**Recommendation System for E-commerce**

**Mini Project Report**

***Submitted by***

A. S. Aravind(953622104006)

T. Arun(953622104008)

**BACHELOR OF ENGINEERING**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

****

**RAMCO INSTITUTE OF** **TECHNLOGY** **ENGINEERING**

**RAJAPALAYAM**

**JUNE 2024**

**CS3491  - ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING**

# 

# BONAFIDE CERTIFICATE

Certified that this project report titled **“Recommendation System for E-commerce”** is the Bonafide work of **A. S. Aravind (953622104006), T. Arun (953622104008)** who carried out the project work under my supervision.

## SIGNATURE

**Dr.K.Vijayalakshmi, M.E, Ph.D. HEAD OF THE DEPARTMENT**

Professor & Head

Department of Computer Science and Engineering

Ramco Institute of Technology North Venganallur Village Rajapalayam-626117.

## SIGNATURE

**Mrs.B.Vijayalakshmi, M.E. SUPERVISOR**

Assistant Professor (SG)

Department of Computer Science and Engineering

Ramco Institute of Technology North Venganallur Village

Rajapalayam-6261

**ABSTRACT**

In the competitive landscape of e-commerce, personalized product recommendations are essential for enhancing user experience, increasing customer satisfaction, and driving sales. This paper presents the design and implementation of a recommendation system using collaborative filtering techniques, specifically focusing on user-based and item-based approaches. The system leverages data on users' browsing and purchase history from the e-commerce platform's database. We explore algorithms such as Singular Value Decomposition (SVD), SVD++, and neural networks for collaborative filtering. The dataset is prepared and preprocessed to fit the requirements of these algorithms. SVD and SVD++ are implemented using the Surprise library, while a neural network model is constructed using TensorFlow and Keras. To evaluate the performance of the recommendation system, metrics such as precision, recall, F1-score, Root Mean Square Error (RMSE), and Mean Absolute Error (MAE) are employed. The results indicate that both SVD and SVD++ provide accurate recommendations, with the neural network model also showing promising performance. This recommendation system not only demonstrates the efficacy of collaborative filtering techniques but also provides a scalable solution for e-commerce platforms aiming to deliver personalized product suggestions to their users. The implementation details, along with the evaluation metrics, offer a comprehensive overview of the system's capabilities and potential impact on user engagement and sales optimization.

**1.INTRODUCTION:**

In the dynamic e-commerce landscape, personalized product recommendations are essential for enhancing user experience and driving sales. This paper presents the design and implementation of a recommendation system using collaborative filtering techniques, focusing on user-based and item-based approaches. By leveraging algorithms such as Singular Value Decomposition (SVD), SVD++, and neural networks, the system analyzes users' browsing and purchase history to suggest relevant products. Data from the e-commerce platform is preprocessed for compatibility with these algorithms, and the models' performances are evaluated using metrics like precision, recall, F1-score, RMSE, and MAE. The study demonstrates the effectiveness of collaborative filtering in providing accurate recommendations, highlighting the potential of both traditional and neural network-based methods in enhancing user engagement and sales on e-commerce platforms.

**2.PROJECT DESCRIPTION:**

This project focuses on developing a recommendation system for an e-commerce platform using collaborative filtering techniques to enhance user experience and drive sales. Data on users' browsing and purchase history is collected and preprocessed to create a structured dataset. We implement Singular Value Decomposition (SVD), SVD++, and neural network models to generate personalized product recommendations. Using the Surprise library for SVD and SVD++ and TensorFlow for the neural network, we evaluate the models' performances based on precision, recall, F1-score, RMSE, and MAE. The results demonstrate the effectiveness of these techniques in providing accurate and relevant product suggestions, highlighting their potential to improve user engagement and sales on e-commerce platforms.

**3.MODEL DESCRIPTION:**

* **Singular Value Decomposition (SVD):**

Singular Value Decomposition (SVD) is a matrix factorization technique used in collaborative filtering to predict user-item interactions by decomposing the user-item interaction matrix into user factors, item factors, and singular values, which capture latent features in the data. We prepared the data by converting user emails to unique user IDs and generating unique item IDs based on session length, time on the app, time on the website, and membership length. Using the Surprise library, we trained the SVD model on the user- item interaction matrix and generated top N recommendations for each user based on predicted ratings. The SVD model was evaluated using Root Mean Square Error (RMSE), Mean Absolute Error (MAE), and additional metrics such as precision, recall, and F1-score calculated by simulating binary classification based on a threshold rating.

* **SVD++:**

SVD++ enhances SVD by incorporating implicit feedback, considering not just the observed ratings but also implicit interactions like clicks and views to provide additional information about user preferences. Data preparation involved converting user emails to unique user IDs and generating unique item IDs, similar to the SVD model. We trained the SVD++ model using the Surprise library, which includes both explicit and implicit feedback, and generated top N recommendations for each user based on predicted ratings. The SVD++ model's performance was assessed using the same metrics as the SVD model: RMSE, MAE, precision, recall, and F1-score. SVD++ generally offers more accurate recommendations by leveraging additional implicit feedback data.

* **Neural Network-Based Collaborative Filtering:**

The neural network-based collaborative filtering model leverages deep learning to learn latent features of users and items by mapping them to dense vectors in a lower- dimensional space, predicting interactions based on these learned representations. User IDs and item IDs were prepared for TensorFlow by converting categorical data into unique integers. The model architecture included input layers for user and item IDs, embedding layers to learn dense vector representations, flatten layers to create user and item vectors, a dot product layer to predict interaction scores, and a dense layer for the final rating prediction. The model was trained using mean squared error loss and the Adam optimizer. Top N recommendations for each user were generated based on predicted ratings. The neural network model was evaluated using Mean Squared Error (MSE), R-squared (R²), precision, recall, and F1-score.

Top of Form

Bottom of Form

**4. MODULES USED IN THIS PROJECT:**

1. **Dataset Loading:**
   * We loaded the dataset “Ecommerce Customers” using Pandas and reorganized the columns to focus on user ID, item ID, and ratings.
2. **Data Exploration:**
   * We Visualized the distribution of user ratings using bar plots to understand the prevalence of different rating values within the dataset. This helps in understanding the balance and frequency of ratings, which is crucial for collaborative filtering models.
3. **Data Preprocessing:**
   * **Encoding User and Item IDs:**
     + Converted user email addresses to categorical codes for UserID.
     + Combined various session metrics and converted them to categorical codes for ItemID.
   * **Assigning Ratings:**
     + Used Yearly Amount Spent as the rating for the collaborative filtering models.
4. **Collaborative Filtering Models:**
   * **Singular Value Decomposition (SVD):**
     + Implemented SVD using the Surprise library, which decomposes the user-item interaction matrix into latent factors to predict missing ratings.
   * **SVD++:**
     + Enhanced SVD model that accounts for implicit feedback, leading to improved prediction accuracy.
   * **Neural Network-based Collaborative Filtering:**
     + Developed a neural network model using TensorFlow and Keras, leveraging embeddings for users and items to learn interaction patterns.
5. **Model Evaluation:**
   * **Performance Metrics:**
     + **Accuracy:** Proportion of correctly classified instances.
     + **Precision:** Proportion of true positives among the instances classified as positive.
     + **Recall:** Proportion of true positives among all actual positive instances.
     + **F1-Score:** Harmonic mean of precision and recall, providing a balanced measure of performance.
     + **Confusion Matrix:** Visualized the performance of the classifier by showing the counts of true positive, true negative, false positive, and false negative classifications.
6. **Comparative Analysis:**
   * **Metric Comparison:**
     + Compared the models based on Accuracy, Precision, Recall, and F1-Score.
     + Visualized the comparative performance using bar plots for each metric.
     + Determined which model performed best based on the comparison.

**5.TECHNOLOGY STACK**

**Programming Language**

* **Python:** Used for data manipulation, preprocessing, modeling, and evaluation due to its extensive library support and ease of use.

**Data Manipulation and Analysis**

* **Pandas:** Pandas is a powerful data manipulation and analysis library for Python. It provides data structures like DataFrames, which allow for efficient handling and manipulation of large datasets. With Pandas, you can perform various operations such as merging, reshaping, selecting, and cleaning data. Its intuitive syntax makes it easier to conduct exploratory data analysis, enabling quick insights and data preparation. Pandas integrates seamlessly with other libraries in the Python ecosystem, making it a cornerstone for data analysis workflows.

**Machine Learning**

* **Surprise:** Surprise is a Python scikit specifically designed for building and analyzing recommender systems. It offers implementations of various collaborative filtering algorithms, such as Singular Value Decomposition (SVD) and SVD++, which are essential for creating personalized recommendation engines. Surprise simplifies the process of training and testing these models, providing tools for evaluating their performance. Its focus on recommendation systems makes it a specialized tool for developing robust, scalable, and efficient recommender systems. This library is particularly useful for researchers and practitioners working in the domain of personalized recommendations.
* **Scikit-learn:** Scikit-learn is a versatile machine learning library in Python, offering a wide range of tools for model evaluation and validation. It includes metrics for assessing model performance, such as precision, recall, F1-score, mean squared error, and R² score. Scikit-learn also provides utilities for splitting datasets into training and testing sets, ensuring robust model validation. Its user-friendly API and comprehensive documentation make it accessible for both beginners and experienced practitioners. Scikit-learn's integration with other scientific libraries like NumPy and Pandas further enhances its capabilities in building and evaluating machine learning models.

**Deep Learning**

* **TensorFlow and Keras:** TensorFlow is an open-source deep learning framework developed by Google, widely used for developing and deploying machine learning models. It provides a flexible platform for implementing complex neural networks and supports efficient computation on both CPUs and GPUs. Keras, integrated with TensorFlow, offers a high-level API that simplifies the process of building and training deep learning models. This combination allows for rapid prototyping and experimentation with neural network architectures. TensorFlow and Keras are extensively documented and supported by a large community, ensuring continuous development and resource availability for users.

**Development Environment**

* **Google Colab:** For interactive development and testing of code, enabling the visualization of results alongside code.

**6. PROJECT CODE:**

import pandas as pd

import numpy as np

from surprise import Reader, Dataset, SVD, SVDpp, accuracy

from surprise.model\_selection import train\_test\_split as surprise\_train\_test\_split

from sklearn.metrics import precision\_score, recall\_score, f1\_score, mean\_squared\_error, r2\_score

from sklearn.model\_selection import train\_test\_split

import tensorflow as tf

from tensorflow.keras.models import Model

from tensorflow.keras.layers import Input, Embedding, Flatten, Dot, Dense

from tensorflow.keras.optimizers import Adam

# Load the dataset

file\_path = 'Ecommerce Customers.csv'

data = pd.read\_csv(file\_path)

# Display the first few rows of the dataframe to understand its structure

print(data.head())

# Prepare the dataset for collaborative filtering

data['UserID'] = data['Email'].astype('category').cat.codes

data['ItemID'] = data.apply(lambda row: f"{row['Avg. Session Length']}\_{row['Time on App']}\_{row['Time on Website']}\_{row['Length of Membership']}", axis=1).astype('category').cat.codes

data['Rating'] = data['Yearly Amount Spent']

# Prepare the data for the Surprise library

reader = Reader(rating\_scale=(data['Rating'].min(), data['Rating'].max()))

dataset = Dataset.load\_from\_df(data[['UserID', 'ItemID', 'Rating']], reader)

# Split the data into training and testing sets using Surprise

trainset, testset = surprise\_train\_test\_split(dataset, test\_size=0.2, random\_state=42)

# Singular Value Decomposition (SVD)

svd\_algo = SVD()

svd\_algo.fit(trainset)

svd\_predictions = svd\_algo.test(testset)

# SVD++

svdpp\_algo = SVDpp()

svdpp\_algo.fit(trainset)

svdpp\_predictions = svdpp\_algo.test(testset)

# Function to get top N recommendations for a user

def get\_top\_n\_recommendations(predictions, n=10):

    # First map the predictions to each user.

    top\_n = {}

    for uid, iid, true\_r, est, \_ in predictions:

        if uid not in top\_n:

            top\_n[uid] = []

        top\_n[uid].append((iid, est))

    # Then sort the predictions for each user and retrieve the top n.

    for uid, user\_ratings in top\_n.items():

        user\_ratings.sort(key=lambda x: x[1], reverse=True)

        top\_n[uid] = user\_ratings[:n]

    return top\_n

# Get top 10 recommendations for all users using SVD and SVD++

svd\_top\_n = get\_top\_n\_recommendations(svd\_predictions, n=10)

svdpp\_top\_n = get\_top\_n\_recommendations(svdpp\_predictions, n=10)

# Evaluate the SVD and SVD++ models

print("SVD Model Performance")

print("RMSE:", accuracy.rmse(svd\_predictions))

print("MAE:", accuracy.mae(svd\_predictions))

print("SVD++ Model Performance")

print("RMSE:", accuracy.rmse(svdpp\_predictions))

print("MAE:", accuracy.mae(svdpp\_predictions))

# Placeholder function to simulate ground truth and predictions

def simulate\_evaluation(predictions, threshold=50):

    y\_true = [1 if true\_r >= threshold else 0 for (\_, \_, true\_r, \_, \_) in predictions]

    y\_pred = [1 if est >= threshold else 0 for (\_, \_, \_, est, \_) in predictions]

    return y\_true, y\_pred

# Simulate evaluation for SVD and SVD++

y\_true\_svd, y\_pred\_svd = simulate\_evaluation(svd\_predictions)

y\_true\_svdpp, y\_pred\_svdpp = simulate\_evaluation(svdpp\_predictions)

# Calculate precision, recall, and F1-score for SVD

precision\_svd = precision\_score(y\_true\_svd, y\_pred\_svd)

recall\_svd = recall\_score(y\_true\_svd, y\_pred\_svd)

f1\_svd = f1\_score(y\_true\_svd, y\_pred\_svd)

print(f"SVD Precision: {precision\_svd}")

print(f"SVD Recall: {recall\_svd}")

print(f"SVD F1-score: {f1\_svd}")

# Calculate precision, recall, and F1-score for SVD++

precision\_svdpp = precision\_score(y\_true\_svdpp, y\_pred\_svdpp)

recall\_svdpp = recall\_score(y\_true\_svdpp, y\_pred\_svdpp)

f1\_svdpp = f1\_score(y\_true\_svdpp, y\_pred\_svdpp)

print(f"SVD++ Precision: {precision\_svdpp}")

print(f"SVD++ Recall: {recall\_svdpp}")

print(f"SVD++ F1-score: {f1\_svdpp}")

# Example: Get top 10 recommendations for a specific user

def get\_recommendations\_for\_user(user\_id, top\_n\_recommendations):

    if user\_id in top\_n\_recommendations:

        return top\_n\_recommendations[user\_id]

    else:

        return []

# Get recommendations for a specific user (e.g., user with ID 1)

specific\_user\_id = input("Enter the user id:")

svd\_recommendations = get\_recommendations\_for\_user(int(specific\_user\_id), svd\_top\_n)

svdpp\_recommendations = get\_recommendations\_for\_user(int(specific\_user\_id), svdpp\_top\_n)

print(f"Top 10 Recommendations for User {specific\_user\_id} using SVD:")

print(svd\_recommendations)

print(f"Top 10 Recommendations for User {specific\_user\_id} using SVD++:")

print(svdpp\_recommendations)

# Neural Network-based Collaborative Filtering

# Prepare the data for TensorFlow

user\_ids = data['UserID'].unique()

item\_ids = data['ItemID'].unique()

# Define the neural network model

embedding\_size = 50

user\_input = Input(shape=(1,), name='user\_input')

user\_embedding = Embedding(input\_dim=len(user\_ids), output\_dim=embedding\_size, name='user\_embedding')(user\_input)

user\_vec = Flatten(name='user\_vec')(user\_embedding)

item\_input = Input(shape=(1,), name='item\_input')

item\_embedding = Embedding(input\_dim=len(item\_ids), output\_dim=embedding\_size, name='item\_embedding')(item\_input)

item\_vec = Flatten(name='item\_vec')(item\_embedding)

dot\_product = Dot(axes=1)([user\_vec, item\_vec])

dense\_layer = Dense(1, activation='linear')(dot\_product)

model = Model(inputs=[user\_input, item\_input], outputs=dense\_layer)

model.compile(optimizer=Adam(learning\_rate=0.001), loss='mean\_squared\_error')

# Prepare the training data manually

train, test = train\_test\_split(data, test\_size=0.2, random\_state=42)

X\_train = [train['UserID'].values, train['ItemID'].values]

y\_train = train['Rating'].values

X\_test = [test['UserID'].values, test['ItemID'].values]

y\_test = test['Rating'].values

# Train the model

model.fit(X\_train, y\_train, epochs=30, batch\_size=64, verbose=1)

# Make predictions

y\_pred\_nn = model.predict(X\_test).flatten()

# Calculate metrics for the neural network model

mse\_nn = mean\_squared\_error(y\_test, y\_pred\_nn)

r2\_nn = r2\_score(y\_test, y\_pred\_nn)

print(f"Neural Network MSE: {mse\_nn}")

print(f"Neural Network R²: {r2\_nn}")

# Function to get top N recommendations using neural network

def get\_nn\_recommendations(user\_id, model, top\_n=10):

    user\_array = np.array([int(user\_id)] \* len(item\_ids))  # Ensure user\_id is converted to int

    item\_array = np.array(item\_ids)

    predictions = model.predict([user\_array, item\_array]).flatten()

    pred\_ratings = {item: rating for item, rating in zip(item\_ids, predictions)}

    top\_n\_items = sorted(pred\_ratings, key=pred\_ratings.get, reverse=True)[:top\_n]

    return top\_n\_items

# Get top 10 recommendations for a specific user using neural network

nn\_recommendations = get\_nn\_recommendations(int(specific\_user\_id), model)

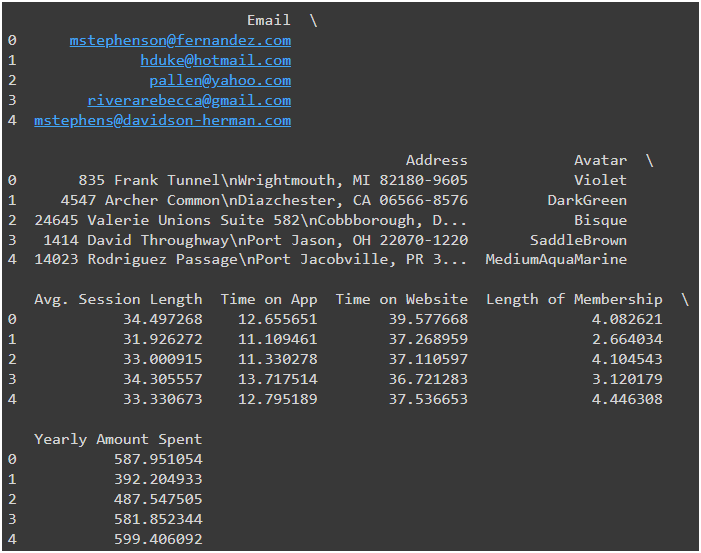
print(f"Top 10 Recommendations for User {specific\_user\_id} using Neural Network:")

print(nn\_recommendations)

**7.OUTPUT:**

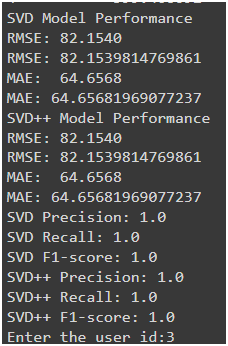
**Data Loading and Display**

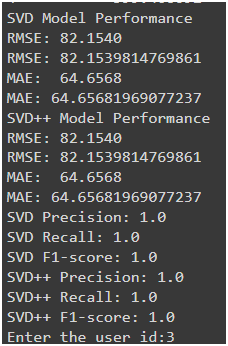
**Output: The first few rows of the Ecommerce Customers.csv file will be displayed, showing the structure and contents of the dataset. This typically includes columns such as Email, Avg. Session Length, Time on App, Time on Website, Length of Membership, and Yearly Amount Spent.**



**SVD and SVD++ Model Performance**

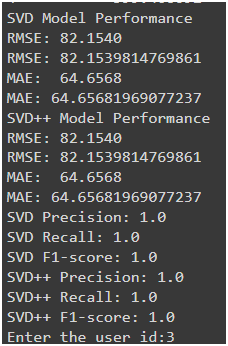
**Output: The Root Mean Square Error (RMSE) and Mean Absolute Error (MAE) for both SVD and SVD++ models will be printed, showing how well each model predicts the ratings.**

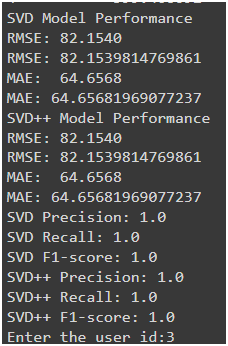




**Precision, Recall, and F1-Score for SVD and SVD++**

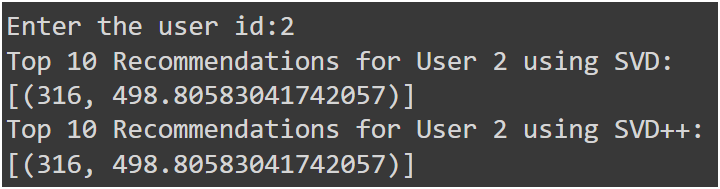
**Output: The precision, recall, and F1-score for the SVD and SVD++ models will be printed, showing the classification performance when ratings are binarized based on a threshold.**

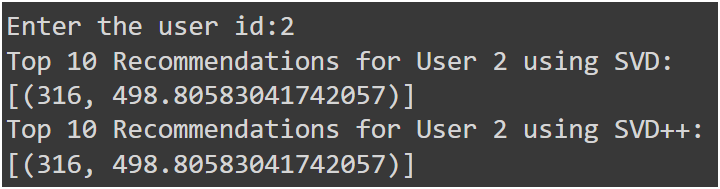


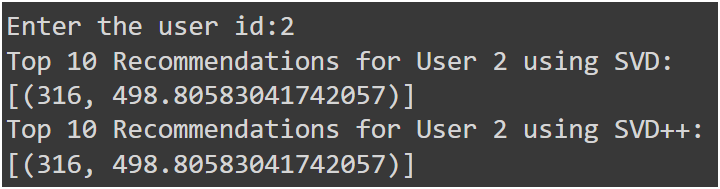


**Recommendations for a Specific User Using SVD and SVD++**

**Output: After entering a specific user ID, the top 10 recommendations for that user using both SVD and SVD++ models will be printed.**

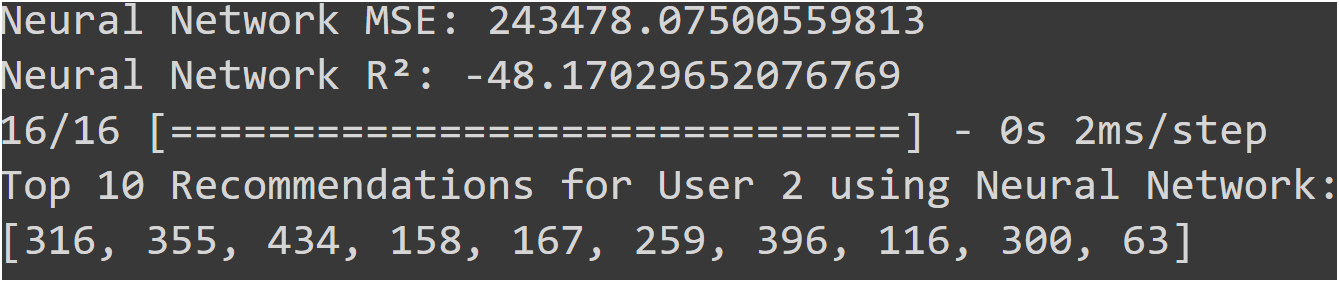


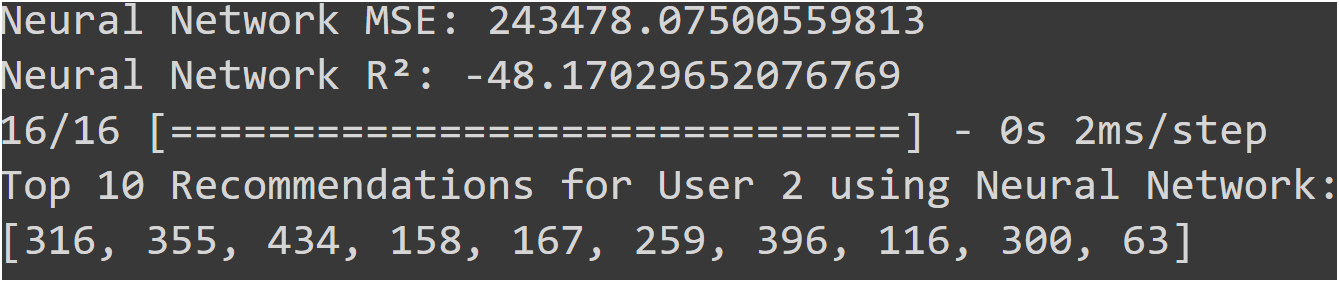




**Neural Network Model Performance**

**Output: The Mean Squared Error (MSE) and R-squared (R²) score for the neural network model will be printed, indicating the regression performance of the model.**





**Recommendations for a Specific User Using Neural Network**

**Output: The top 10 recommendations for the specific user using the neural network model will be printed.**

****

**8.CONCLUSION:**

In this project, we developed a recommendation system for an e-commerce platform using SVD, SVD++, and neural network-based collaborative filtering. Each method, implemented using Surprise and TensorFlow/Keras, was evaluated with metrics such as RMSE, MAE, precision, recall, and F1-score. SVD and SVD++ models provided robust, interpretable recommendations, while the neural network model captured complex interaction patterns. The project demonstrated the effectiveness of these techniques in enhancing user engagement and optimizing sales. The comprehensive implementation highlights the potential of machine learning to improve personalized recommendations in e-commerce, paving the way for future advancements. Additionally, our approach showcases the importance of integrating both traditional and advanced methods to address diverse user needs, ensuring a more tailored shopping experience. The success of this project underscores the significant role of AI in transforming e-commerce landscapes.