Documentation

```
NAME
    collaborative_filtering

CLASSES
    builtins.object
        Collaborate

    class Collaborate(builtins.object)
     |  Collaborate(M)
     |
     |  Class to perform collaborative filtering with and without baseline approach
     |
     |  Methods defined here:
     |
     |  __init__(self, M)
     |      Initialize utility (ratings) matrix
     |      Note: Matrix needs to have items as rows and users as columns
     |
     |      Input:
     |      M (numpy.ndarray): Input Matrix
```

```
 |
 |  comp(self, k=2, baseline=False)
 |      Fills gaps in utility matrix using CF predictors
 |
 |      Input:
 |      k (int): Nearest neighbours taken based on similarity (default = 2)
 |      baseline (bool): Toggle baseline offset (default = False)
 |
 |  predictor(self, user, item, k=2, baseline=False)
 |      estimates rating for a given input user and item.
 |
 |      Input:
 |      user (int): Index of User
 |      item (int): Index of Item
 |      k (int): Nearest neighbours taken based on similarity (default = 2)
 |      baseline (bool): Toggle baseline offset (default = False)
 |
 |  ----------------------------------------------------------------------
 |  Data descriptors defined here:
 |
 |  __dict__
 |      dictionary for instance variables (if defined)
 |
 |  __weakref__
 |      list of weak references to the object (if defined)

DATA
    INT_MIN = -2147483648
    maxsize = 2147483647



NAME
    svd

CLASSES
    builtins.object
        SVDAlgorithm

    class SVDAlgorithm(builtins.object)
     |  Methods defined here:
     |
     |  __init__(self)
     |      Initialize self.  See help(type(self)) for accurate signature.
     |
     |  eigen_decomposition(self, M)
     |      Returns Eigen values and corresponding eigen vectors arranged in descending order.
     |
     |      @params:
     |      M: Input numpy matrix
     |
     |      Output:
     |      Returns list - sorted_eigen_values, sorted_eigen_vectors
     |      sorted_eigen_values - list of sorted eigen_values
     |      sorted_eigen_vectors - numpy matrix containing eigen vectors
     |
     |  svd(self, M, dimension_reduction=1.0)
     |      Applies Singular Value Decomposition to input matrix M - minimum reconstruction
     |      error of M expressed as U, sigma and V such that M = U * sigma * V
     |
     |      Supports dimensionality reduction where least values of sigma are removed along with
     |      their corresponding U columns and V rows.
     |
     |      @params:
     |      M : Input numpy matrix M
     |      dimension_reduction: Reduce the dimensions. Recommended range: 0.8 - 1.0
     |
     |      Output:
     |      Returns list - U, sigma, V
     |      sigma - singular values of M
     |
     |  ----------------------------------------------------------------------
     |  Data descriptors defined here:
     |
     |  __dict__
     |      dictionary for instance variables (if defined)
     |
     |  __weakref__
     |      list of weak references to the object (if defined)




    NAME
        cur


    FUNCTIONS
        column_selection(M, m_square_sum, c, repeat_allowed=False)
            Column selection algorithm


            Input:
            M: Input numpy matrix M
            m_square_sum: Sum of squares of elements of M
```

c: number of columns to select

repeat: Repetition allowed

cur(M, c, r, dim_red=None, repeat=None)

CUR function returns C,U,R

Input:

M: input numpy array

c: Number of column selections

r: Number of row selections

repeat: Repetition allowed

NAME

similarity_funcs

FUNCTIONS

pearson_sim(M, x, y)

Pearson correlation coefficient of two rows M(x) and M(y)

Input:

```
        M (numpy.ndarray): Input Matrix

        x (int): Index of first item

        y (int): Index of second item



NAME
    error_funcs

FUNCTIONS
    rmse(M, M_p)
        Computes Root Mean Square Error.

        Input:
        @M - Actual numpy array
        @M_p - Predicted numpy array

        Returns: Root Mean square error - float

    spearman_correlation(M, M_p)
        Returns Spearman score for the prediction.
        Formula: 1 - [sum(diff(predicted - actual)^2) / n((n^2)-1)]

        Input:
        @M - Actual numpy array.
        @M_p - Predicted numpy array.

        Returns:
        Spearman score - float

    top_k(k, M, M_p, ignore=True)
        Returns precision of predicted results in top k ratings.
        Input:
        @M - Actual numpy array.
        @M_p - Predicted numpy array.
        @ignore - Ignores already rated values.

        Returns:
        Precision of predictions in top K - float
```

```
NAME
    data_handling

CLASSES
    builtins.object
        CleanData

    class CleanData(builtins.object)
     |  CleanData(filename=None)
     |
     |  Helper class to structure dataset. Save's final matrix into a .npy file
     |
     |  Methods defined here:
     |
     |  __init__(self, filename=None)
     |      Class initialized with 135359 * 220970 given in the dataset documentation.
     |      Link: http://files.grouplens.org/datasets/movielens/ml-100k.zip
     |
     |  process(self, limit_users=None)
```

```
 |      Initializes output matrix and fills the matrix with ratings according to the dataset.
 |
 |      Input:
 |      @limit - number of entries in the dataset to be considered.
 |
 |      Output:
 |      Dataframe and numpy array saved as 'data_df.csv' and 'data_np.npy' respectively.
 |
 |  read_data(self, filename)
 |      Returns a pandas dataframe of the dataset with columns labelled as 0,1,2.
 |
 |  ----------------------------------------------------------------------
 |  Data descriptors defined here:
 |
 |  __dict__
 |      dictionary for instance variables (if defined)
 |
 |  __weakref__
 |      list of weak references to the object (if defined)

DATA
    MOVIE_ID = 1
    RATINGS = 2
    USER_ID = 0
```

The documentation for this class was generated from the following file:

- C:/Users/BEJJANKI ADITYA/PycharmProjects/IR-3/collaborative_filtering.