# Movie Recommendation System

## Project Overview

This project implements a comprehensive movie recommendation system that provides personalized movie suggestions based on user preferences. It utilizes various methodologies, including collaborative filtering, autoencoders, deep learning, and transformer architectures, to enhance recommendation accuracy and user experience.

## Table of Contents

## Technologies Used

- **Programming Languages:** Python
- **Libraries:**
  - TensorFlow
  - scikit-learn
  - Pandas
  - NumPy
  - Matplotlib
  - FuzzyWuzzy
  - OpenAI API (if applicable)
- **Data Storage:** Pandas DataFrames
- **Modeling Techniques:**
  - K-Nearest Neighbors (KNN)
  - Autoencoders
  - Neural Collaborative Filtering (NCF)
  - Transformer Models

## Dataset

The project utilizes the MovieLens dataset for training and testing the recommendation models. The dataset contains user ratings for a wide range of movies, which is used to create user-item interaction matrices.

## Key Features

- **Personalized Recommendations:** Provides tailored movie suggestions based on user preferences and historical data.
- **Multiple Recommendation Approaches:** Implements KNN, autoencoders, NCF, and transformer models to enhance recommendation accuracy.
- **Performance Evaluation:** Evaluates model performance using precision and recall metrics, visualizing results for easy comparison.
- **User-friendly Interface:** (If applicable) Created a simple interface for users to interact with the recommendation system.

## Project Structure

perl
Copy code
```
movie-recommendation-system/
│
├── data/                      # Contains dataset files
│   └── movies.csv
│   └── ratings.csv
│
├── notebooks/                 # Jupyter notebooks for exploratory 
data analysis
│   └── data_exploration.ipynb
│
├── models/                    # Model implementations
│   └── knn_model.py
│   └── autoencoder_model.py
│   └── ncf_model.py
│   └── transformer_model.py
│
├── requirements.txt           # Python package dependencies
└── README.md                  # Project documentation
```

## Methodologies

1. **Data Processing:**
    - Loaded and cleaned the MovieLens dataset to create user-item interaction matrices.
    - Handled missing values and transformed the data to ensure compatibility with different models.
2. **Modeling Approaches:**
    - **K-Nearest Neighbors (KNN):**
        - Implemented KNN to identify similar users and items based on ratings, achieving baseline metrics of precision 0.5 and recall 0.1.
    - **Autoencoders:**
        - Developed autoencoder models to learn user and item embeddings, capturing latent features and improving recommendation quality.
    - **Neural Collaborative Filtering (NCF):**
        - Designed and trained an NCF model to capture complex interactions between users and items, enhancing recommendation accuracy.
    - **Transformer Model:**
        - Leveraged transformer architectures to utilize attention mechanisms for improved modeling of user preferences.
3. **Evaluation:**
    - Assessed the performance of each model using precision and recall metrics.
    - Visualized the results using Matplotlib to compare the effectiveness of different methodologies and inform optimizations.

## Installation

Clone the repository:
bash
Copy code

```
git clone
https://github.com/Vikramjeetsingh07/Movie-Recomendation-system-using-KNN-Autoencoders-NCF-transformers
cd movie-recommendation-system
```

1.

Install the required packages:
bash
Copy code

```
pip install -r requirements.txt
```

2.

## Usage

Load the dataset and preprocess it:
python
Copy code
```
import pandas as pd
from your_preprocessing_module import preprocess_data

movies = pd.read_csv('data/movies.csv')
ratings = pd.read_csv('data/ratings.csv')
user_item_matrix = preprocess_data(movies, ratings)
```

1.

Train and evaluate models:
python
Copy code
```
from your_model_module import train_knn_model,
train_autoencoder_model, train_ncf_model, train_transformer_model

knn_model = train_knn_model(user_item_matrix)
autoencoder_model = train_autoencoder_model(user_item_matrix)
ncf_model = train_ncf_model(user_item_matrix)
transformer_model = train_transformer_model(user_item_matrix)
```

2.

Generate recommendations:
python
Copy code
```
recommendations = knn_model.recommend(user_id)
```

3.

## Evaluation

The models are evaluated using precision and recall metrics. Results are visualized to assess the strengths and weaknesses of each approach, leading to informed optimizations.

## Results

- **KNN:** Precision: 0.5, Recall: 0.1

- **Autoencoders:** (Add specific metrics)
- **Neural Collaborative Filtering:** (Add specific metrics)
- **Transformer Model:** (Add specific metrics)

(You can include plots or visual representations of results here.)

## Contributing

Contributions are welcome! Please feel free to submit a pull request or raise an issue to discuss improvements and enhancements.

## License

This project is licensed under the MIT License - see the LICENSE file for details.