HOMEWORK 4

NAME : ISHANA SHINDE , KAVYA SUDHA KOLLU
[redacted]
RMSE SCORE: 0.82
RANK: 260

README :

1. I used Google Collab Pro(to get more RAM) for this assignment.
2. Use !pip install on Collab to install the python libraries and packages.
3. Upload the training data and test data files train.dat, test.dat and open it as Movie_data and Movie_test_data.
4. Run the cells on Google Collab Pro.

Introduction:

Recommender systems are utilized in different ways for different objectives. These systems can be used to recommend movies, books, music, etc. The objective of this assignment is to build a Movie Recommender System to predict the 5-star rating a movie will get for a given user.

Approach:

Main steps include

1. Reading data into dataframes
2. Features used for implementation
3. Model Selection
4. Choosing the best model.

Data Preprocessing:

The training dataset consists of UserID, MovieID, and Ratings as the main features. Along with the training data there are additional files containing details of Movie actors, directors, genres, and tags.

1. **Reading data into dataframes:**

- For this part the data files used were train.dat and test.dat.
- Both the train and test data are read by pd.read_table() with the separator being '\t' from pandas library as a Dataframe.
- The training dataset consists of 3 features - UserID, MovieID and Ratings.
- The features from train data are stored into Movie data.

- Rating from train data is stored in Movie_rating and is dropped from Movie data.
- Movie_genres is added into genre_data which is added as a new column into Movie_data.

2. **Features selection for implementation and dimensionality reduction:**

- Movie id and user id are considered as the key features for the implementation.
- **PCA** - Principal component analysis was used in order to reduce dimensionality. Use **StandardScaler** to help you standardize the dataset's features onto unit scale (mean = 0 and variance = 1)and then pca with 2 components is applied to fit and transform the data. This gave us an rmse score of 0.85 which was higher than the original one so was discarded.
- We performed **matrix factorization** on Movie data to generate latent features when multiplying two different kinds of entities.For this we are using the Non-negative Matrix Factorization. Once matrix factorization was done to train and test data we passed the data to the XGB model and the RMSE value for the model was 0.98 so was discarded.
- **Genres** have also been considered which were added as a new column but the output file generated gave an rmse value of 0.89 so has been discarded.

Model Selection:

We used different models like KNN, XGboost, Stochastic Gradient Descent, Decision trees, Naive Bayes etc. to fit and predict the ratings for the movies:

1. **KNN**
   For K Nearest Neighbor we used the regression model(sklearn KNeighborsRegressor)as the ratings column contains continuous data. The parameters for this model were n_neighbors = 300 and metric = 'cosine' where it computes the cosine distance. This model gave a RMSE score of 0.99 on cross validation and 1.23 on miner.

2. **XGBoost**
   XGBoost is a powerful approach for building supervised regression models.XGBoost dominates structured or tabular datasets on classification and regression predictive modeling problems. For XGBoosts we used XGBRegressor. It was imported into the code as ***import xgboost as xg*** The model was passed in with different parameters like n_jobs = 15, random_state =

15, n_estimators = 1000. ***The model gave us a RMSE score of 0.82 and this was the best RMSE score achieved.***

3. **Stochastic Gradient Descent**
   In Stochastic Gradient Descent, the gradient of the loss is estimated each sample at a time and the model is updated along the way with a decreasing strength schedule (aka learning rate). Here we use the SGDRegressor by sklearn along with that we first use the StandardScalar function to scale the data. The parameters used for this model are max_iterations = 1000 and the stopping criterion. The RMSE score for this model is 0.99.

4. **Decision Tree**
   We implemented decision trees as well for our model selection by using the DecisionTreeRegressor(). It is a 1D Regression model. The decision trees are used to fit a sine curve with additional noisy observation. As a result, it learns local linear regressions approximating the sine curve. The RMSE for this model was 0.98.

5. **Naive Bayes**
   Bayesian regression allows a natural mechanism to survive insufficient data or poorly distributed data by formulating linear regression using probability distributors rather than point estimates. The output or response ‘y’ is assumed to be drawn from a probability distribution rather than estimated as a single value.The Naive Bayes Regression model BayesianRidge() used on the preprocessed data gave a RMSE score 1.00.

6. **AdaBoostRegressor**
   An AdaBoost regressor AdaBoostRegressor() is a meta-estimator that begins by fitting a regressor on the original dataset and then fits additional copies of the regressor on the same dataset but where the weights of instances are adjusted according to the error of the current prediction. As such, subsequent regressors focus more on difficult cases.The RMSE for this model was 1.01.

7. **Random Forest Regressor**
   A random forest RandomForestRegressor() is a meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.The RMSE for the model was 0.97.

CONCLUSION:

The final result of this assignment is a prediction file with ratings and an RMSE score of 0.82. As the dataset was very large and with a large amount of features there is still more room for improvement in terms of feature extraction. For cold start users we can incorporate the additional files like the movie tags where we can use TF-IDF Vectorization and maybe compute the similarity using Dice coefficient.