Ministry of Science and Higher Education of the Republic of Kazakhstan
L.N. Gumilyov Eurasian National University

Faculty of Information Technology
Department of Information Systems

COURSEWORK

ON THE SUBJECT
"Mathematical Foundations of Intelligent Systems"
For third-year students of the specialty 6B06103 – Information Systems

Topic: **Building a recommendation system for e-commerce based on user behavior data**

Completed by:

Student of group IS-33

Zhumabek Akbayan

Coursework Supervisor:

Prof., Zhukabayeva T. K.

Full name, signature

Members of the Commission:

Assoc. Prof. Muhanova A.A.

Full name, signature

PhD, Serikbayeva S.K.

Full name, signature

Full name, signature

Grade
«_____» _____2024

Astana, 2024

Content

Introduction

In today's world, where the number of available movies and shows is growing day by day, users are faced with the problem of choosing the right content. In a sea of movies, shows and new releases, it is difficult to find exactly what each person is interested in. This is where recommendation systems that use user behavior to provide personalized recommendations come to the rescue. The goal of this project is to develop a movie recommendation system that analyzes data on user actions and suggests the most appropriate movies.

Recommendation systems have long been an integral part of digital platforms such as Netflix, YouTube, and Amazon. They help users quickly navigate huge catalogs of content by suggesting exactly the movies and TV series they might like. The technology is based on analyzing historical data about users' preferences, what movies they have watched, what ratings they gave and what genres they prefer. In this way, the system can understand each user's preferences and make customized suggestions [9].

The goal of this project is to develop such a movie recommendation system using user behavior data. We will work with data on movie ratings, genres, number of views and other tags that will allow us to create accurate recommendations. The system will be based on various machine learning models, such as collaborative filtering, which is based on analyzing user similarities, and content filtering, which uses information about the movies themselves to create recommendations [12].

This task is important not only in the context of the entertainment industry, but also for broader applications in other areas such as e-commerce and social media. Recommendation systems improve user experience by helping people find interesting products or content that matches their tastes. This leads to increased user engagement, better user experience and more time spent on the platform. With technological advancements and the introduction of more sophisticated machine learning techniques such as deep neural networks, these systems can become even more accurate and efficient in the future, overcoming limitations such as the cold-start problem or extensive database.

Mathematical concepts play a key role in the design of such systems. Data mining techniques such as matrix factorization, similarity metrics (cosine similarity, Pearson correlation) and other algorithms are used to predict which movie a user will like. These techniques help analyze user behavior and look for patterns, which allows recommending movies that match users' preferences. Evaluating system performance is also important: using accuracy, completeness, and F1 metrics, we can evaluate how well the system manages recommendations and how relevant they are to each individual user.

Recommendation systems are becoming not only an important part of entertainment platforms, but also a key element for businesses, as they help to increase user engagement, improve user satisfaction, and drive business growth. In this paper, we investigate how such systems can be built, implemented, and analyze which machine learning approaches and techniques are most effective for creating high-quality recommendations.

Thus, the development of a movie recommendation system not only contributes to improving the user experience in the entertainment industry but is also a good example of the application of modern machine learning and artificial intelligence techniques. In the future, these systems will continue to evolve and improve, helping users find the most interesting content and providing deeper and more personalized interactions with platforms.

1 Literature Review

1.1 Description of the subject area

A recommendation system is an algorithmic tool used to provide users with objects of interest, such as movies, merchandise, music, and other resources. They play a key role in the field of information technology, especially in the field of e-commerce, entertainment services and social networks. The main task of these systems is to analyze user behavior and provide on this basis products or content that best meet the interests of each individual user.

In the case of the movie recommendation system, the challenge is to narrow down the selection of millions of movies and TV shows by providing users with movies and TV shows that they might like. This is important because due to the vast number of options available, users often face the problem of choosing, and this can reduce their participation in the platform. The recommendation system helps to solve this problem by increasing user satisfaction and retention on the platform. This is achieved by analyzing data such as previous views, ratings, viewing time, genres, and other parameters.

Depending on the technology used, the recommendation system is divided into several types. The main ones are collaborative filtering, content filtering and a hybrid model combining their elements. It is important to note that such systems are widely used not only in the film industry, but also in other areas, such as e-commerce, for the personalized provision of goods or services.

There are several ways to create a recommendation system, each of which has its own characteristics and areas of application. One of the most popular methods is collaborative filtering. This method assumes that if two users had similar tastes in the past, then they are likely to have similar preferences in the future. Collaborative filtering can be of two types: user-defined and object-based. In the first case, the system searches for users with similar interests and provides movies that these users like. In the second case, she finds films with similar characteristics (for example, by genre or subject) and provides them to viewers who have shown interest in similar films.

Content filtering is based on an analysis of the characteristics of the content itself. For example, the system can provide films with similar genres, actors or directors. Unlike collaborative filtering, content filtering does not require a lot of data about other users. This method is especially useful when the user has left insufficient data about their preferences, or the system uses new objects that are not available in the context of other users.

The hybrid model combines the two previous methods to improve the accuracy of recommendations. These systems can use collaborative filtering and content filtering at the same time, which allows you to take advantage of these two technologies and minimize their disadvantages. For example, a hybrid model can start with collaborative filtering for users with a sufficient history of interaction with the system, and then move on to content filtering for inexperienced users who do not yet have preferences.

One of the most popular models for recommendation systems is matrix factorization. This method is used to reduce the dimensionality of the data and extract hidden factors that explain user behavior. An example of this is the SVD algorithm. It decomposes the matrix of user interactions with objects (for example, movies) into several simpler matrices, allowing you to efficiently extract hidden preferences.

Another popular model is the k-nearest Neighbor algorithm (KNN), which analyzes the similarity between users or movies based on their characteristics. The system can offer the user a movie that was liked by users who look like him.

With the development of technology and the use of machine learning, more sophisticated methods such as neural networks and deep learning have become popular. These models make it possible not only to analyze explicit data about user preferences, but also to identify hidden patterns and relationships that cannot be noticed by traditional methods. Examples of such methods are recurrent neural networks (RNN) and long-term short-term memory (LSTM) neural networks, which consider dynamic changes in user preferences over time.

The recommendation system is very important in the film industry, especially on streaming platforms such as Netflix, Amazon Prime and Hulu. They help users find movies and TV shows that might interest them based on an analysis of their previous

views, ratings, and even behavior such as skipping movies or adding them to a list to watch. Such systems have become an integral part of the user experience, as they simplify the search for content and significantly increase audience engagement.

For example, Netflix uses a hybrid system that combines collaborative filtering with content filtering. This allows you to recommend movies based on the preferences of users with similar tastes and characteristics of the content itself (such as genre, actors and direction). In addition, Netflix also considers temporal aspects, such as seasonal changes in the interests of users, and uses machine learning algorithms to improve the accuracy of recommendations.

The use of recommendation systems in the film industry is also associated with the improvement of business processes. For example, they help platforms understand audience preferences, which can be used to recommend content, improve marketing strategies, or choose the direction of creating new movies and TV series. In some cases, the system can predict the potential success of a film based on previous user preferences, which allows investment companies and producers to make more informed decisions about financing projects.

In general, recommendation systems play a key role in the film industry, improving the user experience, helping to find suitable content and significantly increasing audience engagement, which, in turn, contributes to revenue growth and platform improvement.

1.2 Methodology

In this project, several key stages of the method were selected to create a movie recommendation system that provides efficient data processing, model building, and end-application creation. The system uses machine learning techniques that focus on data analysis and building algorithms that can make recommendations based on user behavior.

The initial stage of methodology development involves the preparation of data, which includes the collection, purification and pre-processing of information. Movie data such as movie ratings, genre, budget, actors, and other characteristics are collected

from the following available sources. An important part of this step is to remove missing data, normalize values, and select related functions to build the model. The data obtained also improves the quality of subsequent analysis and modeling by eliminating errors and duplicating values. The next stage involves the study and analysis of data that classifies and analyzes the distribution of numerical characteristics. This will help us understand how different variables relate to movie ratings and user preferences. Using graphs and tables to visualize your data, we can better understand the structure of information and identify patterns that can be useful in building models.

To make recommendations, one of the machine learning methods is used - collaborative filtering based on the similarity between users and movies. More sophisticated algorithms such as random forests or neural networks are also used to improve the accuracy of recommendations. This method was chosen because it allows you to efficiently process large amounts of data and obtain high-quality forecasts. Indicators such as accuracy, completeness, and possible errors (standard error) are used to assess the quality of a model that measures the accuracy of a forecast.

To implement the recommendation system, a set of tools and technologies are used to ensure efficient data processing, model generation and user interaction. The Python programming language was chosen because of its flexibility and extensive libraries for data analysis, machine learning, and visualization.

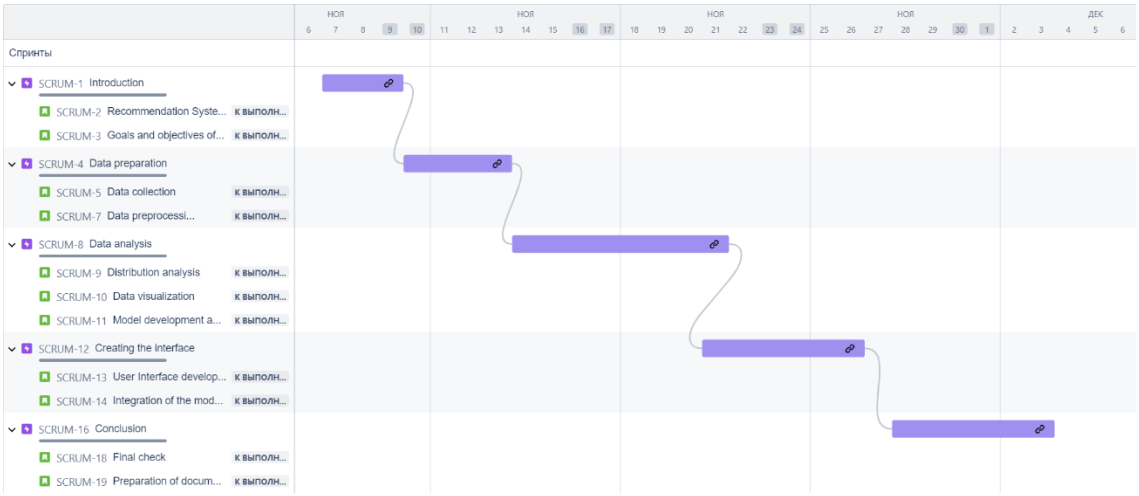
Libraries such as pandas are used to process data and perform operations. Skikit learning provides an opportunity to implement machine learning algorithms, including model training, testing and evaluation. Tensorflow or Keras libraries can be used to create more complex models, such as neural networks. Matflotlives and samples are used to visualize data and analyze results.

Python is used to collect and process data using the appropriate libraries. The Streamlite library is used as an end-user interface to quickly create web applications with an intuitive interface that allows users to interact with the recommendation system.

Thus, the design method includes the stages of data preparation, analysis, selection and construction of models, training and testing of models, as well as the creation of client interfaces.

Also, to monitor the implementation of each stage or process, a Jira, Gantt chart is used, that is, this chart is used to present a project implementation plan for creating a recommendation system for films (see fig. 1.2.1).

Figure 1.2.1 Diagram Gantt



2 Dataset

2.1 Data understanding

In the course project, the basis for building a recommendation system for films was a carefully selected dataset that contains information about films, their characteristics, as well as user preferences. Data is a key element of the system, as it determines the accuracy and quality of recommendations.

Understanding data is the first and one of the most important steps in working with any machine learning project. Understanding the data The main goal was to explore the structure and content of the selected dataset, which will become the basis for development. At this stage, a study of the structure and content of the dataset is carried out.

The main focus in data understanding is on studying the structure and content of the dataset used to build a recommendation system for movies. In this course work, the dataset was selected in kagle, which is a social network of data processing and machine learning specialists. The selected dataset consists of information about more than 5,000 films and includes many useful characteristics that help to better understand their popularity, characteristics and audience preferences. The main columns in the dataset:

– Title of the film: The names of the films that will be presented in the recommendations.

– Genres: A list of genres of each film, such as drama, comedy, action, which allow you to take into account the tastes of users.

– IMDb Score (imdb_score): An average rating based on user reviews, which is one of the key indicators of the film's popularity.

– Directors and actors (director_name, actor_1_name and others): Data about directors and main actors, which can also affect user preferences.

– Budget and box office (budget, gross): Financial indicators that give an idea of the level of the film and its success with the audience.

– Duration (duration): The length of the movie, which may be important for some users.

– Release year (release_year): A time factor reflecting changes in audience preferences.

Also, according to the selected dataset, it is important to understand how the data is distributed and what patterns can be identified, which is why data analyses are carried out. Special attention in the analysis was paid to the distribution of movie ratings and user preferences. For example, there are often popular genres such as dramas, comedies and action films that enjoy increased audience interest. We also analyzed data on the popularity of films in different years to understand how the time factor affects the choice of viewers.

A real data set is rarely ideal, so data processing is always necessary. Certain tasks were performed during the preprocessing of the data:

Elimination of missing values, since the lack of information about the movie or the user's rating can distort the model's operation. The omissions were replaced with average or median values, or deleted altogether if there were too many of them.

Converting categorical data, such as genres or countries, to numeric formats using methods such as one-hot encoding. This allows machine learning models to correctly interpret such data.

Scaling numeric data such as budget or movie duration. The use of methods such as normalization or standardization ensured the correct participation of this data in the training process of the model.

Identification and processing of emissions. For example, films with abnormally high or low ratings that could influence the model were processed using statistical characteristics analysis.

Analyzing these data will help identify hidden patterns to build an effective recommendation model.

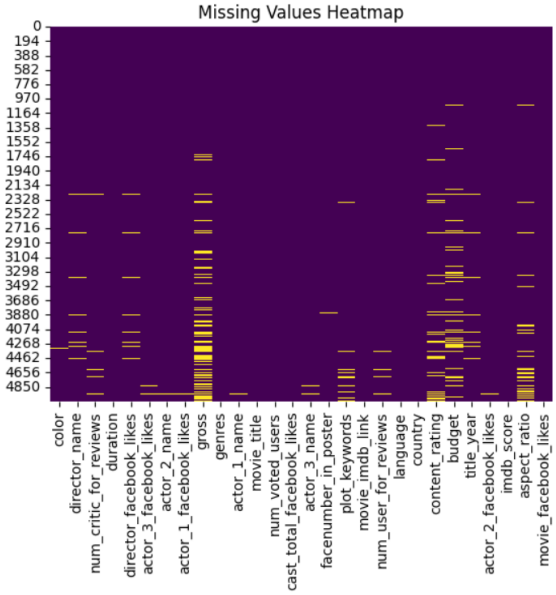
2.2 Data processing

Data processing is a key part of any machine learning project that affects the quality of the model and its ability to make accurate predictions. As part of the development of a recommendation system for films, data processing includes many

stages, starting from data purification and ending with their conversion to a format suitable for working with machine learning algorithms.

One of the first steps was to identify and process missing values in the data. For example, columns such as budget and box office receipts are missing in the dataset. This data is critically important because it reflects the financial performance of films that may be related to their popularity. For numeric data, such as a budget, the gaps are filled with the median value. The median is preferable to the average value because it is less sensitive to deviations. In categorical data, for example, in movie genres, missing values are replaced with the value "Unknown" or similar to avoid information loss (see fig. 2.2.1).

Figure 2.2.1 Heat map of data gaps



Repeated entries in the data can significantly distort the results of the analysis and training of the model. It checks for duplicates here. Because the same movie can be presented twice in different rows, but with slightly different data, which requires combining or deleting them (see fig. 2.2.2).

Outliers are values that differ significantly from the rest of the data and can lead to incorrect modeling results. For example, films with zero budgets or box office receipts in the trillions of dollars are clearly anomalies. Visualization techniques such as boxplot are used to identify deviations, as well as numerical approaches such as interquartile range (IQR). Deviations will either be deleted if they did not contain

useful information, or replaced with more realistic values if such data could be obtained (see fig. 2.2.3).

Figure 2.2.2 Handling emissions

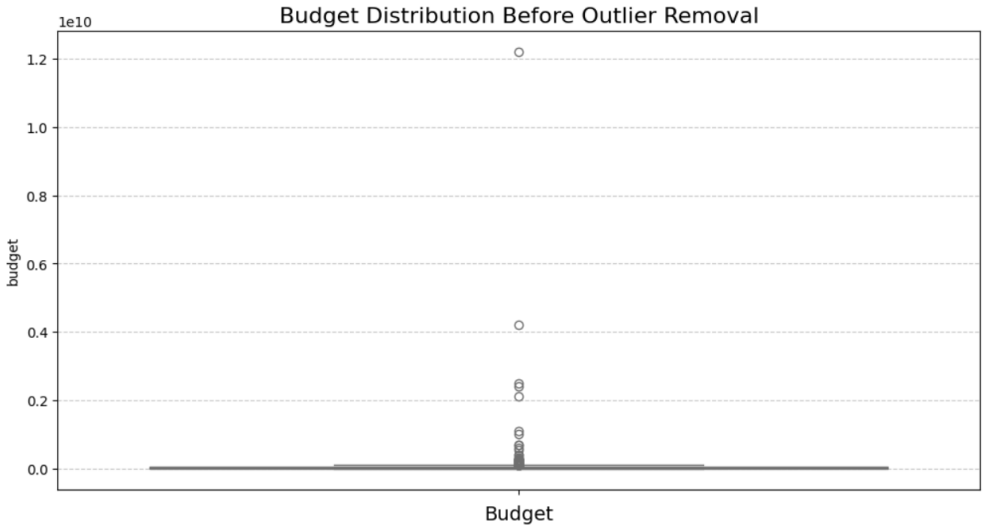
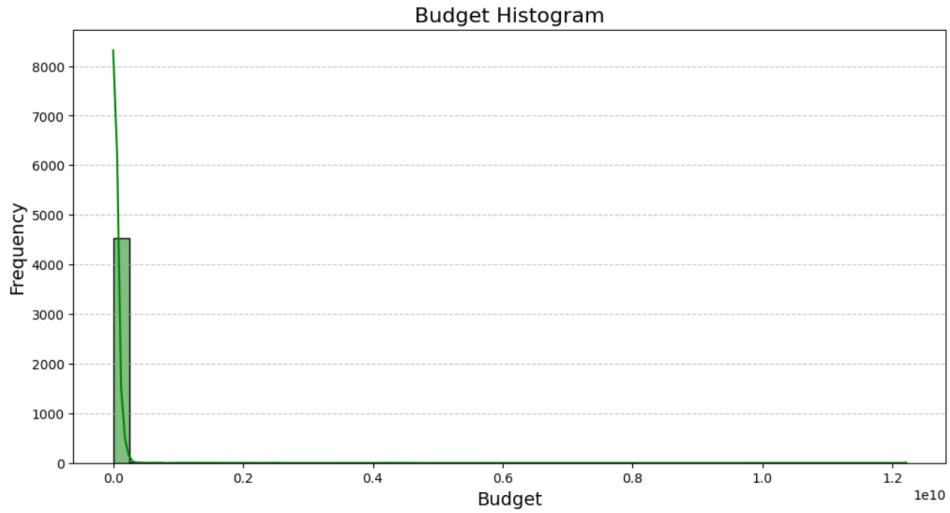


Figure 2.2.3 Handling emissions histogram



Categorical data, such as genres or languages of films, have been converted into numerical form so that they can be used in machine learning algorithms. A single encoding method is used for this purpose. For example, if a movie belongs to the comedy genre, a new column is created in which the value will be 1 if the genre is present, and 0 if it is absent. This approach allows you to store complete information about genres without losing sight of their multicategory (a film can belong to several genres at once).

To improve the efficiency of the models, new features are added that were not in the original dataset. For example:

– The profitability of a film: Calculated as the ratio of box office receipts to the budget. This allows the model to take into account the financial success of the films.

– Popularity: This indicator was determined based on the average rating of the film and the number of ratings that helped determine the films in demand.

– Number of genres: A function has been introduced that reflects the number of genres the film belongs to.

Numerical characteristics such as budget and box office receipts have been normalized using the minimum and maximum scaling method. This will bring the values to a single range (from 0 to 1), which is especially important for data scale-sensitive algorithms such as KNN or gradient boosting.

For training and testing models, the data is divided into two parts. The training sample was used to build the model, and the test sample was used to assess its quality. This separation allows you to check how well the model works with data that it has not previously seen.

Finally, the processed data was saved in CSV format. This simplifies subsequent work with them and provides convenient integration with the stages of building a model and developing an application.

Data processing is an integral part of the project, as the success of the entire system depends on its quality. The prepared data becomes a reliable basis for further analysis and the construction of a highly effective system of recommendations for films.

3. Model Implementation

3.1 Theoretical foundation

The implementation of the recommendation system relies heavily on the principles of machine learning and data science, paying special attention to algorithms that allow you to offer personalized content based on user behavior. The recommendation system for this project is based on collaborative filtering and content-based filtering techniques, which form the theoretical basis of how the system generates recommendations.

Collaborative filtering is a method in which the system predicts user preferences based on the preferences of other users. It is assumed that if two users had similar interests in the past, they will have similar preferences in the future. There are two main types of collaborative filtering: user-defined and subject-based. With collaborative user-based filtering, the system recommends movies based on user preferences whose tastes match those of the target user. For example, if two users like romantic comedies, the system will prompt the other to watch other romantic comedies that one user liked. On the other hand, collaborative element-based filtering recommends movies that are like those that the user has already rated or liked. If a user watches an action movie and likes it, the system will recommend other action movies or films with similar characteristics, for example, with the same director or actors.

Although collaborative filtering can be effective, it faces some limitations. One of the serious problems is the problem of "cold start", when the system hardly recommends products for new users or new movies, based on which there is not enough data to make recommendations. Content-based filtering is used to solve this problem. Content-based filtering focuses on the features of the elements themselves, rather than on user interaction. This method allows you to select films based on their characteristics, such as genre, director, actors, or keywords. For example, if a user likes sci-fi movies with a certain actor, the system will suggest other sci-fi movies featuring the same actor. Even if the user is new or the movie is unfamiliar to other users, content-based filtering can still provide accurate recommendations based on the attributes of the movie.

One of the key advantages of content-based filtering is its ability to generate recommendations for new or little-known films that have not yet been rated or widely viewed. However, it also has its limitations. For example, if a user prefers to watch only one type of movie (e.g. action movies), content-based filtering can lead to over-specialization, offering only similar recommendations and potentially skipping movies that may be of interest. To overcome this, a hybrid approach is often used, combining both collaborative filtering and content-based filtering, which provides a more balanced and accurate recommendation system [2].

In addition to these filtering methods, a random forest algorithm is used to predict the success of a film based on its characteristics. A random forest is an ensemble method, that is, it combines several decision trees to improve prediction accuracy. Each decision tree considers a subset of the available functions and makes a prediction based on them. By combining the results of many different decision trees, Random Forest creates a more robust model that is less susceptible to overfitting. In the context of a movie recommendation system, Random Forest can be used to predict various outcomes, such as box office success, user ratings, or the overall popularity of a movie. For example, Random Forest can predict that a movie with a big budget, famous actors and a popular director is likely to have good box office receipts. Conversely, it can be predicted that films with a smaller budget and less star power will have low box office receipts. Such predictive modeling can be invaluable for making recommendations that consider both user preferences and the potential success of films.

By combining collaborative filtering, content-based filtering, and random forest, the system can offer a more comprehensive approach to recommendations. Collaborative filtering helps to recommend movies based on the tastes of similar users, while content-based filtering focuses on the characteristics of the movies themselves. Random Forest expands the capabilities of the system by providing forecasts based on the characteristics of the film, which are useful not only for making personalized recommendations, but also for evaluating the success or failure of the film by combining collaborative filtering, content-based filtering and random forest, the system can offer a more comprehensive approach to recommendations. Collaborative

filtering helps to recommend movies based on the tastes of similar users, while content-based filtering focuses on the characteristics of the movies themselves. Random Forest expands the capabilities of the system by providing forecasts based on the characteristics of the film, which are useful not only for making personalized recommendations, but also for evaluating the success or failure of the film. Together, these approaches work synergistically, providing users with relevant and attractive recommendations for watching movies [17].

As for evaluating the effectiveness, the success of the model is measured using various indicators. The standard error (RMSE) is the main evaluation indicator for this system, as it helps to assess how well the model predicts user ratings. RMSE is especially useful because it punishes large errors more strongly, which means that if the model makes large errors in forecasting, then RMSE will reflect this more accurately. Lower As for evaluating effectiveness, the success of the model is measured using various indicators. The standard error (RMSE) is the main evaluation indicator for this system, as it helps to assess how well the model predicts user ratings. RMSE is especially useful because it punishes large errors more strongly, which means that if the model makes large errors in forecasting, then RMSE will reflect this more accurately. A lower RMSE indicates a better match between the predicted and actual estimates, showing that the recommendation system makes accurate predictions. In addition to RMSE, other evaluation indicators such as accuracy, responsiveness and F1 score can be used to evaluate the effectiveness of the recommendation system in developing appropriate proposals. The accuracy shows how many of the recommended movies were relevant to the user, while the review shows how many of them were recommended. The F1 indicator provides a balance between accuracy and recall, making it a comprehensive indicator for evaluating overall system performance.

The machine learning model implemented in this project is designed to be scalable and adaptable for machine learning, which allows it to process large datasets and make accurate predictions for a wide range of users.

3.2 Machine learning model implementation and performance

This part of the course examines the process of implementing a machine learning model for a movie recommendation system, as well as evaluating its effectiveness. The main purpose of the machine learning model is to predict which movies might be of interest to the user based on the available data. This project uses several methods and approaches to implement the model and its subsequent evaluation [14].

To implement the machine learning model, first import the necessary libraries. Libraries such as sklearn, pandas, seaborn are used for text classification, and other models are imported from libraries for the recommendation system, respectively. These libraries are necessary for analyzing the dataset, as well as visualization, at the same time they will help you understand the dataset. First, the SVM algorithm is used as a template. Importing libraries helps to read our dataset.

Pandas is a library designed to work with data in tabular form, which allows you to effectively manipulate and analyze data. Seaborn and Matplotlib are popular data visualization libraries that provide convenient ways to create graphs and diagrams to visually display information. NumPy is used to perform mathematical operations with arrays of data, which is important when processing large amounts of information. TfidfVectorizer from the scikit-learn library is used to convert text data into numeric vectors, which allows you to work with text information such as movie descriptions. Finally, train_test_split from the same library helps to divide the data into training and test samples, which is important for evaluating the model (see fig.3.2.1).

```
import pandas as pd
import seaborn as sns
import numpy as np
rom sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
df = pd.read_csv('movie_metadata.csv')
df.head()
print(df.shape)
```

Figure 3.2.1 Reading the dataset

	color	director_name	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_likes	actor_2_name	actor_1_facebook_likes	gross	
0	Color	James Cameron	723.0	178.0	0.0	855.0	Joel David Moore	1000.0	760505847.0	Action|Adv
1	Color	Gore Verbinski	302.0	169.0	563.0	1000.0	Orlando Bloom	40000.0	309404152.0	Action
2	Color	Sam Mendes	602.0	148.0	0.0	161.0	Rory Kinnear	11000.0	200074175.0	Actior
3	Color	Christopher Nolan	813.0	164.0	22000.0	23000.0	Christian Bale	27000.0	448130642.0	
4	NaN	Doug Walker	NaN	NaN	131.0	NaN	Rob Walker	131.0	NaN	

5 rows × 28 columns

Analysis using a correlation matrix is useful for identifying relationships between numerical features in your dataset. It helps to understand which factors can influence each other and possibly be useful for building predictive models such as linear regression or other machine learning algorithms.

The graph of the heat map shows a matrix of correlations between various numerical features. The color scale indicates the degree of connection between the variables:

– The bright red color indicates a strong positive correlation.

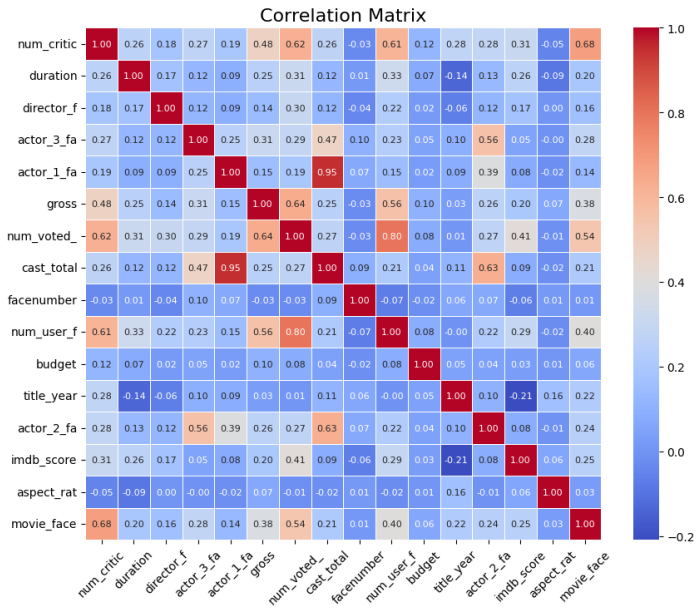
– The bright blue color has a strong negative correlation.

– White and light blue shades indicate low correlation or lack of connection.

Visualization makes it easy to see which columns in the dataset are strongly related to each other and which are not. For example, the graph may show that "num_critic_for_reviews" and "gross" have a highly positive correlation, which suggests that an increase in the number of critics is associated with an increase in box office receipts (see fig. 3.2.2).

```
numerical_cols = df.select_dtypes(include=['float64', 'int64']).columns
df.columns = [col[:10] for col in df.columns]
numerical_cols = df.select_dtypes(include=['float64', 'int64']).columns
corr_matrix = df[numerical_cols].corr()
plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt='.2f', linewidths=0.5, annot_kws={'size': 8})
plt.title('Correlation Matrix', fontsize=16)
plt.xticks(fontsize=10, rotation=45)
plt.yticks(fontsize=10, rotation=0)
plt.show()
```

Figure 3.2.2 Correlation matrix of the dataset

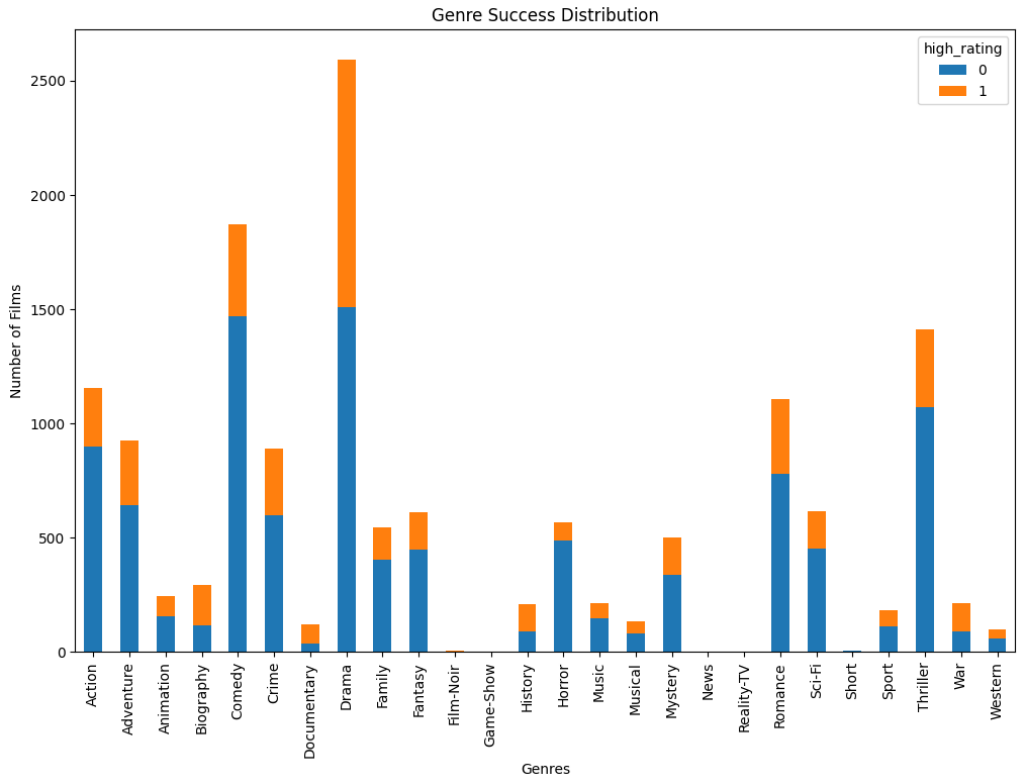


Let's see which genres are more common among successful films. To do this, an analysis is carried out by dividing films by their genres and evaluating which of them are most often associated with high ratings. Then the number of successful films for each genre is calculated to see which directions are most popular (see fig. 3.2.3).

Based on this data, a visual chart is created that shows how many successful and unsuccessful films belong to each genre. This makes it easy for us to notice trends, for example, which genres are most often highly appreciated by viewers and critics. This approach helps to understand in which direction success is most often achieved, which can be useful for further analysis or decision-making.

```
df_genres = df['genres'].str.get_dummies(sep='|')
df_genres['high_rating'] = (df['imdb_score'] > 7).astype(int)
genre_success = df_genres.groupby('high_rating').sum()
genre_success.T.plot(kind='bar', figsize=(12, 8), stacked=True)
plt.title('Genre Success Distribution')
plt.xlabel('Genres')
plt.ylabel('Number of Films')
plt.show()
```

Figure 3.2.3 Distribution by genre



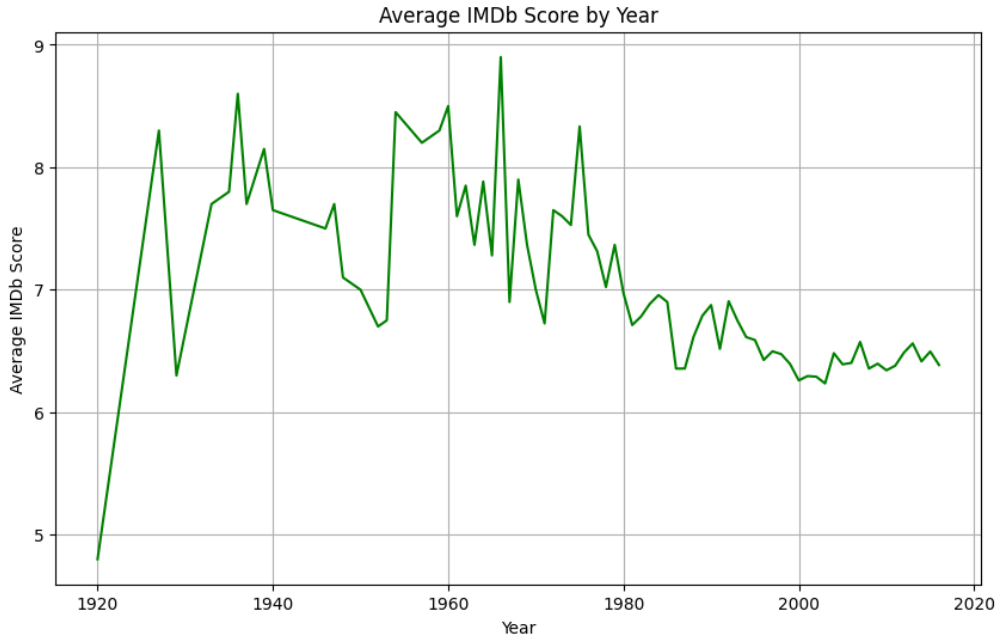
The following analysis analyzes how the average rating of films (according to IMDb) changed depending on the year of their release. To do this, the data is grouped by year, and for each year the average rating of all films released during this period is calculated. The results are then visualized using a graph (see fig. 3.2.4).

On the graph, the X-axis represents the years, and the Y-axis shows the average rating. The line running through the dots illustrates a general trend: for example, whether there were periods when films generally received higher or lower ratings. Such an analysis helps to identify changes in audience preferences or the evolution of film quality over time. The graph is supplemented with a grid for the convenience of data analysis.

```
df_avg_rating = df.groupby('title_year')['imdb_score'].mean()
plt.figure(figsize=(10, 6))
sns.lineplot(x=df_avg_rating.index, y=df_avg_rating.values, color='green')
plt.title('Average IMDb Score by Year')
plt.xlabel('Year')
plt.ylabel('Average IMDb Score')
plt.grid(True)
```

plt.show()

Figure 3.2.4 Movie ratings by year



The analysis of the importance of the success factors of the film is to find out the key factors that affect the rating and success. This analysis shows machine learning using Random Forest. At the initial stage of the analysis, the success criterion will be determined and the data will be preprocessed. Determining the success criterion, that is, we will set a threshold by which we will divide films into "successful" and "unsuccessful", for example, films with box office receipts above the budget are considered successful. Data preprocessing, key features are selected here and prepared for the model. A new target attribute is also created to help indicate whether the movie was successful, for example, a binary value of 1 or 0 (see fig. 3.2.5).

```
import pandas as pd
import numpy as np
data = pd.read_csv('movie_metadata.csv')
data['success'] = np.where(data['gross'] > data['budget'], 1, 0)
features = ['budget', 'duration', 'gross', 'title_year', 'genres']
data = data[features + ['success']]
data = pd.get_dummies(data, columns=['genres'], drop_first=True)
data.dropna(inplace=True)
X = data.drop(columns=['success'])
```

y = data['success']

data.head()

Figure 3.2.5 Analysis of the importance of the success factors of the film

	budget	duration	gross	title_year	success	genres_Action|Adventure	genres_Action|Adventure|Animation|Comedy|Crime|Family|Fantasy	genres_Action|Adve
0	237000000.0	178.0	760505847.0	2009.0	1	False	False	
1	300000000.0	169.0	309404152.0	2007.0	1	False	False	
2	245000000.0	148.0	200074175.0	2015.0	0	False	False	
3	250000000.0	164.0	448130642.0	2012.0	1	False	False	
5	263700000.0	132.0	73058679.0	2012.0	0	False	False	

5 rows × 918 columns

The Random Forest classification model will be used to train a model for predicting the success of a film. Random Forest is an ensemble learning technique that builds several decision trees during training and produces a class that is the mean prediction (regression) or the mode of the classes (classification) of the individual trees. It is reliable and accurate for a range of applications since it minimizes overfitting by building trees from arbitrary subsets of features and data. The model is evaluated using accuracy, confusion matrix and classification report and as a result it turns out performance indicators of the model (see table 3.2.1).

from sklearn.model_selection import train_test_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

model = RandomForestClassifier(n_estimators=100, random_state=42)

model.fit(X_train, y_train)

y_pred = model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)

print(f'Accuracy: {accuracy:.4f}')

print('Confusion Matrix:')

print(confusion_matrix(y_test, y_pred))

print('Classification Report:')

print(classification_report(y_test, y_pred))

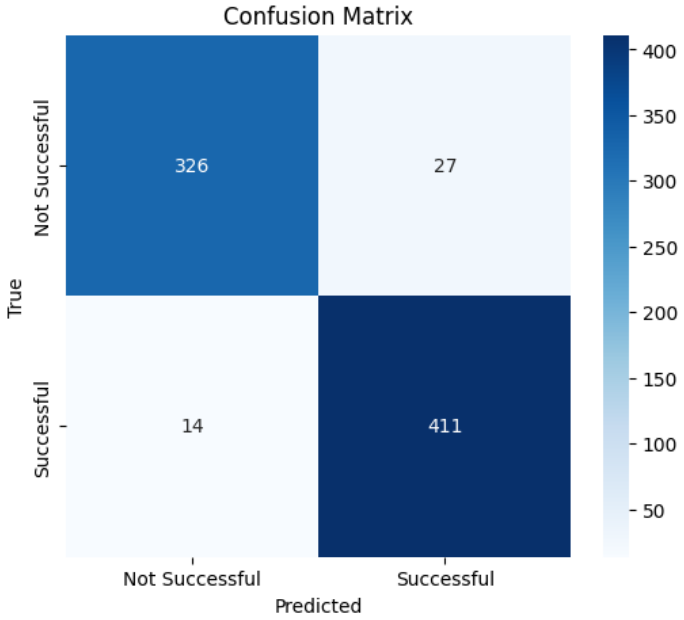
Table 3.2.1 Performance indicators of the model

Classification	Precision	Recall	F1-score	Support
0	0.96	0.92	0.94	353
1	0.94	0.97	0.95	425
Accuracy			0.95	778
Macro avg	0.95	0.95	0.95	778
Weighted avg	0.95	0.95	0.95	778

The next step is to visualize the error matrix. The error matrix shows how many films were correctly predicted to be successful or unsuccessful, and how many mistakes were made. To do this, the seaborn library is used to display the error matrix as a heat map (see fig. 3.2.6).

```
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['Not Successful', 'Successful'], yticklabels=['Not Successful', 'Successful'])
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix')
plt.show()
```

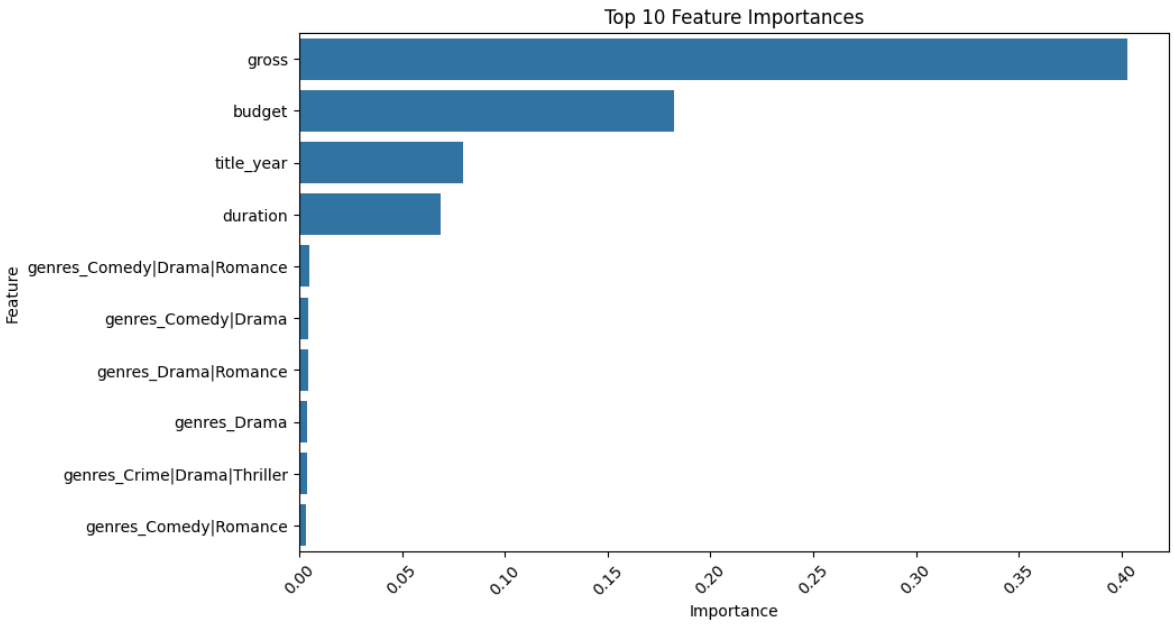
Figure 3.2.6 Visualization of the error matrix



Visualizing the importance of features will help us see which features have the greatest impact on the model's predictions. Random Forest automatically calculates the importance of each feature. After the calculation, we visualize it to understand which signs play a key role (see fig. 3.2.7).

```
import matplotlib.pyplot as plt
import seaborn as sns
feature_importances = model.feature_importances_
features_list = X.columns
top_n = 10
top_features = sorted(zip(feature_importances, features_list), reverse=True)[:top_n]
top_importances, top_feature_names = zip(*top_features)
plt.figure(figsize=(10, 6))
sns.barplot(x=top_importances, y=top_feature_names)
plt.title('Top 10 Feature Importances')
plt.xlabel('Importance')
plt.ylabel('Feature')
plt.xticks(rotation=45)
plt.show()
```

Figure 3.2.7 Visualization of the importance of features



The result of the analysis shows that the model classifies films very accurately as successful or unsuccessful, and the accuracy and completeness indicators are very high for both classes. This indicates the successful operation of the model and a good selection of features to predict the success of the film.

4 System development

4.1 Database implementation

For the work of the recommendation system, a pre-prepared dataset of films in JSON format used, which is a convenient format for storing and processing data. This dataset contains essential information necessary for building recommendations, including parameters such as movie genres, their ratings, links to posters, and descriptions. The database files connected to the system as follows:

```
import json
with open('C:/Users/User/Desktop/movrecom/movie_data.json', 'r+', encoding='utf-8') as f:
data = json.load(f)
with open('C:/Users/User/Desktop/movrecom/movie_titles.json', 'r+', encoding='utf-8') as f:
movie_titles = json.load(f)
```

The basic movie data downloaded from the movie_data.json file. This file contains vital information about each movie, including its characteristics, genres, IMDb rating and links to movie pages. These data are the basis for building a recommendation model that works using the KNN (K-nearest neighbors) method. The KNN method helps to find similar films based on various attributes, such as genre and rating.

The movie_titles.json file connected, which contains movie titles and links to their pages on IMDb. This allows the system to display additional information about the films to the user, such as description, cast, rating, and poster. Using this data, the system can display to the user details about the films that were recommended because of the model's operation. How the database used:

– After connecting the JSON files, the data is stored in the variables data and movie_titles. These variables are used in the model to find similar movies based on the movie or genre selected by the user.

– The system downloads the data at the start of the application and uses it to operate the KNN model.

– During recommendation generation, data from the database filtered and analyzed in real time.

The database is the backbone of the recommendation system, as it contains all the information needed to build recommendations. The JSON format provides simple storage, easy processing, and quick connection to the application. This approach allowed us to realize a flexible and scalable system that supports work with many films and provides quick access to data.

4.2 Recommendation system creation

The main part of the project is dedicated to the creation of a film recommendation system and focuses on the design and implementation of a recommendation system for films. The main goal of the system is to provide the user with recommendations that will match their preferences and interests based on film data. The system allows recommendations to be selected in two ways: based on the selected film or based on genre and rating preferences. The implementation of the system consisted of several key steps, including the use of the K nearest neighbours (KNN) algorithm, working with IMDb data and creating a user-friendly interface for user interaction.

In order to suggest films similar to the user's preferences, the K nearest neighbours (KNN) algorithm was chosen. This classification method uses an approach where for each film, the Euclidean distance between its features and those of other films in the database is calculated. Based on these distances, nearest neighbours (similar films) are identified and then suggested to the user.

The KNN algorithm has been implemented manually, which allows a flexible control of the process and a better understanding of how the algorithm works. The system takes two types of recommendations as input:

– Recommendations based on the selected film: The user selects a film and the system finds the films most similar in terms of genres and IMDb rating.

– Recommendations based on genres and rating: The user can select one or more film genres and a minimum rating, and the system will suggest films that match these criteria.

To implement the algorithm, the K-NearestNeighbours class was created, which includes several methods:

– A method to calculate the Euclidean distance between two objects.

– A method for sorting films by similarity to the selected one.

– A method to generate a list of nearest neighbours and analyse them.

To improve the quality of recommendations and increase usability, the system connects to an external source of information - IMDb. Using the BeautifulSoup library, the application extracts film data from the IMDb website, such as, film posters, information about the director and actors, brief plot descriptions, additional ratings.

The Streamlit library was used to create a user-friendly and interactive user interface, which makes it easy to develop web applications with intuitive controls [8]. The interface includes:

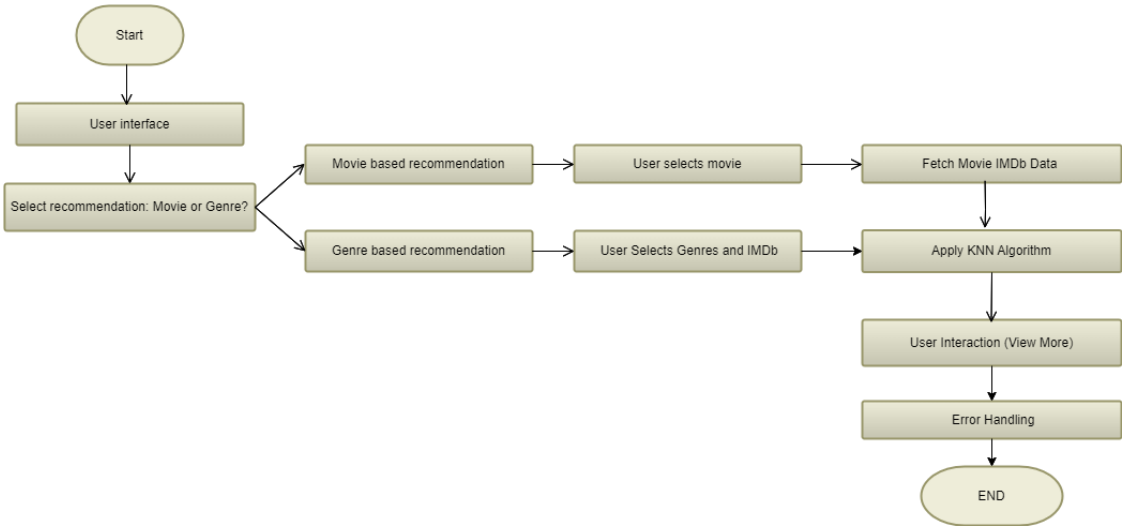
– Drop-down lists where the user can select a film or genres for films.

– Sliders and numeric fields where the user can adjust recommendation parameters such as minimum rating or number of recommended films.

– Dynamic display of results: the system displays a list of recommended films with their posters, a brief description, information about the director and actors, and links to IMDb for more details [9].

To fully understand how the algorithm is used Flowchart Diagram is used, it is a diagram that represents a process, system or computer algorithm (see fig. 4.2.1).

Figure 4.2.1 Flowchart Diagram of Movie Recommendation System



The movie recommendation system implemented in the course work uses the K-NearestNeighbours algorithm, which analyses movie data to suggest movies that match the user's interests. Combined with functionality to extract data from IMDb and an intuitive interface, the system provides users with accurate and visually appealing recommendations, improving their experience of interacting with the application.

5 Results and Discussion

In the process of developing a film recommendation system based on the K-Nearest Neighbors (KNN) algorithm, the goal was to provide the user with personalized recommendations given the selected films or genres. It is important to note that this approach was based on data obtained from a real source, IMDb, which allowed us to consider parameters such as genres, ratings, and additional metadata such as cast and plot descriptions.

Using the KNN algorithm as the main model for recommendations has shown to be effective in the context of films, where the selection of recommendations directly depends on the similarity of the selected film to other items in the database. This approach allows not only recommending films based on a single user's choice, but also based on genres, which is important for users who like multiple categories of films. The algorithm based on computing the Euclidean distance between objects is quite simple and straightforward, but at the same time it also has limitations such as high dependence on data quality [12].

The results of the system showed that the recommendations based on the selected film meet the user's expectations in most cases. Once the user selects a film, the system finds similar films based on various attributes such as genres and IMDb rating. However, it is worth noting that the recommendations do not always perfectly match user preferences. This is because KNN focuses on the ‘proximity’ of the data, which does not always guarantee a complete match of viewers' tastes.

Also, the system, which offered recommendations based on the selected genres and IMDb scores, showed satisfactory results around filtering. When the user specifies multiple genres and a minimum rating, the system correctly filters the films matching these parameters, resulting in more accurate and personalized recommendations. This enables the user to receive suggestions of films that match their interests while considering the right qualities such as high rating or genre affiliation.

Several tests were conducted to evaluate the accuracy of the system, and while KNN proved to be effective, we also identified several issues. One of them is the dependence on the selected data: the more information about films (e.g. reviews,

number of views, etc.), the more accurate the recommendations will be. However, this project used a limited set of features, which affected accuracy.

Another important topic of discussion is the lack of consideration for dynamic user preferences. In the current system, recommendations are based solely on static data (genres, IMDb rating). However, user preferences can change depending on time of day, mood, current trends in cinema and other factors. Future work could integrate data on current user preferences or use online learning techniques to allow the system to adapt to changes in preferences in real time.

Opportunities to improve the system include adding a feature that would allow the user to rate the films they have watched, as well as using hybrid recommendation models. Filteringstems that combine collaborative filtering techniques with content filtering can significantly improve the accuracy of recommendations, as they consider both similarities between users and similarities between the films themselves.

As noted in studies such as Desrosiers and Karypis (2011), hybrid systems can be particularly useful in situations where a single approach cannot fully fulfil all user needs. They allow combining the strengths of different methods and compensating for their weaknesses. In the context of film recommendations, this may mean that the system will be able to consider more factors, including the opinions of other users, the interests of a particular user, and even current trends in the industry [1].

Additionally, the developed KNN-based recommendation system has shown its practical applicability in the task of selecting films for users. However, to improve the accuracy and quality of recommendations, it is necessary to expand the dataset and consider more sophisticated approaches, such as using hybrid models or algorithms considering dynamic user preferences.

Conclusion

In conclusion, this term paper focuses on the development of a film recommendation system based on various machine learning and data analysis techniques. The work used a set of approaches including the K-nearest neighbours (KNN) algorithm, as well as data analysis techniques such as investigating the relationships between different movie characteristics, genre analysis, and investigating the year-to-year variation of average movie ratings.

To construct recommendations, we used film data, which includes information about genre, rating, and other characteristics, and applied various machine learning techniques to predict users' preferences. In addition, the analysis revealed the popularity of different genres among successful films and also tracked the changes in the average rating of films over time, which is useful information to further improve the recommendations.

The main contribution of the work is the creation of an effective recommendation system that takes into account user preferences based on genre, rating and other criteria, and offers different ways to interact with the user through film or genre selection. In addition to KNN, other machine learning techniques have also been used to improve the quality of recommendations and evaluate their effectiveness.

The results of the work showed that machine learning techniques such as KNN and data analysis techniques such as learning correlations and distributions can significantly improve the performance of the recommendation system, creating more accurate and personalised recommendations for users. This system can be further refined and improved in the future by using more sophisticated algorithms such as Random Forest or gradient bousting, and by integrating with other data sources such as user feedback or social media.

Thus, this work has not only shown the practical value of using machine learning to build recommendation systems, but has also opened up opportunities for further development and expansion of the system's functionality, taking into account new methods and approaches in the field of data analysis and machine learning.

References

1. Desrosiers, C., & Karypis, G. (2011). A comprehensive survey of neighborhood-based recommendation methods. Proceedings of the 2009 IEEE International Conference on Data Mining Workshops, 1-11.

2. Mahesh, Batta. "Machine learning algorithms-a review." International Journal of Science and Research (IJSR).[Internet] 9.1 (2020): 381-386.

3. Jordan, M. I., & Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. Science, 349(6245), 255-260.

4. Raschka, S., & Mirjalili, V. (2017). Python machine learning: Machine learning and deep learning with python. Scikit-Learn, and TensorFlow. Second edition ed, 3, 17.

5. Hao, J., & Ho, T. K. (2019). Machine learning made easy: a review of scikit-learn package in python programming language. Journal of Educational and Behavioral Statistics, 44(3), 348-361.

6. Jain, M., Rajput, H., Garg, N., & Chawla, P. (2020, July). Prediction of house pricing using machine learning with Python. In 2020 International conference on electronics and sustainable communication systems (ICESC) (pp. 570-574). IEEE.

7. Khorasani, Mohammad, Mohamed Abdou, and Javier Hernández Fernández. "Streamlit use cases." Web Application Development with Streamlit: Develop and Deploy Secure and Scalable Web Applications to the Cloud Using a Pure Python Framework. Berkeley, CA: Apress, 2022. 309-361.

8. Richards, T. (2023). Streamlit for Data Science: Create interactive data apps in Python. Packt Publishing Ltd.

9. Ko, H., Lee, S., Park, Y., & Choi, A. (2022). A survey of recommendation systems: recommendation models, techniques, and application fields. Electronics, 11(1), 141.

10. Isinkaye, Folasade Olubusola, Yetunde O. Folajimi, and Bolande Adefowoke Ojokoh. "Recommendation systems: Principles, methods and evaluation." Egyptian informatics journal 16.3 (2015): 261-273.

11. Goyani, Mahesh, and Neha Chaurasiya. "A review of movie recommendation system: Limitations, Survey and Challenges." ELCVIA: electronic letters on computer vision and image analysis 19.3 (2020): 0018-37.

12. Choi, Sang-Min, Sang-Ki Ko, and Yo-Sub Han. "A movie recommendation algorithm based on genre correlations." Expert Systems with Applications 39.9 (2012): 8079-8085.

13. Furtado, F., and A. Singh. "Movie recommendation system using machine learning." International journal of research in industrial engineering 9.1 (2020): 84-98.

14. Biancalana, Claudio, et al. "Context-aware movie recommendation based on signal processing and machine learning." Proceedings of the 2nd Challenge on Context-Aware Movie Recommendation. 2011. 5-10.

15. Ricci, F., Rokach, L., Shapira, B., & Kantor, P. B. (2011). Recommender Systems Handbook. Springer.

16. Burke, R. (2002). Hybrid Recommender Systems: Survey and Experiments. User Modeling and User-Adapted Interaction, 12(4), 331–370.

17. Su, X., & Khoshgoftaar, T. M. (2009). A Survey of Collaborative Filtering Techniques. Advances in Artificial Intelligence, 2009, Article ID 421425.

18. Aggarwal, C. C. (2016). Recommender Systems: The Textbook. Springer.

19. Zhang, S., Yao, L., Sun, A., & Tay, Y. (2019). Deep Learning Based Recommender System: A Survey and New Perspectives. ACM Computing Surveys, 52(1), 1–38.

20. Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix Factorization Techniques for Recommender Systems. IEEE Computer, 42(8), 30–37. https://doi.org/10.1109/MC.2009.263

Appendices

Appendix 1. Movie Recommender System

Figure 1.1 Main page

Figure 1.2 Film Title Recommendation Page

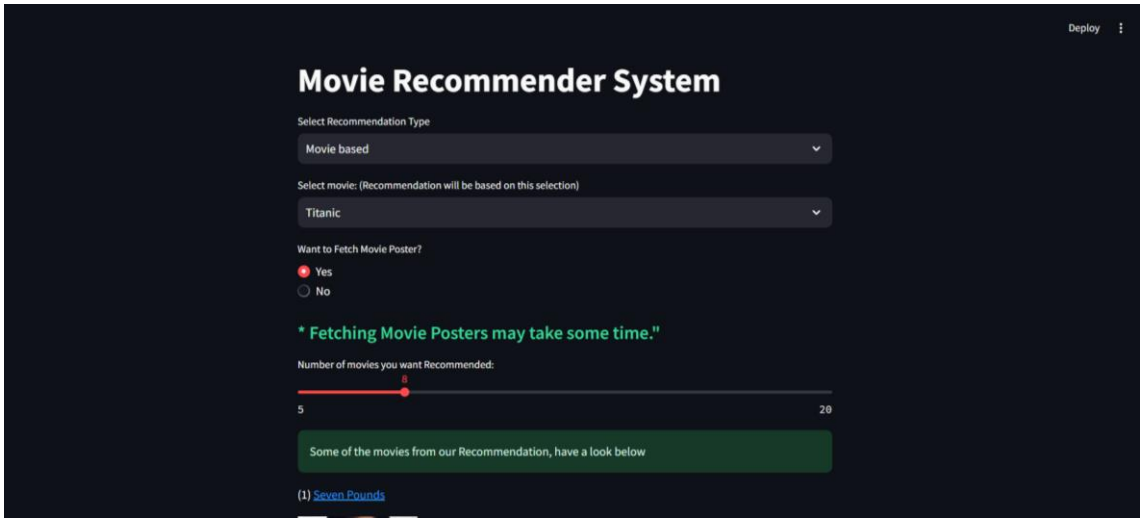
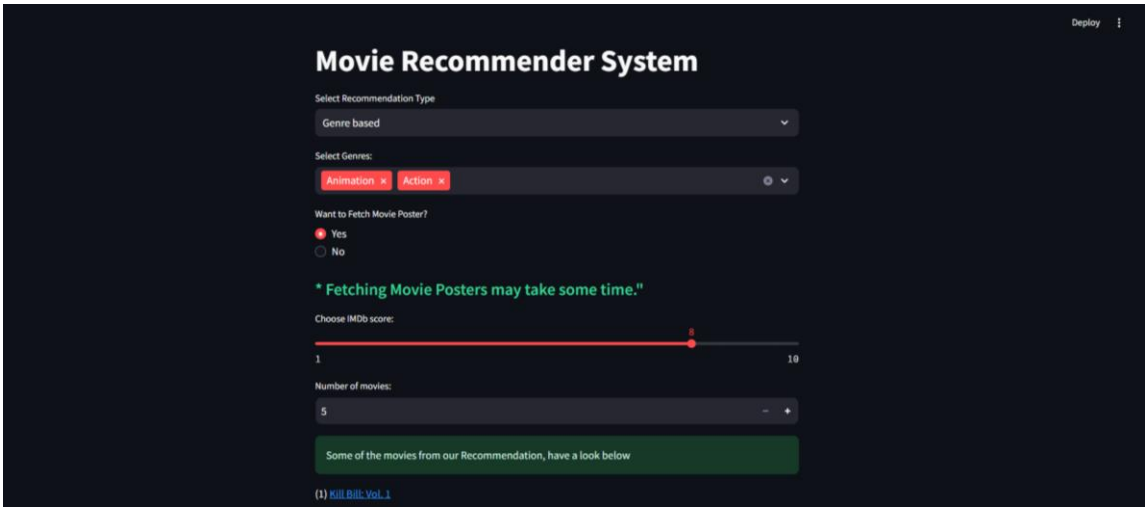
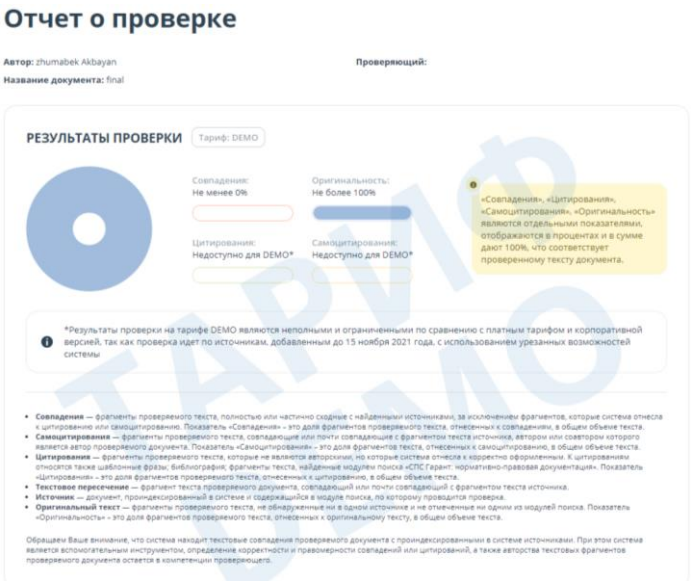


Figure 1.3 Genres recommendation page



Appendix 2. Anti-plagiarism

Figure 2.1 Plagiarism Analysis



Отчет о проверке

Автор: zhumabek Akbayan
Проверяющий:
Название документа: final

РЕЗУЛЬТАТЫ ПРОВЕРКИ Тариф: DEMO

Совпадения: Не менее 0%
Оригинальность: Не более 100%
Цитирования: Недоступно для DEMO*
Самоцитирования: Недоступно для DEMO*

«Совпадения», «Цитирования», «Самоцитирования», «Оригинальность» являются отдельными показателями, отображаются в процентах и в сумме дают 100%, что соответствует проверенному тексту документа.

*Результаты проверки на тарифе DEMO являются неполными и ограниченными по сравнению с платным тарифом и корпоративной версией, так как проверка идет по источникам, добавленным до 15 ноября 2021 года, с использованием урезанных возможностей системы

- **Совпадения** — фрагменты проверяемого текста, полностью или частично сходные с найденными источниками, за исключением фрагментов, которые система отнесла к цитированию или самоцитированию. Показатель «Совпадения» - это доля фрагментов проверяемого текста, отнесенных к совпадениям, в общем объеме текста.
- **Самоцитирования** — фрагменты проверяемого текста, совпадающие или почти совпадающие с фрагментом текста источника, автором или соавтором которого является автор проверяемого документа. Показатель «Самоцитирования» - это доля фрагментов текста, отнесенных к самоцитированию, в общем объеме текста.
- **Цитирования** — фрагменты проверяемого текста, которые не являются авторскими, но которые система отнесла к корректно оформленным. К цитированиям относятся также шаблонные фразы; библиография; фрагменты текста, найденные модулем поиска «СПС Гарант: нормативно-правовая документация». Показатель «Цитирования» - это доля фрагментов проверяемого текста, отнесенных к цитированию, в общем объеме текста.
- **Текстовое пересечение** — фрагмент текста проверяемого документа, совпадающий или почти совпадающий с фрагментом текста источника.
- **Источник** — документ, проиндексированный в системе и содержащийся в модуле поиска, по которому проводится проверка.
- **Оригинальный текст** — фрагменты проверяемого текста, не обнаруженные ни в одном источнике и не отмеченные ни одним из модулей поиска. Показатель «Оригинальность» - это доля фрагментов проверяемого текста, отнесенных к оригинальному тексту, в общем объеме текста.

Обращаем Ваше внимание, что система находит текстовые совпадения проверяемого документа с проиндексированными в системе источниками. При этом система является вспомогательным инструментом, определение корректности и правомерности совпадений или цитирований, а также авторства текстовых фрагментов проверяемого документа остается в компетенции проверяющего.