ALPEREN KÜTÜK-180101030

YUNUS ZİYA ARIKAN-180101020

COMPUTER ENGINEERING

Project Report: Amazon Product Recommendation System

Introduction:

The primary objective of this project was to design and implement an effective recommendation system for electronic products available on Amazon, leveraging user reviews and ratings. This system aims to enhance the shopping experience by suggesting relevant products to users based on their past interactions. The project explored two main approaches: a popularity-based recommendation system and collaborative filtering-based recommendation systems, analyzing their performance and suitability for real-world application. A recommendation system is an advanced technology that utilizes machine learning and data analysis to provide personalized suggestions to users. It operates by collecting and analyzing user behavior, user preferences, and historical user item interactions.

By applying complex algorithms and statistical models, recommendation engines are capable of predicting and presenting users with items, services, or content that align with their interests and preferences.

There are different types of recommendation systems, including collaborative filtering methods, content-based filtering, and user-based collaborative filtering.Recommender system technology is widely used across various industries, particularly in e-commerce, streaming platforms, news and media, and digital marketing, to enhance user engagement, boost user trust, increase sales, and improve overall customer satisfaction.

We worked about E-Commerce Electronics.

How does a recommendation system work?

Step 1: Collecting user data

Step 2: Analyzing data

Step 3: Filtering

Step 4: Generating recommendations

PURPOSE:

Machine learning, as a subcategory of artificial intelligence, enables recommender systems to recognize patterns and relationships in large historical datasets, such as understanding complex aspects of user's behavior.

For generating tailored content, recommendation systems rely on specific training data and algorithms.

4 Types of Recommendation Systems

1. Collaborative filtering recommender systems

User-based Collaborative Filtering

Item-based Collaborative Filtering

2. Content based recommender systems

Content-based filtering focus on the attributes of items and give you recommendations based on the similarity between them.

3. Hybrid recommender systems

4. Deep learning-based recommendations

(We used Collaborative filtering recommender systems)

Benefits of recommendation systems:

Recommendation systems are now an essential component of various digital platforms, ranging from e-commerce websites to streaming services.

Each platform benefits differently from the recommender system. We all can observe the effects of the recommendation system on a daily basis. Recommender systems lead to a better user experience and significantly reduce the time spent searching for the best items.

Personalized user experience

Increased sales and revenue

Enhanced user engagement

Targeted advertising

Amazon's recommendation system:

Amazon is renowned for its highly accurate product recommendations in its online store. The company leverages advanced technologies, including:

1-Artificial intelligence algorithms:

These algorithms analyze vast amounts of data to predict what products a user might be interested in.

2- Machine learning: Amazon uses machine learning to continuously improve and refine its recommendation system based on user behavior and feedback.

3- Collaborative filtering: Generates recommendations to other users based on the behavior of similar users.

4- Personalized recommendations: Amazon's system tailors product suggestions based on individual user behavior, browsing history, and purchase history.

5- Item-to-Item collaborative filtering: Instead of segmenting users into groups, Amazon uses collaborative filtering method to match each product to a set of similar products. This way, when a particular user views a product, they get recommendations of similar items.

RESULT:

Data Preparation and Exploratory Data Analysis

1. Data Loading and Cleaning:

The dataset utilized consists of user ratings for a variety of electronic products on Amazon. This dataset was imported into the analysis environment for preprocessing.

Initial data cleaning involved handling missing values by either removing records with significant gaps or imputing missing data where feasible. Additionally, ensuring consistent data types across the dataset was critical for accurate analysis.

Duplicate entries were identified and removed to prevent skewing the analysis. This cleaning process was essential to ensure the data's integrity and reliability for subsequent steps.

2. Exploratory Data Analysis (EDA):

EDA was conducted to uncover underlying trends and patterns within the dataset. Key steps included plotting the distribution of ratings to understand user sentiment towards products.

The analysis showed that a majority of users rated only a few products, indicating a typical long-tail distribution common in e-commerce datasets.

Dataset Analysis:

The dataset used for this project provided a rich source of information on user interactions with electronic products on Amazon. Here are some key insights from the dataset analysis.

1. Rating Distribution:

The dataset included ratings ranging from 1 to 5 stars. The distribution revealed a high concentration of ratings at 4 and 5 stars, suggesting generally positive user experiences with the products.

A notable observation was the relatively fewer 1 and 2-star ratings, which could indicate either high product quality or a bias towards positive feedback among users.

2. User Engagement:

Analysis of user engagement highlighted that a small proportion of users were highly active, contributing a large number of ratings. These power users play a crucial role in the recommendation system's data pool.

The majority of users, however, rated only a handful of products. This sparse rating pattern is typical in large-scale recommendation systems and presents challenges for collaborative filtering algorithms, which rely on sufficient user data to make accurate predictions.

3. Product Popularity:

The dataset revealed certain products that were highly popular, receiving a large number of ratings. These products often had high average ratings, making them strong candidates for recommendations in a popularity-based system.

Conversely, many products had very few ratings, reflecting the long-tail phenomenon where a vast number of items receive limited user attention.

4. User-Product Interaction Matrix:

The interaction matrix, representing users and their rated products, was highly sparse, with most entries being zero (indicating no rating). This sparsity is a common challenge in recommendation systems, requiring sophisticated algorithms to infer preferences from limited data points.

Data Subset Creation:

To enhance computational efficiency and focus the analysis on more relevant data, a subset of the original dataset was created. This subset was based on the following criteria:

Users who have rated at least 50 products.

Products that have received at least 50 ratings.

This filtering reduced the dataset size significantly, enabling more efficient computation while retaining the interactions most indicative of user preferences.

Popularity-Based Recommendation System:

The popularity-based recommendation system offers a straightforward approach to suggesting products, relying solely on overall product popularity. Key steps included:

Computing Average Ratings:

For each product, the average rating was computed to serve as a measure of its popularity.

Products were then ranked based on their average ratings, with the highest-rated products being recommended to users.

Evaluation:

The effectiveness of the popularity-based recommendation system was evaluated using the Root Mean Square Error (RMSE) metric, comparing predicted ratings with actual user ratings.

On the test dataset, this model achieved an RMSE value of 1.0815. Although simple, this approach provides a baseline for comparison with more sophisticated recommendation methods.

Collaborative Filtering-Based Recommendation System:

Collaborative filtering methods aim to provide personalized recommendations by leveraging similarities between users and items. Two key algorithms were implemented and evaluated:

1. KNNWithMeans Algorithm:

This algorithm employs the K-nearest neighbors approach to identify users with similar rating patterns. Ratings from these similar users are averaged (with mean normalization) to predict ratings for items not yet rated by the active user.

The KNNWithMeans model achieved an RMSE value of 0.9668, indicating a significant improvement over the popularity-based approach.

2. SVD and SVDpp Algorithms:

Singular Value Decomposition (SVD) and its variant SVDpp were employed to capture latent factors in the user-item interaction matrix.

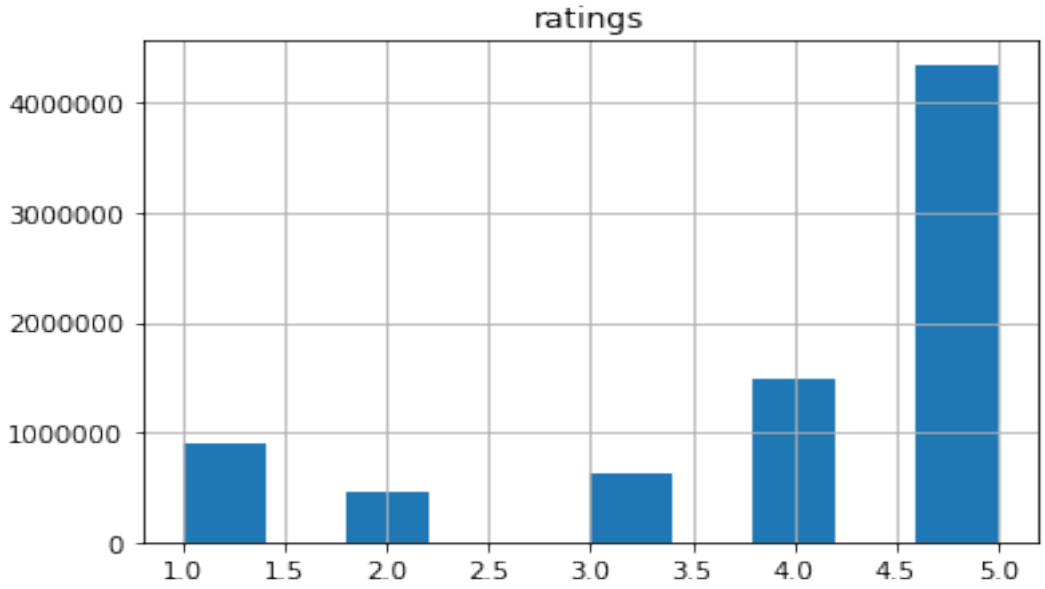
Hyperparameters for these models were optimized using GridSearchCV, ensuring the best possible performance.

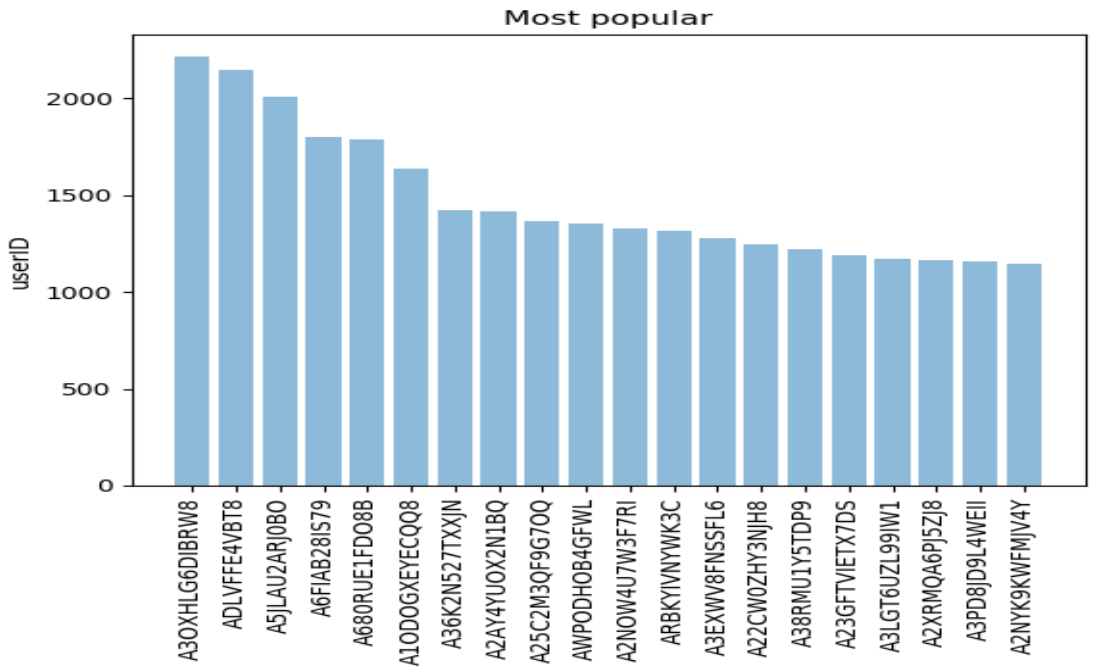
The SVD model achieved an RMSE value of 0.9573, while the SVDpp model slightly improved this to 0.9570. These results demonstrate a high level of accuracy in predicting user ratings, outperforming the simpler models.

Conclusion and Recommendations:

Performance Comparison:

The collaborative filtering methods, particularly SVD and SVDpp, significantly outperformed the popularity-based recommendation system. This underscores the effectiveness of personalized recommendation techniques in capturing user preferences and providing more accurate suggestions.





In [63]:

```
import sklearn.metrics as metric
from math import sqrt
MSE = metric.mean_squared_error(pred_df['true_ratings'], pred_df['predicted_ratings'])
print('The RMSE value for Popularity Recommender model is', sqrt(MSE))
```

```
The RMSE value for Popularity Recommender model is 1.0815377502945511
```

The RMSE value for Popularity Recommender model is 1.0815

SVD

In [116…

```
def evaluate_algorithms(data):
    algorithms = {
        'KNN Basic': KNNBasic(),
        'KNN With Means': KNNWithMeans(),
        'KNN With ZScore': KNNWithZScore(),
        'SVD': SVD(),
        'SVDpp': SVDpp(),
    }
```

In [117…

```
param_grid = {
    'n_epochs': [20, 25], # number of iterations
    'lr_all': [0.007, 0.009, 0.01], # learning rate
    'reg_all': [0.4, 0.6] # regularization term
    }
```

In [118…

```
gs_svd = GridSearchCV(SVD, param_grid, measures=['rmse', 'mae'], cv=5, n_jobs=-1) # cross-validation
gs_svd.fit(data)

#Perform grid search for SVDpp
gs_svdpp = GridSearchCV(SVDpp, param_grid, measures=['rmse', 'mae'], cv=5, n_jobs=-1)
gs_svdpp.fit(data)

#Best results
print('Best RMSE for SVD:', gs_svd.best_score['rmse'])
print('Best parameters for SVD:', gs_svd.best_params['rmse'])
print('Best RMSE for SVDpp:', gs_svdpp.best_score['rmse'])
print('Best parameters for SVDpp:', gs_svdpp.best_params['rmse'])
```

```
Best RMSE for SVD: 0.9572690395866879
Best parameters for SVD: {'n_epochs': 20, 'lr_all': 0.007, 'reg_all': 0.4}
Best RMSE for SVDpp: 0.9569773590616748
Best parameters for SVDpp: {'n_epochs': 20, 'lr_all': 0.007, 'reg_all': 0.4}
```

Popularity Recommender Model (RMSE)

In [120…
```
MSE = metric.mean_squared_error(pred_df['true_ratings'], pred_df['predicted_ratings'])
print('The RMSE value for Popularity Recommender model is', sqrt(MSE))
```

```
The RMSE value for Popularity Recommender model is 1.0815377502945511
```

Collaborative Filtering Recommender Model (RMSE)

In [121…
```
print(len(testset))
type(testset)
```

```
19004
```

Out[121… `list`

KNNWithMeans

In [122…
```
# Evalute on test set
test_pred = algo_user.test(testset)
test_pred[0]
```

Out[122… `Prediction(uid='A2XHOLOLKGV1FE', iid='B000NDA5E0', r_ui=4.0, est=2.789473684210526, details={'actual_k': 0, 'was_impossible': False})`

In [96]:
```
# compute RMSE
accuracy.rmse(test_pred) #range of value of error
```

```
RMSE: 1.0016
```

Out[96]: `1.0015950381186367`

SVD

In [123…
```
best_svd = gs_svd.best_estimator['rmse']
best_svd.fit(trainset)
```

Out[123… `<surprise.prediction_algorithms.matrix_factorization.SVD at 0x1ccc29438b0>`

In [124…
```
#Train best SVDpp model
predictions_svd = best_svd.test(testset)
print('Test RMSE for SVD:', accuracy.rmse(predictions_svd))
```

```
RMSE: 0.9675
Test RMSE for SVD: 0.9675483811431123
```

In [128…
```
# Use the "best model" for prediction
gs.test(testset)
accuracy.rmse(gs.test(testset))
```

```
RMSE: 0.8608
```

Out[128… `0.8608374806356534`

The RMSE value for Collaborative Filtering model, byKNNWithMeans is 1.0015950381186367 and SVD is 0.9672121554841631. After parameter tuning of SVD it is 0.8612

Get top - K (K = 5) recommendations. Since our goal is to recommend new products to each user based on his/her habits, we will recommend 5 new products.

```
from collections import defaultdict
def get_top_n(predictions, n=5):

    # First map the predictions to each user.
    top_n = defaultdict(list)
    for uid, iid, true_r, est, _ in predictions:
        top_n[uid].append((iid, est))

    # Then sort the predictions for each user and retrieve the k highest ones.
    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
```

```
top_n = get_top_n(test_pred, n=5)
```

```
# Print the recommended items for each user
for uid, user_ratings in top_n.items():
    print(uid, [iid for (iid, _) in user_ratings])
```

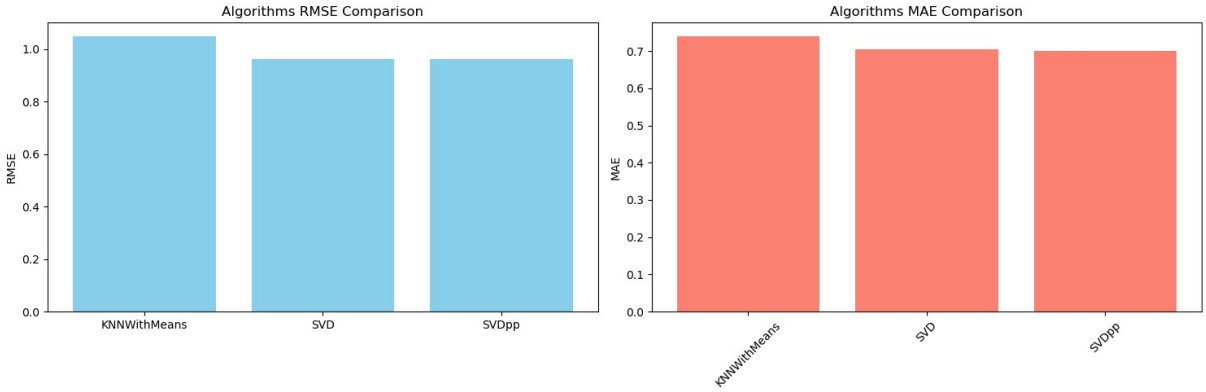
```
A2XHOLOLKGV1FE ['B0076LY7UK', 'B0077R2B9M', 'B004T1YA5W', 'B000NDA5E0', 'B00429N18S']
A1PL7QILVQV3IF ['B007M4M3NO', 'B000B8IHDS', 'B00829THK0', 'B000JE7GPY', 'B000R6QAHY']
AY6A8KPYCE6B0 ['B0000E1717', 'B004YIFP9K', 'B0056YNA1Q', 'B00CD8ADKO', 'B0041OMWNY']
A2J0IBS4PFRO2C ['B000OK2X6K', 'B0007Z1M50', 'B000A3WS84', 'B0000C4G79', 'B003VNKKRG']
A21T0D2F7SKG5S ['B001GO5TU0', 'B006QB1RPY', 'B000BH5OW6', 'B0002Y5WXE', 'B00009R6VZ']
A1U5NWJOYH2QQH ['B008J127VC', 'B0001WXTF0', 'B000A4AVQO', 'B00011Y1MQ', 'B0007P11M4']
```

```
Computing the msd similarity matrix...
Done computing similarity matrix.
Computing the msd similarity matrix...
Done computing similarity matrix.
Computing the msd similarity matrix...
Done computing similarity matrix.
Computing the msd similarity matrix...
Done computing similarity matrix.
Computing the msd similarity matrix...
Done computing similarity matrix.
Evaluating RMSE, MAE of algorithm KNNWithMeans on 5 split(s).

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    1.0465  1.0511  1.0397  1.0527  1.0546  1.0489  0.0053
MAE (testset)     0.7405  0.7466  0.7324  0.7392  0.7383  0.7394  0.0046
Fit time          0.07    0.10    0.07    0.08    0.08    0.08    0.01
Test time         0.14    0.17    0.14    0.15    0.14    0.15    0.01
Evaluating RMSE, MAE of algorithm SVD on 5 split(s).

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    0.9539  0.9635  0.9673  0.9742  0.9539  0.9625  0.0078
MAE (testset)     0.7005  0.7082  0.7061  0.7110  0.7028  0.7057  0.0037
Fit time          0.45    0.41    0.43    0.44    0.47    0.44    0.02
Test time         0.04    0.04    0.04    0.04    0.05    0.04    0.00
Evaluating RMSE, MAE of algorithm SVDpp on 5 split(s).

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    0.9627  0.9615  0.9659  0.9582  0.9698  0.9636  0.0040
MAE (testset)     0.6998  0.6974  0.7065  0.6988  0.7019  0.7009  0.0032
Fit time          1.96    2.51    2.89    1.94    4.29    2.71    0.86
Test time         0.43    0.44    0.42    0.60    0.43    0.46    0.07
KNNWithMeans: RMSE = 1.0489, MAE = 0.7394
SVD: RMSE = 0.9625, MAE = 0.7057
SVDpp: RMSE = 0.9636, MAE = 0.7009
```



In conclusion, the developed suggestion system promises an important hope in using user reviews effectively to provide personalized product recommendations. This project ultimately creates a strong basis for further progress in suggestion systems, which aim to create a uninterrupted and enjoyable shopping experience for Amazon users.