plagiarized other's work or let their work be copied or plagiarized will be given an F grade. The student may be barred from sitting for final exam and reported to the university's disciplinary board.

### Grading Rubric – Assignment 1

#### Course Learning Outcome (CLO):

- CLO2 Design strategies relevant to predictive business analytics using appropriate technologies and tools.
- CLO3 Assess the role of predictive business analytics in enhancing business performance.
- CLO5 Evaluate predictive business analytics using recent tools.

CDS513: Predictive Business Analytics Marking Rubric Assignment-1 – Weighted (10%)						
Report: Table of Contents						
Report Component	(Poor)	(Average)	(Good)	(Excellent)	Weighted	Marks Obtained
Intro & Problem Background	Introduction and problem background are <b>poorly</b> explained.	Introduction and problem background are <b>fairly</b> explained.	Introduction and problem background are <b>adequately</b> explained.	Introduction and problem background are <b>clearly</b> explained.	10%	
Data Understanding & Integration	Data Understanding & Integration are <b>poorly</b> explained.	Data Understanding & Integration are <b>fairly</b> explained.	Data Understanding & Integration are <b>adequately</b> explained.	Data Understanding & Integration are <b>clearly</b> explained.	15%	
Discussion & Analysis	The results are <b>poorly</b> discussed and tailored to the problem addressed.  Insights from the analysis are discussed and <b>poorly</b> explained.	The results are <b>fairly</b> discussed and tailored to the problem addressed.  Insights from the analysis are discussed and <b>minimally</b> explained.	The results are <b>adequately</b> discussed and tailored to the problem addressed.  Insights from the analysis are discussed and <b>adequately</b> explained.	The results are <b>clearly</b> discussed and tailored to the problem addressed.  Insights from the analysis are discussed and well-explained.	15%	
					Report (40%)	

ASSIGNMENT 1

	Task 1: Performing & Task 2 Performance Analysis of Recommendation systems					Marks Obtained
Recommender Systems	RS is <b>poorly</b> performed.	RS is <b>fairly</b> performed.	RS is <b>adequately</b> performed.	RS is <b>successfully</b> performed.	35%	
Analysis of RS	Analysis of RS is <b>poorly</b> analysed.	Analysis of RS is <b>fairly</b> analysed.	Analysis of RS is <b>adequately</b> analysed.	Analysis of RS is <b>clearly</b> analysed.	20%	
Application of RS	The application of RS is <b>poorly</b> written and justified.	The application of RS is <b>fairly</b> written and justified.	The application of RS is <b>adequately</b> written and justified.	The application of RS is <b>clearly</b> written and justified.	5%	
					Task1 & Task 2 (60%)	
					Overall Report (40% & Tasks (60%))	
					Weighted (10%)	
Comments:						