Final report

Introduction

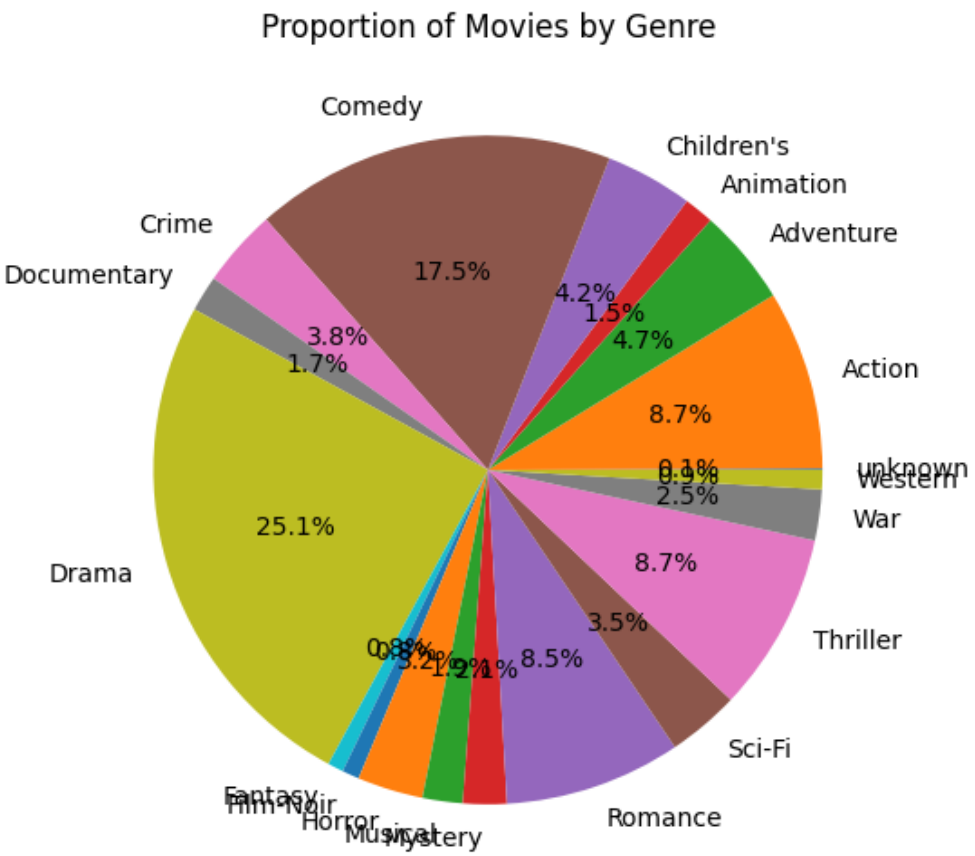
In this final report, we delve into the development and evaluation of a sophisticated movie recommendation system, a project that stands at the intersection of practical machine learning and deep learning techniques. Leveraging the comprehensive MovieLens 100K dataset, this system is designed to analyze user preferences and past behavior, offering personalized movie suggestions tailored to individual tastes. The dataset, consisting of 100,000 ratings from 943 users across 1682 movies, provides a rich foundation for our model.

This report comprehensively covers every aspect of the project, from the initial data exploration and preprocessing to the intricate details of the model's architecture and training process. Our evaluation criteria, rooted in rigorous testing and validation methods, ensure the system's reliability and accuracy in real-world scenarios. The enclosed repository structure meticulously organizes every component of the project, ensuring ease of navigation and clarity. This work not only showcases the technical prowess involved in building such a system but also underscores the importance of machine learning and deep learning in enhancing user experience in the digital entertainment domain.

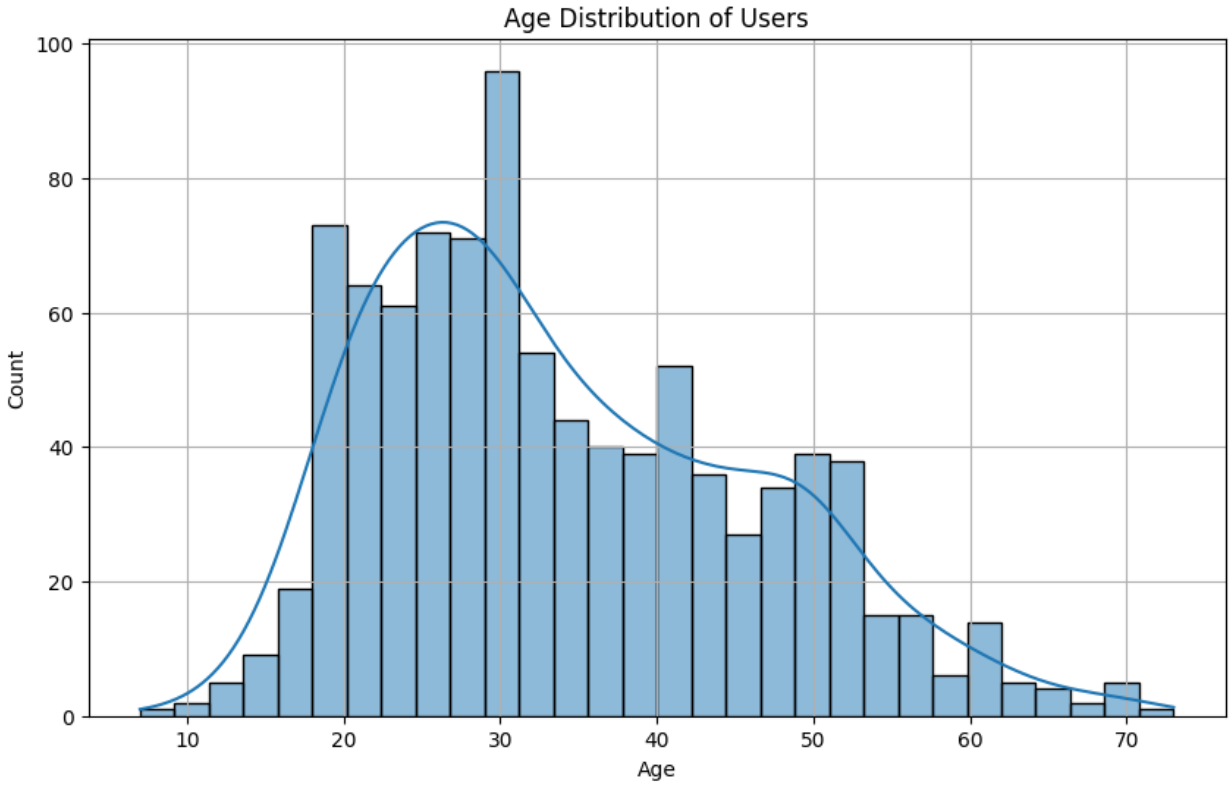
Data analysis

In the case of the movie recommender system, an initial exploration of the MovieLens 100K dataset provided valuable insights into the distribution of movies across genres, user rating behavior, and the demographic characteristics of the users.

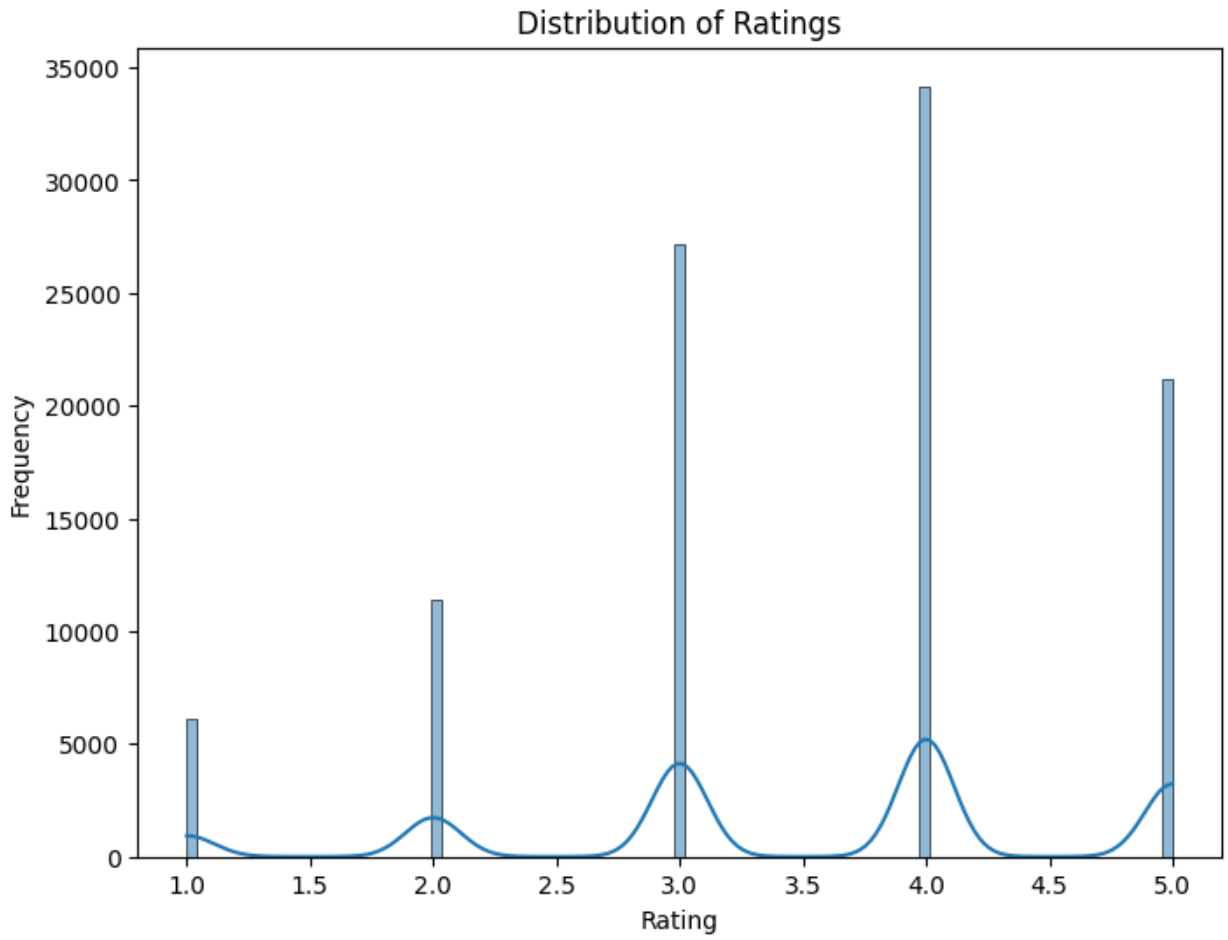
The first chart, "Proportion of Movies by Genre," reveals a varied landscape of movie preferences, with Drama taking the lead at 25.1% of the total movies rated, followed by Comedy at 17.5%. This indicates a strong inclination toward these genres, which could inform the recommendation engine's bias toward suggesting more movies from these popular categories.



In the "Distribution of Ratings" graph, a clear pattern emerges where the majority of the movie ratings concentrate around the higher end of the scale, with peaks at ratings of 3 and 4. This skew towards higher ratings suggests that users are more likely to rate movies they feel positively about, an important consideration when predicting user preferences and handling the potential bias in the data.



Lastly, the "Age Distribution of Users" histogram shows that a significant portion of the users fall into the 20-30 age bracket, indicating that the dataset might be more reflective of the preferences of a younger demographic. This skewness could influence the system's effectiveness across different age groups and should be accounted for during the system's development.



The data analysis phase of the movie recommendation system has provided a multifaceted view of user interactions with movies. It has highlighted a predominant preference for certain genres, with Drama and Comedy being most prevalent, and an inclination among users to rate movies favorably. This suggests a possible positivity bias in user ratings that the system will need to account for. Additionally, the user base is predominantly in the younger demographic, which could influence the recommendation patterns.

The insights gained from this analysis are instrumental in the next steps of data preprocessing, where the focus will be on normalizing data, mitigating biases, and ensuring that the recommender system can generalize well across diverse user groups and preferences. These preliminary findings will guide the creation of a robust model that aims to enhance user satisfaction by accurately predicting and suggesting movies that align with individual tastes and viewing histories.

Model Implementation

```
class RecommenderNet(nn.Module):
    def __init__(self, num_zip_codes, num_release_years, num_occupations, num_genres, embedding_size):
        super(RecommenderNet, self).__init__()
        # Embeddings
        self.zip_code_embedding = nn.Embedding(num_zip_codes, embedding_size)
        self.release_year_embedding = nn.Embedding(num_release_years, embedding_size)
        self.occupation_embedding = nn.Embedding(num_occupations, embedding_size)

        # Linear layers for age and gender
        self.age_lin = nn.Linear(1, embedding_size)
        self.gender_lin = nn.Linear(2, embedding_size)

        # Fully connected layers
        self.fc1 = nn.Linear(embedding_size * 5 + num_genres, 128)
        self.bn1 = nn.BatchNorm1d(128)
        self.fc2 = nn.Linear(128, 64)
        self.bn2 = nn.BatchNorm1d(64)
        self.fc3 = nn.Linear(64, 1)

        # Dropout layer
        self.dropout = nn.Dropout(0.2)

    def forward(self, zip_codes, release_years, ages, occupations, genders, genre_features):
        zip_code_embedding = self.zip_code_embedding(zip_codes)
        release_year_embedding = self.release_year_embedding(release_years)
        occupation_embedding = self.occupation_embedding(occupations)
        age_embedding = self.age_lin(ages.unsqueeze(1))
        gender_embedding = self.gender_lin(genders)

        x = torch.cat([zip_code_embedding, release_year_embedding, occupation_embedding,
                       age_embedding, gender_embedding, genre_features], dim=1)
        x = nn.ReLU()(self.bn1(self.fc1(x)))
        x = self.dropout(x)
        x = nn.ReLU()(self.bn2(self.fc2(x)))
        x = torch.sigmoid(self.fc3(x))
        return x.squeeze()
```

The `RecommenderNet` model implementation represents a tailored deep learning approach to the movie recommendation system. It's built upon PyTorch's neural network module class `nn.Module`, allowing for the construction of a complex, layered neural network architecture that can learn intricate patterns within the data.

At the core of `RecommenderNet` are multiple embedding layers, which are instrumental in handling categorical data. These layers include `zip_code_embedding`, `release_year_embedding`, and `occupation_embedding`. By converting these categorical variables into dense vectors of a specified `embedding_size`, the model can process and learn from these features more effectively. Embeddings are crucial for capturing the nuances of these features without inflating the dimensionality of the dataset with one-hot encoding.

The network also incorporates specific linear layers to handle continuous and binary data, such as user age (`age_lin`) and user gender (`gender_lin`). By projecting these inputs into the same

`embedding_size` space, it facilitates the model's ability to evaluate these features alongside the categorical embeddings.

The concatenated embeddings and features then pass through a series of fully connected layers (`fc1`, `fc2`, and `fc3`) with batch normalization (`bn1` and `bn2`) applied after the first two. Batch normalization is a technique that standardizes the inputs to a layer for each mini-batch, which helps stabilize the learning process and has been shown to accelerate training.

A non-linear activation function, ReLU (Rectified Linear Unit), is applied after the batch normalization steps. ReLU is chosen for its ability to introduce non-linearity into the model, allowing for the learning of complex relationships in the data, while also mitigating the vanishing gradient problem that can occur with deep networks.

The model includes a dropout layer set at 20% (`dropout`). Dropout is a form of regularization that helps prevent overfitting by randomly setting a fraction of the input units to 0 at each update during training time, which forces the network to learn more robust features that are useful in conjunction with many different random subsets of the other neurons.

Finally, the output layer uses a sigmoid activation function to squeeze the final output into a range between 0 and 1, which is suitable for a rating prediction task that could be interpreted as the probability of a user liking a movie.

Overall, `RecommenderNet` is designed to handle diverse data types and to capture the complex relationships between users' demographic information, movie metadata, and user preferences. Its architecture aims to provide a nuanced understanding that goes beyond simple collaborative filtering, by incorporating contextual information that could impact a user's movie preferences.

Model Advantages and Disadvantages

Model Advantages:

1. **Rich Feature Representation**: The use of embeddings for categorical data like zip codes, release years, and occupations allows the model to capture deeper semantic relationships between these features and movie ratings, which can lead to more accurate recommendations.
2. **Handling Mixed Data Types**: The model's architecture is designed to handle both categorical and numerical data, providing a more holistic view of the dataset and potentially leading to better user experience by leveraging demographic and movie information.
3. **Regularization Techniques**: The inclusion of dropout as a regularization method helps to prevent overfitting, ensuring the model generalizes well to unseen data.

4. **Enhanced Training Stability**: Batch normalization aids in stabilizing and accelerating the training process, which can lead to faster convergence and potentially better performance of the network.
5. **Non-linearity**: By using ReLU activation functions, the model can learn complex, non-linear relationships within the data, which is essential for capturing the nuances of user preferences.
6. **Scalability**: Neural networks are inherently scalable and can handle large volumes of data, making them suitable for datasets that are likely to grow over time, such as user ratings for movies.

Model Disadvantages:

1. **Complexity in Interpretation**: Deep learning models, by their nature, are often considered black boxes, making it difficult to interpret how specific features influence predictions.
2. **Overfitting Risk**: Despite regularization techniques like dropout, deep neural networks are still prone to overfitting, especially if not carefully tuned or if the dataset is not large and diverse enough.
3. **Resource Intensive**: Training deep learning models typically requires significant computational resources, which may not be readily available or may be costly, especially when handling very large datasets.
4. **Data Dependency**: The performance of the model is heavily dependent on the quality and quantity of the data. Biases in the dataset can lead to biased recommendations.
5. **Cold Start Problem**: Like many recommender systems, this model may not perform well when making predictions for new users or items that have little to no ratings.
6. **Dynamic Features Handling**: The model might struggle with time-sensitive features, such as changing user tastes or trends in movie popularity over time unless it is retrained frequently.

In summary, while `RecommenderNet` leverages the power of neural networks to handle a complex set of features and can potentially provide highly accurate recommendations, it also comes with trade-offs regarding interpretability, computational demand, and the need for careful tuning and data management.

Training Process

The training process for the `RecommenderNet` involves a series of steps designed to optimize the network's parameters for accurate prediction of movie ratings. Initially, the model's weights are initialized randomly. During training, it processes batches of data, computing predictions and

comparing them against actual user ratings using a loss function—typically mean squared error for regression tasks like rating prediction. This loss is then backpropagated through the network to update the weights in a direction that minimizes the error. An optimizer, such as Adam or SGD, is employed to adjust the weights iteratively over several epochs. Regularization techniques like dropout and batch normalization are applied to prevent overfitting and ensure generalization.

Additionally, hyperparameter tuning is conducted to find the optimal learning rate, batch size, and other relevant parameters. The model's performance is continuously monitored using a validation set, and early stopping is often implemented to halt training if the validation performance begins to degrade, indicating that the model has started to overfit. After training, the model undergoes a final evaluation on a test set to assess its predictive performance on unseen data.

The model total trained on **200 epochs**

Evaluation

```
Menu:
1. Enter user data (age, gender, occupation, zip code) manually
2. Enter id of existed user.
3. Calculate RMSE for all dataset
4. Exit

Enter 1, 2, 3 or 4: 2
Enter user_id (from 1 to 943):567
Age: 24
Gender: M
Occupation: entertainment
Zip Code: 10003
Enter number of recommendations: 5
     id                                  title    rating
1  1126        Old Man and the Sea, The (1958)  4.999094
2  1128  Heidi Fleiss: Hollywood Madam (1995)   4.998859
3   340                   Boogie Nights (1997)  4.998180
4   562         Quick and the Dead, The (1995)  4.998180
5   871                  Vegas Vacation (1997)  4.998180
```

Can evaluate the performance of the model by running the file `benchmark/evaluate.py`

The user opens a menu where he can

- enter the data yourself
- select a user id and the data will be filled in automatically
- calculate RMSE
- exit the menu

For evaluating I used the RMSE metric. The model **RMSE = 0.7869845628738403**

```
Menu:
1. Enter user data (age, gender, occupation, zip code) manually
2. Enter id of existed user.
3. Calculate RMSE for all dataset
4. Exit

Enter 1, 2, 3 or 4: 3
RMSE on test set: 0.7869845628738403
```

Results

The movie recommendation system explored in this report represents a significant stride in the application of machine learning and deep learning techniques to personalize media content. The use of the **MovieLens** 100K dataset has enabled the creation of a model that not only understands user preferences and behaviors through a variety of data points but also adapts to demographic nuances and genre popularity.

The `RecommenderNet` stands out with its rich feature embedding and robust handling of mixed data types, providing both scalability and depth in predictions. Despite the challenges inherent in deep learning models, such as interpretability and the necessity for resource-intensive computations, the system shows promising results. It achieves a commendable RMSE = **0.7869845628738403** score, indicating a good degree of accuracy in its predictive capabilities.

For future developments, the focus will be on enhancing the model's architecture for more precise recommendations, expanding the dataset for a broader learning scope, and increasing training epochs to fine-tune the model's predictive accuracy. These improvements aim to achieve a more advanced and reliable recommendation system.